U. S. Army Research Office

Report No. 81-3

August 1981

ARO (D)
R-81-

PROCEEDINGS OF THE 1981 ARMY NUMERICAL ANALYSIS AND COMPUTERS CONFERENCE

Sponsored by the Army Mathematics Steering Committee

HOST

U. S. Army Missile Command
Redstone Arsenal, Alabama
26-27 February 1981

PROPERTY OF U.S. ARMY
STINFO BRANCH
BRL, APG, MD. 21005

Approved for public release; distribution unlimited.
The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

U. S. Army Research Office
P. O. Box 12211
Research Triangle Park, North Carolina

FOREWORD

These Proceedings preserve in print most of the invited addresses and contributed papers of the 1981 Army Numerical Analysis and Computers Conference. The Army Mathematics Steering Committee (AMSC) sponsors these meetings on behalf of the Office of the Chief of Research, Development and Acquisition. Members of this committee insist that the guest lecturer be internationally known scientists who are effective researchers and are presently working in frontier fields of current interest. They feel that the addresses by the invited speakers as well as the contributed papers by Army personnel will stimulate the interchange of ideas among the scientists attending said meetings.

Under the date of 15 October 1980, Colonel Robert J. Feist, Acting Director, US Army Missile Command, issued a formal invitation to hold the 1981 conference at his installation. Part of his letter to Dr. Jagdish Chandra, Chairman of the AMSC is quoted below:

> The Army Mathematics Steering Committee is invited to hold its 1981 Numerical Analysis and Computers Conference at the U. S. Army Missile Command. The dates 25-26 February 1981 are suggested as being a suitable time for this purpose.
>
> The Army Missile Command is looking forward to offering this opportunity for mathematicians and other scientists doing research for the Army to share their ideas with each other and with this command.
>
> The MICOM point of contact in making further arrangements is Dr. B. Z. Jenkins, Autovon 746-7279.

This is the second in this series of conferences to have as its host the U. S. Army Missile Command. The 1978 conference was held at Redstone Arsenal and had Dr. S. H. Lehnigk as its Chairman on Local Arrangements. This year Dr. B. Z. Jenkins served in this capacity. Both of these gentlemen are members of the AMSC, and both of them did an excellent job of handling the many details needed to conduct meetings of this size.

The theme of this year's conference was "Mathematical Software". Not only did the invited speakers treat this important area but many of the contributed papers emphasized it. Preceding the conference on 24-25 February 1981 a tutorial on "Software Reliability" was offered by Professors B. Littlewood of the City University of London (on leave at George Washington University) and V. Basili of the University of Maryland. Another special event was an evening session on the "UNIX Operating System" for certain computers manufactured by the Digital Equipment Corporation. The speakers for this meeting were Drs. B. Henriksen and Fred Bunn of the Ballistic Research Laboratory, together with Professor R. Riesenfeld of the University of Utah. The names of the invited speakers and the titles of their addresses are noted on the following page.

Speaker and Affiliation	Title of Address
Dr. W. J. Cody Argonne National Laboratory	OBSERVATIONS ON THE MATHEMATICAL SOFTWARE EFFORT
Professor B. F. Caviness Rensselaer Polytechnic Institute	ALGEBRAIC COMPUTATION
Professor S. F. McCormick Colorado State University	NUMERICAL SOFTWARE FOR FIXED POINT MICROPROCESSOR APPLICATIONS AND FOR FAST IMPLEMENTATION OF MULTIGRID TECHNIQUES
Professor L. J. Osterweil University of Colorado	STRATEGY FOR INTEGRATING PROGRAM TESTING AND ANALYSIS TOOLS

The success of this conference was due to many scientists, including the active and enthusiastic members of the audience, the chairpersons of the sessions and the authors of the papers. The members of the AMSC would like to thank these gentlemen for taking time to prepare papers for these proceedings so that persons unable to attend this symposium can profit by their contributions to the scientific literature.

TABLE OF CONTENTS*

Title	Page
Foreword	iii
Table of Contents	v
Agenda	ix
OBSERVATIONS ON THE MATHEMATICAL SOFTWARE EFFORT W. J. Cody	1
SOFTWARE RELIABILITY ESTIMATION THROUGH FITTING A POPUPATION PROCESS TO DATA Marc R. Stromberg	17
EXAMPLE APPLICATIONS OF SOFTWARE RELIABILITY ESTIMATION Leslie F. Claudio and Donna L. Branch	41
PROBABILISTIC PROGRAM ESTIMATES - COMPARISON OF SIMULATED RESULTS USING BETA VIS-A-VIS TRIANGULAR ACTIVITY DISTRIBUTIONS Conrad W. Faber	59
ON THE DISTRIBUTION OF A LINEAR COMBINATION OF MULTINOMIAL VARIABLES John C. Conlon ..... AMSAA	85
SIMULATING THE ARMY MILITARY CONSTRUCTION PROCESS James H. Johnson	99
NUMERICAL SOLUTION OF THE BOUNDARY LAYER EQUATIONS AT THE SHOCK/ SURFACE INTERFACE BEHIND A HEMISPHERICAL BLAST WAVE Richard J. Pearson and James E. Danberg ..... BRL	117
A SIMPLE MODEL FOR PREDICTING THE BLAST LOADS ON BOX-LIKE STRUCTURES Klaus O. Opalka ..... B.R.L.	143
FRONT TRACKING FOR HYPERBOLIC CONSERVATION LAWS: A PROGRESS REPORT James Glimm and Oliver McBryan	165
DISCUSSION OF A VERTICALLY AVERAGED HYDRODYNAMIC MODEL USING BOUNDARY FITTED COORDINATES Billy H. Johnson and Joe F. Thompson	169
THE GEM CODE: DIRECT SOLUTIONS OF ELLIPTIC AND MIXED PROBLEMS WITH NON-SEPARABLE 5- AND 9-POINT OPERATORS Patrick J. Roache	189

*This Table of Contents lists only the papers that are published in this Technical Manual. For a list of all the papers presented at the 1981 Army Numerical Analysis and Computers Conference see the copy of the Agenda.

TITLE PAGE

A NEW VARIATIONAL METHOD FOR INITIAL VALUE PROBLEMS, USING PIECEWISE HERMITE POLYNOMIAL SPLINE FUNCTIONS
C. N. Shen and Julian J. Wu 195

TIME-STEPPING METHODS FOR SECOND-ORDER EVOLUTION EQUATIONS
Vassilios A. Dougalis and Steven M. Serbin 213

ON NUMERICAL BOUNDARY CONDITIONS FOR HYPERBOLIC SYSTEMS
Max D. Gunzburger and William J. Layton 221

EXTRAPOLATING METEOROLOGICAL DATA FOR ARTILLERY APPLICATIONS
Abel J. Blanco 233

COMPUTER-AIDED SOLUTION OF THE BACTERIAL SURVIVAL EQUATIONS IN MICROBIOLOGY II. - EIGENVALUES FOR HEAT DIFFUSION EQUATION FOR FINITE CYLINDERS AND PLATES BY COMPUTER
Chia Ping Wang and Ari Brynjolfsson 247

ALGORITHMS FOR SPARSE, SYMMETRIC, DEFINITE QUADRADIC λ-MATRIX EIGENPROBLEMS
David S. Scott and Robert C. Ward 261

SOFTWARE FOR ORDERING LARGE SPARSE LEAST SQUARES PROBLEMS PRIOR TO GIVENS REDUCTION
David Hume, James Litsey, and Robert Plemmons 267

TRANSVERSE CURRENTS AND OHMIC LOSSES OF MICROSTRIP OBTAINED FROM A MATRIX FORMULATION WHICH FACILITATES THEIR NUMERICAL CALCULATION
Peter J, McConnell and Robert L. Brooke 283

ASPECTS OF ALGERBRAIC COMPUTATION
B. F. Caviness 291

NUMERICAL SOFTWARE FOR FIXED POINT MICROPROCESSOR APPLICATIONS AND FOR FAST IMPLEMENTATION OF MULTIGRID TECHNIQUES
Steve McCormick 303

SAFE LIFE DESIGN OF GUN TUBES - SOME NUMERICAL METHODS AND RESULTS
Anthony P. Parker, Kim A. Sleeper, and Christopher P. Andrasic .. 311

GUN TUBE FATIGUE LIFE ESTIMATES- INFLUENCE OF RESIDUAL STRESS, CRACK GROWTH LAW AND LOAD SPECTRA
Donald M.Neal, Anthony P. Parker, and Eward M. Lenoe 335

NUMERICAL PREDICTION OF RESIDUAL STRESSES IN A AUTOFRETTAGED TUBE OF COMPRESSIBLE MATERIAL
P. C. T. Chen 351

DYNAMIC GUN TUBE BENDING ANALYSIS
Richard A. Lee, Jonathan F. Kring, and Dana S. Charles 363

TITLE PAGE

FINITE ELEMENT MODELING OF THE VULNERABILITY OF AN M-15 LAND MINE USING AN EXPLICIT INTEGRATION SCHEME
Frederick H. Gregory 375

ON THE ACCURACY OF FLOW RULE APPROXIMATIONS USED IN STRUCTURAL AND SOLID RESPONSE COMPUTER PROGRAMS
Joseph M. Santiago 413

MICROCOMPUTER IMPLEMENTATION OF CONTROL ALGORITHMS FOR WEAPON POINTING AND STABILIZATION
Jonathan Korn, Edward Carroll, Ronald Johnson, and Barry Kullack. 443

DATA ANALYSIS OF INTRUSION SIGNATURES FOR PHYSICAL SECURITY APPLICATIONS WITH A MINICOMPUTER
Wen-Wu Shen 471

NUMERICAL SOLUTION TO BEAM VIBRATIONS UNDER A MOVING COUPLE
Julian J. Wu 523

ANALYTICAL SOLUTIONS IN NUMERICAL ANALYSIS
J. L. Harris 549

A STRATEGY FOR INTEGRATING PROGRAM TESTING AND ANALYSIS
Leon Osterweil 555

A MODIFIED KROENIG-PENNEY MODEL
Francis E. Council, Jr. 609

COMPUTATION OF MATRIX CHAIN PRODUCTS
T. C. Hu and M. T. Shing 615

ATTENDANCE LIST 629

AGENDA FOR THE

1981 ARMY NUMERICAL ANALYSIS AND COMPUTERS CONFERENCE
26-27 February 1981
Huntsville, Alabama

Thursday
26 February 1981

0830-0845 WELCOMING REMARKS - Dr. W. C. McCokle, Technical Director, US Army Missile Command and Director, US Army Missile Laboratory

0845-0945 KEYNOTE ADDRESS

CHAIRPERSON - Dr. B. Z. Jenkins, US Army Missile Command, Redstone Arsenal, Alabama

SPEAKER - Dr. W. J. Cody, Argonne National Laboratory, Argonne, Illinois

TITLE - OBSERVATIONS ON THE MATHEMATICAL SOFTWARE EFFORT

0945-1015 BREAK

1015-1155 TECHNICAL SESSION I

CHAIRPERSON - Dr. Jonathan Kring, US Army Tank-Automotive Command, Warren, Michigan

SOFTWARE RELIABILITY ESTIMATION THROUGH FITTING A POPULATION PROCESS

Messrs. Marc R. Stromberg, Leslie F. Claudio and Ms. Donna L. Kaspersen, US Army Electronics Proving Ground, Ft. Huachuca, Arizona

EXAMPLE APPLICATIONS OF SOFTWARE RELIABILITY ESTIMATION

Messrs. Marc R. Stromberg, Leslie F. Claudio and Ms. Donna L. Kaspersen, US Army Electronics Proving Ground, Ft. Huachuca, Arizona

PROBABILISTIC PROGRAM ESTIMATES - COMPARISON OF SIMULATED RESULTS USING BETA VIS-A-VIS TRIANGULAR ACTIVITY DISTRIBUTIONS

Mr. Conrad M. Faber, US Army Aviation Research & Development Command, St. Louis, Missouri

THE DISTRIBUTION OF A LINEAR COMBINATION OF MULTINOMIAL VARIABLES

Dr. John C. Conlon, US Army Materiel Systems Analysis Activity, Aberdeen Proving Ground, Maryland

SIMULATING THE ARMY MILITARY CONSTRUCTION PROCESS, A HIERARCHIAL NETWORK WHICH PROCESSES CHARACTERIZED PROJECTS IN A DECISION FUNCTION ENVIRONMENT

Mr. J. H. Johnson, US Army Construction Engineering Research Laboratories, Champaign, Illinois

1015-1155 TECHNICAL SESSION II

CHAIRPERSON - Mr. Anthony Parker, US Army Materiels and Mechanics Research Center, Watertown, Massachusetts

NUMERICAL SOLUTION OF THE BOUNDARY LAYER EQUATIONS AT THE SHOCK/WALL INTERFACE BEHIND A HEMISPHERICAL BLAST WAVE

Drs. Richard J. Pearson and James E. Danberg, Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland

A SIMPLE MODEL FOR PREDICTING THE BLAST LOADS ON BOX-LIKE STRUCTURES

Dr. Klaus Otto Opalka, Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland

TRACKING OF DISCONTINUITIES IN TWO DIMENSIONAL FLUID FLOW--A PROGRESS REPORT

Professor James Glimm, The Rockefeller University, New York, New York

DISCUSSION OF A VERTICALLY AVERAGED HYDRODYNAMIC MODEL USING BOUNDARY FITTED COORDINATES

Dr. Billy H. Johnson, US Army Engineer Waterways Experiment Station, Vicksburg, Mississippi

HELE-SHAW FLOW WITH SUCTION

Professor Gunter H. Meyer, Georgia Institute of Technology, Atlanta, Georgia

1155-1330 LUNCH

1330-1510 TECHNICAL SESSION III

CHAIRPERSON - Dr. John C. Conlon, US Army Materiel Systems Analysis Activity, Aberdeen Proving Ground, Maryland

THE GEM CODE: DIRECT SOLUTIONS OF ELLIPTIC AND MIXED PROBLEMS WITH NON-SEPARABLE 5- AND 9-POINT OPERATORS

Dr. Patrick J. Roache, Ecodynamics Research Associates, Inc., Albuquerque, New Mexico

A NEW VARIATIONAL METHOD FOR INITIAL VALUE PROBLEMS, USING PIECEWISE HERMITE POLYNOMIAL SPLINE FUNCTIONS

Drs. C. N. Shen and Julian J. Wu, Benet Weapons Laboratory, Watervliet Arsenal, Watervliet, New York

TIME-STEPPING METHODS FOR SECOND-ORDER EVOLUTION EQUATIONS

Drs. Steven M. Serbin and Vassilios A. Dougalis, University of Tennessee, Knoxville, Tennessee

NUMERICAL BOUNDARY CONDITIONS FOR HYPERBOLIC SYSTEMS

Professor Max D. Gunzburger, The University of Tennessee, Knoxville, Tennessee

1330-1510 TECHNICAL SESSION IV

CHAIRPERSON - Dr. Billy H. Johnson, US Army Engineer Waterways Experiment Station, Vicksburg, Mississippi

EXTRAPOLATING METEOROLOGICAL DATA FOR ARTILLERY APPLICATIONS

Mr. Abel J. Blanco, US Army Atmospheric Sciences Laboratory, White Sands Missile Range, New Mexico

COMPUTER-AIDED SOLUTION OF THE BACTERIAL SURVIVAL EQUATIONS IN MICROBIOLOGY II - EIGENVALUES FOR HEAT DIFFUSION EQUATIONS FOR FINITE CYLINDERS AND PLATES BY COMPUTER

Drs. Chia Ping Wang and Ari Brynjolfsson, US Army Natick Research and Development Laboratories, Natick, Massachusetts

SOLVING QUADRATIC-MATRIX PROBLEMS WITHOUT FACTORIZATION

Dr. Robert C. Ward, Union Carbide Corporation-Nuclear Division, Oak Ridge, Tennessee

SOFTWARE FOR ORDERING LARGE SPARSE LEAST SQUARES PROBLEMS PRIOR TO GIVEN REDUCTION

Professors R. J. Plemmons and D. Hume, The University of Tennessee, Knoxville, Tennessee

TRANSVERSE CURRENT DISTRIBUTIONS AND OHMIC LOSSES OF MICROSTRIP OBTAINED FROM A MATRIX FORMULATION WHICH FACILITATES THEIR NUMERICAL COMPUTATION

Drs. Peter J. McConnell and Robert L. Brooke, US Army Mobility Equipment Research & Development Command, Ft. Belvoir, Virginia

1510-1540 BREAK

1540-1640 GENERAL SESSION I

CHAIRPERSON - Dr. Paul T. Boggs, US Army Research Office, Research Triangle Park, North Carolina

ALGEBRAIC COMPUTATION

Professor B. F. Caviness, Rensselaer Polytechnic Institute, Troy, New York on leave at General Electric Company, Research & Development Center, Schenectady, New York

1930-2100 SPECIAL SESSION ON "UNIX" IN THE RESEARCH ENVIRONMENT

CHAIRPERSON - Dr. Bruce Henriksen, Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland

OVERVIEW OF "UNIX"

Dr. Bruce Henriksen, Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland

"UNIX" TANK WARS

Dr. Fred Bunn, Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland

"UNIX" SOFTWARE SUPPORT FOR ALPHA_1

Professor Richard Riesenfeld, Computer Science Department, University of Utah, Salt Lake City, Utah

Friday
27 February 1981

0830-0930 GENERAL SESSION II

CHAIRPERSON - Dr. Norman Coleman, US Army Armament Research and Development Command, Dover, New Jersey

TITLE TO BE ANNOUNCED

Professor S. F. McCormick, Colorado State University, Ft. Collins, Colorado

0930-1000 BREAK

1000-1140 TECHNICAL SESSION V

CHAIRPERSON - Dr. Steven M. Serbin, University of Tennessee, Knoxville, Tennessee

SAFE LIFE DESIGN OF GUN TUBES - SOME NUMERICAL METHODS AND RESULTS

Messrs. Anthony P. Parker, Kim A. Sleeper and Christopher P. Andrasic, US Army Materials and Mechanics Research Center, Watertown, Massachusetts

GUN TUBE FATIGUE LIFE ESTIMATES - INFLUENCE OF RESIDUAL STRESS, CRACK GROWTH LAW AND LOAD SPECTRA

Messrs. Donald Neal, Anthony P. Parker and Edward M. Lenoe, US Army Materials and Mechanics Research Center, Watertown, Massachusetts

NUMERICAL PREDICTION OF RESIDUAL STRESSES IN AN AUTOFRETTAGED TUBE OF COMPRESSIBLE MATERIAL

Dr. P. C. T. Chen, Benet Weapons Laboratory, Watervliet Arsenal, Watervliet, New York

DYNAMIC GUN TUBE BENDING ANALYSIS

Drs. Richard Lee, Jonathan Kring and Lt. Steve Charles, US Army Tank-Automotive Command, Warren, Michigan

1000-1140 TECHNICAL SESSION VI

CHAIRPERSON - Mr. J. L. Harris, US Army Missile Command, Redstone Arsenal, Alabama

FINITE ELEMENT MODELING OF THE VULNERABILITY OF AN M-15 LAND MINE USING AN EXPLICIT INTEGRATION SCHEME

Dr. Frederick H. Gregory, Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland

OPTIMUM AIMING OF ARTILLERY INDIRECT FIRE

Mr. Richard S. Sandmeyer, US Army Materiel Systems Analysis Activity, Aberdeen Proving Ground, Maryland

ALGORITHMS BASED ON SINC APPROXIMATIONS

Professor Frank Stenger, The University of Utah, Salt Lake City, Utah

ON THE ACCURACY OF FLOW RULE APPROXIMATIONS USED IN STRUCTURAL AND SOLID RESPONSE COMPUTER PROGRAMS

Dr. Joseph M. Santiago, Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland

1140-1300 LUNCH

1300-1400 TECHNICAL SESSION VII

CHAIRPERSON - Mr. Richard S. Sandmeyer, US Army Materiel Systems Analysis Activity, Aberdeen Proving Ground, Maryland

MICROCOMPUTER IMPLEMENTATION OF CONTROL ALGORITHMS FOR WEAPON POINTING AND STABILIZATION

Dr. Jonathan Korn, Alphatech, Inc., and Messrs. Edward Carroll, Barry Kullack and Ronald Johnson, US Army Armament Research and Development Command, Dover, New Jersey

DATA ANALYSIS OF INTRUSION SIGNATURES FOR PHYSICAL SECURITY APPLICATION WITH A MINICOMPUTER

Dr. Wen-Wu Shen, US Army Mobility Equipment Research and Development Command, Ft. Belvoir, Virginia

NUMERICAL SOLUTION TO BEAM VIBRATIONS UNDER A MOVING COUPLE

Dr. Julian J. Wu, Benet Weapons Laboratory, Watervliet Arsenal, Watervliet, New York

1300-1400 TECHNICAL SESSION VIII

CHAIRPERSON - Dr. Joseph M. Santiago, Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland

NUMERICAL METHODS FOR THE APPLICATION OF DIGITAL FILTERS TO THE ANALYSIS OF BALLISTIC DATA

Dr. James M. Walbert, Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland

ANALYTICAL SOLUTIONS IN NUMERICAL ANALYSIS

Mr. J. L. Harris, US Army Missile Command, Redstone Arsenal, Alabama

1400-1500 GENERAL SESSION III

CHAIRPERSON - Mr. M. A. Hirschberg, Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland

STRATEGY FOR INTEGRATING PROGRAM TESTING AND ANALYSIS TOOLS

Professor L. J. Osterweil, University of Colorado, Boulder, Colorado

1500 ADJOURN

Observations on the Mathematical Software Effort

W. J. Cody *

Applied Mathematics Division
Argonne National Laboratory
Argonne, Illinois 60439

Abstract. John Rice introduced the term 'mathematical software' in 1969 to denote "the set of algorithms in the area of mathematics." Beginning with a meeting at Purdue University the following year, work on mathematical software has attracted ever more talented people and has steadily gained in professional recognition. In this paper we discuss certain milestones, both successes and disappointments, which mark this rise. We also examine the present work spectrum and discuss new problems arising from advancing technology and changing work patterns.

1. Introduction. John Rice coined the term 'mathematical software' in 1969, and focussed attention on the subject the following year with a symposium held as part of a Special Year in Numerical Analysis at Purdue University. The movement spawned by that first meeting has been fruitful. In 1969 only a few individuals worked on what we now call mathematical software, and only a few fortunate computer sites had access to decent numerical programs. Today many talented people work in the field, large collections of good numerical software are widely available, and specialized meetings are common.

A description of the mathematical software effort is difficult because it is so broad. Its domain is that nebulous region between the discovery of numerical algorithms and the consumption of numerical software. On the one hand numerical analysts devise new computational methods, and on the other hand individuals wish to apply effective methods to their immediate problems. It is the job of the mathematical software effort to bridge the gap by packaging numerical analysts' work in software appealing to the consumer. Strictly speaking, work on mathematical software is limited to tasks related to the implementation of numerical algorithms. In practice the spectrum of activities is surprisingly wide because the process of implementation is itself worthy of study. In addition to obvious concerns with program design and testing, there are major concerns with programming practices, documentation standards, software organization and distribution methods. Other activities involve the development of programming tools to partially automate design, implementation, testing and maintenance of software, and work on the computational environment, including the design of arithmetic systems and programming languages properly supportive of good numerical software. Major contributions have been made in each of these areas by individuals who consider their primary interest to be mathematical software.

The published proceedings of the Purdue meeting contain Rice's appraisal of the mathematical software effort as it stood in 1970 [45], including a chronological account of progress. This paper is a similar appraisal of the effort as it stands today. Instead of updating the chronological record, however, we discuss what we consider to be major milestones marking progress to this point. We

* This work was supported by the Applied Mathematical Sciences Research Program (KC-04-02) of the Office of Energy Research of the U.S. Department of Energy under Contract W-31-109-Eng-38.

then examine current problems in the field and future challenges posed by an advancing technology. This work was inspired by a panel session on the same subject that ended the week-long International Seminar on Problems and Methodologies in Mathematical Software Production held this past November in Sorrento, Italy, under the sponsorship of The University of Naples and the C.N.R. We gratefully acknowledge the contributions of our fellow panelists, B. Ford, T. J. Dekker, M. Gentleman, J. N. Lyness and P. C. Messina, and a responsive audience. With the benefit of leisurely reflection we have reorganized and expanded some of their ideas and combined them with our own thoughts on the matter. We alone are responsible for the selection and expression of the opinions that follow, however.

The reader should be aware that the views presented below may be colored by personal bias, and that other views exist. The surveys and suggestions for future research in [24,28,39,47,48] are especially recommended to the interested reader.

2. The Past. Many people associate the beginning of the mathematical software effort with Rice's 1969 call for a meeting at Purdue University [44]. The roots go back further, however. While it would be trite to trace them to the first numerical subroutine libraries, we detect an emerging concern for software quality in the early 1960's. By then individuals at The University of Toronto, The University of Chicago, Stanford University, Bell Laboratories and Argonne National Laboratory were critically examining software and advertising their findings through technical reports and discussions at computer user group meetings. The ideas and evaluation techniques were not well enough established for publication in refereed journals, however, and efforts were hampered by poor communications. Often workers at one location were completely unaware of similar work elsewhere. Yet each of these computing centers developed outstanding program libraries by contemporary standards.

In early 1966 J. F. Traub organized SICNUM, the Special Interest Committee on Numerical Mathematics. The group grew quickly, and by midyear when the first informal SICNUM Newsletter appeared it had a membership of almost 1000. Two articles in the first Newsletter typify SICNUM's interests. The first announced the establishment of a working group "to investigate testing and certification techniques for numerical subroutines," and the second announced a SICNUM sponsored evening session at the 1966 National ACM Conference. The session included a panel discussion "in the area of machine implementation of numerical algorithms." By constantly emphasizing efforts to improve the quality of numerical software, SICNUM and its successor SIGNUM set the stage for Rice's 1969 call for a symposium.

In his call Rice defined mathematical software as "computer programs which implement widely applicable mathematical procedures" [44]. This contrasts with the definition he later included in the published proceedings, "the set of algorithms in the area of mathematics" [45]. These two definitions illustrate the fundamental confusion between algorithms and computer programs that plagued the early development of numerical software. The realization that an implementation is different from the underlying algorithm marks the emergence of mathematical software as a separate field of endeavor.

That difference was not widely understood in 1969. Despite early admonitions from G. Forsythe [22] and other prominent researchers, most numerical analysts still believed that their work was finished when they had defined an algorithm. Computer programming was a job for computer programmers; numerical analysts only programmed when it was necessary for their research (and pure mathematicians never programmed). A university professor seeking advancement and tenure shied away from working on numerical software. As a result most of the early work was concentrated in government and industrial laboratories with only a few selfless university people involved. Unfortunately, the same attitudes are still common. While work on mathematical software has gained some professional stature and there are more talented people involved in the effort today than were involved in 1969, many others still do not dare to become involved if they seek promotion. This is still especially true at many universities.

Three software projects that greatly influenced the mathematical software effort began about the time of the Purdue meeting. Each project, IMSL, NAG and NATS, resulted in a widely-used collection of high-quality numerical software. Certain software collections were publically available before this. Computer user's groups had organized program repositories by the early 1960's, and the IBM Scientific Subroutine Package (SSP) was available on the IBM 7094, for example. Although these collections contained a few good programs, their general reputations were deservedly notorious. The IMSL, NAG and NATS collections were the first to combine quality with wide distribution.

IMSL, International Mathematical and Statistical Libraries, Inc., was founded in 1971 by some of the people involved in the IBM SSP effort. It delivered the first purely commercial numerical subroutine library to IBM customers a year later. By mid 1973 the library had also been delivered to UNIVAC and CDC customers, and for the first time the same library of numerical programs became available on a variety of computing equipment. This enabled numerical programmers to write and distribute applications programs without worrying about the availability of a decent support library. Today IMSL supports most major computers. The success of this venture is suggested by the number of computing centers now relying on IMSL and its competitors for their core library, thus freeing local personnel to develop the specialized programs necessary for their own work.

IMSL's main competition comes from the NAG and, to a lesser extent, PORT libraries. NAG, originally Nottingham Algorithms Group but now Numerical Algorithms Group, was organized about 1970 in Great Britain as a cooperative venture between universities using ICL 1906A computers. Supported by heavy government subsidies, NAG extended its coverage to other machines and now seeks to become self-supporting. The PORT library is a product of Western Electric arising from the early library work at Bell Laboratories. It is not aggressively marketed, and is therefore not as widely used as the IMSL and NAG libraries.

The NATS project, National Activity to Test Software, was conceived in 1970 and funded in 1971 by the National Science Foundation and the Atomic Energy Commission to study problems in producing, certifying, distributing and maintaining quality numerical software [5]. This was a cooperative effort between personnel at Argonne National Laboratory, Stanford University, The University of Texas at Austin and scattered test sites to examine software production as a research

problem. Intrinsic to this effort was the production of two software packages, the EISPACK collection of matrix eigensystem programs and the FUNPACK collection of special function programs. The project formally ended with the distribution of extended second releases of both packages in 1976 [11,23,51].

By any measure, the NATS project was a spectacular success. Not only did it produce superior software, but it also pioneered in organizational and technical achievements that are still being exploited. For example, the project developed an early system for automated program transformation and maintenance [50] that led directly to current research on the TAMPR system [6]. We believe that the NATS aids were developed before similar aids for program transformation were developed at JPL [31] and within the IMSL [3] and NAG [19] projects. They were certainly the first to be successfully used in a software project. Important as such technical achievements were, however, they were overshadowed by the organizational concepts the project developed. Machura and Sweet recently stated [36], "The most important lesson learned from the EISPACK project is that the development and distribution of quality software can be achieved by the joint efforts of several different organizations." Before the NATS success software was typically developed with the limited resources of one organization; since the NATS success cooperative ventures have become common.

None of this would have mattered if the NATS software had not been superior. Fortunately, the software produced by the project was well received and is still considered to be some of the best available. EISPACK, in particular, set and met high standards for performance, transportability and documentation. It has become a paradigm for thematic numerical software collections with the term 'PACK' now implying all that is good in numerical software. Attesting to EISPACK's influence, the following PACKs in addition to FUNPACK either exist or are in advanced planning stages: ELLPACK [46], FISHPAK [2], ITPACK [26], LINPACK [17], MINPACK [37], PDEPACK [42], QUADPACK [43], ROSEPACK [14], SPARSPAK [25], TESTPACK [9] and TOOLPACK [40]. While many of these are superb packages, the use of a 'PACK' name does not automatically instill quality.

It is disappointing that the NATS experience was not fully exploited. Attempts to establish a central organization for software production based on the NATS concept [15,16] failed for various political and technical reasons. This denied segments of the numerical software community access to experienced people and important resources. Many of the projects mentioned above had to rely on their own resources to coordinate production, certification and distribution of their software, duplicating similar capabilities already developed in other projects.

The first Purdue symposium was followed by two other important meetings. SIGNUM sponsored a meeting in 1971 in Ljubljana, Yugoslavia, concurrent with the 1971 IFIP Congress, that ultimately led to the establishment in late 1974 of WG 2.5, the IFIP Working Group on Numerical Software. Members of the working group now represent numerical software interests in language and hardware standardization efforts, often with detailed advice from the group as a whole. In addition the working group has organized several international workshops on software topics and has drafted and published several technical reports.

The other important meeting was the second mathematical software symposium held at Purdue in 1974. While its influence was not as great as that of the

first meeting, it did lead to the establishment of the ACM Transactions on Mathematical Software with John Rice as editor. Since its appearance in early 1975 with papers from the Purdue meeting, TOMS has complemented the SIGNUM Newsletter by providing an outlet for refereed numerical software papers.

The second Purdue meeting was also noteworthy for the first open discussion of the BLAs, or Basic Linear Algebra Subprograms [32]. As the name implies, the BLAs are a collection of Fortran subprograms implementing low level operations, such as the dot product, from linear algebra. The project was originally organized in 1973 as a private effort to reach consensus on names, calling sequences and functional descriptions for such programs, but it quickly became a cooperative effort officially sanctioned by ACM-SIGNUM. Once conventions had been agreed on, it was possible for linear algebra programs to do fundamental operations in a uniform way. This was already a significant accomplishment, but the group also prepared efficient implementations of the BLAs routines for most popular computers. The project's most important contribution, however, was the concept of establishing popular 'conventions' as opposed to official standards. Language designers are reluctant to augment standard languages to include something useful to only a small group. Even if that is done, years pass before the new feature is available in compilers. The establishment of private conventions outside language standards is a more reasonable approach, and the BLAs project demonstrates that it is also a practical one. As with NATS, this lead has not been fully exploited.

It is difficult to assess the importance of events in the immediate past, but we believe that the recently proposed IEEE standard for floating-point arithmetic will prove to be important. One major disappointment in numerical work has been the general lack of progress in designing clean computer arithmetic systems. High quality software is supposed to be fail-safe and transportable; it must work properly regardless of quirks in the host arithmetic system. Software production is seriously hampered when computer arithmetic violates simple arithmetic properties such as

$$1.0 * X = X$$
$$X * Y = Y * X,$$

and

$$X + X = 2.0 * X.$$

There exist mainframes of recent design in which each of these properties fails for appropriate floating-point X and Y. Worse yet, on some machines there exist floating-point $X > 0.0$ such that

$$1.0 * X = 0.0,$$
$$X + X = 0.0,$$

or

$$[\mathrm{sqrt}(X)]^2 = \text{overflow or underflow.}$$

All these anomalies are traceable to engineering economies [12]. Computer designers repeatedly ignore complaints about such mathematical atrocities, and new anomalies seem to appear with each new machine.

That may be changing however. By 1977 technology had advanced to the point where small microprocessor manufacturers considered adding floating-point

arithmetic to their chips. In an unprecedented move they turned to numerical analysts for advice. The result was the formation of a special subcommittee of the IEEE Computer Society to draft a standard for binary floating-point arithmetic. The draft they produced [1] is radically different from existing arithmetic systems. Not only is it free of anomalies, but it also contains new features specifically requested and designed by numerical analysts with software experience. The first chips based on this proposed standard have now appeared [29], and the first microcomputers are being delivered [30].

These, then, are the milestones leading to where we stand today: the early work at isolated computing centers, the establishment of SICNUM, the two Purdue Symposia, the establishment of commercial numerical software libraries, the NATS project and the EISPACK package, the establishment of IFIP WG 2.5 and of TOMS, the BLAs, and the drafting of a standard for floating-point arithmetic. Each of these events added something new and important to the movement. There have also been some disappointments. We mention in particular the failure to achieve full professional recognition for software work, especially at universities, the failure to fully exploit the NATS experience, and the general lack of progress in mainframe arithmetic design.

3. The Present. While the problems we face today are similar to those we faced ten years ago, the solutions have become more complicated. We are still concerned about the production of high-quality transportable software, but we expect more from such software now than we did in the past. Therefore it is more difficult to produce.

The last section pointed to many thematic numerical software packages. Some such as EISPACK and LINPACK are complete, while others such as MINPACK and QUADPACK are still under development. We believe it is significant that most of the early success involved linear algebra programs. It is true that linear algebra is a fundamental mathematical tool for other problem areas, such as optimization and partial differential equations, and that good software for these other problems was not likely to be produced until good linear algebra programs were ready. But it is also true that linear algebra had reached an algorithmic maturity that invited software production. The algorithms were well developed, well understood and backed by error analysis that clearly displayed the limitations of software implementations. Because the production of EISPACK required minimal algorithmic work the producers could concentrate on recasting algorithms to enhance desirable software attributes. The effort thus produced a significant software package within three years of funding. In contrast, the MINPACK effort required about five years to produce its first small package. This lengthy development time reflects the difficulty of the task and is likely to be typical of future projects. As in many other fields, prominent researchers in optimization do not agree on the best algorithms; new methods frequently appear accompanied by confusing claims of superiority over existing methods and programs. The situation is common in a vigorous, dynamic research field, but it does not encourage the quick production of high-quality software. All the 'easy' implementations may have been done already.

Despite these difficulties, we believe that some additional problem areas could be harvested for software now. We are frankly puzzled by the lack of an effort in ordinary differential equations, for example. Existing algorithms

seem to be well enough understood, but no group has emerged with the necessary dedication and support.

There is one other little-understood aspect of successful numerical software projects that we believe to be important. Part of the variation in quality in the numerous PACKs previously mentioned is due to an improper appreciation of a fundamental lesson from the NATS project. We stated above that linear algebra was in a good algorithmic position when the EISPACK work began. That does not mean the field was stagnant, however; new algorithms were being introduced. The project deliberately ignored new work because it felt that algorithms had to prove themselves before being included. Further, the project found that there is a one to two year delay between the completion of the first pass at software and its final release. This time is spent iteratively testing, revising and documenting to insure that the package does what it claims. Thus there must be a one to two year moratorium on the introduction of new material into the package. This simple discovery has far-reaching implications. Algorithmic researchers find it almost impossible to observe such a moratorium; they are intent on wide distribution of their latest discoveries. Further, they cannot effectively polish software they feel to be inferior. Therefore, control over software projects should be vested in individuals who understand and are dedicated to software production rather than in individuals who primarily produce algorithms. Algorithm producers should be involved in software packaging, but they should not control it.

There is another advantage to this approach. Software packages require a uniformity of style to simplify documentation and maintenance. As EISPACK demonstrated, different programs may contain large segments of code that can be rendered almost identical, e.g., by using similar variable names and identical labels. The elements of a package also must adopt a uniform philosophy for detecting and reporting errors. The necessary surgery to produce package uniformity is best done by someone with no particular attachment to the original programs.

Aside from algorithmic development, the most difficult problem facing us today is testing. There are two fundamentally different reasons for testing, hence two fundamentally different approaches. On the one hand algorithm creators want to show that their creations are in some way superior to existing algorithms, and they approach performance testing as a contest. The tests they design specifically highlight whatever advantage the new algorithm may have; there is usually no attempt to uncover weaknesses in the algorithm or its implementation.

On the other hand, the selection of software for general use requires complete performance evaluation. Usually some duplication of purpose is acceptable in building a library, for example, so the concern is more with eliminating unacceptable programs and in matching programs to problem characteristics than in determining the 'best' program. Tests for this purpose should aggressively exercise a program in ways that will detect weaknesses, display strengths, explore robustness and probe problem-solving ability. We liken this type of testing to a physical examination. Inevitably the results of such testing will be used to compare programs, but the original intent is that a program be examined in isolation to stand or fall on its own merits.

Designing and implementing test programs is an important numerical problem that has been neglected in the rush to produce software for other purposes. Software testing locates weaknesses and leads to improvements in the next software generation. Yet, except for the ELEFUNT package of transportable Fortran test programs for the elementary functions [13] and collections of test programs for optimization software [9,38], no thematic test packages exist to our knowledge. Some test materials are distributed with various PACKs mentioned earlier, but these are not intended for general use.

The trouble is that we know little about how to test most types of software. Accuracy tests, for example, are usually battery tests exercising programs on someone's haphazard collection of problems. Not only is this time-consuming, but there is little purpose behind what is done and the mass of data gathered may be incomprehensible even to those who gathered it. We must find a better way. We must back off from the problem and critically examine what we are doing; every test should have a purpose. We must find understandable and useful ways to present test results. (Note in this regard the clever use of Chernoff faces [10] to summarize evaluations of software for solving systems of nonlinear equations [27]).

There are some leads in the literature that may prove useful. J. Lyness and J. Kaganove show that numerical software falls into two broad classes [34]. Class 1 (precision bound) programs implement methods, called 'finite decision methods', that guarantee to produce results in a finite number of steps. Elementary function programs are examples of class 1 programs. The accuracy achieved in class 1 programs usually approaches limits imposed by the computer arithmetic system. All other programs are class 2 (heuristic bound) programs implementing 'unreliable exact arithmetic algorithms'. The algorithms are such that useful results are not guaranteed in a finite number of steps even with exact arithmetic. Results that are produced are usually limited in accuracy by the algorithm and not by machine arithmetic. Quadrature and optimization programs are usually class 2.

The importance of this classification is that while accuracy test results for class 1 programs vary with the operating system, compiler and machine, properly structured accuracy tests for class 2 programs produce system-independent results when the accuracy achieved is sufficiently above machine limits. Thus certain types of accuracy tests for class 2 software need be done only once and only on one system.

But accuracy testing is just part of a complete test package; efficiency and robustness are also important. Because class 2 programs frequently require user-supplied software with an unpredictable effect on timing, other measures of efficiency, such as the number of accesses to the user-supplied program, must be used. Where efficiency varies significantly from problem to problem, it is important to explore efficiency as a function of the problem space. In its most elegant form to date, efficiency testing has been combined with accuracy testing and parameterization of a problem space to produce 'performance profiles'. The prototype work on automatic quadrature programs [35] produced curves combining probability of success and expected number of integrand evaluations as functions of requested accuracy for specific parameterized problem families. Curves for a problem family with features similar to those in a particular application should be useful in selecting a program for that application based on balancing

requested accuracy and predicted cost against the probability of success. The concepts of software classification and performance profiles exemplify the abstract assault on evaluation procedures that we believe is essential to progress in this area. Except for [33], these ideas have not been exploited beyond the work cited.

Concerns for numerical software have spawned important work in other fields as well. For example, research on the TAMPR system [6] for automated program transformation and maintenance was specifically motivated by early NATS work. TAMPR is intended to accept programs in certain standard languages, map them into abstract forms, make transformations on these abstract forms, and finally recover specific realizations of the transformed programs in standard languages again. The transformations are limited conceptually only by our ability to describe what must be done. An early version of the system was used to realize all versions of the LINPACK programs from complex single precision prototypes, for example. This application included enforcing formatting conventions and selectively implanting either calls to BLAs routines or inline coding with BLAs functionality, depending on the particular target computer host. Ultimately the capabilities may include automatic translation from one programming language to another by simply specifying different source and target languages in the first and last steps.

TAMPR is only one of many useful tools now under development. The TOOLPACK project is working on an extensive collection of software tools specifically designed to simplify the writing, testing, analyzing and maintaining of numerical software. The package is to combine the capabilities of TAMPR with those of formatters like POLISH [18], static analyzers like DAVE [41] and PFORT [49], dynamic analyzers like NEWTON [20], and other as yet unspecified tools including text editors. Specification of the package is still incomplete, but there is agreement that the package will be portable and that package elements will be compatible in data requirements. Release of a prototype version for evaluation and comment is tentatively set for late 1982.

We earlier mentioned the work of the IEEE on standardization of binary floating-point arithmetic for microprocessors. That is only one instance of a wide concern for computer arithmetic. The IEEE has recently established a second subcommittee to draft a radix and format-independent floating-point standard that will be upward compatible with the previous effort. Although the new draft is again intended for microprocessors, its inclusion of non-binary arithmetics should interest designers of larger equipment.

The fruits of such standardization efforts will not become widely available for some time, however. In the meantime we are forced to write software for existing computers. We can improve the portability of software among such machines by explicitly including environmental dependencies in the source code. There have been several attempts to establish a fundamental set of parameters describing arithmetic systems for this purpose. IFIP WG 2.5 published one proposal [21] that has proven unsatisfactory in many respects and has not been widely used. A second proposal [8] related to Brown's model for floating-point arithmetic [7] has received important support in some areas. The entire arithmetic model is imbedded in ADA [52], for example, much to the consternation of some numerical analysts. We return to that in a moment. Still a third proposal is being considered by the ANSI X3J3 Fortran Standards Committee for inclusion

in the next Fortran standard. This proposal defines certain parameters and reserves their names in the same way that SIN is a reserved name. The parameter names are then aliases for numerical values appropriate to the particular host environment. The difference between this approach and the ADA approach is that here only the names are specified; the numerical values provided are implementation dependent. While the parameters are based on a model of an arithmetic system, the model is not imposed by the standard. Thus the details of the model used in a particular situation can be chosen to fit the circumstances. When portability is crucial the model can be chosen to conservatively estimate machine parameters; when local performance is important the model can be chosen to closely approximate the local system. Such flexibility is not available in the ADA approach where the model specified must be conservative to be universal.

The activities and concerns just outlined are typical of the mathematical software effort today. Several large software projects are underway; others are planned. There are many ancillary activities aimed at improving the environment for software production and use. But there are also difficult problems that are not being addressed. We are not making much progress in testing methodology, for example.

4. The Future. Prediction of the future is always risky. Nevertheless we present a few guesses at what lies ahead. We expect that the quantity and quality of numerical software will continue to increase and that the activities just described will flourish in the future. Advancing technology and even the present success of the numerical software effort pose problems that must be overcome, however.

The most significant problem we face plagues every technical field and has been with us for a long time - communications at all levels. As we become more specialized we lose touch with one another and especially with potential customers.

Good communications with customers is crucial. Superb software is worthless unless software consumers are persuaded to use it. It is not enough to make users aware of software existence, though that is a difficult task in itself; consumer lethargy must be overcome at the same time. Consumers are reluctant to modify running programs unless they are convinced that the software they are currently using is inferior enough to endanger their work and that the new software will remove that danger. Open literature publications have never solved this type of communications problem. The consumers we must reach are applications people who do not read numerical analysis or mathematical software literature. We must find other ways to reach them.

Several years ago both the Albuquerque and Livermore branches of Sandia Laboratories inserted library monitors in their operating systems [4]. These monitors provided information on who was using which routines and on the values of certain parameters in the initial calls to those routines. This information proved valuable to both the librarians and the users. It led to improvement of frequently used programs and provision of new special purpose programs for problems previously solved with general purpose routines. It also permitted personal contact when it appeared that a program was being misused, when program bugs were found, or when better programs became available. Of course, diplomacy

and tact were essential in these contacts. In a few cases users objected when they felt their privacy was being invaded or they did not appreciate proffered advice. Sandia Livermore Laboratories augmented personal contact with an advertising campaign in which new programs were featured on posters prominently placed in all terminal rooms. Such efforts are noteworthy, rare and insufficient.

Today we face a revolution in the way computers are being used. The small 'personal' computer is becoming common at Argonne, and elsewhere as well, we suspect. While it is often acquired for monitoring experiments and gathering data, the temptation to use it for numerical purposes is strong. This is especially true when the cost of using a central computing facility grows and the 'free' personal machine would otherwise sit idle. Such usage is not necessarily bad, because smaller machines are approaching the hardware capabilities of larger machines of only a few years ago. Software is the problem. Owners of such machines frequently write their own software or obtain it from friends. In this respect they operate as large computing centers used to twenty years ago. The software movement has completely lost whatever contact it may have had with these users and that contact will be difficult to regain.

One possibility may be to contribute to the journals many of these people read. Byte, Personal Computing and the like are often sources of information for such users. While some of the articles in these journals are written by highly qualified people, much of the numerical advice is amateurish, reflecting techniques that lost favor long ago. We cannot legitimately complain about this situation unless we are willing as a profession to provide the proper advice and software through these journals. We must be the ones to initiate communications.

Unfortunately, we are also losing whatever communication we had with users of the larger machines. Often the original motivation for numerical software work was provided by users with applications that were endangered by poor computer programs. As our effort has matured many of us have become more concerned with software production for the sake of production and less concerned about the real needs of users. We have tended to communicate among ourselves and to neglect the users. Perhaps that behavior pattern is typical of a new field. We hope that it will change in our field.

At the technical level we find challenges posed by new computer hardware. We have only begun to work on algorithms and software for parallel and vector machines, and now we are faced with microprocessors as well. Their coming is an exciting event for numerical software people. The IEEE arithmetic standard provides computational capability that was not previously available at any level. In addition to sophisticated handling of underflow and overflow, standard-conforming systems must provide square root and mod functions, among others, that are as accurate as the usual arithmetic operations. Some early implementations of the standard include square root in the hardware where it becomes no more expensive to use than an ordinary division operation. This combination of speed and accuracy in square root coupled with other features must influence our selection of algorithms. I believe we will see dramatic changes in algorithms, software and even computer languages as these new microprocessors become common.

Overall we view the future with confidence and expectation. We will probably never satisfactorily solve the communications problem, but we expect that the quality of numerical software will continue to improve and that software production will become easier as new tools and hardware appear.

References.

[1] ACM SIGNUM Newsletter, Special Issue on the Proposed IEEE Floating-Point Standard, October, 1979.

[2] J. Adams, P. Swarztrauber and R. Sweet, FISHPAK: Efficient FORTRAN Subprograms for the Solution of Separable Elliptic Partial Differential Equations, Version 3, Nat. Center for Atmos. Res., Boulder, Colo., 1978.

[3] T. J. Aird, "The IMSL Fortran converter: an approach to solving portability problems", Portability of Numerical Software, Lecture Notes in Computer Science, Vol. 57, W. Cowell (ed.), Springer Verlag, New York, Heidelberg, Berlin, 1977, pp. 368-388.

[4] C. B. Bailey and R. E. Jones, "Usage and argument monitoring of mathematical library routines", TOMS 1, 1975, pp. 196-209.

[5] J. M. Boyle, W. J. Cody, W. R. Cowell, B. S. Garbow, Y. Ikebe, C. B. Moler and B. T. Smith, "NATS, a collaborative effort to certify and disseminate mathematical software", Proceedings 1972 National ACM Conference, Vol. II, Association for Computing Machinery, New York, 1972, pp.630-635.

[6] J. M. Boyle and M. Matz, "Automating multiple program realizations", Proceedings of the M.R.I. International Symposium XXIV: Computer Software Engineering, Polytechnic Press, Brooklyn, New York, 1977.

[7] W. S. Brown, A Simple but Realistic Model of Floating-Point Computation, Computing Science Technical Report No. 83, Bell Laboratories, Murray Hill, N.J., 1980.

[8] W. S. Brown and S. I. Feldman, Environmental parameters and basic functions for floating-point computation", TOMS 6, 1980, pp. 510-523.

[9] A. Buckley, "A portable package for testing minimization algorithms", to appear in proceedings of COAL Conference on Mathematical Programming, Testing and Validating Algorithms and Software, held at Boulder, Colo., Jan. 5-6, 1981.

[10] H. Chernoff, "The use of faces to represent points in k-dimensional space graphically", Jour. Amer. Stat. Ass. 68, 1973, pp. 361-368.

[11] W. J. Cody, "The FUNPACK package of special function subroutines", TOMS 1, 1975, pp. 13-25.

[12] W. J. Cody, "Basic concepts for computational software", Applied Mathematics Division Technical Memorandum 360, Argonne National Laboratory, November, 1980 (to appear in Proceedings of International Seminar on

Problems and Methodologies in Software Production, Sorrento, Italy, November 3-8, 1980).

[13] W. J. Cody and W. Waite, Software Manual for the Elementary Functions, Prentice Hall, Englewood Cliffs, N.J., 1980.

[14] D. Coleman, P. Holland, N. Kadeen, V. Klema and S. C. Peters, "A system of subroutines for iteratively reweighted least squares computations", TOMS 6, 1980, pp. 327-336.

[15] W. R. Cowell and L. D. Fosdick, "Mathematical software production", Mathematical Software III, J. R. Rice (ed.), Academic Press, 1977, pp. 195-224.

[16] W. R. Cowell and L. D. Fosdick, A Program for Development of High Quality Mathematical Software, Report CU-US-079-75, Department of Computer Science, University of Colorado, Boulder, Colo., 1975.

[17] J. J. Dongarra, J. R. Bunch, C. B. Moler and G. W. Stewart, LINPACK User's Guide, SIAM, Philadelphia, 1979.

[18] J. Dorrenbacher, D. Paddock, D. Wisneski and L. D. Fosdick, POLISH, a Program to Edit Fortran Programs, Report CU-CS-050-76 (Rev.), Dept. of Comp. Sci., Univ. of Colo., Boulder, Colo., 1976.

[19] J. J. Du Croz, S. J. Hague and J. L. Siemieniuch, "Aids to portability within the NAG project", Portability of Numerical Software, Lecture Notes in Computer Science, Vol. 57, W. Cowell (ed.), Springer Verlag, New York, Heidelberg, Berlin, 1977, pp. 390-404.

[20] J. Feiber, R. N. Taylor and L. J. Osterweil, NEWTON - A Dynamic Testing System for Fortran 77 Programs; Preliminary Report, Univ. of Colo., Dept. of Comp. Sci. Tech. Note, Nov. 1980.

[21] B. Ford, "Parameterization of the environment for transportable numerical software", TOMS 4, 1978, pp. 100-103.

[22] G. Forsythe, "Algorithms for scientific computation", Comm. ACM 9, 1966, pp. 255-256.

[23] B. S. Garbow, J. M. Boyle, J. J. Dongarra and C. B. Moler, Matrix Eigensystem Routines - EISPACK Guide Extension, Lecture Notes in Computer Science, Vol. 51, Springer Verlag, New York, Heidelberg, Berlin, 1977.

[24] C. W. Gear, Numerical Software: Science or Alchemy?, Report UIUCDCS-R-79-969, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Ill., 1979.

[25] A. George, J. Liu and E. Ng, User Guide for SPARSPAK: Waterloo Sparse Linear Equations Package, Research Report CS-78-30, Dept. of Comp. Sc., University of Waterloo, Waterloo, Ontario Canada, 1980.

[26] R. G. Grimes, D. R. Kincaid, W. I. MacGregor and D. M. Young, ITPACK Report: Adaptive Iterative Algorithms Using Symmetric Sparse Storage, Report CNA-139, Center for Numerical Analysis, University of Texas at Austin, 1978.

[27] K. L. Hiebert, "An outline for comparison testing of mathematical software -- illustrated by comparison testings of software which solves systems of nonlinear equations", to appear in proceedings of COAL Conference on Mathematical Programming, Testing and Validating Algorithms and Software, held at Boulder, Colo., Jan. 5-6, 1981.

[28] R. E. Huddleston (ed.), Program Directions for Computational Mathematics, unnumbered report, Dept. of Energy, Washington, D.C., 1979.

[29] Intel, The 8086 Family User's Manual, Numerics Supplement, Intel Corp., Santa Clara, Cal., 1980.

[30] V. C. Klema, private communication, January, 1981.

[31] F. T. Krogh, "Features for Fortran Portability", Portability of Numerical Software, Lecture Notes in Computer Science, Vol. 57, W. Cowell (ed.), Springer Verlag, New York, Heidelberg, Berlin, 1977, pp. 361-367.

[32] C. L. Lawson, R. J. Hanson, D. R. Kincaid and F. T. Krogh, "Basic linear algebra subprograms for Fortran usage", TOMS 5, 1979, pp. 305-323.

[33] J. N Lyness, "A bench mark experiment for minimization algorithms", Math. Comp. 33, 1979, pp. 249-264.

[34] J. N. Lyness and J. J. Kaganove, "Comments on the nature of automatic quadrature routines", TOMS 2, 1976, pp. 65-81.

[35] J. N. Lyness and J. J. Kaganove, "A technique for comparing automatic quadrature routines", Computer Journal, 1977, pp. 170-177.

[36] M. Machura and R. A. Sweet, "A survey of software for partial differential equations", TOMS 6, 1980, pp. 461-488.

[37] J. J. More, B. S. Garbow and K. E. Hillstrom, User Guide for MINPACK-1, Report ANL-80-74, Argonne National Laboratory, Argonne, Illinois, 1980.

[38] J. J. More, B. S. Garbow and K. E. Hillstrom, "Testing unconstrained optimization software", to appear TOMS 7, March 1981.

[39] A. H. Morris, Jr., Development of Mathematical Software and Mathematical Software Libraries, Report NSWC TR 79-102, Naval Surface Weapons Center, Dahlgren, Virginia, 1979.

[40] L. J. Osterweil, "TOOLPACK - an integrated system of tools for mathematical software development", to appear in proceedings of COAL Conference on Mathematical Programming, Testing and Validating Algorithms and Software, held at Boulder, Colo., Jan. 5-6, 1981.

[41] L. J. Osterweil and L. D. Fosdick, "DAVE - a validation, error detection and documentation system for Fortran programs", Software Practice and Experience 6, 1976, pp. 473-486.

[42] PDEPACK: Partial Differential Equations Package User's Guide, Scientific Computing Consulting Services, Manhattan, Kan., 1975.

[43] R. Piessens, E. De Doncker, C. W. Uberhuber and H. J. Stetter, Detailed Test Results for Automated General Purpose Integration over Finite or Infinite Intervals, Unnumbered Report, Applied Mathematics and Programming Division, Katholieke Universiteit Leuven, Heverlee, Belgium, 1978.

[44] J. R. Rice, "Announcement and call for papers, Mathematical Software", SIGNUM Newsletter 4, No. 3, October 1969, p. 7.

[45] J. R. Rice (ed.), Mathematical Software, Academic Press, New York, 1971.

[46] J. R. Rice, "ELLPACK: a research tool for elliptic partial differential equations software", Mathematical Software III, J. R. Rice (ed.), Academic Press, New York, 1977, pp. 319-341.

[47] J. R. Rice, "Software for numerical computation", Research Directions in Software Technology, P. Wegner (ed.), MIT Press, Cambridge, Mass., 1979, pp.688-708.

[48] J. R. Rice, C. W. Gear, J. M. Ortega, B. N. Parlett, M. Schultz, L. F. Shampine and P. Wolfe, Numerical Computation, Panel Report for the COSERS Project, SIGNUM Newsletter special issue, 1978.

[49] B. G. Ryder, "The PFORT verifier", Software Practice and Experience 4, 1974, pp.359-377.

[50] B. T. Smith, J. M. Boyle and W. J. Cody, "The NATS approach to quality software", Software for Numerical Mathematics, D. J. Evans (ed.), Academic Press, New York, 1974, pp. 393-405.

[51] B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema and C. B. Moler, Matrix Eigensystem Routines - EISPACK Guide, Lecture Notes in Computer Science, Vol. 6, Second Edition, Springer Verlag, New York, Heidelberg, Berlin, 1976.

[52] B. A. Wichman, Tutorial Material on the Real Data-Types in ADA, Final Technical Report, U.S. Army European Research Office, London, November 1980.

SOFTWARE RELIABILITY ESTIMATION THROUGH FITTING A POPULATION PROCESS TO DATA *

Marc R. Stromberg
Bell Technical Operations Corporation
Software/Computer Evaluation Facility
600 North Garden Avenue
Sierra Vista, Arizona 85635

ABSTRACT. Characteristics of software reliability can be stated in terms of the mean functions of various transition counting processes associated with a birth-death process.

Many of these counting processes are shown to share the attribute of Poisson processes that the higher moment functions can be expressed in terms of the mean.

Such a dependence provides a computational basis for estimation of the mean functions, hence of software reliability, by any method (such as generalized least squares) which uses only a few moments.

1. Introduction. Software, unlike hardware, can in principle evolve to perfection through the discovery and removal of error. Large computer programs often go through a phase of testing for faults (errors, discrepancies from intended function) and concurrent software modification for removal of discovered faults. Software reliability measurement, if viewed as an estimate of the error content of a computer program, must account for both discovery and removal of errors. A birth and death process is an appropriate model for the discovery and removal process, if the emphasis is on the active (discovered but not yet removed) faults.

Active faults arrive through testing a program with "typical" input, and depart through the efforts of debuggers to remove them. A population process can trace the evolution of software reliability by modeling the decline of the active fault population toward eventual extinction.

To be effective, a model for the active fault population must have the properties that it can fit data observed of a particular program, and that useful conclusions can be drawn from the model which are not obvious from the data.

This paper considers conditions on the active fault population process that allow both fitting the process to data and subsequent computations relevant to reliability.

From data consisting of recorded times of faults and times of repairs, the procedure to be described attempts to estimate, among other things, the one-dimensional distributions of the count of active faults.

* Sponsored by the United States Army under Contract Number DAEA18-77-C-0134.

2. Notation. $X:\Omega x T \longrightarrow E^n$ is a stochastic process defined on the measure space Ω (with complete probability measure Pr) and on the parameter set $T=[o,\infty)$, with values in euclidean space E^n. X has random variables $X_t(\cdot)=X(\cdot,t)$ and sample functions $X_w(\cdot)=X(w,\cdot)$.

The letters i,j,k represent integers or elements of L_n, the integer lattice points of E^n.

For $j\in L_n$ and $I \subseteq T$ any interval, $H_j(I) = \{w\in\Omega : wxI\subseteq X^{-1}(j)\}$, $H_j = \bigcup_{I \subseteq T} H_j(I)$, and $H(I) = \bigcup_{j\in L_n} H_j(I)$.

R represents any irreflexive relation on L_n, and $H_R = \bigcup_{(j,k)\in R} H_j \cap H_k$.

$A_j(w) = \cup\{I : w\in H_j(I)\}$ for $j\in L_n$,
$\underline{A}_R(w) = \cup\{A_j(w): j \in \text{support } R\}$, and
$\overline{A}_R(w) = \cup\{A_k(w): k \in \text{range } R\}$, for $w\in H_R$.

For $t\in T$, D_t is the set of $w\in\Omega$ such that the sample function X_w has a discontinuity at t.

3. Assumptions. We consider hypotheses on X that are suited both to computing, from data, estimates of the one-dimensional distributions of X, and to computing distributions of random variables associated with processes that enumerate particular discontinuities of the sample functions of X.

X itself will be a birth-death (or similar) process, with software faults corresponding to births and software repairs (fault removals) corresponding to deaths. A fault will be said to be active if software input conditions have caused the fault to be executed at least once and it has not yet been removed, and activated if the fault is or has been active.

The sample function value $X_w(t)$ is the number of active faults at time t.

Among the counting processes related to X that will be of interest are the count of activated faults, the count of repairs, and the count of active fault population extinctions (that is, the number of times the only active fault is removed).

X is assumed to satisfy:

A1. $Pr(D_t) = o$ for all $t\in T$; that is, X has no fixed points of discontinuity;
A2. $\pi_i \circ X_w: T \longrightarrow E^1$ is an integer valued step function with unit jump for all $w\in\Omega$, where $\pi_i: E^n \longrightarrow E^1$ is the ith projection, $1\leq i\leq n$;

A3. X is a Markov process, and

A3.1 there are piecewise right continuous functions $q_{jk}(\cdot)$ such that

$$p_{jk}(t,u)= \begin{cases} q_{jk}(u)(u-t)+o(u-t), & j\neq k \\ 1-q_{jj}(u)(u-t)+o(u-t), & j=k \end{cases}$$

where $q_{jj}(u) = \sum_{k\neq j} q_{jk}(u)$

for $j,k\in L_n$, $t<u$ for the Markov transition functions

$$p_{jk}(t,u) \triangleq \Pr\left\{X_u=k \mid X_t=j\right\} ;$$

A3.2 $P(j,t)\triangleq \Pr\left\{X_t=j\right\}$ is left continuous for all $j\in L_n$; and

A4. $\Pr(E(t,u)) = o(u-t)$, where $E(t,u)$ is the set of sample functions with at least two discontinuities on (t,u).

We will call X refinable if X satisfies A1 through A4.

4. Properties of a refinable process. To select particular discontinuities of X, we define the following processes. For any irreflexive relation R on L_n, and $s\geq o$, define $N^s:\Omega\times T\longrightarrow E^1$ by $N^s(w,t)$ = the number of discontinuities $u\in(s,t)$ of the sample function $X_w(\cdot)$ such that $(X_w(u-),X_w(u+))\in R$, if $t>s$; and $N^s(w,t) = o$ if $t\leq s$.

By A1, $N^s(\cdot,t) = N^s(\cdot,u)+N^u(\cdot,t)$ almost surely for $s\leq u\leq t$.

For future reference (and to call N^s a counting process) we note

Lemma 1. For $y,z\in T$, $y<z$, $N^y(\cdot,z):\Omega\longrightarrow E^1$ is a random variable measurable on the sample space of $\left\{X_t:t\in(y,z)\right\}$ (the σ-algebra generated by $X_t^{-1}(B)$ for $t\in(y,z)$, Borel sets $B\subseteq E^n$).

Proof: For $r,s\in T$ and $j,k\in L_n$ define

$$A_{rs}(j,k)= \bigcap_{q\in(r,s)} H_j[r,q)\cup H_k(q,s]$$

where the intersection is over rational $q\in(r,s)$ and where $H_j[r,q) = H_j(I)$ for $I=[r,q)$.

For $n>o$ let Q_n be any ordered sequence of rationals $r_1<s_1<\ldots< r_n<s_n$, and for irreflexive relation R defining N^y let

$$A(Q_n,R)= \bigcap_{i=1}^{n} \bigcup_{(j,k)\in R} A_{r_is_i}(j,k).$$

Finally, let $M_n^R(y,z) = \bigcup_{Q_n \subset (y,z)} A(Q_n,R)$.

Then $M_n^R(y,z)$ differs from the set $\{w\in\Omega: N^y(w,z) \geq n\}$ by the set $\bigcup_{q\in T} D_q$ (union over rational q) of measure o. QED.

Of course, the process N^s does not contain more information than X, however the following definition is useful.

Definition. For fixed $s\geq o$ and irreflexive relation R on L_n, the refinement of $X:\Omega\times T\longrightarrow E^n$ (satisfying A1-A4) is $\widetilde{X}:\Omega\times T\longrightarrow E^{n+1}$ where $\widetilde{X}(w,t) \triangleq (X(w,t), N^s(w,t))$.

The following theorem justifies calling $\widetilde{X}$ a refinement of X.

Theorem 1. If X is a refinable process, then $\widetilde{X}$ also is. The transition intensities of X are inherited by $\widetilde{X}$, in the sense of Corollary 1.1.

Proof: The discontinuities of $\widetilde{X}_w$ coincide with those of X_w, so $\widetilde{X}$ satisfies A1, A2 and A4. To show A3, interpret "equal" as "equal almost surely", and let $Y_{u,k}$ be the characteristic function of the set $X_u^{-1}(k)$ and let $Z_{u,n}^t$ be the characteristic function of the set $\{w: N_u^t(w)=n\}$.

For $s<t<u$, $N_u^s = N_t^s + N_u^t$, so

$Z_{u,n}^s = \sum_{i=o}^{n} Z_{t,i}^s \, Z_{u,n-i}^t$. $Y_{u,k}\, Z_{u,n}^s$ is equal to the characteristic function of the set $\widetilde{X}_u^{-1}(k,n)$. Now $Z_{t,i}^s$ is measurable on the sample space of N_t^s, and the sample space of $\{\widetilde{X}_r: r\leq t\}$ is equal to the sample space of $\{X_r: r\leq t\}$. Therefore, $E[Y_{u,k}\, Z_{u,n}^s | \widetilde{X}_r: r\leq t] =$

$\sum_{i=o}^{n} Z_{t,i}^s \, E[Y_{u,k}\, Z_{u,n-i}^t | X_r: r\leq t]$.

$Y_{u,k}\, Z_{u,n-i}^t$ is measurable on the sample space of $\{X_r: r\geq t\}$, so by the Markov property of X,

$E[Y_{u,k}\, Z_{u,n}^s | \widetilde{X}_r: r\leq t] = \sum_{i=o}^{n} Z_{t,i}^s \, E[Y_{u,k}\, Z_{u,n-i}^t | X_t]$.

The last sum is measurable on the sample space of $\widetilde{X}_t$ and so

$E[Y_{u,k}\, Z_{u,n}^s | \widetilde{X}_r: r\leq t] = E[Y_{u,k}\, Z_{u,n}^s | \widetilde{X}_t]$,

that is, $\Pr\{\widetilde{X}_u=(k,n) | \widetilde{X}_r: r\leq t\} = \Pr\{\widetilde{X}_u=(k,n) | \widetilde{X}_t\}$.

For $t\leq s<u$, $E[Y_{u,k}\, Z_{u,n}^s | \widetilde{X}_r: r\leq t] = E[Y_{u,k}\, Z_{u,n}^s | X_r: r\leq t]$ as above.

By the Markov property of X, the last expectation is $E[Y_{u,k}\, Z_{u,n}^s | X_t]$ as before, which is measurable on the sample space of $\widetilde{X}_t$, and the result follows.

If $t<u\leq s$, $Y_{u,k}\, Z^s_{u,n} = o$ if $n\geq 1$, and

$E[Y_{u,k}\, Z^s_{u,o}|\widetilde{X}_r: r\leq t] = E[Y_{u,k}|X_r: r\leq t]$

$= E[Y_{u,k}|X_t] = E[Y_{u,k}\, Z^s_{u,o}|\widetilde{X}_t]$, so $\widetilde{X}$ is a Markov process.

To show A3.2, let $A_t = X_t^{-1}(j)$, $B_t = \widetilde{X}_t^{-1}(j,n)$ and $P(j,n,t) = \Pr\left\{\widetilde{X}_t=(j,n)\right\}$ for arbitrary $t\in T$, and $j\in L_k$ and n an integer, both being fixed but arbitrary.

By A3.2 $\Pr(A_t\backslash A_u)=q_{jj}(u)(u-t)P(j,t)+o(u-t)$ for $t<u$, so $\lim_{t\to u-} \Pr(A_t\backslash A_u)=o$. Since $P(j,t)$ is left continuous,

$o=\lim_{t\to u-} |P(j,t)-P(j,u)|$

$=\lim_{t\to u-} |\Pr(A_t\backslash A_u)-\Pr(A_u\backslash A_t)|$ and $\lim_{t\to u-} \Pr(A_u\backslash A_t)=o$.

Now, $B_t\backslash B_u \subseteq A_t\backslash A_u \cup C_{t,u}$ where $C_{t,u} = \left\{w: X_t=j, X_u=j, N^s_t=n, N^s_u\neq n\right\}$ and

$$\Pr(C_{t,u}) = \begin{cases} o(u-t), & s<u \\ o, & t<u\leq s \end{cases}$$

by A1, A4 and definition of N^s. Therefore, $\lim_{t\to u-} \Pr(B_t\backslash B_u)=o$ and similarly for $B_u\backslash B_t$, so $\lim_{t\to u-} P(j,n,t) = P(j,n,u)$ and $\widetilde{X}$ satisfies A3.2. For the proof that $\widetilde{X}$ satisfies A3.1, see Corollary 1.1. QED.

Corollary 1.1 Suppose X is refinable, and let $\widetilde{P}_{jk}(u,t)=\Pr\left\{\widetilde{X}_t=(k,n)\,|\,\widetilde{X}_u=(j,m)\right\}$. Then,

1. If $s\leq u<t$ then $\widetilde{P}_{jk}(u,t)=\begin{cases} P_{jk}(u,t)+o(t-u), & n=m+1 \text{ and } (j,k)\in R \text{ or} \\ & n=m \text{ and } (j,k)\notin R, \\ o(t-u) & \text{otherwise.} \end{cases}$

2. If $u<t\leq s$ then $\widetilde{P}_{jk}(u,t)=\begin{cases} P_{jk}(u,t), & n=m=o, \\ o & \text{otherwise.} \end{cases}$

Proof: Let $\Lambda = \widetilde{X}_u^{-1}(j,m)$, and define $\widetilde{P}_{jk}(u,t)$ as required if $\Pr(\Lambda)=o$. If $s\leq u<t$,

$$\Pr\left\{\widetilde{X}_t=(k,n),\widetilde{X}_u=(j,m)\right\} = \int_\Lambda Y_{t,k}\, Z^s_{t,n}\, dPr$$

$$=\int_\Lambda \sum_{i=o}^{n} Z^s_{u,i} E[Y_{t,k}\, Z^u_{t,n-i}|X_r: r\leq u]\,dPr = \begin{cases} A(n-m), & o\leq m\leq n, \\ o & \text{otherwise,} \end{cases} \quad \text{where}$$

$A(i)=\int_\Lambda E[Y_{t,k}\, Z^u_{t,i}|X_r:\leq u]dPr$, by the proof of Theorem 1. Since $s\leq u$,

$A(i)=\int_\Lambda Y_{t,k}\, Z^u_{t,i}dPr=o(t-u)$ if $i\geq 2$ by A4.

Now, $\int_\Lambda E[Y_{t,k}|X_r:r\leq u]dPr=A(o)+A(1)+o(t-u)$, again by A4, and $A(1)=o(t-u)$ if $(j,k)\notin R$ and $A(o)=o(t-u)$ if (j,k) R, by A4 and definition of N^s.

Therefore, since $E[Y_{t,k}|X_r:r\leq u]=Pr\left\{X_t=k|X_u=j\right\}$ a.e. on Λ,

$Pr\left\{X_t=(k,n),\widetilde{X}=(j,m)\right\}$ =

$$\begin{cases} Pr\left\{X_t=k|X_u=j\right\} Pr\left\{\widetilde{X_u}=(j,m)\right\}+o(t-u), & \text{if } n-m=1 \text{ and } (j,k)\in R \text{ or} \\ & \quad n-m=o \text{ and } (j,k)\notin R, \\ o(t-u) & \text{otherwise,} \end{cases}$$

and the conclusion follows.

If $u<t\leq s$, $Pr\left\{\widetilde{X_t}=(k,n),\widetilde{X_u}=(j,m)\right\}=o$, if $m>o$ or $n>o$. For $n=o$, the result is immediate. QED.

The following definition is useful for application of the theorem.

Definition. The irreflexive relation R is an attachment of the process X if for almost all $w\in H_R$ there is $(j,k)\in R$ such that
$\sup A_j(w)=\sup \underline{A}_R(w)=\inf \overline{A}_R(w)=\inf A_k(w)$.

Lemma 2. If R is an attachment of the refinable process X, then for almost all $w\in H_R$ there is a unique discontinuity t of X_w such that $(X_w(t-),X_w(t+))\in R$.

Proof: $t=\sup \underline{A}_R(w)=\inf \overline{A}_R(w)$. QED.

If R is an attachment of X, define $H_R(t,u)=\bigcup_{(j,k)\in R} X_t^{-1}(j)\cap X_u^{-1}(k)$ for $t<u$, and define $\eta_R:H_R\longrightarrow T$ by $\eta_R(w)=\inf \overline{A}_R(w)$. Then

Lemma 3. If R is an attachment, $\eta_R\in(t,u)$ for almost all $w\in H_R(t,u)$.
Proof: $H_R(t,u)\subseteq H_R\cup D_t\cup D_u$, and $t<\sup \underline{A}_R(w)$ and $u>\inf \overline{A}_R(w)$ for $w\in D_t\cup D_u$. QED.

η_R induces a real Borel measure μ_R defined by $\mu_R(E)=Pr(\eta_R^{-1}(E))$ for Borel sets $E\subseteq T$.

The cumulative distribution function of μ_R, F_R, defined by $F_R(t)=\mu_R([o,t))$, satisfies $F_R(u)-F_R(t)=\mu_R(t,u)$ for $t<u$ by A1. By A4, $Pr(H_R(t,u))=F_R(u)-F_R(t)+o(u-t)$

and therefore,

Proposition 1. If R is a finite attachment, then $F_R'(u)=\sum_{(j,k)\in R} q_{jk}(u)P(j,u)$ for all but finitely many u>o.

Proof. $F_R'(u)$ exists if and only if the two sided limits $\lim_{t\to u-} \frac{Pr(H_R(t,u))}{u-t}$ and

$\lim_{t\to u+} \frac{Pr(H_R(u,t))}{t-u}$ exist and are equal.

Since $Pr(H_R(t,u))=\sum_{(j,k)\in R} Pr\left\{X_u=k \mid X_t=j\right\} Pr\left\{X_t=j\right\}$

$=\sum_{(j,k)\in R} q_{jk}(u)P(u,j)+o(u-t)$, the conclusion follows by A3.1 and A3.2. QED.
If R is a relation on L_m, define R_n as the relation $\left\{(j_n,k_{n+1}):(j,k)\in R\right\}$ on L_{m+1} where j_n is the element of L_{m+1} with m+1 st coordinate n and which coincides with j on L_m. Then

Lemma 4. If $X:\Omega\times T\longrightarrow E^m$ is refinable, then for fixed s≥o and irreflexive relation R on L_m, R_n is an attachment of $\widetilde{X}$ for every n≥o.

Proof: Let $S=\bigcup_{q\in T} D_q$ where q is rational, and choose $w\in H_{R_n}\setminus S$ and rational

r,t with $r\in \underline{A}_{R_n}(w)$ and $t\in \overline{A}_{R_n}(w)$.

Then r<t since $N^S(w,t)>N^S(w,r)$. Arbitrary choice of r,t then shows sup $\underline{A}_{R_n}(w) \le$ inf $\overline{A}_{R_n}(w)$.

Since $w\notin D_r\cup D_t$, there is $u\in(r,t)$ with $(X_w(u-), X_w(u+))\in R$. By selection of r and t, u necessarily satisfies $(\widetilde{X_w}(u-), \widetilde{X_w}(u+))\in R_n$, and

$u\le$sup $\underline{A}_{R_n}(w)\le$inf $\overline{A}_{R_n}(w)\le u$. QED.

Simplifying the notations for H_{R_n} and η_{R_n} to H_n and η_n, define $\eta_n:H_n\longrightarrow T$ as done following Lemma 2, with induced measure μ_n and distribution functions F_n. Note that the sets $\left\{w:N^S(w,t)\ge n+1\right\}$ and $\eta_n^{-1}([o,t))$ are equal almost surely so that

Lemma 5. $E[N_t^s]=\sum_{n\geq 1} Pr\{N_t^s\geq n\}=\sum_{n\geq o}\mu_n[s,t)=\sum_{n\geq o}F_n(t)$.

Proposition 2. For $s\geq o$ and finite irreflexive relation R on the state space of the refinable process X, if q_{jk} is a right continuous real function for $(j,k)\in R$, then

$$E[N_t^s]=\int_s^t \sum_{(j,k)\in R} q_{jk}(u)P(j,u)du \text{ for } t\geq s.$$

Proof: Let $P(j,n,t)=Pr\{X_t=j_n\}$. Then by Proposition 1, Lemma 4, and Theorem 1, $F_n'(u)=\sum_{(j,k)R} q_{jk}(u)P(j,n,u)$ for $u>s$.

$F_n'(s)=o$ for $n\geq 1$ and the right hand derivative of F_o at s is the upper derivative $\overline{D}\mu_o(s)$ of μ_o at s with respect to open segments, with $\overline{D}\mu_o(s)=\sum_{(j,k)R} q_{jk}(s)P(j,o,s)$.

Since $\overline{D}\mu_n$ is finite everywhere for $n\geq o$, μ_n is absolutely continuous with respect to Lebesgue measure for $n\geq o$ and

$$F_n(t)=\int_s^t \sum_{(j,k)R} q_{jk}(u)P(j,n,u)du.$$

Then
$$E[N_t^s]=\sum_{n\geq o} F_n(t)=\int_s^t \sum_{(j,k)\in R} q_{jk}(u)P(j,u)du \text{ by Lemma 5. QED.}$$

For the next lemma, the following notation will be used. If $\{R_i\}$ is any collection of relations on the state space of X let $\eta_{n,i}=\eta_{R_i(n)}:H_{R_i(n)}\longrightarrow T$ for the process $\widetilde{X_i}$ defined via R_i, n and s as done previously, following Proposition 1. Use $\eta_{n,i}$ to induce Borel measures $\mu_{n,i}$ and $\mu_i=\sum_{n\geq o}\mu_{n,i}$,

let $F_{n,i}(t)$ and $F_i(t)$ be the distribution functions for $\mu_{n,i}$ and μ_i respectively, and let $N_{t,i}^s$ be the counting process associated with $\widetilde{X_i}$. Then

Lemma 6. If $R_1\subseteq R_2\subseteq \ldots$ is an ascending chain of relations whose union is R, then $\lim_{i\to\infty} E[N_{t,i}^s]=E[N_t^s]$ where N_t^s is the process defined via R and s.

Proof: Let $M_{n,i}(y,z)=\bigcup_{Q_n\subset (y,z)} A(Q_n,R_i)$,

as defined for Lemma 1, and check that

1. $M_{n,i}(y,z) \subseteq M_{n,j}(y,z)$ if $R_i \subseteq R_j$ for $n \geq 1$, and
2. $M_n(y,z) = \bigcup_{i \geq 1} M_{n,i}(y,z)$ where $M_n(y,z) = \bigcup_{Q_n \subset (y,z)} A(Q_n,R)$.

Let $B_{1,n} = M_{n,1}(s,t)$ and
$B_{i,n} = M_{n,i}(s,t) - M_{n,i-1}(s,t)$ for $i \geq 2$, $n \geq 1$,
and note that $\sum_{i=1}^{k} Pr(B_{i,n}) = Pr(M_{n,k}(s,t)) = \mu_{n-1,k}[s,t)$.

Then $E[N_t^s] = \sum_{n \geq 0} \mu_n[s,t) = \sum_{n \geq 0} \sum_{i \geq 1} Pr(B_{i,n+1}) = \lim_{k \to \infty} \sum_{n \geq 0} \mu_{n,k}[s,t)$

$= \lim_{k \to \infty} F_k(t) = \lim_{k \to \infty} E[N_{t,k}^s]$. QED.

Corollary 2.1 If R is any irreflexive relation on the state space of X and q_{jk} is a right continuous real valued function for $(j,k) \in R$, then

$$E[N_t^s] = \int_s^t \sum_{(j,k) R} q_{jk}(u) P(j,u) du \quad \text{for } t \geq s \ .$$

Proof: Assume R infinite and let $R_1 \subset R_2 \subset \ldots$ be an ascending chain of finite relations whose union is R.

Then Proposition 2 gives
$E[N_{t,i}^s] = \int_s^t g_i(u) du$

where $g_i(u) = \sum_{(j,k) R_i} q_{jk}(u) P(j,u)$.

Now, $\{g_i(u)\}$ is a monotone sequence with

$g_i(u) \longrightarrow \sum_{(j,k) \in R} q_{jk}(u) P(j,u)$ and so

$$\int_s^t g_i(u) du \longrightarrow \int_s^t \sum_{(j,k) \in R} q_{jk}(u) P(j,u) du \quad \text{by monotone}$$

convergence. But $\int_s^t g_i(u) du \longrightarrow E[N_t^s]$ by Lemma 6. QED.

Corollary 2.2 Under the hypotheses of Corollary 2.1, the distribution function for η_n satisfies

$$F_n(t)=\int_s^t \sum_{(j,k)\in R} q_{jk}(u)P(j,n,u)du.$$

Proof: The proof of Lemma 6 includes the fact that

$$\mu_n[s,t)=\lim_{k\to\infty} \mu_{n,k}[s,t)=\lim_{k\to\infty} F_{n,k}(t)$$

The argument of corollary 2.1 shows this limit to be

$$\int_s^t \sum_{(j,k)\in R} q_{jk}(u)P(j,n,u)du.$$

Generating equations. We will now use the previous results to develop expressions for higher moments of the random variables N_t^s defined for particular $s\geq o$ and irreflexive relation R on L_k, for a refinable process $X:\Omega x T\longrightarrow E^k$.

If $j\in L_k$, let $j(r)$ be the rth integer coordinate of j and let $A(j)=\left\{i\in L_k : \max\left\{|i(r)-j(r)| : 1\leq r\leq k\right\}=1\right\}$ be the lattice points adjacent to j.

By A2, A3.1, and A4, $\sum_{k\notin A(j)} P_{jk}(u,t)=o(t-u)$ for $t>u$.

Corollary 1.1 then gives the following cases.
If $s<u<t$,

(1) $$P(j,n,t)=P_{jj}(u,t)P(j,n,u)+\sum_{(i,j)\notin R} P_{ij}(u,t)P(i,n,u)+$$

$$\sum_{(i,j)\in R} P_{ij}(u,t)P(i,n-1,u)+o(t-u),\ n>o,$$

and

$$P(j,o,t)=P_{jj}(u,t)P(j,o,u)+\sum_{(i,j)\notin R} P_{ij}(u,t)P(i,o,u)+o(t-u).$$

If $u<t\leq s$,

(2) $P(j,n,t)=o$, $n>o$ and

$$P(j,o,t)=P_{jj}(u,t)P(j,o,u)+\sum_{i\in A(j)} P_{ij}(u,t)P(i,o,u)+o(t-u).$$

All of the above sums can be assumed to be finite.
Equations (1) and (2) give the following derivatives, using + and - to denote right- and left-hand derivatives, respectively, where q_{ij} and $P(j,n,\cdot)$ satisfy appropriate continuity conditions.

(3) If $t>s$,

$$P'(j,n,t)=-q_{jj}(t)P(j,n,t)+\sum_{(i,j)\in R} q_{ij}(t)P(i,n,t)+\sum_{(i,j)\in R} q_{ij}(t)P(i,n-1,t) \quad , n>0 \ ,$$

$$P'(j,o,t)=-q_{jj}(t)P(j,o,t)+\sum_{(i,j)\in R} q_{ij}(t)P(i,o,t);$$

(4) If $t=s$,

$P'(j,n,s)=o \quad , n>1,$

$P'(j,1,s)_+= \sum_{(i,j)\in R} q_{ij}(s)P(i,o,s),$

$P'(j,1,s)_-=o \quad ,$

$P'(j,o,s)_+= -q_{jj}(s)P(j,o,s)+\sum_{(i,j)\in R} q_{ij}(s)P(i,o,s),$

$P'(j,o,s)_-= -q_{jj}(s)P(j,o,s)+\sum_{i\in A(j)} q_{ij}(s)P(i,o,s);$

(5) If $t<s$,

$P'(j,n,t)=o \quad , n>o,$

$P'(j,o,t)= -q_{jj}(t)P(j,o,t)+\sum_{i\in A(j)} q_{ij}(t)P(i,o,t).$

For $j\in L_k$, define $G(j,x,t)= \sum_{n\geq o} x^n P(j,n,t)$, $|x|\leq 1$, so that

$G(x,t)= \sum_{j\in L_k} G(j,x,t)$ is the probability generating function for the

random variable N_t^s. Then

Proposition 3. If $q_{ij}(t)$ are bounded functions for $i,j\in L_k$, then

(6) $\frac{\partial}{\partial t}G(j,x,t)= -q_{jj}(t)G(j,x,t)+\sum_{(i,j)\in R} q_{ij}(t)G(i,x,t)$

$+x\cdot \sum_{(i,j)\in R} q_{ij}(t)G(i,x,t) \quad$, for $|x|\leq 1$, $t\geq s$.

Proof: For fixed $x<1$, boundedness of the $q_{ij}(t)$ implies uniform convergence

of $\sum_{n\geq o} x^n P'(j,n,t)$

$= -q_{jj}(t)P(j,o,t)+\sum_{(i,j)\in R} q_{ij}(t)P(i,o,t)$

$+ \sum_{n\geq 1} x^n [-q_{jj}(t)P(j,n,t)+\sum_{(i,j)\in R} q_{ij}(t)P(i,n,t)$

$+\sum_{(i,j)\in R} q_{ij}(t)P(i,n-1,t)]$, from (3) and (4).

The left side of the equation is therefore $\frac{\partial}{\partial t} G(j,x,t)$.
Absolute convergence is also guaranteed by hypothesis, and the right side can be rearranged to give (6). For x=1, (6) is an application of A3.1, A3.2, and A4. QED.

Equations (6) will be called the generating equations.

Now

(7) $G(x,t)= \sum_{n\geq 0} x^n \Pr\left\{N_t^s=n\right\}$ and for $t\geq s$,

(8) $\sum_{n\geq 0} x^n \frac{d}{dt} \Pr\left\{N_t^s=n\right\} = -F_0'(t) + \sum_{n\geq 1} x^n (F_{n-1}'(t)-F_n'(t))$

$= \sum_{n\geq 0} x^n [\sum_{j\in L_k} \sum_{(i,j)\in R} q_{ij}(t)P(i,n-1,t)$

$+ \sum_{(i,j)\notin R} q_{ij}(t)P(i,n,t) - q_{jj}(t)P(j,n,t)]$,

applying (3) and (4) and interpreting derivatives as right hand derivatives at s.

The proof of Proposition 2 shows that $|F_{n-1}'(t)-F_n'(t)| \leq 2 \frac{d}{dt} E[N_t^s]$, so if the latter quantity is a bounded function, the result of term by term differentiation of (7) with respect to t is uniformly and absolutely convergent for any $|x|<1$. If the functions $q_{ij}(t)$ are also bounded, then application of (6) and rearrangement of (8) show that

$$\frac{\partial}{\partial t} G(x,t) = \sum_{j\in L_k} \frac{\partial}{\partial t} G(j,x,t), \quad |x|<1.$$

Finally, we can apply a power series argument to (8) and conclude

Lemma 7. If $\sum_{j\in L_k} q_{jj}(t)P(j,t)$ is a bounded function, then

(9) $$\frac{\partial^{i+1}}{\partial x^i \partial t} G(x,t) = \sum_{j\in L_k} \frac{\partial^{i+1}}{\partial x^i \partial t} G(j,x,t) \text{ for } |x|<1, i\geq 0.$$

Proof: The hypothesis guarantees the preceding argument, since the conclusion of Proposition 3 is true under this condition also. QED.
We are finally in a position to formally develop moments of the random variables N_t^s.

Differentiating the generating equations (6) with respect to x and summing over L_k produces, applying (9),

$$\frac{\partial^2}{\partial x\,\partial t}G(x,t)=\sum_{j\in L_k}[-q_{jj}(t)\frac{\partial}{\partial x}G(j,x,t)+\sum_{i\in A(j)}q_{ij}(t)\frac{\partial}{\partial x}G(i,x,t)]$$

$$+(x-1)\sum_{j\in L_k}\sum_{(i,j)\in R}q_{ij}(t)\frac{\partial}{\partial x}G(i,x,t)$$

$$+\sum_{j\in L_k}\sum_{(i,j)\in R}q_{ij}(t)G(i,x,t)$$

The first sum over L_k is zero by A3.1. Integrating, we get

$$\frac{\partial}{\partial x}G(x,t)=(x-1)\int_s^t\sum_{(i,j)\in R}q_{ij}(u)\frac{\partial}{\partial x}G(i,x,u)du$$

$$+\int_s^t\sum_{(i,j)\in R}q_{ij}(u)G(i,x,u)du.$$

An application of dominated convergence then gives

$$\lim_{x\to 1-}\frac{\partial}{\partial x}G(x,t)=\int_s^t\sum_{(i,j)\in R}q_{ij}(u)P(i,u)du,$$

which is a restatement of Corollary 2.1.
We can apply the same procedure to find

$$\frac{\partial^2}{\partial x^2}G(x,t)=(x-1)\int_s^t\sum_{(i,j)\in R}q_{ij}(u)\frac{\partial^2}{\partial x^2}G(i,x,u)du$$

$$+2\int_s^t\sum_{(i,j)\in R}q_{ij}(u)\frac{\partial}{\partial x}G(i,x,u)du \quad , \text{ so}$$

$$(10)\quad \lim_{x\to 1-}\frac{\partial^2}{\partial x^2}G(x,t)=2\int_s^t\sum_{(i,j)\in R}q_{ij}(u)Q_s(i,u)du$$

where by Proposition 3., $Q_s(j,t)=\lim_{x\to 1-}\frac{\partial}{\partial x}G(j,x,t)$ for $j\in L_k$ satisfies

(11) $Q_s(j,t)' = -q_{jj}(t)Q_s(j,t) + \sum_{i \in A(j)} q_{ij}(t)Q_s(i,t) + \sum_{(i,j) \in R} q_{ij}(t)P(i,t)$

with initial conditions $Q_s(j,s)=o$, for $j \in L_k$, $t \geq s$.
We therefore have

Proposition 4. If $\sum_{j \in L_k} q_{jj}(t)P(j,t)$ is a bounded function

for $t \geq s$, then $Cov(N_s^o, N_t^o) = E[N_s^o] - E[N_s^o]E[N_t^o] + K(o,s) + K(o,t) - K(s,t)$

where $K(x,y) = \int_x^y \sum_{(i,j) \in R} q_{ij}(u)Q_x(i,u)du$

and where $Q_x(j,t)$ satisfies (11) (for $s=x$).

Proof: for $o \leq s \leq t$,

$$Var[N_t^s] = \lim_{x \to 1-} [\frac{\partial}{\partial x}G(x,t) + \frac{\partial^2}{\partial x^2}G(x,t) - (\frac{\partial}{\partial x}G(x,t))^2]$$

by properties of probability generating functions. Also, since $N_t^o = N_s^o + N_t^s$ almost surely, $Cov(N_s^o, N_t^o) = (Var[N_t^o] + Var[N_s^o] - Var[N_t^s])/2$,
and the result follows from (10). QED.

Application. If X is a birth-death process whose sample functions give the number of active faults in software as functions of time, then the one dimensional discrete distributions of X can be computed by solving the system of equations (6) for x=1. The system is

(12) $P'(j,t) = -q_{jj}(t)P(j,t) + \sum_{i \in A(j)} q_{ij}(t)P(i,t)$, $j \geq o$,

with initial conditions $P(i,o)=o$, $i \geq 1$, and $P(o,o)=1$, where the states of X are the non-negative integers.

The solution of (12) requires that the intensity functions $q_{ij}(t)$ be specified. The results above can be used to generate estimates of the $q_{ij}(t)$ and ultimately of the distributions $\{P(i,t),\ t \geq o,\ i \geq o\}$ under hypotheses on the $q_{ij}(t)$.

For example, assume that

(13) $q_{ij}(t) = \lambda(t)$ for all pairs (i,j) with $i \geq o$ and $i = j-1$, so that the intensity of transitions corresponding to faults activations is a function of time but not of state. Then

(14) $E[N_t^o] = \int_o^t \sum_{i \geq o} \lambda(u)P(i,u)du$

$= \int_0^t \lambda(u)du$, where N_t^o is the random variable for the count of activated faults, by Corollary 2.1.

Denoting $E[N_t^o]$ as a function of time by $F(t)$, we have

(15) $\lambda(t) = F'(t)$.

Therefore, if the mean function of the activated fault counting process, $F(t)$, were known, we would know the intensity functions for fault activation.

Similarly, if $R(t)$ is the mean function for the repair counting process, then by Corollary 2.1,

(16) $R(t) = \int_0^t \sum_{i \geq 1} \mu_i(u)P(i,u)du$

where $\mu_i(t)$ is the intensity function for transitions from a state of i active faults to a state of i-1 active faults, for $i \geq 1$.

If we assume that there is a finite number, K, of debuggers making repairs and that therefore the intensities $\mu_i(t)$ satisfy

(17) $$\mu_i(t) = \begin{cases} i\mu(t) & , i \leq K \\ K\mu(t) & , i > K \end{cases}$$

for some function of time $\mu(t)$, then (12) becomes

(18) $R(t) = \int_0^t \mu(u)[K + \sum_{i=0}^{K} (i-K)P(i,u)]du$

and

(19) $\mu(t) = R'(t)/(K + \sum_{i=0}^{K} (i-K)P(i,t))$.

Assuming (13) and (17), we could solve the system (12) for the distributions of the active fault count if only we knew the mean functions $F(t)$ and $R(t)$ for the counting processes of activated faults and of repairs.

For purposes of estimation, one may as well assume, given (13), that $F(t)$ is the mean function of a Poisson process. An estimate of $F(t)$ could then be determined by well known methods. We therefore will consider a method of estimating $R(t)$.

The counting process for software repairs will almost certainly have correlated increments, especially if K is finite in (17), and a likelihood function for estimation of $R(t)$ will be difficult if not impossible to give expression in closed form. A next best procedure for estimating $R(t)$ is to take advantage of the first two moments of the repair counting process, and to match moments with a Normal process.

To estimate $R(t)$ we then want to minimize an expression of the form

(20) $\sum_{i,j \leq M} (C_i - R(t_i))V_{ij}(C_j - R(t_j))$

where C_i is a count of repairs made prior to t_i, $i=1,\ldots,M$, and where V_{ij} is an entry of the covariance matrix $[\mathrm{Cov}(N^0_{t_i}, N^0_{t_j})]$ of the repair counting process.

Proposition 4 suggests an iterative scheme for the estimation of R(t) by minimizing (20). We assume that an estimate of R(t) can be made by estimating a vector of parameters $(\alpha_1,\alpha_2,\ldots,\alpha_n)=\bar{\alpha}$ upon which R(t) depends. For practical purposes, the scheme then takes the following form. Given an initial estimate $(\alpha_1^{(1)},\alpha_2^{(1)},\ldots,\alpha_n^{(1)})=\bar{\alpha}^{(1)}$ and assuming an estimate of F(t), apply (13), (15), (17), and (19) to compute the solution of (12). Next, apply Corollary 1.1, computing solutions of (11), and apply Proposition 4 to compute an estimate of V, the covariance matrix of the repair counting process.

Minimize (20) for this V for a new estimate $(\alpha_1^{(2)},\alpha_2^{(2)},\ldots\alpha_n^{(2)})=\bar{\alpha}^{(2)}$, and finally, generate a sequence $\left\{\bar{\alpha}^{(i)}\right\}$ of parameter vectors, with $\bar{\alpha}^{(i+1)}$ generated from $\bar{\alpha}^{(i)}$ as $\bar{\alpha}^{(2)}$ was generated from $\bar{\alpha}^{(1)}$.

If the sequence $\left\{\bar{\alpha}^{(i)}\right\}$ is convergent then we will have succeeded in estimating R(t) by matching the first two moments of the repair counting process with a Normal process. Note: the sequence $\left\{\bar{\alpha}^{(i)}\right\}$ may converge slowly. In practice therefore, the sequence is subjected to a Steffensen acceleration. For every n+2 elements of the sequence $\left\{\bar{\alpha}^{(i)}\right\}$, an approximate Jacobian matrix is constructed using finite differences, etc., and an element $\mathbb{F}$ of the accelerated sequence of parameter vectors is generated which becomes the starting vector of a new $\bar{\alpha}^{(i)}$ sequence. In addition, to save computer time, elements of the $\bar{\alpha}^{(i)}$ sequence are not required to actually minimize (20) for various V but only to be terminal elements of a truncated sequence of iterates in an attempt to minimize (20) by a Newton-Raphson method. Finally, it is some $\mathbb{F}$ that, at convergence, is tested as a minimum of (20).

Given estimates of the mean functions F(t) and R(t), the intensities $q_{ij}(t)$ in (12) can in principle be computed. By Theorem 1., we may then refine the process X by various transition counting processes and compute various quantities pertinent to software reliability. For the following applications, we assume X to be a refinable process, and that some method such as outlined above has been used to generate estimates of F(t), R(t), and the $q_{ij}(t)$.

Distribution of Time to Next Repair

For $s \geq o$, we may consider the process $\left\{N_t^s,\ t \geq o\right\}$ counting the number of repair transitions subsequent to time s. We count repairs via the relation $\left\{(i,i-1),\ i \geq 1\right\}$ and define the process $\left\{N_t^s, t \geq o\right\}$ accordingly.

Corollary 2.2 then states that

$$(21)\quad F_n(t) = \int_s^t \sum_{i \geq 1} \mu_i(u) P(i,n,u)\,du,$$

where $F_n(t)$ is the distribution function for time to the n+1 st repair occurring after time s, where the intensity functions $\mu_i(t)$ are precisely the same as in (16) for $t>s$, and where the $P(i,n,t)$ satisfy the system of equations under (3), (4) and (5) above, with the intensities in (12). In particular, the $P(i,n,t)$ satisfy the system

$$(22)\quad P'(i,n,t)=\mu_{i+1}(t)P(i+1,n-1,t)+\lambda_{i-1}(t)P(i-1,n,t)$$

$$-(\mu_i(t)+\lambda_i(t))P(i,n,t),\ n>o, i>o,$$

$$P'(o,n,t)=\mu_1(t)P(1,n-1,t)-\lambda_o(t)P(o,n,t),\ n>o,$$

$$P'(i,o,t)=\lambda_{i-1}(t)P(i-1,o,t)-(\mu_i(t)+\lambda_i(t))P(i,o,t),\ i>o, \text{ and}$$

$$P'(o,o,t)=-\lambda_o(t)P(o,o,t), \text{ for } t \geq s,$$

with initial conditions

$$(23)\quad P(i,n,s)=o,\ n>o,\ i \geq o,$$
$$P(i,o,s)=P(i,s),\ i \geq o.$$

In case n=o in (21) we need only solve equations of (22) for n=o, that is, independently of solutions of (22) for n>o.

Distributions of Time to Last Repair and Time to Last Fault

The probability P(s), that there are no repairs in (s,∞) is, from (21)

with n=o

$$(24)\quad P(s) = 1-\int_s^\infty \sum_{i\geq 1} \mu_i(u)Q_s(i,u)du,$$

where $Q_s(i,t) \triangleq P(i,o,t)$ of the system (22) with $Q_s(i,s)=P(i,s)$.

We can re-write (24) as

$$(25)\quad P(s) = \lim_{t\to\infty} \sum_{i\geq o} Q_s(i,t),$$ and therefore the conditional

distribution of time to last repair as

$$(26)\quad D(t) = \frac{P(t)-P(o)}{1 - P(o)} .$$

In the case K=∞ in (17) and assuming (13) so the increments of the repair counting process are Poisson distributed,

$$(27)\quad Q_s(i,t) = \frac{(F(t)-R(t))^i}{i!} e^{R(s)-F(t)} ,$$

and D(t) becomes

$$(28)\quad D(t) = \frac{e^{R(t)-R(\infty)} - e^{-R(\infty)}}{1-e^{-R(\infty)}} .$$

Similar arguments show that the time-to-last-fault distribution is given by (28), with F in place of R, assuming (13).

We may similarly show that the conditional distribution of time to nth from last fault is

$$(29)\quad D_n(t) = \frac{\int_{F(\infty)-F(t)}^{F(\infty)} u^n e^{-u}du}{\int_o^{F(\infty)} u^n e^{-u}du} .$$

Distribution of Time to Next Extinction

The distribution of time to n+1 st transition after time s from one state

of the active fault process to an adjacent state is, by Corollary 2.2,

$$(30)\quad F_n(t) = \int_s^t q_{ij}(u)P(i,n,u)du.$$

The distribution of time to next extinction of the active fault population is of particular interest in relation to software reliability. By (30), this distribution is

$$(31)\quad F_o(t) = \int_s^t \mu_1(u)P(1,o,u)du,$$

where, by (3), (4) and (5), $P(i,o,t)$ satisfies

$$(32)\quad P'(i,o,t)=\mu_{i+1}(t)P(i+1,o,t)+\lambda_{i-1}(t)P(i-1,o,t) \\ -(\mu_i(t)+\lambda_i(t))P(i,o,t),\ i>o, \\ P'(o,o,t)=-\lambda_o(t)P(o,o,t),$$

with $P(i,o,s)=P(i,s)$ for $i\geq o$.

The functions $\mu_i(t)$, $i\geq 1$ and $\lambda_i(t)$, $i\geq o$ are exactly the same as for equation (12), for $t>s$.

Covariance Function of the active Fault Count

Define relations $A=\{(i,i+1),\ i\geq o\}$, $B=\{(i,i-1),\ i\geq 1\}$, $C=A\cup B$, and let $Q_s(i,t)$ be the solutions of (11) for $R=A$, $\overline{Q}_s(i,t)$ the solutions of

(11) for $R=B$, and similarly define $\bar{\bar{Q}}_s(i,t)$ for $R=C$, for $i\geq o$, $o\leq s<t$.

Let $K(x,y)=\int_x^y \sum_{(i,j)\in A} q_{ij}(u)Q_x(i,u)du$,

$\overline{K}(x,y)=\int_x^y \sum_{(i,j)\in B} q_{ij}(u)\overline{Q}_x(i,u)du$, and

define $\bar{\bar{K}}(x,y)$ analogously.

Let $J(x,y)=2K(x,y)+2\overline{K}(x,y)-\bar{\bar{K}}(x,y)$, and set $D_x(i,y)=2Q_x(i,y)-\bar{\bar{Q}}_x(i,y)$

for $i\geq o$, $u\geq x$. Observe that $D_x(i,y)=\bar{\bar{Q}}_x(i,y)-2\overline{Q}_x(i,y)$, applying (11).

Then $J(x,y)=\int_x^y \sum_{i\geq o}(\lambda_i(u)-\mu_i(u))D_x(i,u)du$

where $\lambda_i(t)=q_{ij}(t)$ for $(i,j)\in A$ and $\mu_i(t)=q_{ij}(t)$ for $(i,j)\in B$ and where $\mu_o(t)=o$.

Next, write $J(o,x)+J(o,y)-J(x,y)$

$= 2J(o,x)+\int_x^y \sum_{i\geq o}(\lambda_i(u)-\mu_i(u))(D_o(i,u)-D_x(i,u))du$, and note that

$D_o(i,t)=iP(i,t)$, $t\geq o$, $i\geq o$ by considering $D_o(i,t)/P(i,t)$ as a conditional expectation or by application of (11), and that $\widetilde{D_s}(i,t) \triangleq D_o(i,t)-D_s(i,t)$ satisfies (12) for $i\geq o$, $t\geq s$ with initial conditions $\widetilde{D_s}(i,s)=iP(i,s)$, $i\geq o$.

Therefore, $J(o,x)+J(o,y)-J(x,y)=2J(o,x)+\sum_{i\geq 1} i\widetilde{D_x}(i,y)-\sum_{i\geq 1} i^2P(i,x)$.

Associating counting processes $\left\{N_t^o,\ t\geq o\right\}$, $\left\{\overline{N}_t^o,\ t\geq o\right\}$ and $\left\{N_t^o+\overline{N}_t^o,\ t\geq o\right\}$ with relations A, B, and C above, we apply simple properties of covariances to conclude

$$\text{Cov}\ (X_s,X_t) = 2\ \text{Cov}\ (N_s^o,N_t^o)+2\ \text{Cov}\ (\overline{N}_s^o,\ \overline{N}_t^o)$$
$$-\text{Cov}(N_s^o+\overline{N}_s^o,\ N_t^o+\overline{N}_t^o)$$

where $X_t=N_t^o-\overline{N}_t^o$ as usual.

Applying Proposition 4 and some algebra,
$\text{Cov}(X_s,X_t) = F(s)+R(s)-(F(t)-R(t))(F(s)-R(s))+ J(o,s)+J(o,t)-J(s,t)$, $t\geq s$,
from which $\sum_{i\geq 1} i^2P(i,s)=2J(o,s)+F(s)+R(s)$.

Using the calculations above, we get

(33) $\text{Cov}\ (X_s,X_t)=\sum_{i\geq 1} i\widetilde{D_s}(i,t)-F(t)-R(t))(F(s)-R(s))$.

Finally, we conclude

Proposition 5. If X is a refinable process and $\sum_{i\geq o}(\lambda_i(t)+\mu_i(t))P(i,t)$ is a bounded function, then for $t\geq s$,

(34) $\text{Cov}\ (X_s,X_t)=\sum_{i\geq 1} iD_s(i,t)$

where $D_s(i,t)$ satisfies equations (12) for $i\geq o$, $t\geq s$, with initial conditions $D_s(i,s)=(i-F(s)+R(s))P(i,s)$, $i\geq o$.

Proof: Observe that $(F(t)-R(t)(F(s)-R(s)) = \sum_{i\geq 1} i(F(s)-R(s))P(i,t)$ and that $D_s(i,t) \triangleq \widetilde{D_s}(i,t)-(F(s)-R(s))P(i,t)$ satisfy (12). Apply (33). QED.

Corollary 5.1. Assume (13) above and that $K=\infty$ in (19). Then

$$\mathrm{Cov}(X_s,X_t)=(F(s)-R(s))e^{-\int_s^t \mu(u)du}, \text{ for } t\geq s,$$

where $\mu(t)=R'(t)/(F(t)-R(t))$.

Proof: Assuming the limiting case of (19), we have $\mu_i(t)=i\mu(t)$ for $i\geq 0$, with $\mu(t)$ as stated. Let $Z(t)=\sum_{i\geq 1} iD_s(i,t)$ where $D_s(i,t)$ is as in Proposition 5.

Then from (12),

$$Z(t)= C+\int_s^t \sum_{i\geq 0}(\lambda_i(u)-\mu_i(u))D_s(i,u)du$$

$$= C-\int_s^t \mu(u)Z(u)du, \text{ where C is a constant, so } Z'(t)=-\mu(t)Z(t),\ t\geq s,$$

Therefore,

$$Z(t)=(F(s)-R(s))e^{-\int_s^t \mu(u)du},$$

solving the differential equation, and using $Z(s)=F(s)-R(s)$. QED.

6. Summary. By requiring the active fault count process X to satisfy appropriate conditions, we can estimate the mean functions of the counting processes for activated faults and for software repairs, as long as the estimation uses only a few moments of the distributions of these processes. The same conditions can then be used to restate the differential equations for the one-dimensional densities of X in terms of the above mean functions, and secondly to state the differential equations and initial conditions for computation of distributions such as time to next repair, time to next fault, time to next extinction of the active fault population, etc., that are important to estimating software reliability.

If the active fault count process is assumed to be a refinable process, then the process can be refined by counting selected transitions occurring after a given time s.

The appropriate choice of refinement can be used to develop higher moments of the activated fault or repair counting process, resulting in an iterative scheme for estimating the mean functions of these counting processes, or to develop distribution functions of random variables interesting from the perspective of software reliability. Since a refinement of a refinable process is itself refinable, one can even develop expressions for such quantities as the distribution of time to first repair subsequent to first extinction, etc.

A computer program in FORTRAN has been written by this author that takes as input the times of software faults and times of software repairs and performs the computations outlined above. The program assumes (13) and (17) for estimating the fault counting process mean function and the repair counting process mean function, is capable of estimating F(t) by maximization of a Poisson likelihood function and of estimating R(t) for a finite number of debuggers by minimization of (20) under the constraint

that the covariance function satisfies Proposition 4.

The program operates as follows. Interval counts of activated faults are formed at selected times from data and a Poisson mean function is estimated via maximization of a likelihood function. Following this, interval counts of repairs are made, forming the C_i of (20). Next, the iterative scheme for minimization of (20), applying Proposition 4, is performed. As a by-product of this minimization values of $\mu(t)$, satisfying (19), and $P(i,t)$, $t \geq o$, $i \geq o$, satisfying (12) are computed. When the iteration converges, these values are retained; the $\mu(t)$ are subsequently used for the solution of systems such as (22) and (32) above, and the distributions $\{P(i,t_j),\ i \geq o\}$ for various t_j are used to establish initial conditions for solution of these systems. Computations involving various distribution functions, applying Corollary 2.2 for various $s \geq o$, are then made.

As a result, the program is also capable of producing values of the following (as functions of time, at various times): conditional mean and standard deviation of time to next fault, mean cumulative activated faults, mean cumulative repairs, the probability that the number of faults remaining to be activated is at most N (for various N), the probability that no new faults will be activated for an interval of length L (for various L), mean and standard deviation of the count of entries (via repair) of a state of N active faults (for various N), mean and standard deviation of the conditional distribution of time to next entry (via repair) of a state of N active faults (for various N), and the discrete distribution of the active fault count.

REFERENCES

1. J. L. Doob, Stochastic Processes, Wiley, New York, 1953.

2. W. A. Fuller, Introduction to Statistical Time Series, Wiley, New York, 1976.

3. P. Henrici, Elements of Numerical Analysis, Wiley, New York, 1964.

4. D. K. Lloyd and M. Lipow, Reliability: Management, Methods, and Mathematics, D. K. Lloyd and M. Lipow, Redondo Beach, CA, 1977.

5. E. Parzen, Stochastic Processes, Holden-Day, San Francisco, CA, 1962.

6. H. L. Royden, Real Analysis, Macmillan, New York, 1968.

7. W. Rudin, Real and Complex Analysis, McGraw-Hill, New York, 1966.

8. G. J. Schick and R. W. Wolverton, An Analysis of Competing Software Reliability Models, IEEE Transactions on Software Engineering, Vol. SE-4, No. 2, 1978.

9. M. L. Shooman, Software Reliability: Measurement and Models, Proceedings of the 1975 Annual Reliability and Maintainability Symposium, 1975.

10. D. L. Snyder, Random Point Processes, Wiley, New York, 1975.

11. A. N. Sukert, An Investigation of Software Reliability Models, Proceedings of the 1977 Annual Reliability and Maintainability Symposium, 1977.

12. A. K. Trivedi and M. L. Shooman, A Many-State Markov Model for the Estimation and Prediction of Computer Software Performance Parameters, Proceedings of the 1975 International Conference on Reliable Software, 1975.

EXAMPLE APPLICATIONS OF SOFTWARE RELIABILITY ESTIMATION *

Leslie F. Claudio
United States Army Electronic Proving Ground
Software and Automation Branch
Fort Huachuca, Arizona 85613

Donna L. Kaspersen
Bell Technical Operations Corporation
Software/Computer Evaluation Facility
600 North Garden Avenue
Sierra Vista, Arizona 85635

ABSTRACT. Methods of software reliability estimation have been applied to data collected on two large software development projects.

Characteristics of software reliability have been derived through application of population process techniques to software failure and repair data. They include the following: mean time between events (faults/repairs), mean time to next fault, number of faults remaining (to be discovered), length of successful mission, and time to last fault.

The results of the application demonstrate the feasibility of the approach. The characteristics provide a basis for judgement of the quality and maturity of software.

1. Introduction. A computer program has been developed for estimating software reliability characteristics using the mean functions of transition counting processes associated with a birth-death process. Data on fault discovery and repair rates for two large software development projects were analyzed with this program; the results are presented here.

The first data used were published (1) for a large software development project, referred to here as System A, at the completion of the test phase. These data are presented in histogram form for faults and repairs as two sets of event counts within each of 18 time units of equal length.

The second data analyzed, designated as System B, were collected on a software development project during the middle testing phase. These data consist of fault discovery times and repair installation times, where the time unit used in the analysis is months of operation time at event occurrence.

For each set of data, mean functions of fault and repair counting processes were developed by assuming a parametric form and fitting the data for parameter estimation. The counting process mean functions were then used to develop estimates of six software reliability characteristics:

(1) D. K. Lloyd and M. Lipow, Reliability: Management, Methods, and Mathematics (Second Edition), p. 519; Redondo Beach, CA, 1977.

* Sponsored by the United States Army under Contract No. DAEA18-77-C-0134.

Mean Time Between Events (MTBF, MTBR)

The instantaneous mean time between faults at time t is the reciprocal of the derivative of the mean number of faults activated per unit time at time t. Similarly, the instantaneous mean time between repairs at time t is the reciprocal of the derivative of the mean number of repairs installed per unit time at time t. This basic estimate leaves the variation of the estimate unknown and, taken alone, is easily misinterpreted.

Mean Time to Next Fault

If a growth in reliability is to occur, it is reasonable to hope that the appearance of new faults is less than certain and that the probability of new faults will decrease. Therefore, we consider the distribution of time to next fault conditioned on the event that a next fault occurs.

Number of Faults Remaining

After the fault mean function is estimated, the expectation of number of faults not yet activated after a time t_1 is $F(\infty)-F(t_1)$. This information is presented as the probability that the number of faults to be activated after time t is at most N, for selected values of N.

Length of Successful Mission

The reliability of software can be expressed in a useful way as the probability that the software will function under typical input conditions for a specific length of time without the appearance of new faults. The input conditions must be assumed to be equivalent to the test conditions with respect only to data input.

Time to Last Fault

An estimate useful for management of the test cycle is an estimate of the time to bring software to a given level of competence. The distribution of time until last fault activation also serves as a lower bound on the time to last repair estimate. This estimate is presented as the mean time to last fault and a table of probabilities that the last fault has been activated prior to given times.

2. Application. The Fault Mean Function was assumed to be that of a Poisson process with the parametric form $F(t)=\alpha(1-e^{-\beta t^{\rho}})$*. With this form, the expectation of total fault activations is $F(\infty)=\alpha$. The parameters β and ρ determine F(t) as the product of a Weibull distribution and a constant.

For System A, the fault mean function was estimated by two methods; maximization of a Poisson likelihood function and by generalized least squares. Each method yielded the same parametric estimates, below to six place accuracy, for both systems.

*For clarification: $F(t) = \alpha(1 - \exp[-\beta t^{\rho}])$.

System	A	B
α =	764.704	898.903
β =	.094	9.259
ρ =	1.183	3.273

For System A, Figure 1 presents the Fault Mean Function with one standard deviation bounds plotted versus observations. Examination shows that each observation unit fell at or within one standard deviation.

Figure 2 presents the same plot for System B. Examination shows that the fit was not as good for System B, with a third of the observations at or slightly exceeding one standard deviation from the mean.

The Repair Mean Function form was selected to constrain the eventual mean total count of repairs to equal the eventual mean total count of faults. The form also satisfies the constraint that the fault mean be an upper bound for the repair mean at all times. The forms of $R(t) = F(t)x$ $(1-e^{-\beta t^{\rho}})$*for System A,and R(t) defined via $R'(t)=\rho' e^{-\beta' t}(F(t)-R(t))$ for System B meet the desired constraints and are convenient, as both have easily computed values and depend on only two parameters. For both System A and B, the repair counting process mean function was estimated by the generalized least squares method resulting in the following parameter estimates.

System	A	B
β'	.151	.169
ρ'	1.029587	15.331

For System A, Figure 3 presents the Repair Mean Function with one standard deviation bounds plotted versus observations. The mean function deviates greatly from the data in interval 12 where nearly 100 repairs were installed within one interval. At interval 15 the data and mean function again coincide. This type of data discontinuity is not unexpected in repair rates as it is common practice to accumulate corrections and install them at a convenient time.

For System B, Figure 4 shows a more consistent repair installation rate during the represented phase of testing.

The instantaneous MTBF of System A, listed as a function of 40 time units in Figure 5, shows a steady increase throughout the 18 time units of test, with a projected dramatic increase if the test and repair phase had continued for 40 units. The MTTR remained relatively stable through the testing inter-

*For clarification: $R(t) = F(t)(1 - \exp[-\beta' t^{\rho'}])$. We assume that F(t) has been previously determined.

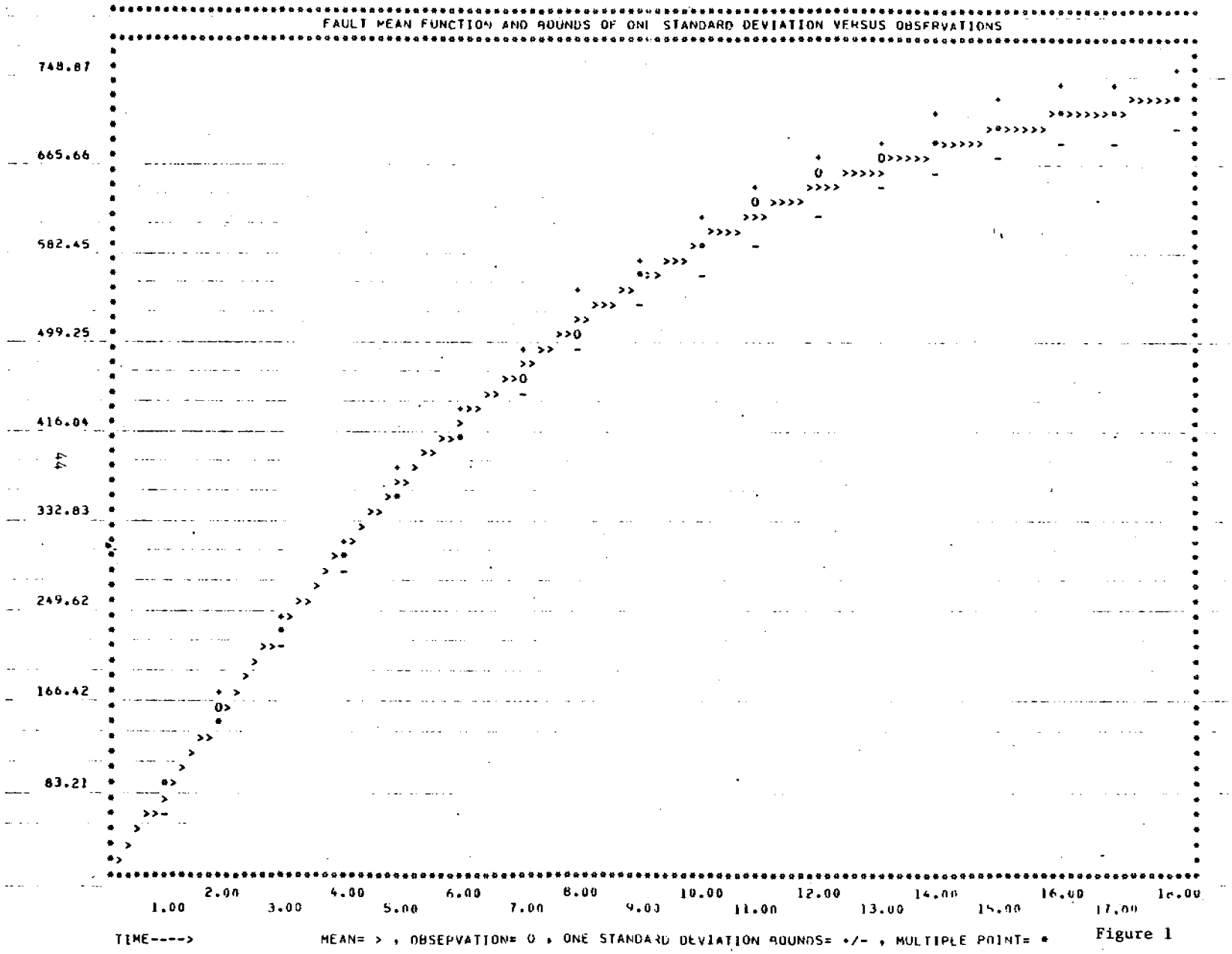

FAULT MEAN FUNCTION AND BOUNDS OF ONE STANDARD DEVIATION VERSUS OBSERVATIONS
748.87
665.66
582.45
499.25
416.04
332.83
249.62
166.42
83.21
1.00
2.00
3.00
4.00
5.00
6.00
7.00
8.00
9.00
10.00
11.00
12.00
13.00
14.00
15.00
16.00
17.00
TIME---->
MEAN= > , OBSERVATION= 0 , ONE STANDARD DEVIATION BOUNDS= +/- , MULTIPLE POINT= *

Figure 1

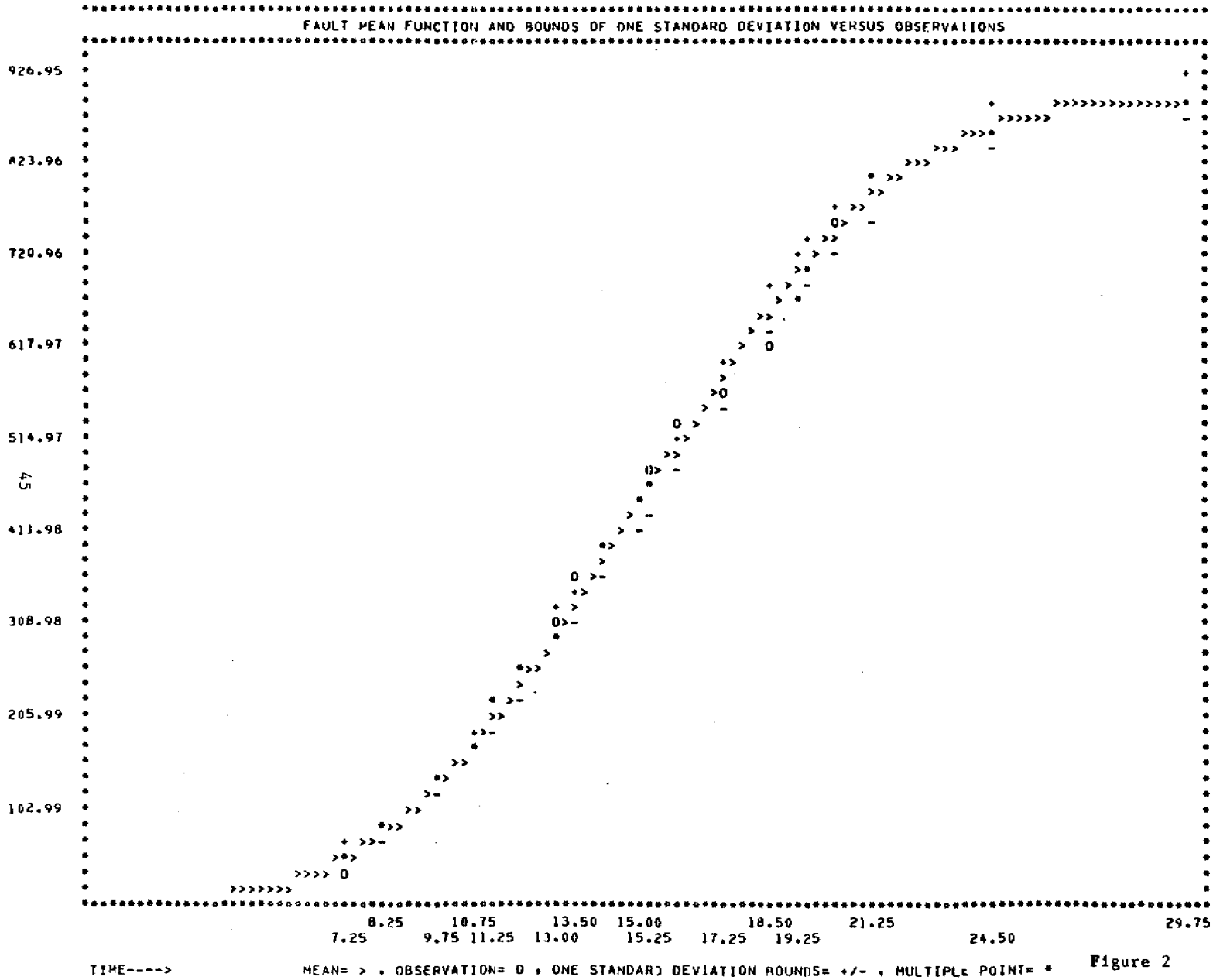

FAULT MEAN FUNCTION AND BOUNDS OF ONE STANDARD DEVIATION VERSUS OBSERVATIONS
926.95
823.96
720.96
617.97
514.97
411.98
308.98
205.99
102.99
45
7.25
8.25
9.75
10.75
11.25
13.00
13.50
15.00
15.25
17.25
18.50
19.25
21.25
24.50
29.75
TIME---->
MEAN= > , OBSERVATION= 0 , ONE STANDARD DEVIATION BOUNDS= +/- , MULTIPLE POINT= *

Figure 2

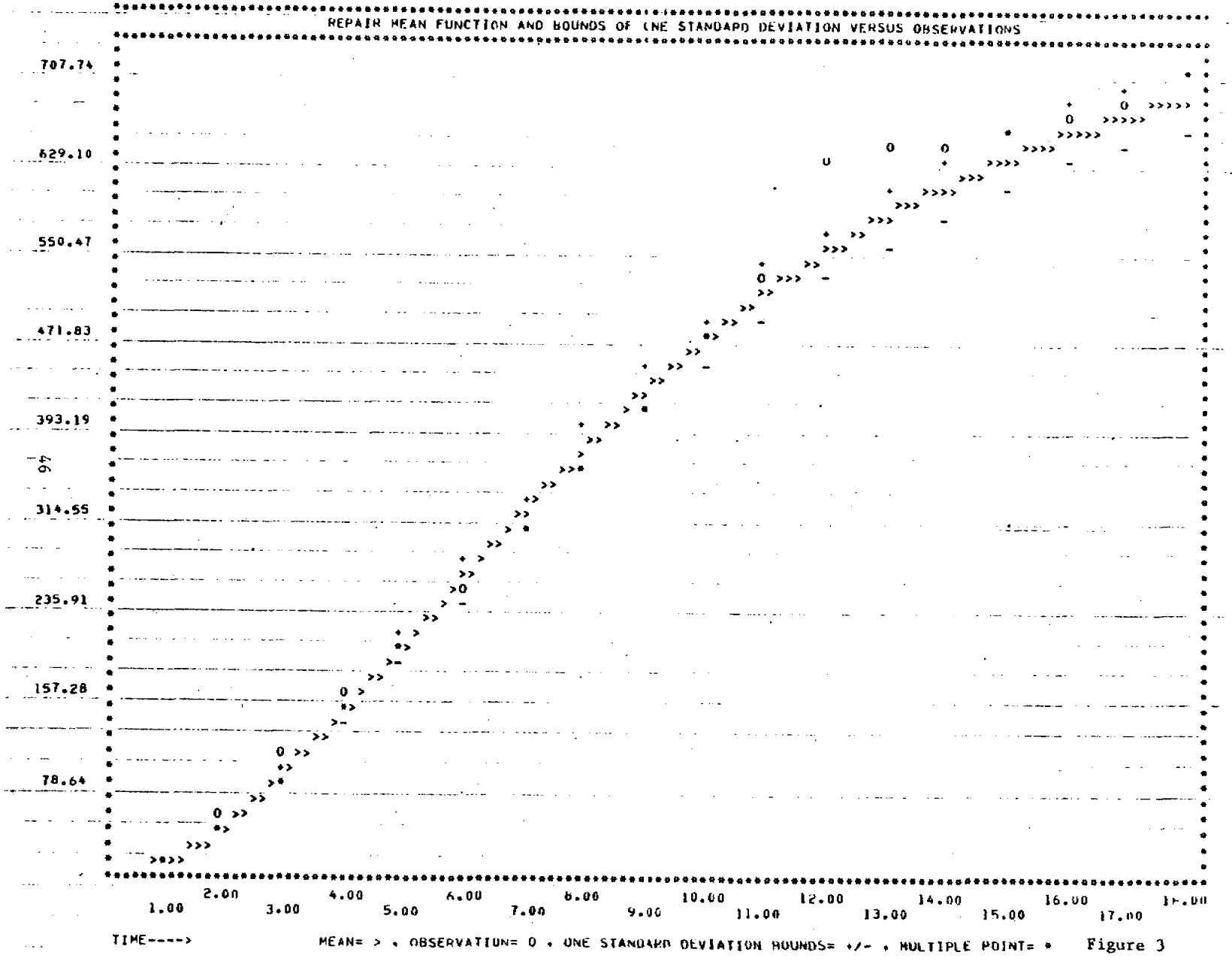

REPAIR MEAN FUNCTION AND BOUNDS OF ONE STANDARD DEVIATION VERSUS OBSERVATIONS
707.74
629.10
550.47
471.83
393.19
314.55
235.91
157.28
78.64
1.00
2.00
3.00
4.00
5.00
6.00
7.00
8.00
9.00
10.00
11.00
12.00
13.00
14.00
15.00
16.00
17.00
TIME---->
MEAN= > , OBSERVATION= 0 , ONE STANDARD DEVIATION BOUNDS= +/- , MULTIPLE POINT= *

Figure 3

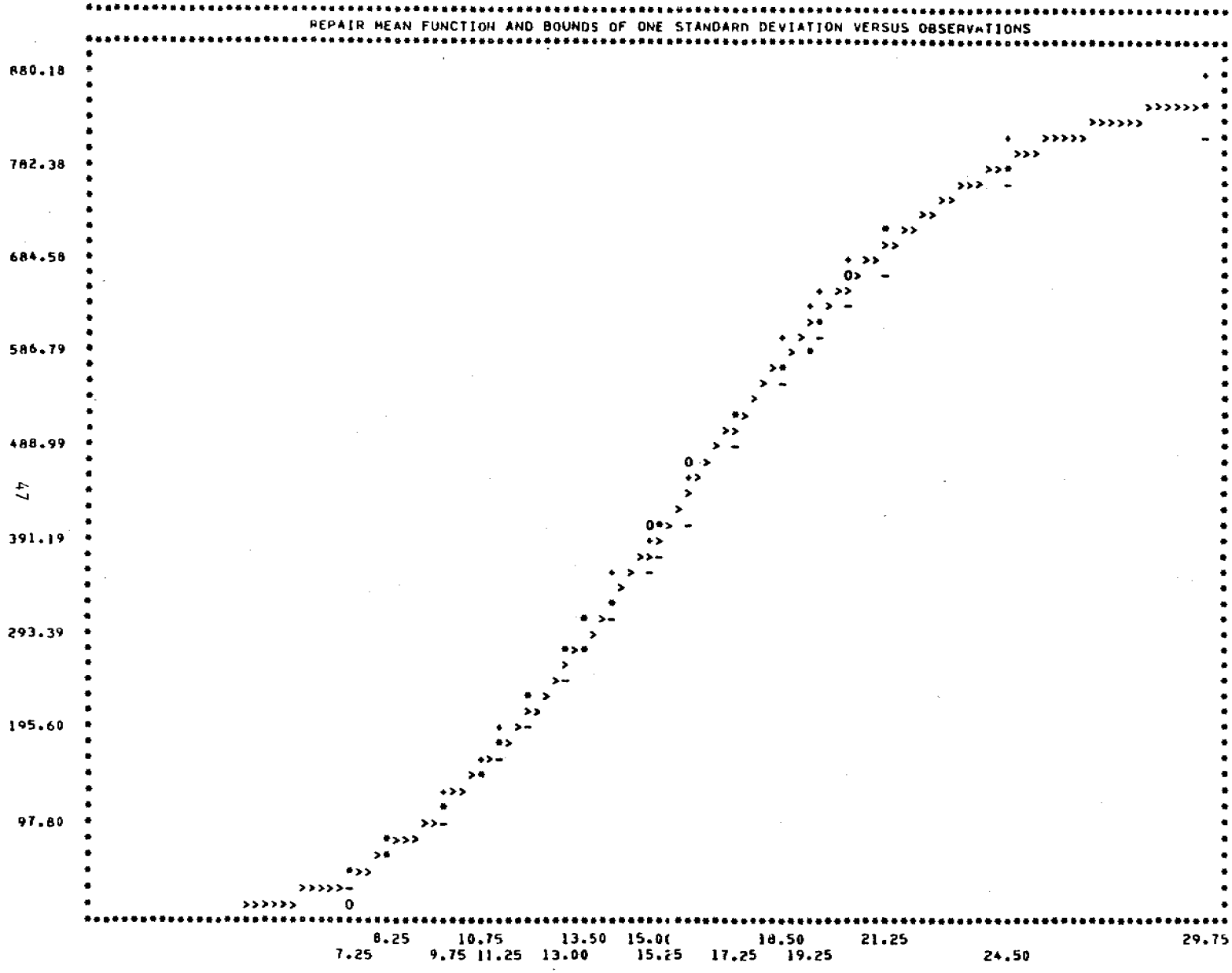

REPAIR MEAN FUNCTION AND BOUNDS OF ONE STANDARD DEVIATION VERSUS OBSERVATIONS
880.18
782.38
684.58
586.79
488.99
391.19
293.39
195.60
97.80
7.25
8.25
9.75
10.75
11.25
13.50
13.00
15.00
15.25
17.25
18.50
19.25
21.25
24.50
29.75
TIME---->
MEAN= > , OBSERVATION= O , ONE STANDARD DEVIATION BOUNDS= +/- , MULTIPLE POINT= *

Figure 4

MEAN TIME BETWEEN FAULTS AND MEAN TIME BETWEEN REPAIRS AT SELECTED TIME POINTS

TIME ABSCISSA	MTBF	MTBR
1.00	.013	.050
2.00	.013	.026
3.00	.014	.020
4.00	.015	.018
5.00	.016	.017
6.00	.019	.017
7.00	.021	.018
8.00	.024	.019
9.00	.028	.020
10.00	.032	.023
11.00	.038	.026
12.00	.044	.029
13.00	.052	.033
14.00	.062	.038
15.00	.073	.044
16.00	.087	.052
17.00	.103	.061
18.00	.123	.071
19.00	.148	.084
20.00	.178	.100
21.00	.214	.118
22.00	.258	.141
23.00	.312	.168
24.00	.377	.200
25.00	.458	.239
26.00	.556	.287
27.00	.677	.344
28.00	.826	.413
29.00	1.009	.496
30.00	1.234	.596
31.00	1.512	.717
32.00	1.855	.864
33.00	2.278	1.041
34.00	2.803	1.255
35.00	3.452	1.514
36.00	4.258	1.827
37.00	5.258	2.206
38.00	6.500	2.663
39.00	8.046	3.216
40.00	9.971	3.885

Figure 5

val but would have been expected to increase if testing had continued for 40 units.

The instantaneous MTBF of System B over 32 months is shown in Figure 6. The test period corresponds to the first 30 units (29.75 time abscissa). The MTBF shows a decline as testing began and faults were activated with a stable low point over eight months before growth began. This testing covered the initial period of system integration and is, therefore, consistent with intuitively expected results as major modules initially interact and interface problems are discovered. The MTTR decreased through the stable fault activities interval and increased as the fault activation rate decreased. This may be interpreted as representing intense debugging effort (long hours, more debuggers, high priority when faults were at a high rate) with a relaxation of effort as the situation was placed under control.

The Mean Time to Next Fault for System A is shown in Figure 7. The issue of whether or not a next fault will occur is not in question until time unit 27 and the mean time to next fault shows a steady growth. The same estimation for System B is shown in Figure 8. The same pattern is evident.

The table of values for estimating the number of faults remaining to be discovered is given at Figure 9 for System A and Figure 10 for System B. The selected values of N are incremented by 5 through the 0-25 range. The probability that there were no more than 25 faults remaining in the software of System A at the end of testing was only .104; for System B it is 1.00. These results appear plausible for System A, but highly implausible for System B unless the System B results are an indication that the testing conditions of integration testing were nearing the end of their effectiveness and would invoke few remaining faults. As System B testing continues under more strenuous conditions, the author expects new fault types to be displayed with a probable new fault mean function for the next test phase.

The expected length of a successful mission was analyzed for both System A and B where the mission is assumed to be using the system under the input conditions of the test period. Figure 11 presents the results for System A for missions with respect to length L, where L=.143 of the time unit. At the end of 40 time units of testing, the probability of executing (or testing) for L time with no new faults is .986. Figure 12 presents the results for System B in terms of L, where L is approximately one day. At the end of testing (29.75 time abscissa) the probability of executing for one day with no new faults is .955.

The Time to Last Fault Distribution for System A, presented at Figure 13 for times from 40 to 60, has a mean of 38.75 time units and a standard deviation of 5.082. Although the testing was stopped at time abscissa 18, the distribution suggests that a test period to 44 time units had an 82.8 percent probability of disclosing all faults and testing to 46.8 would have had 90.5 percent probability of disclosing all of the faults. This information would have been useful to take into consideration with the cost of testing and intended use of the system. For System B, Figure 14, the mean was 31.284 months and the standard deviation was 1.425. The test period (29.75) had an estimated 18 percent probability of disclosing all faults; a test period of 34 months would have had a 94 percent probability of disclosing all faults.

MEAN TIME BETWEEN FAULTS AND MEAN TIME BETWEEN REPAIRS AT SELECTED TIME POINTS

TIME ABSCISSA	MTBF	MTBR
7.25	.043	.049
8.25	.033	.038
9.75	.024	.028
10.75	.021	.024
11.25	.019	.022
12.00	.018	.020
13.00	.016	.019
13.50	.016	.018
14.25	.015	.018
15.00	.015	.017
15.25	.015	.017
16.00	.015	.017
17.25	.016	.018
18.50	.018	.019
19.25	.019	.021
19.50	.020	.021
20.25	.023	.023
21.25	.027	.027
24.50	.067	.049
29.75	.775	.169
29.95	.875	.177
30.15	.990	.186
30.35	1.123	.195
30.55	1.276	.205
30.75	1.454	.215
30.95	1.660	.225
31.15	1.899	.236
31.35	2.178	.247
31.55	2.503	.259
31.75	2.884	.271
31.95	3.330	.284
32.15	3.855	.297

Figure 6

PROBABILITY OF A NEXT FAULT AND CONDITIONAL MEAN AND STANDARD DEVIATION OF TIME TO NEXT FAULT

POINT	TIME ABSCISSA	PROBABILITY OF A NEXT FAULT	CONDITIONAL MEAN TIME TO NEXT FAULT	STANDARD DEVIATION
1	1.00	1.000	.013	.013
2	2.00	1.000	.013	.013
3	3.00	1.000	.014	.014
4	4.00	1.000	.015	.015
5	5.00	1.000	.016	.016
6	6.00	1.000	.019	.019
7	7.00	1.000	.021	.021
8	8.00	1.000	.024	.024
9	9.00	1.000	.028	.028
10	10.00	1.000	.033	.033
11	11.00	1.000	.038	.038
12	12.00	1.000	.045	.045
13	13.00	1.000	.053	.053
14	14.00	1.000	.062	.063
15	15.00	1.000	.074	.075
16	16.00	1.000	.088	.089
17	17.00	1.000	.105	.107
18	18.00	1.000	.126	.129
19	19.00	1.000	.152	.157
20	20.00	1.000	.184	.191
21	21.00	1.000	.223	.234
22	22.00	1.000	.272	.288
23	23.00	1.000	.333	.359
24	24.00	1.000	.410	.454
25	25.00	1.000	.509	.586
26	26.00	1.000	.638	.774
27	27.00	.999	.807	1.036
28	28.00	.997	1.025	1.372
29	29.00	.992	1.294	1.763
30	30.00	.980	1.608	2.172
31	31.00	.957	1.947	2.565
32	32.00	.923	2.292	2.917
33	33.00	.874	2.621	3.216
34	34.00	.812	2.921	3.461
35	35.00	.741	3.185	3.657
36	36.00	.664	3.410	3.810
37	37.00	.585	3.598	3.928
38	38.00	.507	3.753	4.018
39	39.00	.434	3.878	4.086
40	40.00	.367	3.977	4.137

Figure 7

PROBABILITY OF A NEXT FAULT AND CONDITIONAL MEAN AND STANDARD DEVIATION OF TIME TO NEXT FAULT

POINT	TIME ABSCISSA	PROBABILITY OF A NEXT FAULT	CONDITIONAL MEAN TIME TO NEXT FAULT	STANDARD DEVIATION
1	7.25	1.000	.043	.042
2	8.25	1.000	.033	.033
3	9.75	1.000	.024	.024
4	10.75	1.000	.021	.021
5	11.25	1.000	.019	.019
6	12.00	1.000	.018	.018
7	13.00	1.000	.016	.016
8	13.50	1.000	.016	.016
9	14.25	1.000	.015	.015
10	15.00	1.000	.015	.015
11	15.25	1.000	.015	.015
12	16.00	1.000	.015	.015
13	17.25	1.000	.016	.016
14	18.50	1.000	.018	.018
15	19.25	1.000	.019	.020
16	19.50	1.000	.020	.020
17	20.25	1.000	.023	.023
18	21.25	1.000	.027	.028
19	24.50	1.000	.068	.070
20	29.75	.851	.814	.930
21	29.95	.810	.859	.960
22	30.15	.764	.899	.984
23	30.35	.715	.934	1.003
24	30.55	.663	.964	1.018
25	30.75	.610	.989	1.029
26	30.95	.556	1.010	1.036
27	31.15	.504	1.026	1.040
28	31.35	.452	1.039	1.042
29	31.55	.403	1.048	1.042
30	31.75	.357	1.055	1.040
31	31.95	.314	1.058	1.036
32	32.15	.275	1.060	1.031

Figure 8

PROBABILITY THAT AT MOST N FAULTS ARE ACTIVATED AFTER SELECTED TIME POINTS

POINT	TIME ABSCISSA	VALUE OF N 0	5	10	15	20	25
1	1.00	0.000	.000	.000	.000	.000	.000
2	2.00	0.000	.000	.000	.000	.000	.000
3	3.00	0.000	.000	.000	.000	.000	.000
4	4.00	0.000	.000	.000	.000	.000	.000
5	5.00	0.000	.000	.000	.000	.000	.000
6	6.00	0.000	.000	.000	.000	.000	.000
7	7.00	0.000	.000	.000	.000	.000	.000
8	8.00	0.000	.000	.000	.000	.000	.000
9	9.00	0.000	.000	.000	.000	.000	.000
10	10.00	0.000	.000	.000	.000	.000	.000
11	11.00	0.000	.000	.000	.000	.000	.000
12	12.00	0.000	.000	.000	.000	.000	.000
13	13.00	0.000	.000	.000	.000	.000	.000
14	14.00	0.000	.000	.000	.000	.000	.000
15	15.00	0.000	.000	.000	.000	.000	.000
16	16.00	.000	.000	.000	.000	.013	.104
17	17.00	.000	.000	.000	.000	.013	.104
18	18.00	.000	.000	.000	.000	.013	.104
19	19.00	.000	.000	.000	.000	.013	.104
20	20.00	.000	.000	.000	.003	.050	.259
21	21.00	.000	.000	.001	.035	.244	.634
22	22.00	.000	.000	.013	.172	.585	.900
23	23.00	.000	.001	.072	.450	.858	.985
24	24.00	.000	.009	.231	.741	.970	.999
25	25.00	.000	.041	.479	.916	.996	1.000
26	26.00	.000	.125	.723	.981	1.000	1.000
27	27.00	.001	.273	.885	.997	1.000	1.000
28	28.00	.003	.464	.962	1.000	1.000	1.000
29	29.00	.008	.653	.990	1.000	1.000	1.000
30	30.00	.020	.802	.998	1.000	1.000	1.000
31	31.00	.043	.900	1.000	1.000	1.000	1.000
32	32.00	.077	.954	1.000	1.000	1.000	1.000
33	33.00	.126	.981	1.000	1.000	1.000	1.000
34	34.00	.188	.993	1.000	1.000	1.000	1.000
35	35.00	.259	.997	1.000	1.000	1.000	1.000
36	36.00	.336	.999	1.000	1.000	1.000	1.000
37	37.00	.415	1.000	1.000	1.000	1.000	1.000
38	38.00	.493	1.000	1.000	1.000	1.000	1.000
39	39.00	.566	1.000	1.000	1.000	1.000	1.000
40	40.00	.633	1.000	1.000	1.000	1.000	1.000

Figure 9

PROBABILITY THAT AT MOST N FAULTS ARE ACTIVATED AFTER SELECTED TIME POINTS

POINT	TIME ABSCISSA	VALUE OF N 0	5	10	15	20	25
1	7.25	0.000	.000	.000	.000	.000	.000
2	8.25	0.000	.000	.000	.000	.000	.000
3	9.75	0.000	.000	.000	.000	.000	.000
4	10.75	0.000	.000	.000	.000	.000	.000
5	11.25	0.000	.000	.000	.000	.000	.000
6	12.00	0.000	.000	.000	.000	.000	.000
7	13.00	0.000	.000	.000	.000	.000	.000
8	13.50	0.000	.000	.000	.000	.000	.000
9	14.25	0.000	.000	.000	.000	.000	.000
10	15.00	0.000	.000	.000	.000	.000	.000
11	15.25	0.000	.000	.000	.000	.000	.000
12	16.00	0.000	.000	.000	.000	.000	.000
13	17.25	0.000	.000	.000	.000	.000	.000
14	18.50	0.000	.000	.000	.000	.000	.000
15	19.25	0.000	.000	.000	.000	.000	.000
16	19.50	0.000	.000	.000	.000	.000	.000
17	20.25	0.000	.000	.000	.000	.000	.000
18	21.25	0.000	.000	.000	.000	.000	.000
19	24.50	.000	.000	.000	.000	.013	.104
20	29.75	.149	.987	1.000	1.000	1.000	1.000
21	29.95	.190	.993	1.000	1.000	1.000	1.000
22	30.15	.236	.996	1.000	1.000	1.000	1.000
23	30.35	.285	.998	1.000	1.000	1.000	1.000
24	30.55	.337	.999	1.000	1.000	1.000	1.000
25	30.75	.390	1.000	1.000	1.000	1.000	1.000
26	30.95	.444	1.000	1.000	1.000	1.000	1.000
27	31.15	.496	1.000	1.000	1.000	1.000	1.000
28	31.35	.548	1.000	1.000	1.000	1.000	1.000
29	31.55	.597	1.000	1.000	1.000	1.000	1.000
30	31.75	.643	1.000	1.000	1.000	1.000	1.000
31	31.95	.686	1.000	1.000	1.000	1.000	1.000
32	32.15	.725	1.000	1.000	1.000	1.000	1.000

Figure 10

PROBABILITIES OF SUCCESSFUL MISSIONS OF LENGTHS WHICH ARE QUARTER MULTIPLES OF L= .143 AT SELECTED TIME POINTS

POINT	TIME ABSCISSA	1/4 L	1/2 L	3/4 L	L	5/4 L	3/2 L	7/4 L	2L
1	1.00	.062	.004	.000	.000	.000	.000	.000	.000
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
11	11.00	.390	.153	.060	.024	.009	.004	.002	.001
12	12.00	.447	.201	.091	.041	.019	.009	.004	.002
13	13.00	.505	.256	.130	.067	.034	.018	.009	.005
14	14.00	.561	.315	.178	.101	.057	.033	.019	.011
15	15.00	.614	.378	.233	.144	.090	.056	.035	.022
16	16.00	.663	.441	.294	.196	.131	.088	.059	.040
17	17.00	.708	.503	.358	.255	.182	.131	.094	.067
18	18.00	.749	.563	.423	.319	.241	.182	.138	.105
19	19.00	.786	.619	.488	.385	.305	.241	.192	.152
20	20.00	.818	.671	.550	.452	.372	.306	.253	.209
21	21.00	.847	.718	.609	.517	.440	.374	.319	.272
22	22.00	.871	.759	.663	.579	.506	.443	.388	.340
23	23.00	.892	.796	.712	.636	.569	.510	.457	.410
24	24.00	.910	.829	.755	.688	.628	.573	.524	.479
25	25.00	.925	.856	.793	.735	.682	.632	.587	.545
26	26.00	.938	.880	.827	.776	.730	.686	.645	.607
27	27.00	.949	.901	.855	.812	.772	.734	.698	.664
28	28.00	.958	.918	.880	.843	.809	.776	.744	.714
29	29.00	.965	.932	.900	.870	.840	.812	.785	.759
30	30.00	.972	.944	.918	.892	.868	.844	.821	.799
31	31.00	.977	.954	.932	.911	.890	.870	.851	.832
32	32.00	.981	.962	.944	.927	.910	.893	.877	.861
33	33.00	.985	.969	.955	.940	.926	.912	.899	.885
34	34.00	.987	.975	.963	.951	.939	.928	.917	.906
35	35.00	.990	.980	.970	.960	.950	.941	.932	.923
36	36.00	.992	.983	.975	.967	.960	.952	.944	.937
37	37.00	.993	.987	.980	.974	.967	.961	.955	.949
38	38.00	.995	.989	.984	.979	.973	.968	.963	.958
39	39.00	.996	.991	.987	.983	.978	.974	.970	.966
40	40.00	.996	.993	.989	.986	.983	.979	.976	.973

Figure 11

PROBABILITIES OF SUCCESSFUL MISSIONS OF LENGTHS WHICH ARE QUARTER MULTIPLES OF L= .036 AT SELECTED TIME POINTS

POINT	TIME ABSCISSA	1/4 L	1/2 L	3/4 L	L	5/4 L	3/2 L	7/4 L	2L
1	7.25	.813	.661	.537	.436	.353	.281	.232	.188
2	8.25	.764	.584	.446	.340	.259	.198	.150	.115
3	9.75	.693	.479	.332	.229	.158	.109	.076	.052
4	10.75	.649	.421	.273	.177	.115	.074	.048	.031
5	11.25	.630	.397	.250	.157	.099	.062	.039	.024
6	12.00	.604	.365	.220	.133	.080	.048	.029	.018
7	13.00	.577	.333	.192	.111	.064	.037	.021	.012
8	13.50	.567	.321	.182	.103	.058	.033	.019	.011
9	14.25	.556	.309	.172	.096	.053	.030	.016	.009
10	15.00	.551	.304	.167	.092	.051	.026	.015	.009
11	15.25	.551	.303	.167	.092	.051	.028	.015	.008
12	16.00	.554	.306	.170	.094	.052	.029	.016	.009
13	17.25	.571	.327	.187	.107	.061	.035	.020	.011
14	18.50	.605	.366	.222	.134	.081	.049	.030	.018
15	19.25	.632	.399	.253	.160	.101	.064	.041	.026
16	19.50	.642	.412	.265	.170	.109	.070	.045	.029
17	20.25	.674	.455	.307	.207	.140	.095	.064	.043
18	21.25	.721	.521	.376	.272	.197	.142	.103	.075
19	24.50	.875	.765	.670	.587	.514	.451	.395	.347
20	29.75	.989	.977	.966	.955	.945	.934	.924	.914
21	29.95	.990	.980	.970	.960	.951	.942	.932	.923
22	30.15	.991	.982	.974	.965	.957	.948	.940	.932
23	30.35	.992	.984	.977	.969	.962	.954	.947	.940
24	30.55	.993	.986	.979	.973	.966	.960	.953	.947
25	30.75	.994	.988	.982	.976	.970	.964	.959	.953
26	30.95	.995	.989	.984	.979	.974	.969	.964	.959
27	31.15	.995	.991	.986	.982	.977	.973	.968	.964
28	31.35	.996	.992	.988	.984	.980	.976	.972	.969
29	31.55	.996	.993	.989	.986	.983	.979	.976	.973
30	31.75	.997	.994	.991	.988	.985	.982	.979	.976
31	31.95	.997	.995	.992	.989	.987	.984	.982	.979
32	32.15	.998	.995	.993	.991	.989	.986	.984	.982

Figure 12

VALUES OF THE TIME-TO-LAST-FAULT DISTRIBUTION
WITH MEAN 38.750 AND STANDARD DEVIATION 5.082

POINT	TIME ABSCISSA	DISTRIBUTION VALUE
102	40.80	.682
103	41.20	.704
104	41.60	.726
105	42.00	.746
106	42.40	.764
107	42.80	.782
108	43.20	.799
109	43.60	.814
110	44.00	.828
111	44.40	.842
112	44.80	.854
113	45.20	.866
114	45.60	.877
115	46.00	.887
116	46.40	.896
117	46.80	.905
118	47.20	.912
119	47.60	.920
120	48.00	.926
121	48.40	.933
122	48.80	.938
123	49.20	.944
124	49.60	.948
125	50.00	.953
126	50.40	.957
127	50.80	.960
128	51.20	.964
129	51.60	.967
130	52.00	.970
131	52.40	.972
132	52.80	.975
133	53.20	.977
134	53.60	.979
135	54.00	.981
136	54.40	.983
137	54.80	.984
138	55.20	.986
139	55.60	.987
140	56.00	.988
141	56.40	.989
142	56.80	.990
143	57.20	.991
144	57.60	.992
145	58.00	.992
146	58.40	.993
147	58.80	.994
148	59.20	.994
149	59.60	.994
150	60.00	.995
151	60.40	.996

Figure 13

VALUES OF THE TIME-TO-LAST-FAULT DISTRIBUTION
WITH MEAN 31.284 AND STANDARD DEVIATION 1.425

POINT	TIME ABSCISSA	DISTRIBUTION VALUE
102	32.79	.827
103	33.11	.866
104	33.43	.897
105	33.75	.921
106	34.07	.941
107	34.39	.956
108	34.71	.967
109	35.03	.976
110	35.36	.982
111	35.68	.987
112	36.00	.991
113	36.32	.993
114	36.64	.995
115	36.96	.997
116	37.28	.998
117	37.61	.998
118	37.93	.999
119	38.25	.999
120	38.57	1.000
121	38.89	1.000
122	39.21	1.000
123	39.53	1.000
124	39.96	1.000
125	40.18	1.000
126	40.50	1.000
127	40.82	1.000
128	41.14	1.000
129	41.46	1.000
130	41.78	1.000
131	42.11	1.000
132	42.43	1.000
133	42.75	1.000
134	43.07	1.000
135	43.39	1.000
136	43.71	1.000
137	44.03	1.000
138	44.36	1.000
139	44.68	1.000
140	45.00	1.000
141	45.32	1.000
142	45.64	1.000
143	45.96	1.000
144	46.28	1.000
145	46.61	1.000
146	46.93	1.000
147	47.25	1.000
148	47.57	1.000
149	47.89	1.000
150	48.21	1.000
151	48.53	1.000

Figure 14

3. Summary. These initial applications of the model demonstrate the usefulness of the estimation method and the consistency and plausibility of the estimation. The accuracy of the estimation will be further verified by continued analysis through the additional testing of system B and by application of the model to additional systems under test.

PROBABILISTIC PROGRAM ESTIMATES - COMPARISON OF SIMULATED RESULTS USING BETA VIS-A-VIS TRIANGULAR ACTIVITY DISTRIBUTIONS

Conrad W. Faber
U.S. Army Aviation Research and Development Command
St. Louis, Missouri 63120

ABSTRACT. When developing probabilistic program estimates for systems in the R&D stage from three point estimates of component parts, the triangular distribution is often assumed for the parts.

The model/method proposed assumes the component distributions are described by beta probability densities and compares the results vis-a-vis assuming triangular distributions.

The model assumes:

a. Germane historical statistical data is not available nor is a bottoms up engineering estimate appropriate.

b. Enough knowledge is available to estimate the general shape of the beta distributions.

c. The user has access to computer facilities.

1. INTRODUCTION. To adequately evaluate the risk (time and/or cost) of a new Army program, the decision-maker needs more than a point estimate, i.e., some measure of how much the estimate could be in error. Although methodology and models exist for developing risk profiles based upon historical data, frequently a data base is not available or is not sufficiently analogous or the physical/performance characteristics of the new system are beyond the reliable range of the data base. Consequently, estimates for the new system frequently depend upon expert opinion of a very few knowledgeable persons.

Often the "expert" is asked to provide the range and most likely (modal) estimates of several activities, e.g., costs to fabricate subsystems, integration and test costs, etc. A common method of combining these activity three point estimates is to assume a triangular distribution for each activity and determine a total system cost by Monte Carlo simulation.

The triangular distribution is completely described by the mode and range. The beta distribution includes one additional shape parameter which offers much greater flexibility. Because of its rigidity, the triangular distribution can easily misrepresent the probabilities within the range as shown by an example in Appendix A.

The proposed method described in this paper suggests the expert select one of nine beta distributions which best describes the general shape of the probability distribution for each activity. Appendix B compares these nine distributions vis-a-vis triangular distributions with the same range and modes. The proposed method also includes a simplistic system estimate and compares the results obtained by using both the beta and triangular distributions.

PROPOSED METHOD

GENERAL:

a. Uncertainty (risk) of an activity* can be described by a probability distribution. This uncertainty relates to potential technical risks and economic factors. Another major source of uncertainty, not covered in this paper, is requirements uncertainty, e.g., changes in performance requirements and quantity procured. For a given risk assessment, requirements are assumed to be fixed. Program uncertainty is a convolution of the activities risks.

b. To determine program uncertainty, a PERT type network must be developed displaying the activities and events and their major interdependencies. This network should be correlated to the elements of the program work breakdown structure. For each activity, a distribution describes the uncertainty involved in that activity. When applicable historical data is available or factors assumed, appropriate distributions should be used. This paper describes a method of estimating activity distributions when the above is not available.

c. Once the activity distributions and parameters are specified, a total program (or intermediate milestone) probability distribution can be derived by Monte Carlo simulation. Three models (RISCA, SOLVNET and VERT) are described in DARCOM Handbook 11-1.1-79, Army Programs: Decision Risk Analysis Handbook. Of the models known by the author, VERT (Venture Evaluation and Review Technique) is the most versatile and is used in the sample case in a subsequent section.

d. A major shortcoming of most risk models is the limited number of distributions built into the basic program and/or the amount of subjective probabilistic data to be requested from the "expert." Since many activity distributions are skewed to the right, i.e., the possible range of an overrun exceeds that of an underrun, the standard normal distribution is not appropriate. Also, the frequently used triangular distribution can easily misrepresent the probability densities as shown by an example in Appendixes A and B. The VERT program allows the analyst a choice of over a dozen density functions. These can be used to describe activities in terms of time and/or cost.

e. For a program with many activities, the Central Limit Theorem (CLT) provides an unbiased estimate of the expected mean and variance of the total program cost** by simply adding the expected values of the activity means and variances since the limiting distribution of additive variables (even from skewed distributions) is normal. However, for a few skewed activities, or domination by a few skewed activities, or intermediate milestone distributions based upon a small number of skewed activities, application of the CLT can give misleading results regarding variance and skewness.

---

* Activity is defined as the time or cost to complete a task whereas an event is a point in time, e.g., start of flight testing.

** Estimates of program time or cost as a function of time generally cannot use the CLT. While the expected mean for time can be computed along the critical path, the distribution for time or cost for several activities requires more sophisticated techniques, e.g., simulation.

SELECTION OF DISTRIBUTION(S):

a.When the analyst must determine the activity distributions from subjective inputs, the analyst usually can only zero in on the general shape of the distribution and its associated parameters. Consequently the following criteria was used to select a distribution or distributions:

1. Simplicity
2. Could be symmetric, skewed left, or skewed right
3. Could have varying degrees of kurtosis, i.e., concentration around mean or mode
4. Could be normalized for computer simulation

b. During the 1960's, several theoretical papers (See References 3 thru 8) where written regarding cost uncertainty. All of the authors of these papers chose the beta probability function to describe activities because of its versatility and simplicity. Because of the large amount of computer core and central processing unit (CPU) time required to run a Monte Carlo simulation, most of these earlier authors advocated the convolution of beta distributions by the method of moments. This method provides a total program cost distribution profile but suffers from the same shortcomings as using the Central Limit Theorem discussed earlier. During the decade of the 1970's, little use has been made of this research. However, with todays high speed computers, a complex network can be simulated via Monte Carlo techniques much more efficiently. (Using the VERT program, a complex network can be simulated 1000 times with under 240K core and under 2 minutes CPU time.)

c. This author evaluated several distributions (triangular, gamma, weibull, beta, et al.) and reached the same conclusion that the beta function could adequately describe most activity distributions and was generally superior to other probability functions vis-a-vis the criteria listed above. The preceeding statement is not meant to suggest the exclusive use of the beta distribution when acquiring subjective inputs. There are situations where other distributions maybe more appropriate, e.g., the Poisson distribution for the expected life of a component or the binomial distribution for either/or situations.

d. The beta probability density function (pdf) is:

$$f(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{(a-1)} (1-x)^{(b-1)}, \text{ where } 0 < x < 1$$

The parameters "a" and "b" determine the degree of skewness and kurtosis. The following transformation of actual high (H) and low (L) points of the range conform to the beta pdf range of 0 thru 1.

$x = (X-L)/(H-L)$, where X is the actual data value.

The computer program, given the beta pdf parameters, randomly selects x and then transforms it to X for each iteration through the network.

e. When obtaining subjective probabilistic information, the author has experienced the best results when the choices available to the experts have the following characteristics:

1. Finite end points which exclude extremely unlikely probabilities
2. Unimodal
3. Continuous rather than discrete
4. Few input parameters required
5. A finite set, with visual illustrations, from which to choose

The beta distribution also met the first four elements of this criteria. To meet criteria five, nine representative beta distributions were selected. The first four are skewed to the right with modes 25% and 40% of the way through the range. Distributions five through seven are symmetric and distributions eight and nine slightly skewed to the left with the modes 60% of the way through the range. These nine distributions are displayed in Appendix C.

INPUT REQUIRED:

a. To determine program or subprogram uncertainties, the activities and events must be defined and their interrelationships established. This is best depicted in a PERT type network. Although it is not the purpose of this paper to describe how to construct this network, the following general comments indicate the flexibility available.

1. Branching probability paths can be constructed, e.g., probabilities of failure causing program stop, sufficient problems to cause major redesign, or adequate success to continue work as originally planned. This branching may be activated by cost and/or time constraints.
2. Activities can be described in terms of time or cost risk. Care should be taken to include the interdependancies of events. Activities may have to be subdivided for this purpose, e.g., design of item A into preliminary design and final design of A because the prelimary design of A is required before item B can be designed.
3. Time uncertainty usually assumes a normal work pace (e.g., a 40 hour work week). Analysis can then determine critical activities which allows management the option of selected overtime or reallocation of resources and awareness of which activities/events to monitor closely.
4. Cost is frequently determined as a linear function of time, i.e., cost = a + bx, where a is a base constant and b is a cost per unit of time. This is based on the close relationship between cost and time where time is a function of technical uncertainty.
5. Activities/events become more specific as a program is defined in more detail. E.g., to monitor the risk in an ongoing program, the activities/events for the next 6 months are in more detail than those farther in the future whereas past activities are now given a fixed number.
6. A well constructed program network may combine elements of all the above.

The inputs required for risk analysis can be readily seen from the developed network. The suggested procedure which follows assumes activity estimates cannot be obtained by traditional parametric statistical relationships.

b. Parameters required to describe an activity's uncertainty, using the beta distribution, are: the lower and upper bounds, the most likely value, and a choice of one of the nine beta distributions shown in Appendix C. Note that there is a redundancy between the most likely value and the beta distribution selected. This redundancy provides a check on the consistency of the information provided.

1. The high (H) or pessimistic bound assumes significant aspects of the activity develop problems but excludes extremely unlikely or catastrophic occurrences such as a tornado destroying a prototype or a national transportation strike. There should be little chance of exceeding this bound - a workable guideline is no more than one chance in a hundred.

2. The low (L) or optimistic bound is defined similarly to the high estimate, except the most favorable conditions exist.

3. The most likely value or mode (M) is that estimate which has the greatest possibility of occurring.

4. Unless the distribution is symmetric around the mean, the mode is different from the mean. The above terms are illustrated by a hypothetical example in Appendix D.

DATA COLLECTION:

a. Unless the person providing the information has experience in this method of estimating, the personal interview method is preferred. Although it may require more time and money, the analyst has more confidence in the reliability of the inputs. When two or more estimators are available, the Delphi technique may be used. Other data collection techniques are discussed in DARCOM Handbook 11-1.1-79, Reference 2.

b. Some general points the interviewer should consider are:

1. He(s) must understand and be able to describe the program, scope of work, and the network in adequate detail to answer questions by the estimator and to ask the right questions.

2. Allow sufficient time for the interview. Try to pick a setting which minimizes interruptions.

3. The mode is the point most likely, i.e., the point with the most chance of being correct. It may not be the mean or expected value. To assist the interviewer, the modes and means are given in Appendix C with the nine beta distributions. Also given are the areas under the decile and quartile tails of the distributions.

4. The low and high points of the range should be reasonable. This includes the possibility that many events could be favorable or unfavorable but nothing catastrophic would happen.

5. Because of the redundancy in input, the interviewer can quickly check for consistency. However, an atmosphere of cooperation should be promoted to minimize defensive reactions. Also, the interviewer should be careful to not introduce bias into the the estimates received.

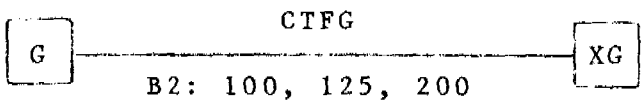
6. The interviewer should remain alert to the estimator's understanding of the process and his knowledge of what is being estimated.

7. As a result of additional data acquired, the program network may need to be revised.

SAMPLE CASE:

a. Situation: A missile system is to be developed using an existing proven system as the base. The only major change will be in the guidance subsystem. The system is composed of five subsystems: airframe (A), propulsion (P), guidance (G), peculiar ground support equipment (PG) and common ground support equipment (CG). No major problems in subsystem interfaces is expected. The first four subsystems will be designed, fabricated or modified (DFM). These four subsystems will then each be component tested and fixed (CTF) as necessary. Meanwhile CG will be acquired (ACG). Next all subsystems will be integrated and fixed (IF) as necessary, followed with a complete system test (ST).

b. The above relationships are shown in Exhibit 1. Event names follow the abbreviations above. Figures below each line indicate the beta type distribution plus the low, mode and high cost estimates. E.g., for the activity CTFG



the guidance subsystem is component tested and fixed as necessary. The cost uncertainty is described by the beta type 2 distribution with a range from 100 to 200 and a mode of 125. Exhibit 2 portrays the same data in tabular form.

c. Analysis:

1. Although the mode is that value which occures most often, it is not the expected value for an activity. (Reference example in Appendix D.) In the Sample Case, the point estimate for the total program, determined by adding the modes, is 1695 whereas the sum of the activity expected values is 1775. The mode method underestimates the costs by 80 units or 5 percent. The mode method typically underestimates a program's cost or time when the program component activities are skewed to the right, i.e., the range of an overrun exceeds that of an underrun. The program total mean value, as determined by simulation, will normally not equal the expected mean value because of the random selection of activity values during simulation; however, the two methods should have totals within $\pm$ 1 percent. Although the point estimate as determined by the expected value or by simulation is superior to the mode method, the decision maker still has no quantification of the uncertainties about the point estimate, i.e., some measure of how much the estimate could be in error.

2. Simulating the Sample Case network by Monte Carlo techniques provides a convolution of the activity probability distributions and provides a measure of the uncertainty around the point estimate. Exhibit 3 displays probabilities and costs for selected events. E.g., using the beta distribution, there is a 75 percent chance that the cost of the program will equal or be less than 1831 units or a 25 percent chance that the cost will exceed 1831 units. Exhibits 4.1 thru 4.3 display the VERT output for the Sample Case for the same events summarized on Exhibit 3. Using the VERT model, output can be generated at any event.

SAMPLE CASE NETWORK

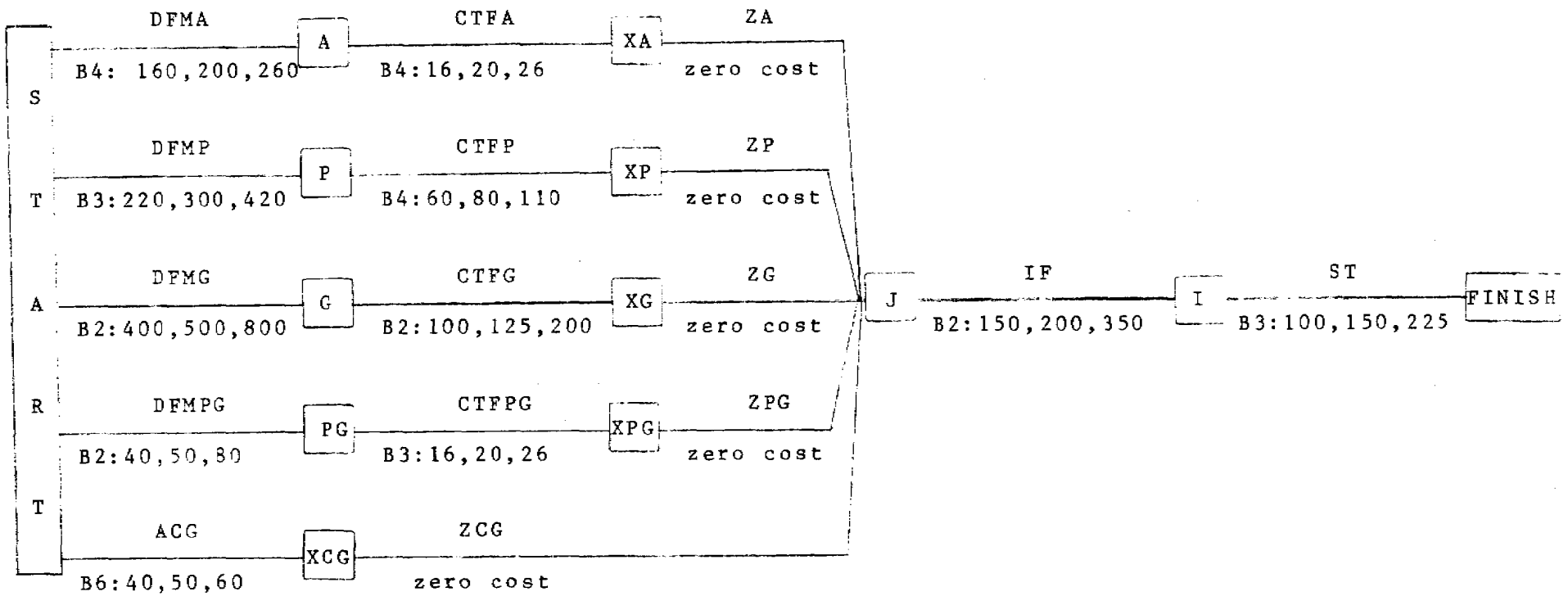


EXHIBIT 1

SAMPLE CASE DATA

Activity	Beta Parameters a	b	L	M	H	E[x]*	Sim**
DFMA	3.	4.	160.	200.	260.	203.	203.
DFMP	2.	2.5	220.	300.	420.	309.	301.
DFMG	2.	4.	400.	500.	800.	533.	536.
DFMPG	2.	4.	40.	50.	80.	53.	53.
ACG	3.	3.	40.	50.	60.	50.	50.
CTFA	3.	4.	16.	20.	26.	20.	20.
CTFP	3.	4.	60.	80.	110.	81.	82.
CTFG	2.	4.	100.	125.	200.	133.	132.
CTFPG	2.	2.5	16.	20.	26.	20.	20.
Subtotal at event "J"			1052.	1345.	1982.	1402.	1397.
IF	2.	4.	150.	200.	350.	217.	215.
ST	2.	2.5	100.	150.	225.	156.	150.
TOTAL PROGRAM			1302.	1695.	2557.	1775.	1762.

* Expected value for the normalized beta distribution is a/(a+b), which is converted by multiplying by the range and adding the lower limit.

** Activity mean values resulting from simulating the activities by 1000 iterations through the network using the VERT model.

EXHIBIT 2

SAMPLE CASE

Probability Points for Convoluted Distributions

Event*	Probability	Beta Dist.	Triangular Dist.**
XG	.10	576	598
	.25	610	638
	mean	668	705
	.75	719	763
	.90	778	837
J	.10	1285	1331
	.25	1330	1385
	mean	1397	1457
	.75	1459	1525
	.90	1524	1595
FINISH	.10	1632	1712
	.25	1689	1764
	mean	1762	1848
	.75	1831	1927
	.90	1899	1994

* Costs are for all activities leading to the event.

XG: Completion of guidance subsystem prior to integration with other subsystems.

J: Cost of all subsystems before system integration.

FINISH: Cost of total program.

** The triangular distribution results assumed the same range and mode for each activity as that used for the beta.

EXHIBIT 3

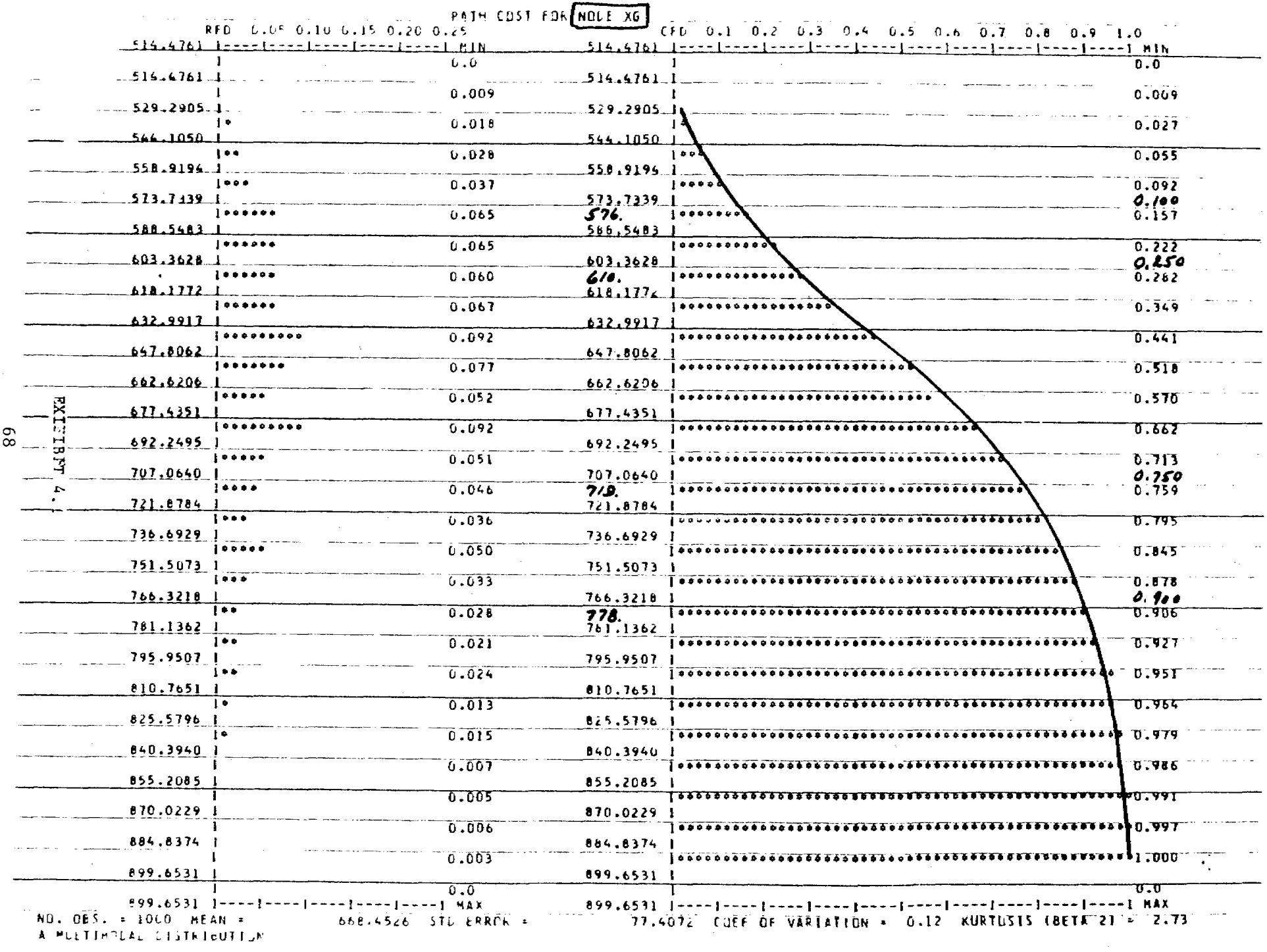
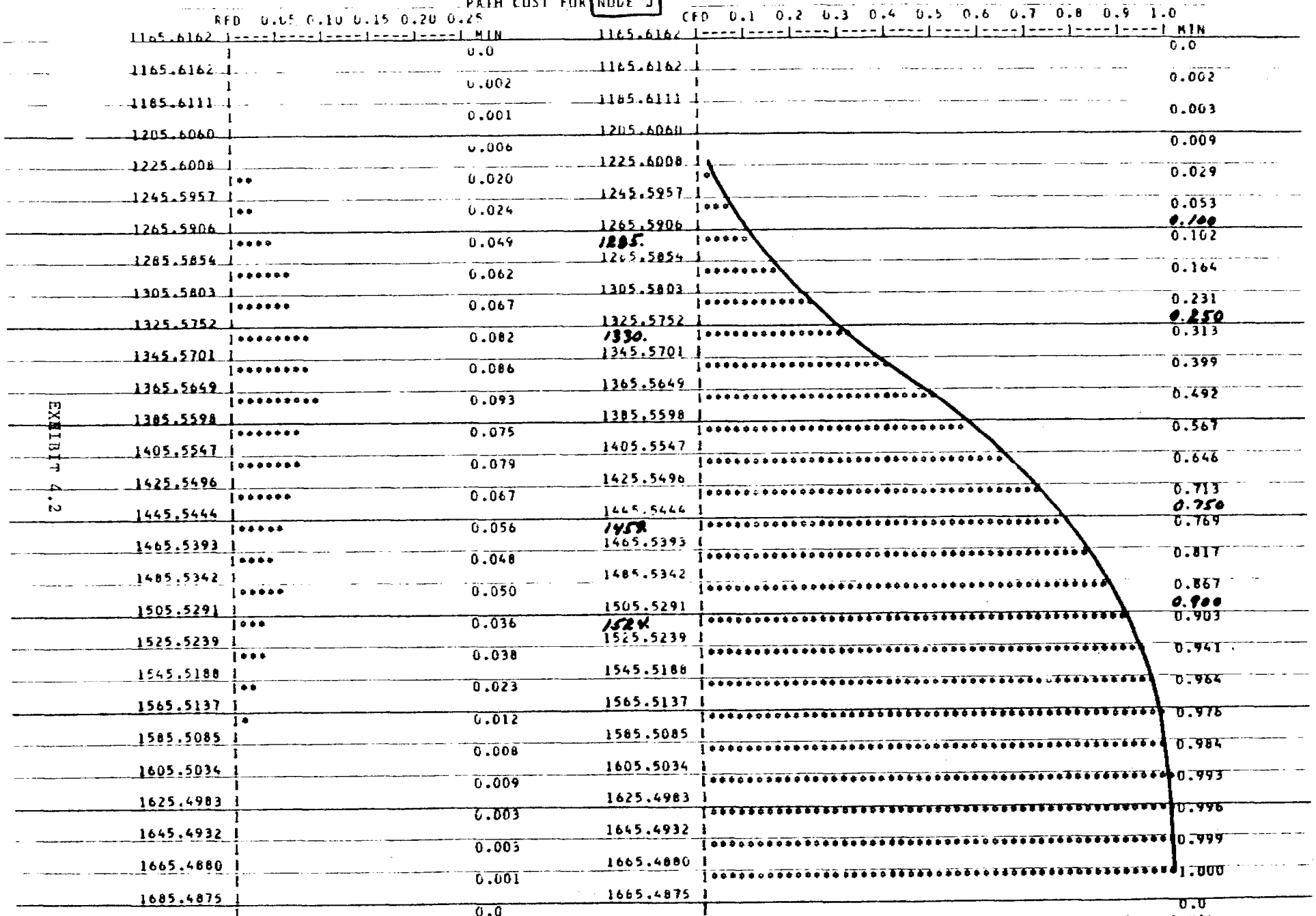

PATH COST FOR NODE XG
RFD 0.05 0.10 0.15 0.20 0.25
CFD 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
514.4761 MIN
514.4761
0.0
514.4761
0.009
529.2905
0.018
544.1050
0.028
558.9196
0.037
573.7339
0.065
588.5483
0.065
603.3628
0.060
618.1772
0.067
632.9917
0.092
647.8062
0.077
662.6206
0.052
677.4351
0.092
692.2495
0.051
707.0640
0.046
721.8784
0.036
736.6929
0.050
751.5073
0.033
766.3218
0.028
781.1362
0.021
795.9507
0.024
810.7651
0.013
825.5796
0.015
840.3940
0.007
855.2085
0.005
870.0229
0.006
884.8374
0.003
899.6531
0.0
899.6531 MAX
0.0
0.009
0.027
0.055
0.092
0.100
0.157
0.222
0.250
0.282
0.349
0.441
0.518
0.570
0.662
0.713
0.750
0.759
0.795
0.845
0.878
0.900
0.906
0.927
0.951
0.964
0.979
0.986
0.991
0.997
1.000
0.0
576.
610.
719.
778.
NO. OBS. = 1000 MEAN = 668.4526 STD ERROR = 77.4072 COEF OF VARIATION = 0.12 KURTOSIS (BETA 2) = 2.73
A MULTIMODAL DISTRIBUTION

EXHIBIT 4.

EXHIBIT 4.2

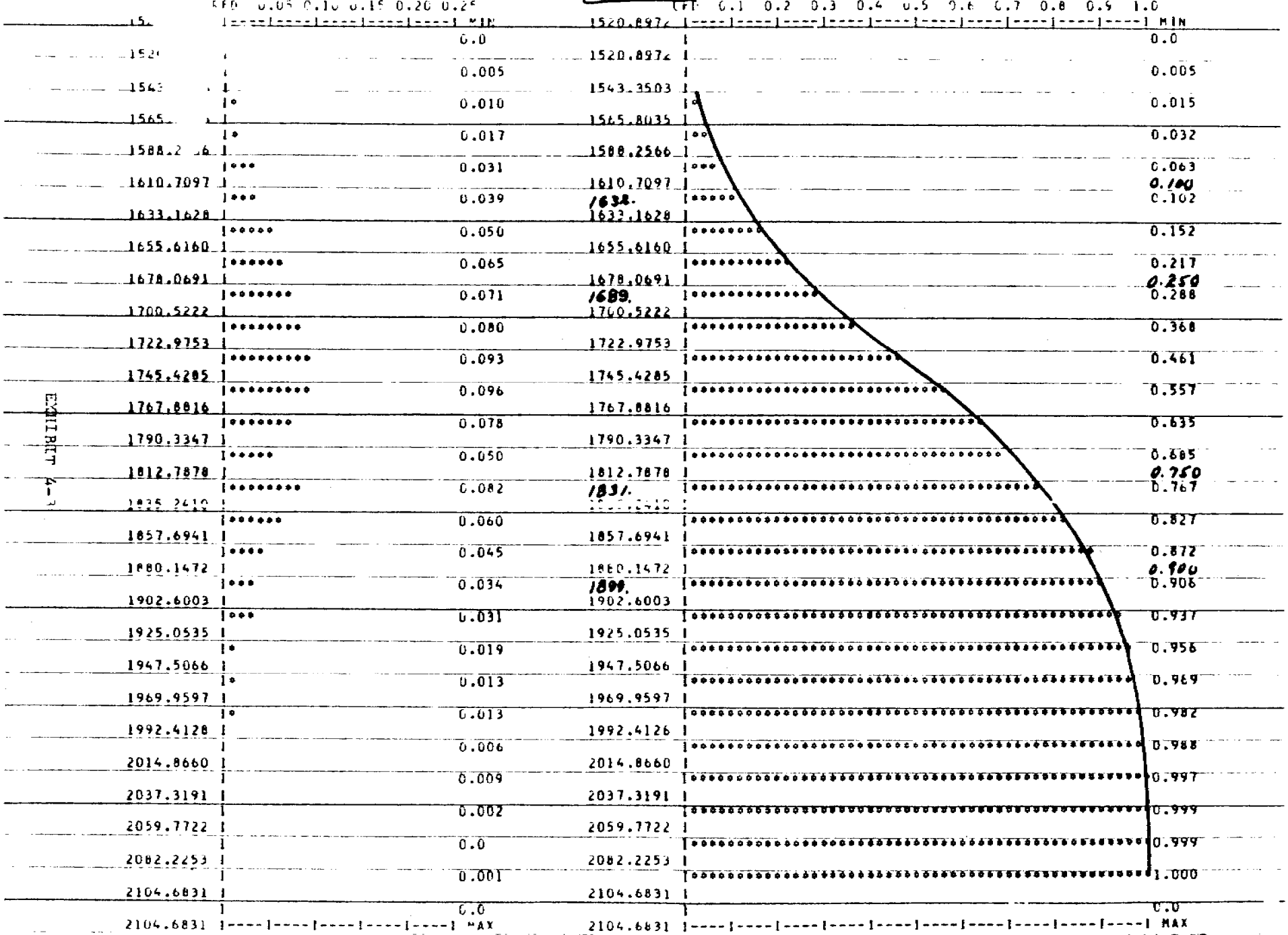
PATH COST FOR NODE J



NO. OBS. = 1000 MEAN = 1396.6648 STD ERROR = 89.4612 COEF OF VARIATION = 0.06 KURTOSIS (BETA 2) = 2.68

MODE = 1371.1433 PEARSONIAN SKEW = 0.26

PATH AND OVERALL COST ARE THE SAME FOR NODE FINISH



NO. OBS. = 1000 MEAN = 1761.4609 STD ERROR = 161.6426 COEF OF VARIATION = 0.06 KURTOSIS (BETA 2) = 2.80
MODE = 1746.6366 PEARSONIAN SKEW = 0.13

EXHIBIT 4-3

CONCLUDING REMARKS:

The desirability and even the necessity for quantifying the uncertainty around the point estimates (time or cost) for new or ongoing programs is becoming a standard procedure within the Department of Army. The professionalism of analysts dictate that they maintain awareness of new and revised techniques. With the increasing availability of efficient and fast computer equipment, former analytical methods can now be performed economically. Although the method proposed in this paper is not new, its use has been limited by unawareness, availability of computer models with random number generators for many probability distributions, and limited computer capability. The later reasons are no longer true and a major purpose of this paper is to promote more widespread awareness of the capabilities available to analysts and decision makers.

Although this method does not reduce the amount of uncertainty in a program, it does attempt to quantify them in a more precise manner. Provided with this additional knowledge, the decision maker should be able to make better decisions and allocations of our available resources.

REFERENCES

1. Faber, Conrad W. "Probabilistic Estimates With Limited Data". Technical Memorandum 81-F-1, U.S. Army Aviation Research Command. November 1980.

2. Army Programs: Decision Risk Analysis (DRA) Handbook. DARCOM Handbook 11-1.1-79, DDC No. 80-367.

3. Fields, David S., et al. "Measuring the Reliability in Cost Analysis (Metrical)". Technical Military Planning Operation, General Electric Company. January 1963.

4. Lundberg, E.D., et al. "A Computerized Technique to Express Uncertainty in Advanced System Cost Estimates". Mitre Corporation. September 1963.

5. Dienemann, Paul F. "Estimating Cost Uncertainty Using Monte Carlo Techniques". Rand Corporation. DDC No AD 813331. April 1966.

6. Durrwachter, Henry W., et al. "Process-- A Probabilistic Cost Estimating System Simulator". HRB-Singer, Inc. DDC No AD 813331. April 1967.

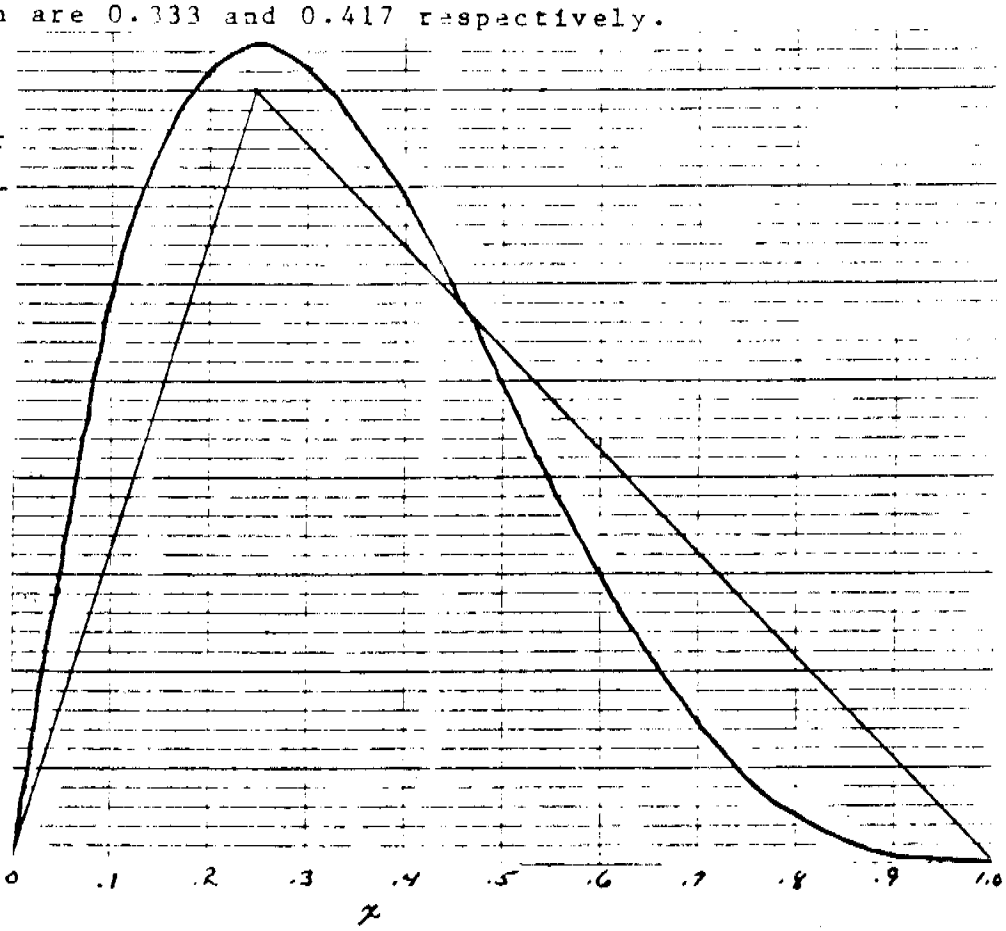
7. Husic, Frank J. "Cost Uncertainty Analysis". Research Analysis Corporation. May 1967.

8. Schlenker, George. "An Analytical Estimation of System Cost Uncertainty". Technical Note 67-3, U.S. Army Weapons Command. September 1967.

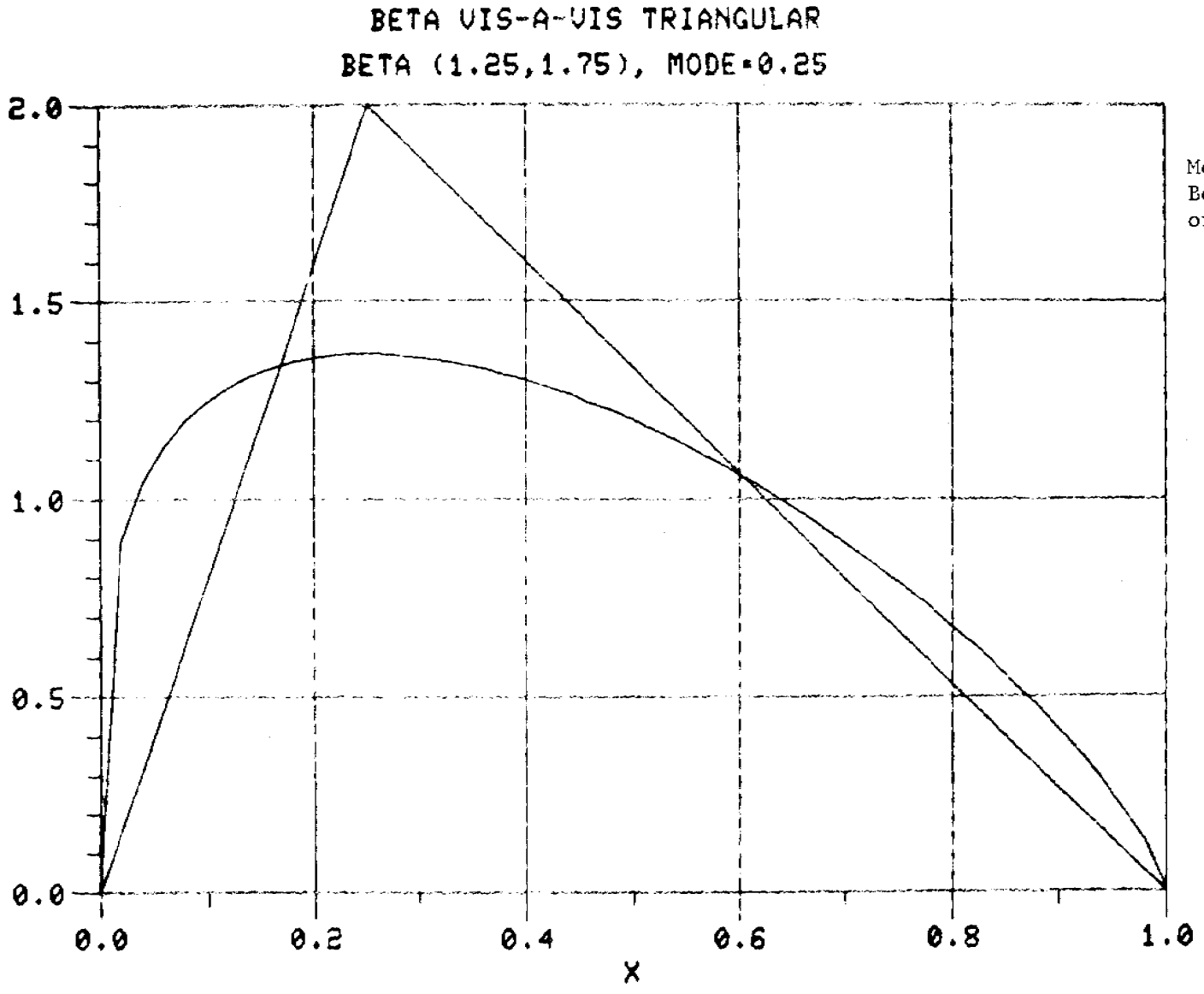
BETA VIS-A-VIS TRIANGULAR DISTRIBUTIONS

SPECIFIC EXAMPLE: Compared are the beta type 2 distribution and a triangular distribution, both with the same range and mode. For this case, as shown on the graph and chart below, the triangular distribution has significantly less area in the low range and more in the high range. Also, the expected value or mean of the beta and triangular distributions shown are 0.333 and 0.417 respectively.

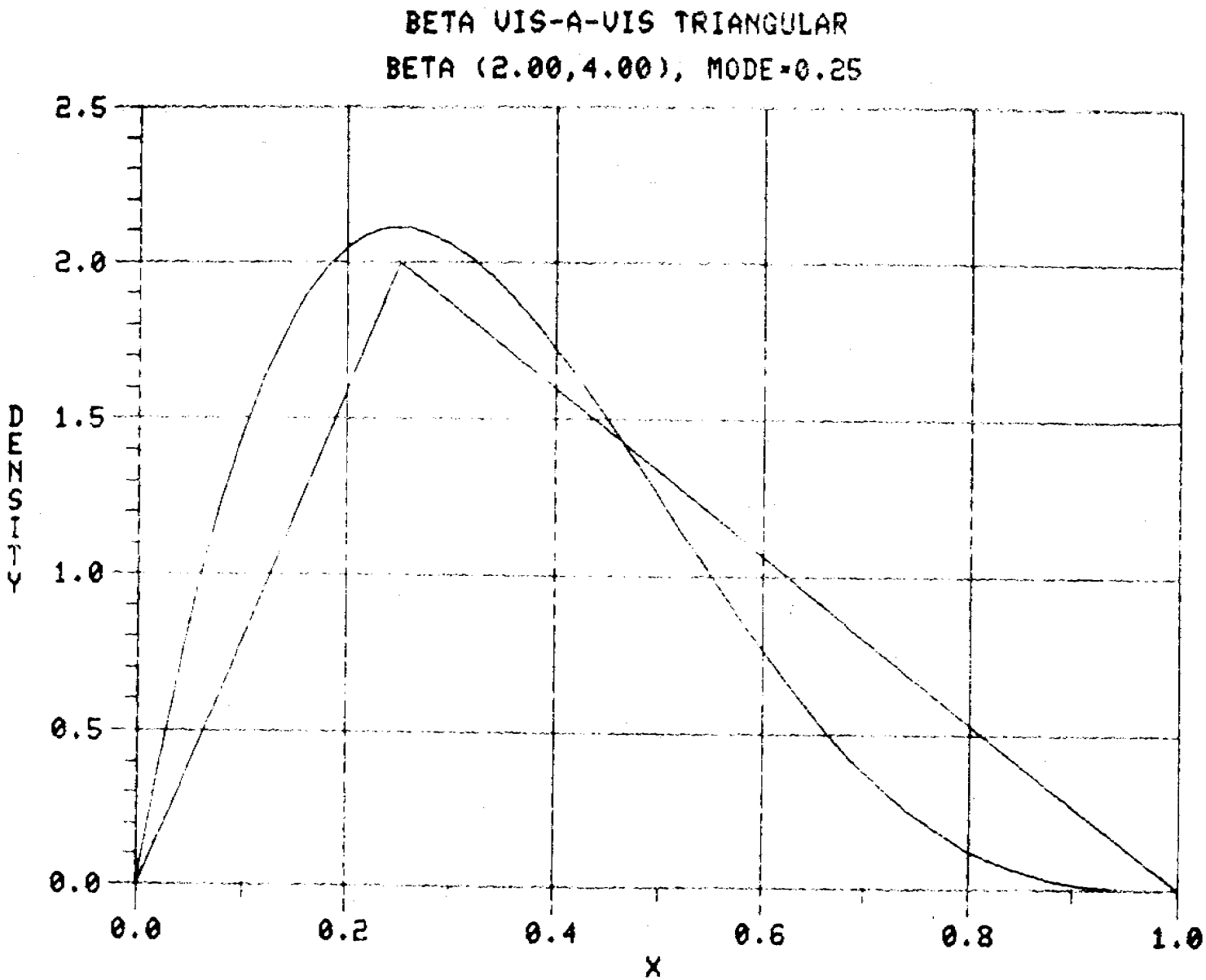
x	Cum. Prob Beta	Cum. Prob Triangular
.00	.000	.000
.10	.081	.040
	.263	.160
.25	.367	.250
.30	.472	.347
.40	.663	.520
.50	.813	.667
.60	.913	.787
.70	.969	.880
.75	.984	.917
.80	.993	.947
.90	.999	.987
1.00	1.000	1.000



GENERAL: The differences described above are for a specific case and will change with different shaped beta distributions. Whereas both distributions include the parameters of range and mode, the beta parameters include a shape parameter which allows greater discretion in describing the uncertainty in an activity. However, under certain conditions, the triangular distribution maybe as accurate as experience will justify.



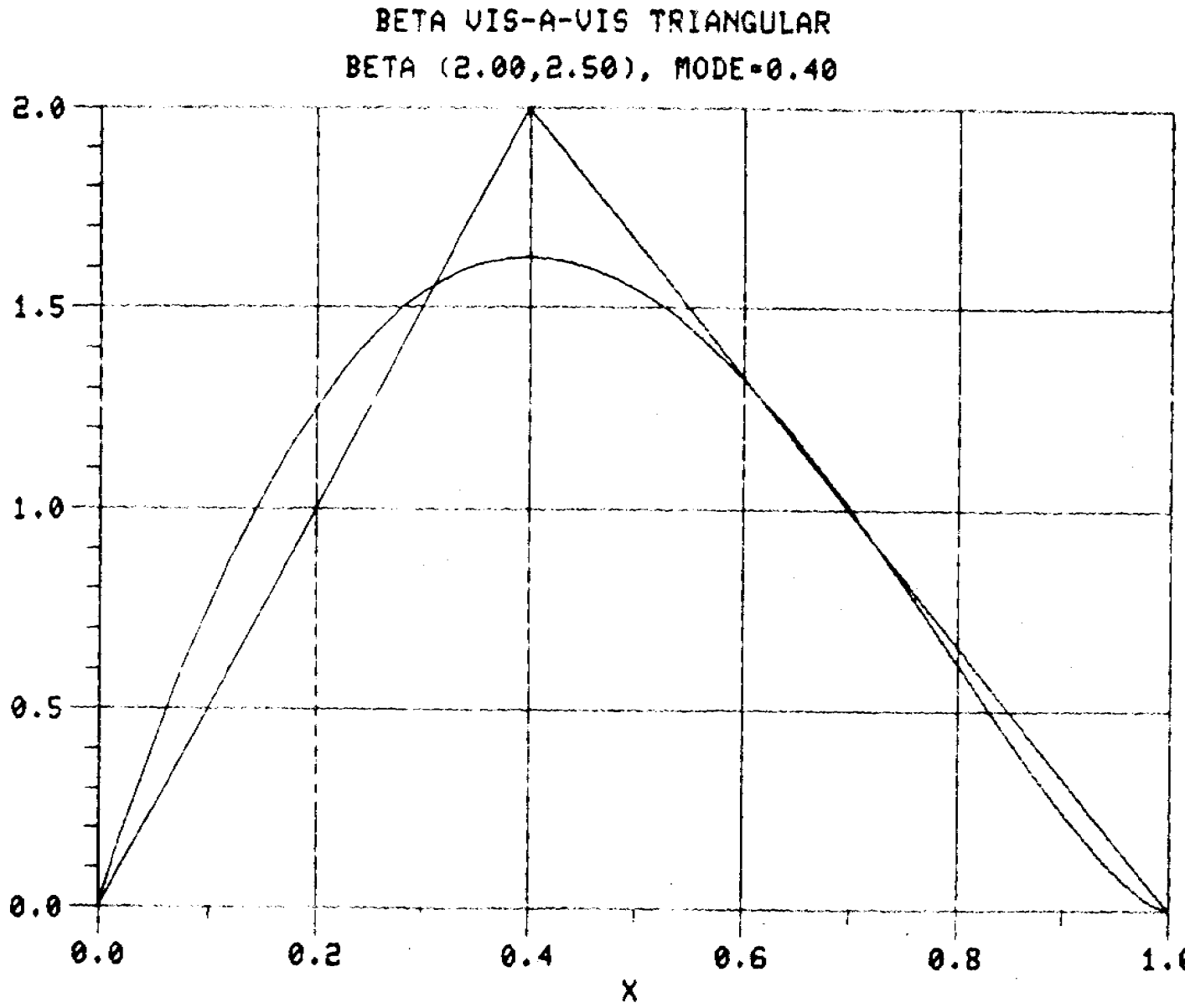
Means same.
Beta has more chance for over or underrun.



Expected value:

Beta	.333
Tri	.417

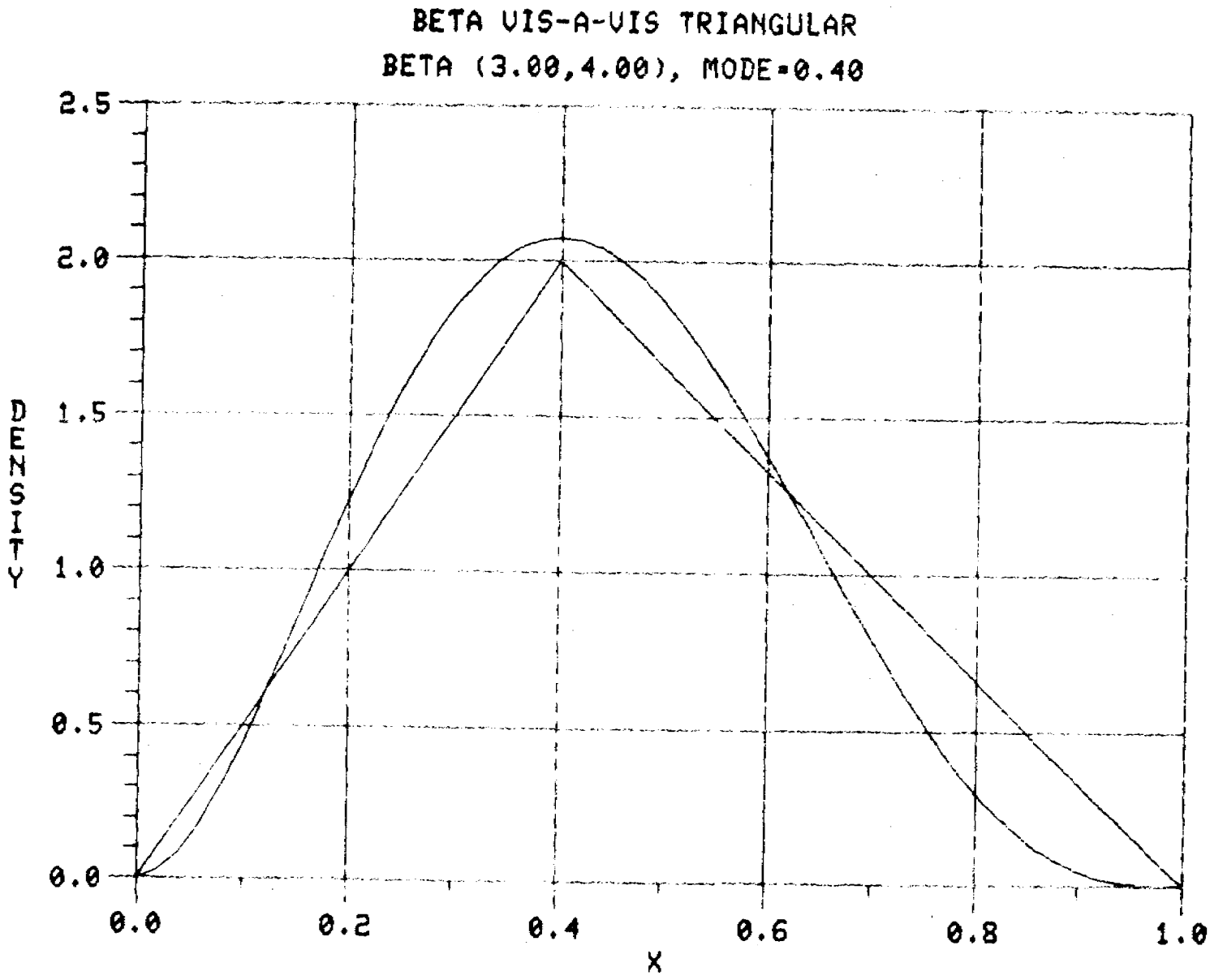
Beta has more chance for underrun, much less for over run.



Expected value:

Beta	.444
Tri	.467

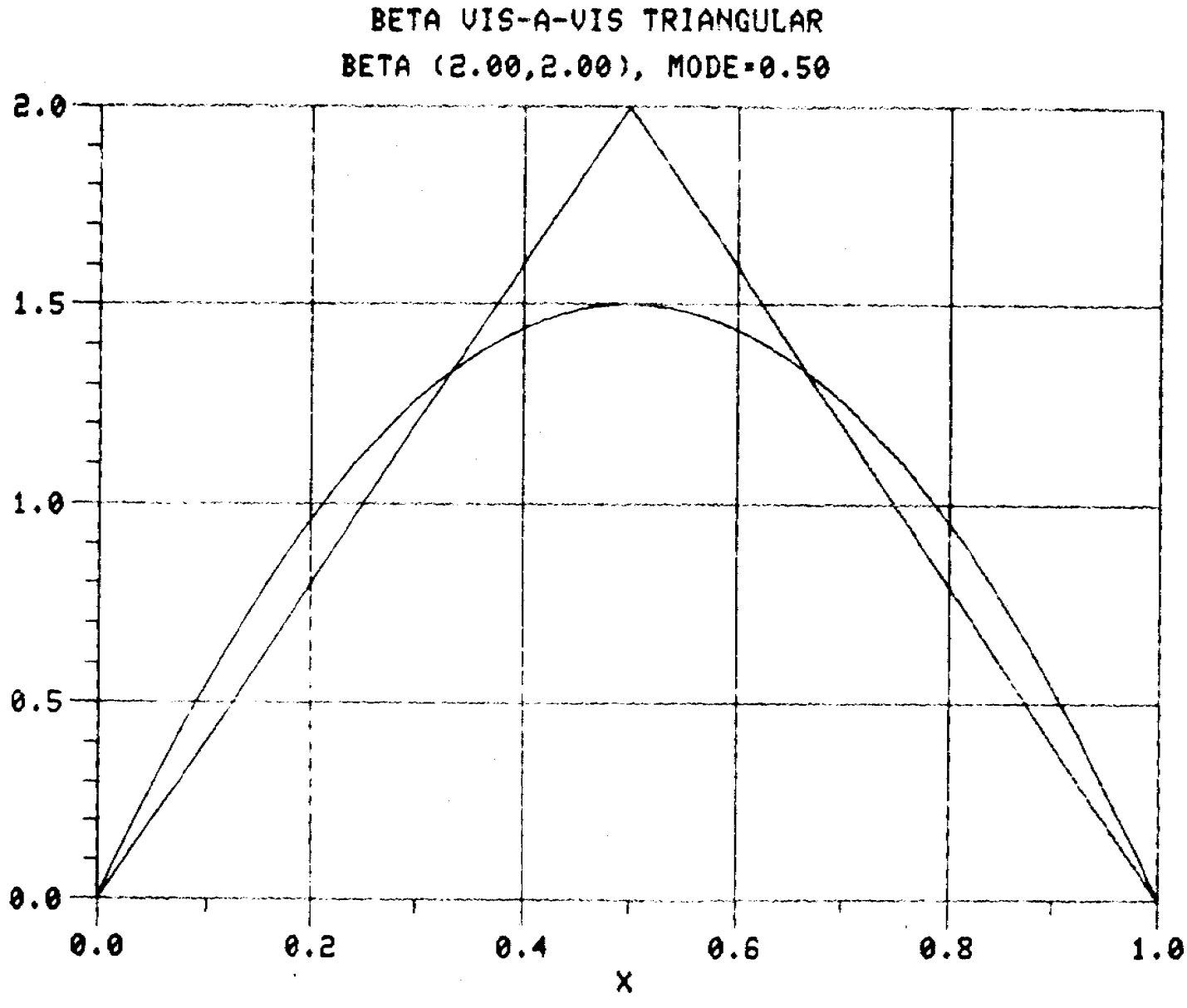
Beta has more chance under-run. Overrun about the same.



Expected value:

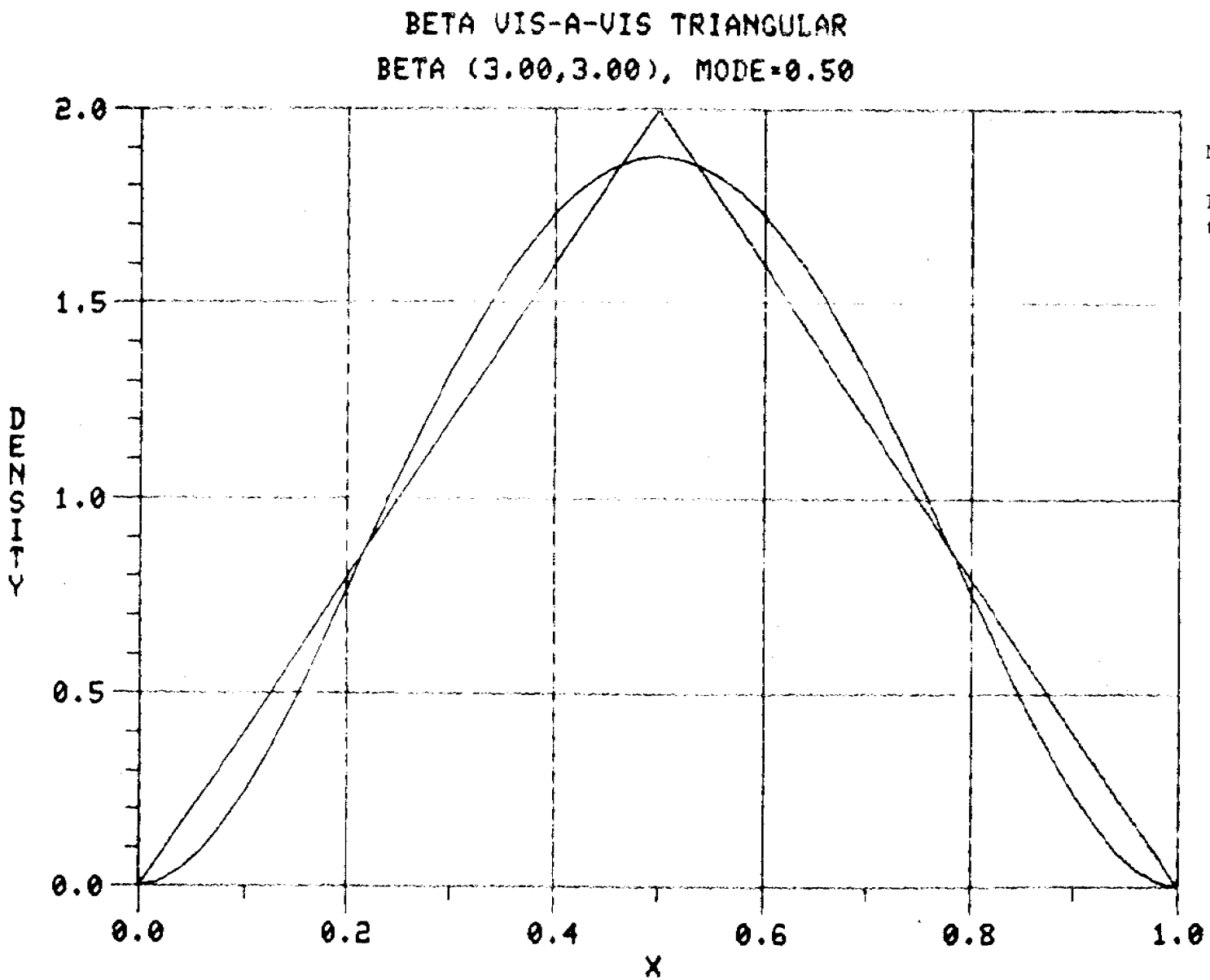
Beta	.429
Tri	.467

Beta has less chance for big overrun.



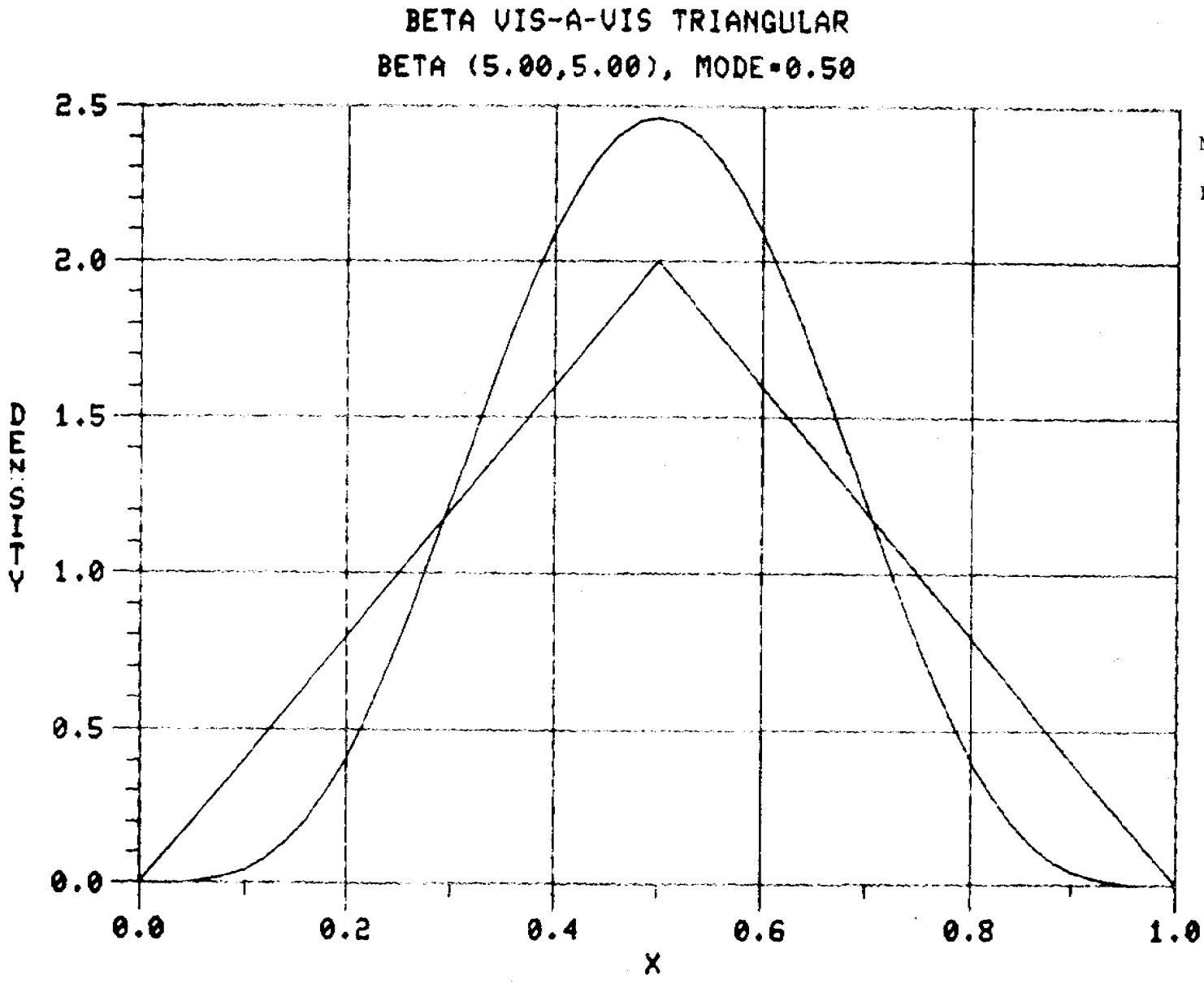
Means same.

Beta distribution has larger variance.



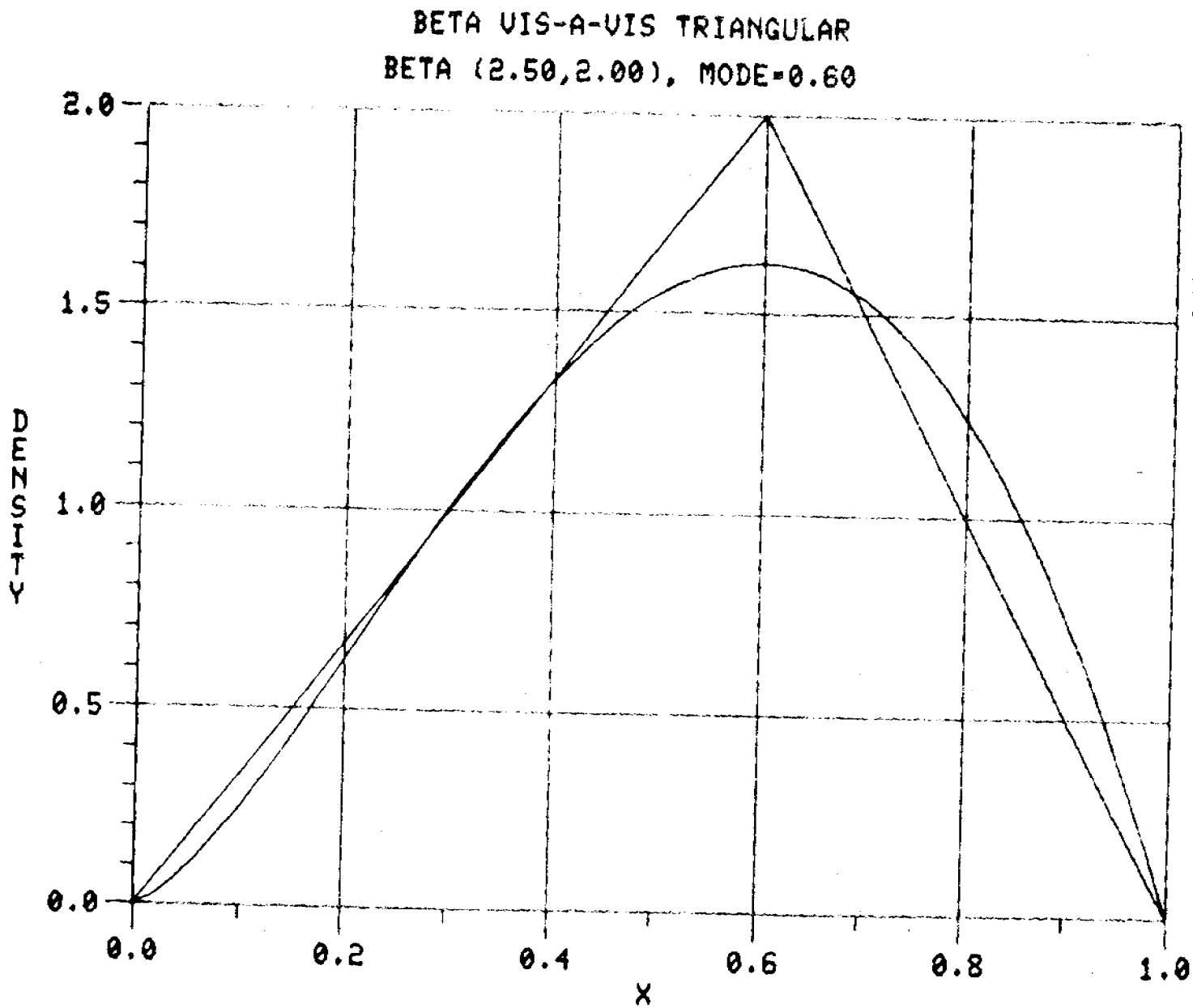
Means same.

Little difference between two distributions.



Means same.

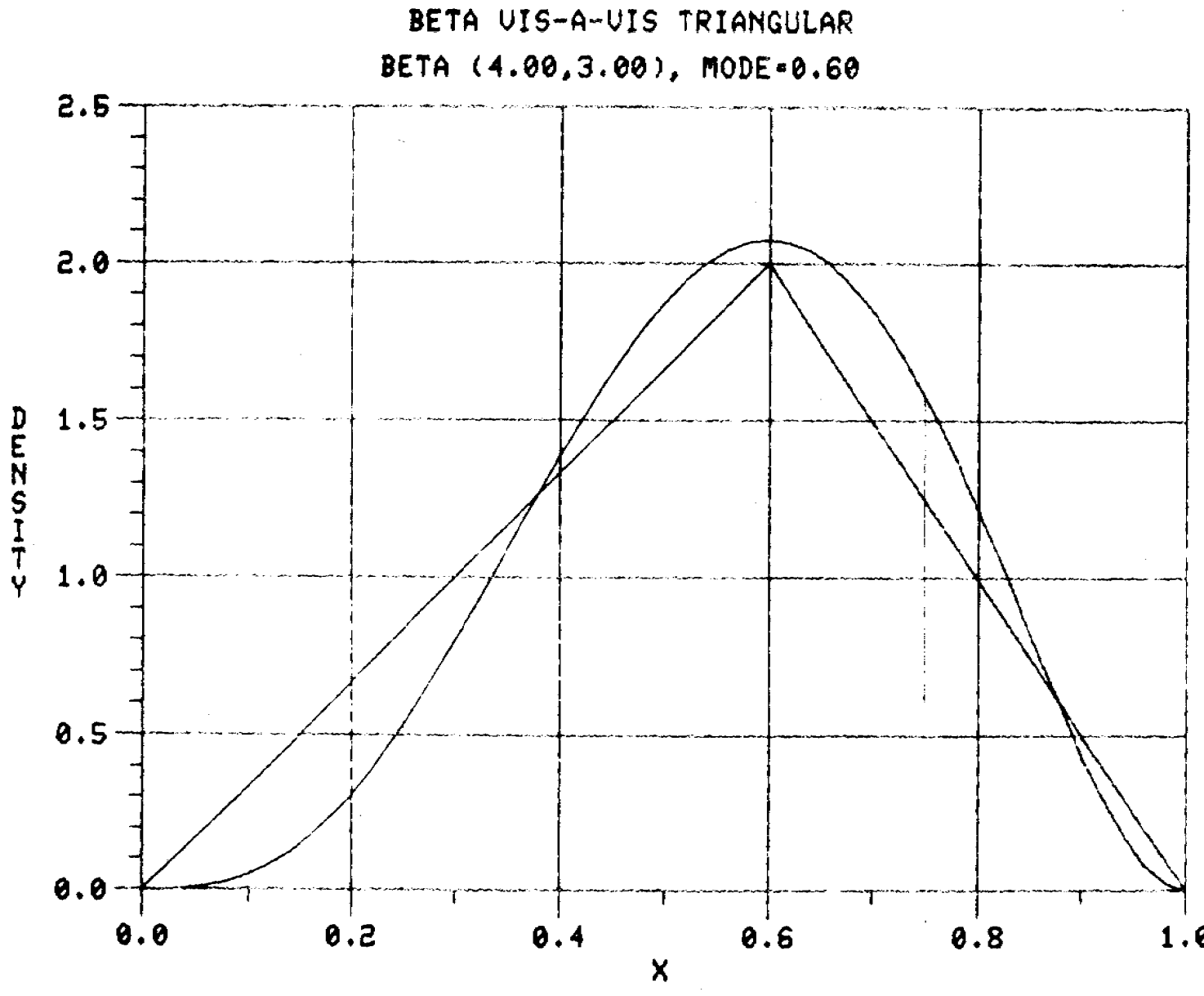
Beta has smaller variance.



Expected value:

Beta	.556
Tri	.533

Beta has more chance of overrun. Underrun about the same.



Expected value:

Beta	.571
Tri	.533

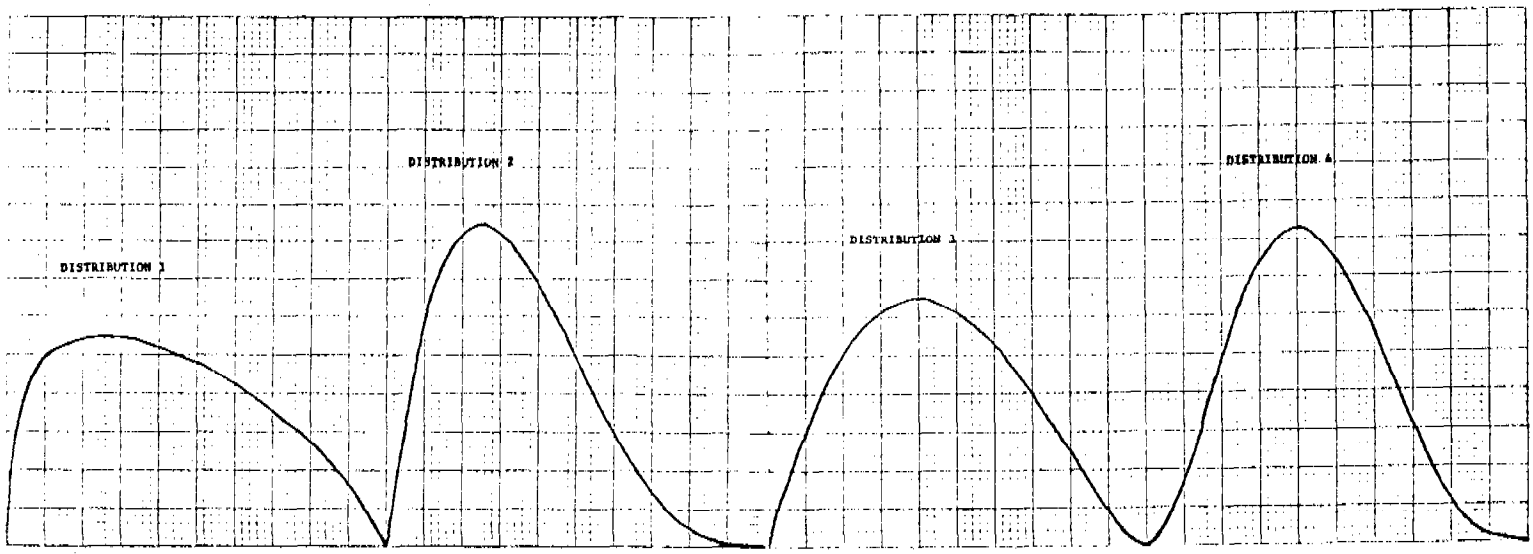
Beta has less chance for underrun.

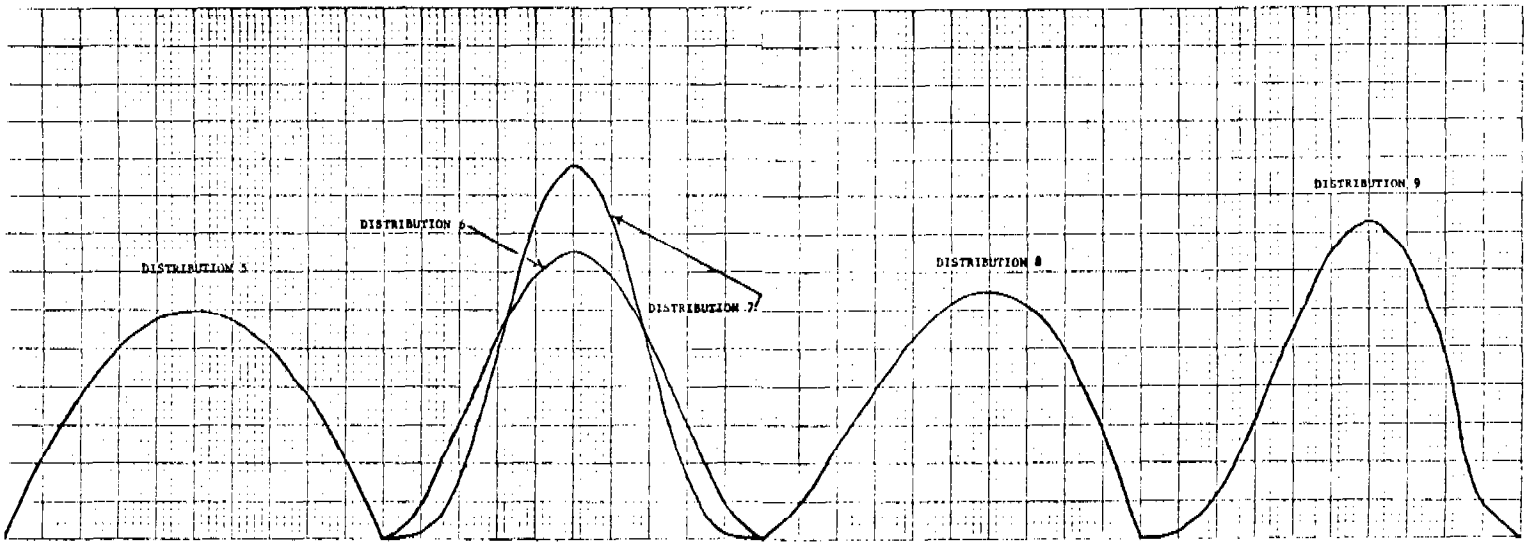
Parameters and Data of Beta Distributions

Dist. Code	Alpha	Beta	Mode	Mean	Pr(.10)*	Pr(.25)*	Pr(.75)*	Pr(.90)*
1	1.25	1.75	.25	.417	.098	.300	.114	.024
2	2	4	.25	.333	.081	.367	.016	.001
3	2	2.5	.4	.444	.039	.208	.090	.010
4	3	4	.4	.429	.016	.169	.038	.001
5	2	2	.5	.500	.028	.156	.156	.028
6	3	3	.5	.500	.009	.104	.104	.009
7	5	5	.5	.500	.001	.049	.049	.001
8	2.5	2	.6	.556	.010	.090	.208	.039
9	4	3	.6	.571	.001	.038	.169	.016

* Denotes the areas in the tails of the distributions from the deciles and quartiles of the range, e.g., Pr(.10) is the probability that the cost or time will be in the lower 10% of the distribution range.

REPRESENTATIVE BETA DISTRIBUTION





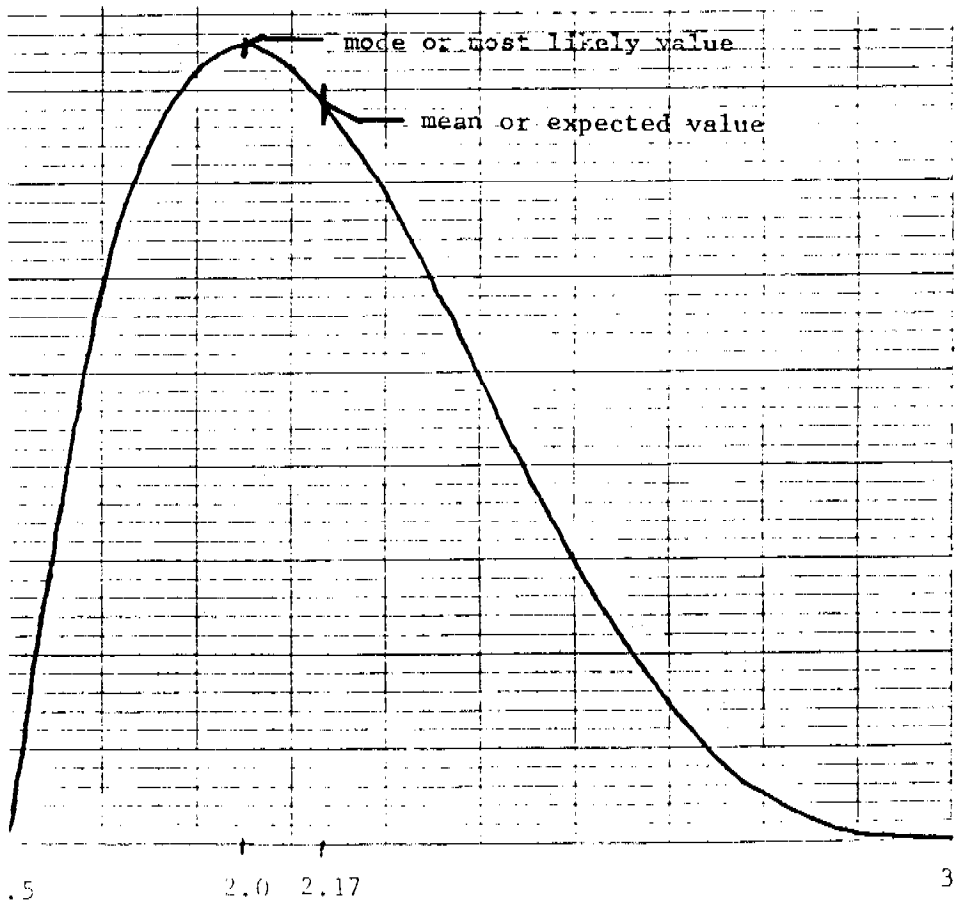
HYPOTHETICAL EXAMPLE

Mode vis-a-vis Mean

PROBLEM: How much should you pay the neighbor boy for mowing your yard?

SITUATION: The price is normally a fixed price arrangement. You consider $2. an hour a fair price and most of the time it takes 2 hours to do the job. Many times there is little rain and the resulting shorter grass can be mowed more quickly. However, if there is some wind, small branches fall on the lawn and the boy must pick them up before he mows. Occasionly, the wind blows down many branches. Extremely rare occurances are ignored, e.g., extended draught or a tornado. You estimate the job will take from 1.5 to 3.5 hours with the most likely time of 2.0 hours and a distribution shaped like that shown below.

CONCLUSION: Since the distribution is a beta distribution with a/b parameters of 2/4, the average expected time is 2.17 hours or $4.34 at $2. per hour.



ON THE DISTRIBUTION OF A LINEAR COMBINATION OF MULTINOMIAL VARIABLES

John C. Conlon
U S Army Materiel Systems Analysis Activity
Aberdeen Proving Ground, Maryland 21005

ABSTRACT. A user-provided subjective comparison of the quality of a service or product as furnished by two different agents is generally done by having a sample of users rate one or other of the agents as better. If the same users are not exposed to the service provided by both agents, a different technique of evaluating the agents is required. In this paper, a technique of having some users rate one agent and different users rate the other on a scale of 0 to K is proposed. A test of hypothesis is given, the distribution of the test statistic is obtained by simulation. The program is listed and some sample output is included.

1. INTRODUCTION. In this report we describe a technique for evaluating a subjective rating of a product or service. A typical situation might be when two performing agents are manufacturing a product or providing a service. Each agent is being rated subjectively as to the quality of the product or service. We develop a parameter useful in comparing the two agents and a statistic for testing the difference. The statistic is asymptotically normal, but we needed to determine its distribution for small sample sizes. We accomplished this using a Monte Carlo simulation.

In Section 2 we describe the problem with the inherent difficulties in solving it analytically. We then detail how we simulate the distribution of the test statistic. In Section 3 we document the functions of the program and its subroutines. In the Appendix, we give a listing of the program, input requirements, and sample output. We also include a graph of the distribution showing how it approaches normality as sample sizes are increased.

2. DESCRIPTION OF THE PROBLEM AND THE SIMULATION. Whenever a subjective rating of the quality of some product/service is performed it usually assumes the following form. Subjects are requested to rate on a scale from zero to "k" the quality of the product or service. The number "k" can be any integer larger than zero with the relationship that the larger the value of k, the finer is the delineation desired. When a number, say n, of subjects rate the product/service, then the numbers of observations in each of the k + 1 categories represent a sample from a (k + 1) - dimensional multinomial distribution.

The situation often arises when two performing agents are providing the product/service and the analyst desires to determine which one is performing better. In some cases it may be impossible to have a subject choose which he thinks to be the better of the two after having tested each one. This may occur when the performing agents are providing the product/service at two different times and the same subjects are not available. In this case it seems reasonable to have n subjects rate performing agent A and m other

subjects rate performing agent B. This provides the analyst with independent samples of size n and m from two multinomial distributions. We may then test for the equivalence of two multinomial distributions. There are some well established procedures for accomplishing this. For our situation this may not be entirely appropriate, since to test the equivalence of two multinomial distributions is actually to test the equivalence of the probabilities of being in each category for the two distributions. As an alternative to the classical approach, consider the parameter, $\sum_{j=0}^{k} a_j p_j$, where p_j is the probability of an observation being in the jth category and a_j is a weighting factor proportional to the desirability of being in category j. We require for convenience that a_0 be zero and that all weights be integer values. A reasonable test to determine the better of the two performing agents is to test the hypothesis, $\sum_{j=0}^{k} a_j(p_j - p_j^*) = 0$.

The statistic chosen for testing the above hypothesis is $\sum_{j=0}^{k} a_j(MN_j - NM_j)$, where N_j and M_j are the number of responses in the jth category, respectively, for each distribution. This statistic was chosen since it always yields an integer value. We denote this statistic Q and study the distribution of Q. The probability that Q does not exceed x, $F_Q(x)$, is

$$\sum_{u \le x} \sum_{\{\underset{\sim}{m},\underset{\sim}{n}:\ \sum_{j=0}^{k} a_j(Mn_j - Nm_j) = u\}} \binom{N}{n_0, n_1, \ldots, n_k} \binom{M}{m_0, m_1, \ldots, m_k} \prod_{j=0}^{k} p_j^{n_j} p_j^{*m_j}.$$

While this is an exact expression for the distribution function F_Q, it is of little value in constructing a test of the null hypothesis, $\sum_{j=0}^{k} a_j(p_j - p_j^*) = 0$. Even when the null hypothesis is true, it is impossible to compute percentage points, the reason being that for any set of parameters, $p_0, p_1, \ldots, p_k$, there are an uncountable number of values for $p_0^*, p_1^*, \ldots, p_k^*$ such that $\sum_{j=0}^{k} a_j(p_j - p_j^*) = 0$.

Since the random vectors N and M are asymptotically multivariate normal, the statistic Q is also asymptotically normal. When the null hypothesis is true, the mean of Q is zero and the variance of Q is

$$\sum_{j=0}^{k} MN\left[a_j^2\left(Mp_j + Np_j^*\right) - \sum_{i=0}^{k} a_i a_j \left(Mp_i p_j + Np_i^* p_j^*\right)\right].$$

In order to construct an approximate test of the hypothesis, $\sum_{j=0}^{k} a_j(p_j-p_j^*) = 0$, using the normal distribution we must use the estimates N_j/N and M_j/M for p_j and p_j^*, respectively, $j = 0, 1, \ldots, k$. Since this is an asymptotic result, a large sample size is required ($N, M \geq 25$ is recommended).

The problem remains as to how to handle the small sample situation. We decided to simulate the distribution of Q for small sample sizes. The simulation involves the following steps (which are explained in more detail later):

- Randomly generate a set of parameters $p_0, p_1, \ldots, p_k$.
- Deterministically construct sets of parameters $p_0^*, p_1^*, \ldots, p_k^*$ such that in each instance $\sum_{j=0}^{k} a_j(p_j-p_j^*) = 0$, or any real number.
- For each set of parameters, one hundred random samples of size N(M) are drawn from the associated multinomial distribution.
- The value of the statistic Q is computed and recorded.

Each of the four steps above is repeated one thousand times with the result that approximately $100{,}000(2^{k-1}-1)$ values of Q are used in the tabulation of the distribution of Q. Let us examine now in detail the steps enumerated above.

The parameters $p_0, p_1, \ldots, p_k$ are generated according to the density $f(p_1,p_2, \ldots,p_k) = k!I(\underline{p}:0\leq \sum_{j=1}^{k} p_j \leq 1)$,[1] with $p_0 = 1 - \sum_{j=1}^{k} p_j$. This distribution was chosen as being uniform on the hyperplane defined by $\{\underline{x}: 0 \leq \sum_{i=1}^{k} x_i \leq 1\}$. To generate the values for $p_0, p_1, \ldots,p_k$, we use the following procedure. The marginal density of p_k is given by

$$f(x) = k(1-x)^{k-1},$$

so that the cumulative distribution function for p_k is given by

$$F(x) = 1-(1-x)^k.$$

The conditional cumulative distribution of p_i given $p_1,p_2, \ldots,p_{i-1}$ is

$$1-[(1-p_1-p_2 - \ldots -p_i)/(1-p_1-p_2 - \ldots -p_{i-1})]^{k-i}.$$

[1] The function I(A) is the characteristic function on the set A.

To generate p_i after having generated $p_1, p_2, \ldots, p_{i-1}$ we generate an observation from the uniform distribution on the interval (0, 1). Let us denote this observation u. The value for p_i is then

$$(1-p_1-p_2- \ldots -p_{i-1})(1-(1-u)^{1/k-i}).$$

After the values for p_1, p_2, ... p_k have been generated, we set p_0 to the quantity $1-\sum_{i=1}^{k} p_i$, so that $\sum_{i=0}^{k} p_i = 1$.

After a set of values for p_0, p_1, ..., p_k has been generated, the next step is to construct values for p_0^*, p_1^*, ...,p_k^* such that $\sum_{i=0}^{k} a_i(p_i-p_i^*) = 0$, or any real number. We chose to select the p_i^*'s deterministically rather than by a random selection process. Our thought was to select sets of values for the p_i^*'s which represent a wide range of possibilities. To this end we chose to select one set for each of the following situations: one non-zero p_i^*, two non-zero p_i^*'s, etc. For the case involving two or more non-zero p_i^*'s we set the constraint that $a_i p_i^* = a_j p_j^*$ for each i,j such that p_i,p_j are non-zero. This is an artificial constraint chosen solely for ease of programming. All possible combinations of each number of non-zero values are attempted although some combinations could result in the sum of the p_i^*'s greater than one. When this occurs, the values are discarded and the next set of p_i^*'s is constructed.

After a single set of values for p_0, p_1, ...,p_k and p_0^*, p_1^*, ..., p_k^* have been established, we then select one hundred samples each of size N and M, respectively, from each multinomial distribution. For each pair of samples the statistic Q is computed and recorded. The results are tabulated and displayed as a discrete distribution function in 140 steps. The length of the steps is a function of the weights and the sample sizes.

3. DESCRIPTION OF THE PROGRAM. The main program receives the input requirements, directs the development of the p_i's and the p_i^*'s, and prepares the results for writing on a permanent file and the output file. We chose to have the data written onto a permanent file so that the distribution can be reaccessed when required and plotted, if desired.

Subroutine WRITE prints the data onto the output file.

Subroutine PROB is responsible for the generation of the p_i's. The intrinsic function, RANF, of course, generates observations from the uniform distribution on the interval (0, 1). Each successive value for p beginning with p_k is computed using the transformation, based on the conditional distribution of p_i

given p_{i+1}, p_{i+2}, ..., p_k, given in Section 2. The values for p_1, p_2, ...,p_k are generated randomly whereas p_0 is set at $1-\sum_{i=1}^{k} p_i$, so that the sum of the p_i's is one.

Subroutine FANCY is responsible for generating the sets of p_i^*'s and for directing the selection of observations from the appropriate multinomial distributions. For each set of values of p_0^*, p_1^*, ..., p_k^* corresponding to values of p_0, p_1, ..., p_k, one hundred samples each of sizes N and M are drawn. The statistic Q is calculated for each sample drawn and the value is sent to subroutine COMP which records the value and stores it. Within the subroutine there is a method for computing all the values for p_0^*, p_1^*, ..., p_k^* in two stages. Firstly, we determine which p_i^*'s will be non-zero. Every non-empty subset of $\{p_1^*, p_2^*, \ldots, p_k^*\}$ is tried. Secondly, we choose the non-zero p_i^*'s so that $\sum_{j=0}^{k} a_j p_j - \sum_{j=0}^{k} a_j p_j^* = 0$, or any real number, and $a_i p_i^* = a_j p_j^*$ for all i,j such that p_i^*, $p_j^* \neq 0$. Once it is determined that $\sum_{i=1}^{k} p_i^* \leq 1$, the parameter p_0^* is set equal to $1-\sum_{i=1}^{k} p_i^*$. At this point the subroutine directs the sampling of observations from the appropriate multinomial distributions. Subroutine MULT is responsible for producing a single observation from a multinomial distribution. Subroutine FANCY calls this routine, receives the observations, and records it. Following this, subroutine FANCY computes the value of the statistic Q and sends it to subroutine COMP which records the value and stores the accumulation of values for final output. As soon as all the possible values for p_0^*, p_1^*, ..., p_k^* within the framework specified have been exhausted, control is transferred back to the main program for generation of a new set of parameters p_0, p_1, ..., p_k.

Subroutine MULT generates a single observation from a multinomial distribution with parameters p_0, p_1, ..., p_k. The interval (0,1) is partitioned into the subintervals $(0,p_0)$, (p_0,p_0+p_1), ..., $(p_0+p_1 + \ldots + p_{k-1}, 1)$. Suppose we label these subintervals 0, 1, ..., k. A random number is generated using the intrinsic function RANF. If the number generated is contained in the interval $(p_0+p_1 + \ldots + p_{i-1}, p_0 + p_1 + \ldots + p_i)$, then an observation for category i is recorded and sent back to subroutine FANCY.

Subroutine COMP receives a realization of the statistic Q from subroutine FANCY. The number of observations of the statistic Q belonging to the interval of integer values of which Q is a member is incremented by one. This subroutine then keeps the accumulation of values for Q which will become the distribution of Q.

4. PROGRAM LISTING

```
      PROGRAM MAIN(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,TAPE8)
      COMMON P(15),N(15),M(15),PX(15),LQ,NBA(140),IVA(140),JM(15),AMN,
     ,XBA(140),IXB,XB,IU1,IU2,.KQ(15)
100   FORMAT(4I10,F10.3)
200   FORMAT(1H,140(I5,5X,F10.5/1X))
300   FORMAT(15I5)
      READ (5,100) IU1,IU2,IRN,IXB,XB
      READ (5,300) (KQ(MS),MS=1,15)
      CALL RANSET(IRN)
      DO 9 MVT=1,140
 9    NBA(MVT)=0
      DO 11 JBN=1,IXB
 11   AA=AA+KQ(JBN)
      AMN=AA/(IXB-1.)
      DO 10 NOP=1,140
      TV=(NOP-71.)*AMN*IU1*IU2/70.-IU1*XB
 10   IVA(NOP)=INT(TV)
      K=1000
      DO 1 I=1,K
      CALL PROB
      A=0.
      DO 2 J=1,IXB
      LXR=KQ(J)
 2    A=A+LXR*P(J)
 1    CALL FANCY(A)
      AL=0.
      AA=0.
      DO 5 LJ=1,140
 5    AL=AL+NBA(LJ)
      XBA(1)=NBA(1)/AL
      DO 4 L=2,140
      MMBA=L-1
      AMX=NBA(L)/AL+XBA(MMBA)
 4    XBA(L)=AMX
      WRITE(8,200) ((IVA(IJU),XBA(IJU)),IJU=1,140)
      CALL WRITE
      STOP
      END

      SUBROUTINE MULT(I,R,J)
      COMMON P(15),N(15),M(15),PX(15),LQ,NBA(140),IVA(140),JM(15),AMN,
     ,XBA(140),IXB,XB,IU1,IU2,KQ(15)
      DIMENSION R(15)
      A=0.
      B=RANF(I)
      DO 1 K=1,IXB
      A=A+R(K)
      IF(B.LE.A) L=K
      IF(B.LE.A) GO TO 2
 1    CONTINUE
 2    J=L
      RETURN
      END
```

```
      SUBROUTINE WRITE
      COMMON P(15),N(15),M(15),PX(15),LQ,NBA(140),IVA(140),JM(15),AMN,
     ,XBA(140),IXB,XB,IU1,IU2,KQ(15)
      WRITE(6,100) ((IVA(I),XBA(I)),I=1,140)
100   FORMAT(1H1,139(21HLESS THAN OR EQUAL TO,
     ,I5,5X,F7,5/1X),21H GREATER THAN,I5.5X,F7,5)
      RETURN
      END

      SUBROUTINE PROB
      COMMON P(15),N(15),M(15),PX(15),LQ,NBA(140),IVA(140),JM(15),AMN,
     ,XBA(140),IXB,XB.IU1,IU2,KQ(15)
      DO 3 J=1,15
3     P(J)=0.
      A=0.
      KF=IXB-1
      DO 1 I=1,KF
      IX=IXB+1-I
      AX=IXB-I
      BX=1./AX
      X=(1-RANF(I))**BX
      Y=1.-X
      P(IX)=(1.-A)*Y
1     A=A+P(IX)
      P(1)=1.-A
      RETURN
      END

      SUBROUTINE CHANGE(J)
      COMMON P(15),N(15),M(15),PX(15),LQ,NBA(140),IVA(140),JM(15),AMN,
     ,XBA(140),IXB,XB,IU1,IU2,KQ(15)
      J=J-1
      L=JM(J)
      L=L-1
      K1=IXB-1
      DO 3 I=J,K1
3     JM(I)=L+J-I
      DO 1 IJ=J,K1
      IX=K1+J-IJ
      IZ=JM(IX)
      IF(IZ.GT.1) L1=IX
      IF(IZ.GT.1) GO TO 4
1     CONTINUE
4     J=L1
      RETURN
      END

      SUBROUTINE COMP(K)
      COMMON P(15),N(15),M(15),PX(15),LQ,NBA(140),IVA(140),JM(15),AMN,
     ,XBA(140),IXB,XB,IU1,IU2,KQ(15)
```

```
      DO 1 I=1,139
      ZJ=(I-71.00)*AMN*IU1*IU2/70.-IU1*XB
      J=INT(ZJ)
      IF(K.LE.J) NBA(I)=NBA(I)+1
      IF(K.LE.J) GO TO 2
 1    CONTINUE
      NBA(140)=NBA(140)+1
 2    RETURN

      SUBROUTINE FANCY(X)
      COMMON P(15),N(15),M(15),PX(15),LQ,NBA(140),IVA(140),JM(15),AMN,
     ,XBA(140),IXB,XB,IU1,IU2,KQ(15)
      L1=2
      K1=IXB-1
      DO 1 I=1,K1
      I1=K1-I+2
 1    JM(I)=I1
      J2=K1
      GO TO 2
 4    L1=JM(J2)
      IF (J2.EQ.1.AND.L1.EQ.1) GO TO 9
 2    GO TO 12
14    IF(L1.LE.1) CALL CHANGE (J2)
      IF(L1.GT.1) JM(J2)=JM(J2)-1
      GO TO 4
12    IN=0
      DO 8 I4=1,K1
      IL1=I4+1
      IF(JM(I4).GT.1) IN=IN+1
 8    PX(IL1)=0.
      B=0.
      DO 5 JK=2,IXB
      IXJ=JK-1
      JXA=JM(IXJ)
      CXQ=KQ(JXA)
      IF(JXA.GT.1) PX(JXA)=(X+XB)/(CXQ*IN)
      IF(JXA.LE.1) PX(JXA)=0.
 5    B=B+PX(JXA)
      IF(B.GT.1) GO TO 14
      PX(1)=1.-B
      DO 7 IV=1,100
      DO 6 LMI=1,IXB
      M(LMI)=0
 6    N(LMI)=0
      IN=JV
      DO 11 J9=1,IU1
      CALL MULT(IN,PX,JV)
      IN=JV
11    M(JV)=M(JV)+1
      IO=IV
      DO 15 K9=1,IU2
      CALL MULT(IO,P,KV)
```

```
      IO=KV
15    N(KV)=N(KV)+1
      NQA=0
      MQA=0
      DO 3 JA=2,IXB
      LM=KQ(JA)
      MQA=MQA+LM*M(JA)
 3    NQA=NQA+LM*N(JA)
      LQ=(NQA*IU1)-(MQA*IU2)
 7    CALL COMP(LQ)
      GO TO 14
 9    RETURN
      END
```

5. INPUT REQUIREMENTS

Input requirements for the program consist of two cards. We use the first card to input the following information:

a. The number of trials for each multinomial distribution,

b. The number of categories,

c. A random number generator initializer, and

d. A value for calculating the power of the test.

We use the second card for assigning weights to the individual categories.

Card 1: Format (4I10, F10.3)

- Number of trials for the multinomial distribution with parameter vector $\underset{\sim}{p}$,

- Number of trials for the multinomial distribution with parameter vector $\underset{\sim}{p}^*$,

- Random number generator initializer. (Any integer number will suffice. Use of the same integer generates a duplicate string of random numbers.),

- The number of categories in both distributions (this number must be an integer between 3 and 15, inclusive), and

- A real number (x) to be used for power calculations. The program generates the distribution of the statistic Q when the parameters have the property

$$\sum_{i=0}^{k} a_i(p_i - p_i^*) = x.$$

Card 2: Format (15I5). Weights are given as integer values with zero being the first weight. Only weights for the number of categories specified on the first card are necessary.

6. SAMPLE DISTRIBUTIONS

N = 10 M = 12 $\sum_{i=0}^{6} a_i(p_i - p_i^*) = -1$

Weights: $a_0 = 0$, $a_1 = 2$, $a_2 = 4$, $a_3 = 6$, $a_4 = 8$, $a_5 = 10$, $a_6 = 12$

u	$P(Q \leq u)$	u	$P(Q \leq u)$	u	$P(Q \leq u)$
-852	.00015	-288	.19522	276	.97670
-840	.00020	-276	.21143	288	.97988
-828	.00022	-264	.23188	300	.98205
-816	.00029	-252	.24966	312	.98462
-804	.00033	-240	.27200	324	.98633
-792	.00042	-228	.29146	336	.98840
-780	.00049	-216	.31562	348	.98968
-768	.00062	-204	.33657	360	.99130
-756	.00073	-192	.36208	372	.99227
-744	.00093	-180	.38386	384	.99353
-732	.00107	-168	.41051	396	.99427
-720	.00134	-156	.43325	408	.99525
-708	.00155	-144	.46065	420	.99581
-696	.00193	-132	.48360	432	.99653
-684	.00223	-120	.51107	444	.99689
-672	.00275	-108	.53372	456	.99745
-660	.00319	-96	.56098	468	.99775
-648	.00392	-84	.58343	480	.99818
-636	.00450	-72	.60985	492	.99840
-624	.00546	-60	.63148	504	.99870
-612	.00633	-48	.65667	516	.99885
-600	.00763	-36	.67711	528	.99911
-588	.00872	-24	.70084	540	.99923
-576	.01042	-12	.72007	552	.99937
-564	.01189	0	.74208	564	.99944
-552	.01411	12	.75968	576	.99956
-540	.01607	24	.77990	588	.99961
-528	.01899	36	.79598	600	.99970
-516	.02157	48	.81420	612	.99974
-504	.02518	60	.82846	624	.99978
-492	.02852	72	.84480	636	.99981
-480	.03309	84	.85736	648	.99986
-468	.03713	96	.87148	660	.99987
-456	.04279	108	.88230	672	.99991
-444	.04792	120	.89466	684	.99992
-432	.05496	132	.90395	696	.99994
-420	.06136	144	.91456	708	.99994
-408	.06991	156	.92241	720	.99995
-396	.07770	168	.93146	732	.99996
-384	.08792	180	.93790	744	.99997
-372	.09709	192	.94534	756	.99997
-360	.10910	204	.95061	768	.99998
-348	.11994	216	.95690	780	.99998
-336	.13402	228	.96130	792	.99999
-324	.14650	240	.96636	804	.99999
-312	.16272	252	.96980	816	1.00000
-300	.17708	264	.97390		

N = 10 M = 5 $\sum_{i=0}^{3} a_i(p_i - p_i^*) = 0$

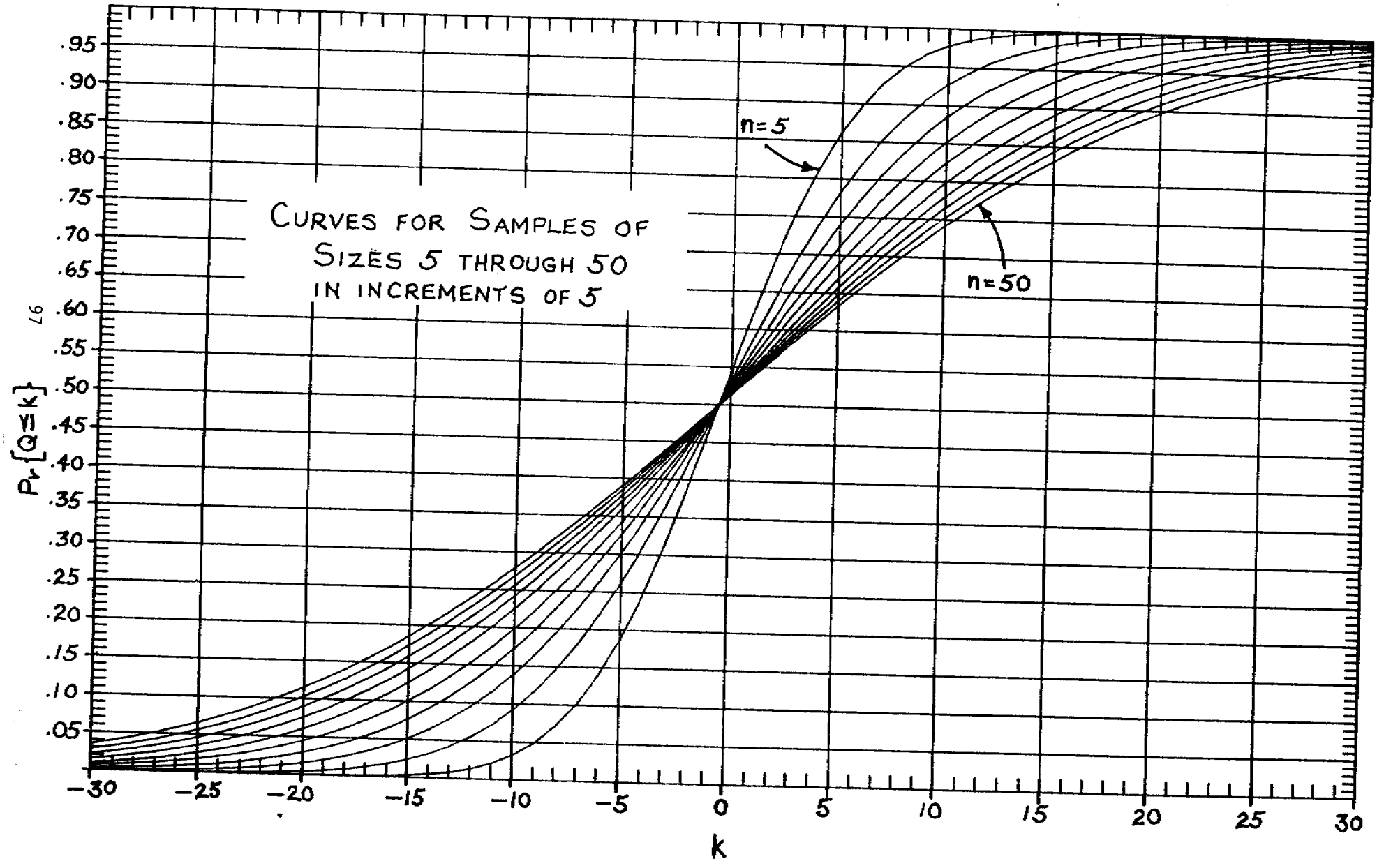
Weights: $a_0 = 0$, $a_1 = 5$, $a_2 = 8$, $a_3 = 9$

u	$P(Q \leq u)$	u	$P(Q \leq u)$	u	$P(Q \leq u)$
-205	.01368	-67	.22514	70	.79115
-202	.01368	-64	.24274	73	.79115
-199	.01572	-61	.24274	76	.80556
-196	.01572	-58	.26095	79	.80556
-193	.01792	-55	.27950	82	.81963
-190	.02032	-52	.27950	85	.83309
-187	.02032	-49	.29957	88	.83309
-184	.02295	-46	.29957	90	.84654
-181	.02295	-44	.31936	93	.84654
-178	.02610	-41	.31936	96	.85771
-176	.02610	-38	.34067	99	.85771
-173	.02910	-35	.36200	102	.86878
-170	.03271	-32	.36200	105	.87898
-167	.03271	-29	.38338	108	.87898
-164	.03626	-26	.38338	111	.88889
-161	.03626	-23	.40542	114	.88889
-158	.04105	-20	.42765	117	.89821
-155	.04616	-17	.42766	120	.90729
-152	.04616	-14	.45028	123	.90729
-149	.05172	-11	.45028	126	.91514
-146	.05172	-8	.47311	129	.91514
-143	.05725	-5	.49519	132	.92299
-140	.06357	-2	.49519	134	.92299
-137	.06357	0	.52051	137	.92996
-134	.07070	2	.52051	140	.93633
-132	.07070	5	.54211	143	.93633
-129	.07835	8	.54211	146	.94221
-126	.07835	11	.56437	149	.94221
-123	.08614	14	.56437	152	.94747
-120	.09494	17	.58516	155	.95222
-117	.09494	20	.60637	158	.95222
-114	.10463	23	.60637	161	.95721
-111	.10463	26	.62808	164	.95721
-108	.11538	29	.62808	167	.96149
-105	.12634	32	.64830	170	.96531
-102	.12634	35	.66750	173	.96531
-99	.13816	38	.66750	176	.96896
-96	.13816	41	.68812	178	.96896
-93	.15006	44	.68812	181	.97222
-90	.16397	46	.70784	184	.97222
-88	.16397	49	.70784	187	.97510
-85	.17757	52	.72618	190	.97783
-82	.17757	55	.74246	193	.97783
-79	.19305	58	.74246	196	.98017
-76	.19305	61	.75841	199	.98017
-73	.20857	64	.75841	202	1.00000
-70	.22514	67	.77492		

$N = M$

$$Q = \sum_{i=0}^{5} i(N_i - M_i)$$

$$\sum_{i=0}^{5} i(p_i - p_i^*) = 0$$



SIMULATING THE ARMY MILITARY CONSTRUCTION PROCESS

James H. Johnson

MC Modeling Study
Facility Systems Division
Construction Engineering Research Laboratory
Champaign, Illinois 61820

Manoochehr Ghiassi*

Department of Mechanical and Industrial Engineering
University of Illinois
Urbana, Illinois 61801

ABSTRACT. A simulation model is being developed at the Construction Engineering Research Laboratory for the support of the continuing assessments of the Army Military Construction (MCA) process by the Corps of Engineers. This MCA process simulation model provides a functional representation and statistical measurements of system response at both the local and global levels, and is intended to be an analytical tool for researchers investigating this complex process and for management in understanding and controlling it. All MCA projects proposed for an Army Budget Year Program are processed by the model through all appropriate performance and approval levels to their final determination. First goals of the model development have related to time and cost level determinations for performance functions, specifically the planning, programming, design, and construction of military facilities. These assessments are ultimately related to the timeliness and economics of the construction product.

1. INTRODUCTION. The Army Military Construction (MCA) process is a term used to identify the web of interacting procedures, management systems, and requirements/regulations which bring a needed construction project from an Army installation's proposal through all intervening development stages to the turnover of the finished facility.[1]

The Corps of Engineers (CE) is the largest construction agency in the world, processing $10 billion in construction for Fiscal Year 1982 (FY82) alone. The military construction portion of this budget is nearly $1 billion and holds consistently at about 10% of the CE budget. The MCA process utilizes this substantial budget to provide needed military facilities; any improvement in the MCA process will be economically significant if it impacts the quality, timeliness, or price of the facilities delivered.

*Visitor at the University of Illinois.

Analyzing the MCA process can be complicated by the multidimensional nature of the process and can be made even more difficult when multipurpose objectives are included in the assessment task. The existing MCA process reflects the past evolutionary needs of Corps construction management and the hierarchical organization which implements this management. This results in back-and-forth transfers of responsibility over many layers of authority. Furthermore, the complexity of the processing systems and the number of projects handled make it difficult to develop an analytical grasp of the entire MCA operations network and the problems such a network can sometimes encounter. A convenient method of visualizing such complex processes is through network representations.[6]

2. MODELING THE MCA PROCESS. Although network representations of the MCA process should be developed at a level of detail appropriate to the requirements of the intended analysis, "real-world" constraints must be considered. There will usually be analytical time limitations, network size limitations, unavailability of functional data (data requests can disrupt field operations), and the hierarchical orientation of the analyst. Even when some of these constraints can be accommodated, the inherent complexity of the MCA process cannot be avoided. It is this complexity in both the representing and the represented that discourages a formalized and consistent approach in assessing MCA processing problems.

A proper modeling of the MCA process depends upon a recognition of the many aspects or "dimensions" associated with this development (Figure 1). Requirements for replicating the hierarchical process with the appropriate logic at an effective level of detail are fundamental to the analysis. In addition, the approach must be reasonably convenient and sufficiently responsive to permit the investigator a development of timely solutions.

The logic of the functional relationships of the MCA process can be "sculptured" into the network format so as to be visibly representative of the process environment and demonstrably responsive in meeting analytical requirements. It was determined that the modeling of the total MCA process will be most conveniently achieved through a "procedural network."[5] A "procedural network" is defined as a network in which the nodes indicate the decision-points/functions and the arcs represent the flow of responsibilities and/or information associated with project development. Hence, this network identifies events of interest (decision-points/functions) by nodes or boxes, with the task processing at any particular decision node dependent upon the stochastic history of the project and the network logic up to that point. The precedence relationship in the network reflects the organizational hierarchy and the typical or required sequences for processing MCA projects. Precedence requirements define the network configuration and are controlled by such factors as management policy, task characteristics, and procedural regulations. As indicated in Figure 1, there must be a hierarchical representation, a level of detail and an extent of coverage developed for the network which respectively reflect the precedence relationships, sensitivity requirements and a range of interest applicable to the particular MCA problem under study.

These precepts have permitted the development of an MCA process network which represents all significant procedural paths that can occur in the process. Figure 2 identifies the scope of this network up to construction contracting. In interpreting this diagram, it should be understood that the flow of one project or all projects can be considered, and project entities or segments of projects may be processed according to local procedural needs. This is a project controlled process. The characteristics of the construction projects processed by the network must control the functional response, the decision points, and the flow paths of projects through the network. These relationships, ideally determined from statistical descriptions of actual MCA process experience are at the least related to facility type (project complexity) and size (project price). The complexity of these factors and the number of projects involved in the process are not easily handled by any means other than by digital computer simulation techniques.

3. SIMULATION OF THE MCA PROCESS. An effective simulation of the MCA process depends on the development of a representative network and an efficient and compatible simulator system. The simulation modeling approach can provide statistical descriptions of the MCA process derived from either direct functional or surrogate representations which are correlatable to the "real-world" process (Figure 3). Thus, the model is an analytical tool providing rapid answers in a standardized form to the analyst, but still subject to his reasoned interpretation. Basic to these assessments are the procedural efficiencies and time/cost/resource-impact considerations associated with the functional performance of the process. Performance-level assessments of this process can only be achieved by a simulation model which is capable of processing sophisticated procedural networks.

A simulation system which could meet all of the requirements associated with analyzing the MCA process was not a "shelf" item. Of the several simulation languages which are available, the Generalized Network Simulator (GNS), as initially developed at the University of Illinois, was selected. This computer simulation program is written in FORTRAN IV and was designed to simulate generalized stochastic networks primarily as applied to manufacturing processes. The GNS approach was adapted and modified at the Construction Engineering Research Laboratory (CERL) to support procedural networks and MCA process assessment requirements. The computerized program permitting this simulation is called the Corps of Engineers Generalized Network Simulator (CEGNS), and represents a parallel development to the Generalized Manufacturing Simulator (GEMS) System.[8] Both CEGNS for simulation procedural networks and GEMS for simulating product-assembly type networks are derived from the GNS concept, as proposed by Hogg, Dessouky, and Tonegawa in 1977.[4]

Simulation Model Features. The MCA process simulation model is seen as being a computerized representation of a procedural network which depicts the functional steps involved in bringing an Army construction project from the proposal state to a finished and accepted facility. The model must simulate the total project flow, including project acceptance/rejection; it must approximate the impact of project characteristics and volume (workload) levels; and it must allow for interactions between performance functions as well as other dollar and time significant effects. This

modeling system will enable the analyst to create a simulation network under nominal rules and restrictions and to apply this algorithm to his problem areas without the inconvenience of diversionary tasks. He should not, for instance, have to face a monumental computer programming job when there is a requirement to simulate an extensively revised network. A good computer simulation program will effect an efficient, dynamic handling of network logic, permitting simulation network definitions as well as problem requirements to be input in relatively simple statements.

The Simulator. The CEGNS simulation system was developed to effectively accomplish general MCA simulation goals without restricting model growth and expanding requirements. The CEGNS and MCA model developments have been iteratively improved by repeated testing and modifications (Figure 4). The resulting model is a functioning version of the MCA model which has evolved with steadily improving sensitivity. The product of this development was a Study Model utilized as a development vehicle for establishing the criteria applicable to future operational versions of the model. A fundamental feature of the CEGNS simulation modeling system is its capability to replicate a procedural network. Figure 5 shows the distributions of actual and simulated AE design times from Sacramento District data for FY80 projects. The distribution predicted by the MCA simulation model in this figure closely approximates the distribution of the "real-world" process.

4. STUDY MODEL. The Study Model has been used to test the assumptions and verify the conclusions of the simulation study. Although the model is incrementally improved as the study advances, its basic features and application goals developed for evaluating the MCA process have not changed.

MCA Assessment Features. Special features of the Study Model include:

(a) Time/cost/resource expenditure measurements for each process function.

(b) Full MCA process network representation.

(c) Global/local frame-of-reference capability.

(d) Special performance-indicator and display outputs.

Other special operating features include a selectability in input and output formats and a choice of data processing aids especially developed for MCA analysis requirements. Printouts of MCA simulation runs will include many useful and informative features for the analyst, such as echo-checks of the input, a listing of projects supplied or generated, in-process traces, and output summaries in statistical format. Statistical summaries can be provided as follows:

(i) The number of projects held-up, lost (if any), and processed at all queue boxes.

(ii) A listing of all projects passing through any selected activity box.

(iii) A histogram of the time-intervals of all projects flowing between specified activity boxes.

(iv) A bar chart of the number of projects waiting to be processed at any specified queue box over any specified time period.

(v) An accumulating event-calendar of projects (number of projects processed vs. absolute time) for any phase(s) of the MCA process selected for analysis.

Example Applications. Trial assessments of the MCA process by the study model are now discussed to illustrate the power of the simulation approach. The first problem to be examined is definitive in nature and initially requires the type of simulation used for evaluating new procedures with limited impact. The second problem proposes a change with a far-reaching impact on all high cost projects; the relaxation of three process constraints are involved; and more sophisticated modeling features are required.

(a) A Simplified Assessment Problem. It is desireable to compare the potentially shorter construction procedures associated with prefabrication and industrialized-building approaches to the traditional procedures which utilize custom designs and on-site preparation of structural assemblies. The procedures will be compared on the basis of overall processing times.

This may be considered first as a problem in establishing basic differences in processing efficiencies (to be followed by global reviews based on the verified precepts). A needed simplification is the processing only of projects relevant to the study. This eliminates the imposition of unwanted variables such as the scheduling assumptions, sequential-processing effects and district workload considerations implied by a total project environment. A very simple initial approach could consider two facility classes (low to high complexity) and two size (price) categories, small and large. This results in an easily controlled study of four projects processed by a minimum of two computer runs, one for each of the construction approaches. The combined results shown in Figure 6 identify the time advantages of the industrialized building approach as an illustration of the analytical use of CEGNS outputs.

(b) Multiple Constraint Problem. A hypothetical proposal is assumed which requires a procedural change in the approval of high cost Architect and Engineering (AE) contracts over $500,000. In the present procedure, all AE contracts greater than $200K are reviewed by the Division Engineer; all AE contracts greater than $500K are reviewed by the Office of the Chief of Engineers (OCE); and all AE contracts of $1 million are also reviewed by the Department of Defense (DOD). In the hypothesized procedure, only the Division Engineer reviews high cost AE contracts, with OCE notified and coordinated with whenever contracts of $1 million or more are processed. A cost benefit determination for the proposed change is requested.

A "mix" of 500 proposed construction projects is statistically generated, each with ten descriptors defining the unique characteristics of each project. Figure 7 gives a sampling of these synthesized projects.

In simulating these projects, it can be seen that in the winnowing assessments by the Major Commands (MACOMs) and OCE, the number of projects has decreased to 380. These "official" FY projects are distributed to five divisions representing the ten districts which process MCA projects. A district was "selected," and 100 of the 380 projects were processed by this district. Three comparative runs were made for each assessment:

(i) Normal configuration run.

(ii) Hypothetical-change configuration run.

(iii) "No restriction" (fast-track) configuration run.

In these three simulation runs, all configuration and branching factors as well as controllable stochastic events are kept the same -- except for the simulated conditional approvals for the projects as required by the problem.

As part of the output for each of the runs, planning durations, programming durations and design times were statistically plotted for all processed projects. (See Figure 8 for a typical output from a normal run as an example of these products.) In addition, a programming output calendar for the 27 projects with an AE fee above $200,000 which completed final design was generated for each run (Figure 9). This project output display illustrates the time between the normal (basic), hypothetical (no upper AE approval), and limiting (no restrictions) runs. As shown in Figure 10, there are more significant savings in projects with longer design processing times. Table 1 summarizes the 27 projects and their impact on the MCA process. For the assumptions made, it was determined that the cost of the net delay per high cost project was approximately $60,600; where "delay" implies the time difference between the normal and hypothetical runs. Under the assumptions made for this hypothetical proposal, $60,600 amounts to approximately a 1% saving, which is probably not sufficiently significant to justify the change -- even if feasible under other considerations. Such results can assist a decision maker in evaluating proposed changes to the MCA process.

Evaluation of Study Model Capabilities. It is evident from Study Model applications and experience that the simulation approach provides a powerful tool for analyzing the MCA process. If sources of real-world response data can be developed, the present model can be advanced further through verification and updating procedures.

Table I

Significance of Hypothetical Change

Project No.	Δ Hrs.	Contract Price $M	AE Fee $K	Adjusted Delay: Days	27 Project Fac.-Use Value $/Day	All 27 Projects Delay $	Hi Cost(AE) Projects Delay $
209	20	5	229	3	715	2145	
86	10	7	275	2	1000	2000	
146	0	6	238	0	860	0	
71	-10	7	264	-2	1000	-2000	
255	-20	7	290	-3	1000	-3000	
103	-10	8	301	-2	1143	-2286	
81	-30	11	433	-4	1590	-6360	
74	-40	7	291	-5	1000	-5000	
309	70	10	206	9	1430	12870	
100	60	11	456	8	1590	12720	
414	70	8	327	9	1143	10287	
122	80	12	251	10	1714	17140	
138	130	12	248	17	1714	29138	
133	150	13	274	19	1860	35340	
113	80	12	488	10	1714	17140	
83	80	11	457	10	1590	15900	
121	40	16	334	5	2290	11450	
75	60	14	286	8	2000	16000	
175*	110	14	561	14*	2000	28000	28000
284	120	19	396	15	2714	40710	
87	-30	18	364	-4	2571	-10284	
213*	242	22	871	31*	3143	97433	97433
105*	120	14	550	15*	2000	30000	30000
35	140	27	225	18	3857	69426	
327	222	13	266	27	1857	50139	
23*	280	22	890	35*	3143	110005	110005
10	0	25	258	0	3571	0	
				PW:		578913	265438*
				Future Worth:		6,669,077	3,057,846
				FW-PW:		6,090,164	2,792,408
		Significance					
				Net Delay $:		$528,660	$242,400
				Net Delay $/Project:		$19,600	$60,600

Net Delay $ = PW $[1 - (1+i)^{-n}]$; i = 8% inflation + 5% interest.

5. CONCLUSIONS. The types of engineering network models which can effectively represent the MCA process and the scope of the computer simulation developments to implement these representations have been brought into focus by the current study. It has been concluded that the CEGNS computer simulation system will support a simulation model of the total MCA process at all required levels of detail. The simulation approach selected will permit rapid assessments of proposed changes in "real world" procedures, and can be used for diagnosing most problems that arise in operations at the management and field performance levels.

Existing Capabilities. The Study Model has replicated MCA procedures from the time of project formulation to construction completion. The Study Model demonstrated the capacity of the CEGNS simulation modeling system to support the required levels of analysis. CEGNS requires relatively little change in the input stream for solving two problems with minor differences. This capability allows the analyst to vary both the input requirements of the problem and the problem itself in order to determine any instability or lack of response in the solution. The adequacy of the structural and processing concepts of the Study Model has been demonstrated. Verification of this model from detailed operations information, plus computer-processing efficiency adjustments, can result in a "Prototype" MCA Process Simulation Model requiring only a final calibration and validation phase before delivery as an "operational" model.[7]

Future Developments. The MCA Simulation Model will have a broad applications potential. The current Study Model has been demonstrated in the research mode and can now be used in comparative investigations such as measuring the relative impact of proposed changes in the MCA process. Experience gained from exercising the Study Model has provided the base (criteria) for developing an MCA Process Simulation Model of greatest benefit to the Corps. This model can contribute to improved procedures-analysis; improved management control (impact forecasting, optimal product scheduling, resource allocation studies, etc); and the assessment of special problem areas. Confidence in the output of an "operational" MCA Process Simulation Model will be strengthened if it is supported by a significant data bank which has been developed in response to determined modeling needs[3]. This data bank should contain key information that correlates external influences and events with the decision functions which impact the MCA process. The data bank can contribute to the creation of a more realistic generation of projects and attribute assignments for these projects. It will be a source for calibration values to be applied to network logic and subsequent adjustments to this logic. Finally, it will be a principal reference in verification arguments.

Future development objectives require the refinement of model capabilities to that of a convenient and useful analytical tool (see Table II). The model must represent program fluctuations or imposed procedural changes with a precision sufficient to match computer responses to the corresponding "real-world" response. These refinements will permit the convincing demonstrations required of an operational system. Only through such demonstrations can the level of confidence be promulgated

which is necessary for the model output to be considered a primary argument in CE evaluations of the MCA process.

Table II
MCA Process Simulation Model Development Objectives

1. Replicate network flow.
2. Perform time/cost/resource expenditure measurements.
3. Approximate the performance response of process functions in the network to real-world response.
4. Incorporate macro/micro network focusing (modules with graded detail).
5. Consider study-support flexibility; provide selectable outputs.
6. Allow for exogenous influences.

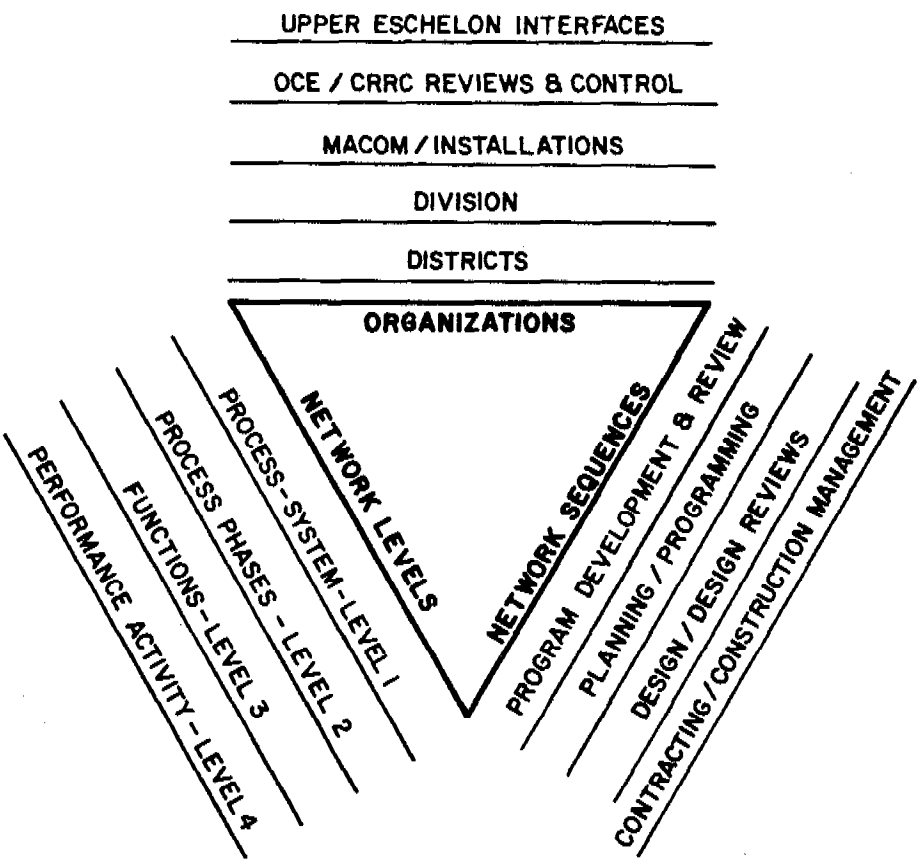


Figure 1. Dimensional ranges of the MCA process.

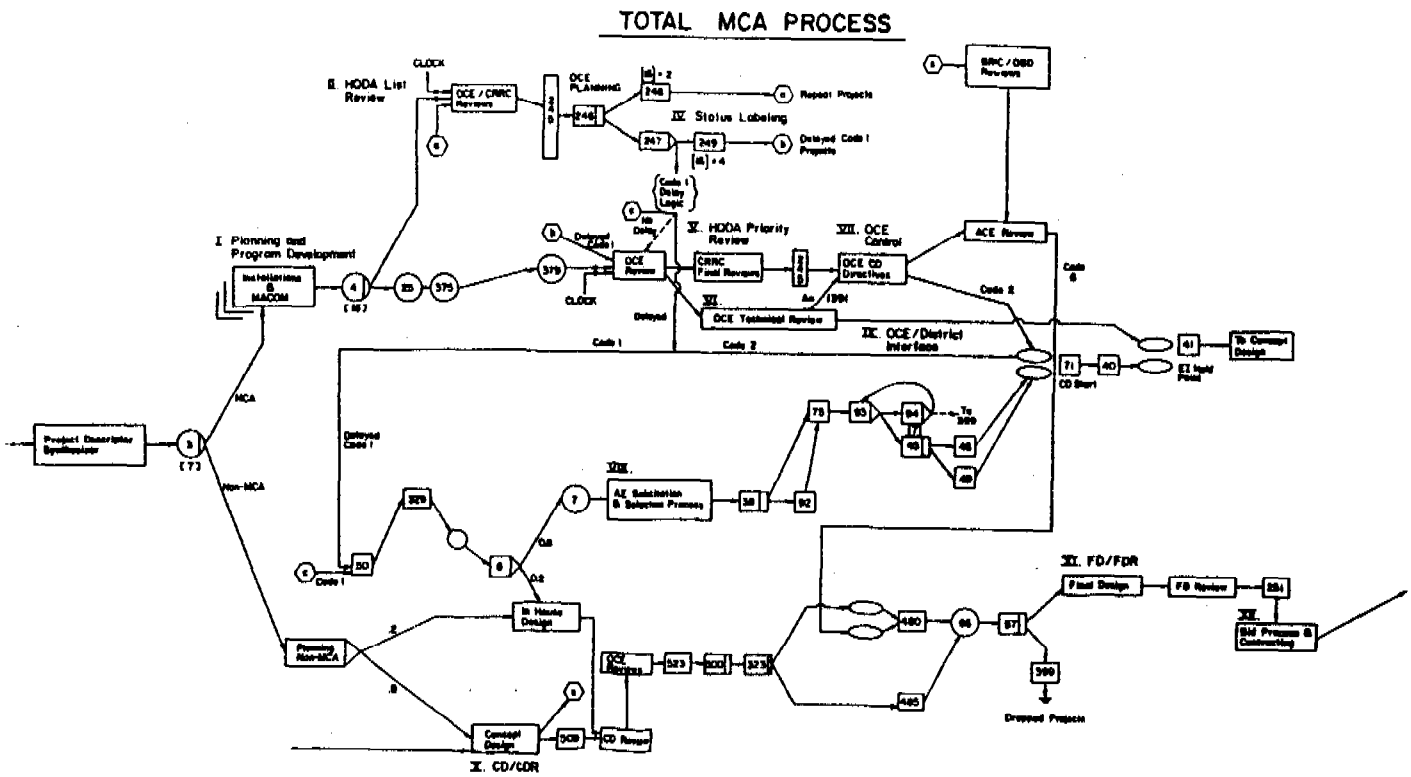


Figure 2. Network diagram of the MCA process to construction contracting.

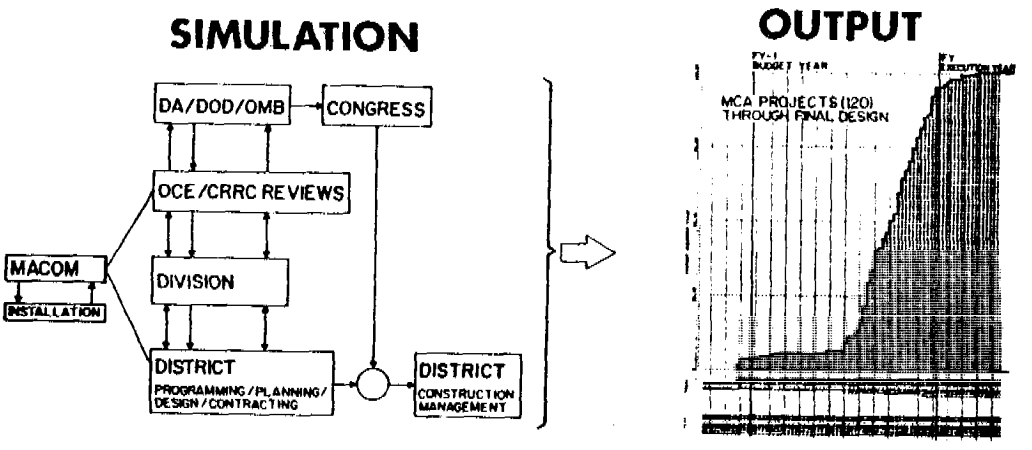


Figure 3. A simulation concept for the MCA process.

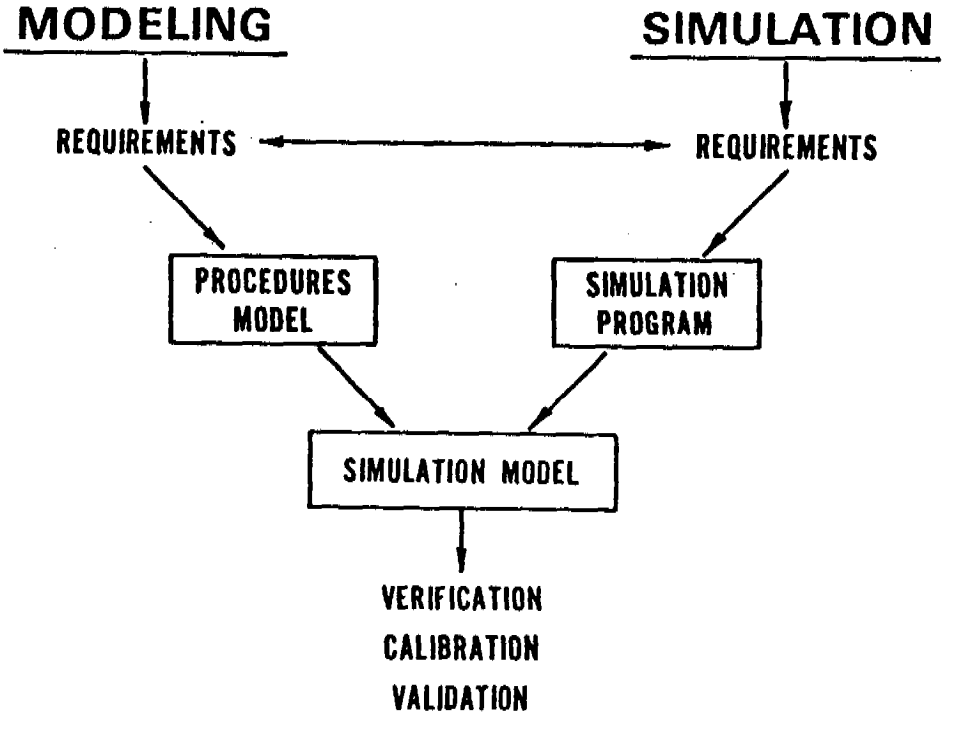


Figure 4. The iterative development of an MCA simulation modeling system.

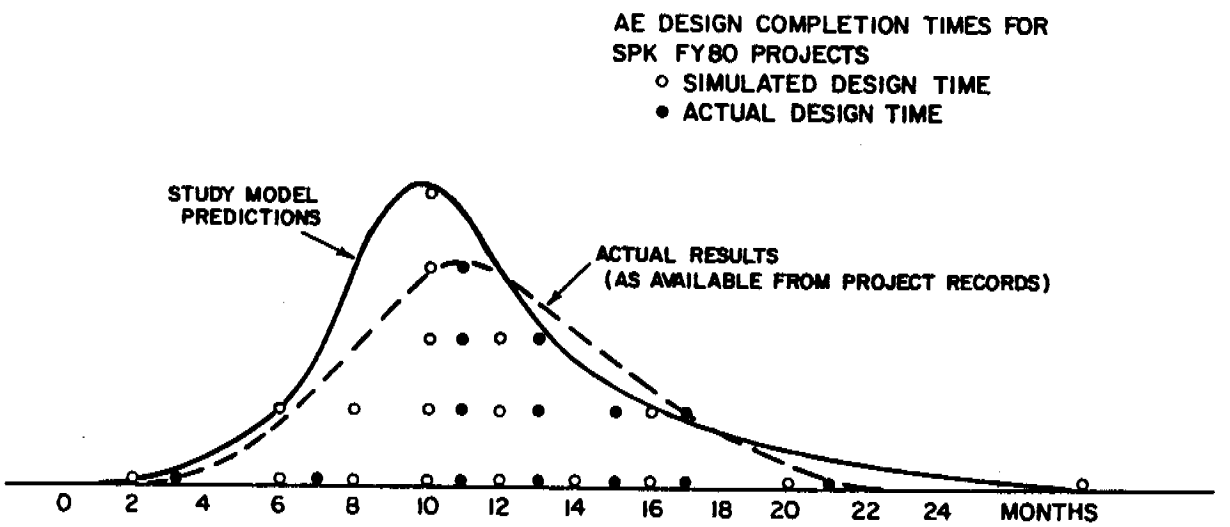


Figure 5. AE design-time distributions for FY80 projects from a Study Model simulation run and from actual performance records.

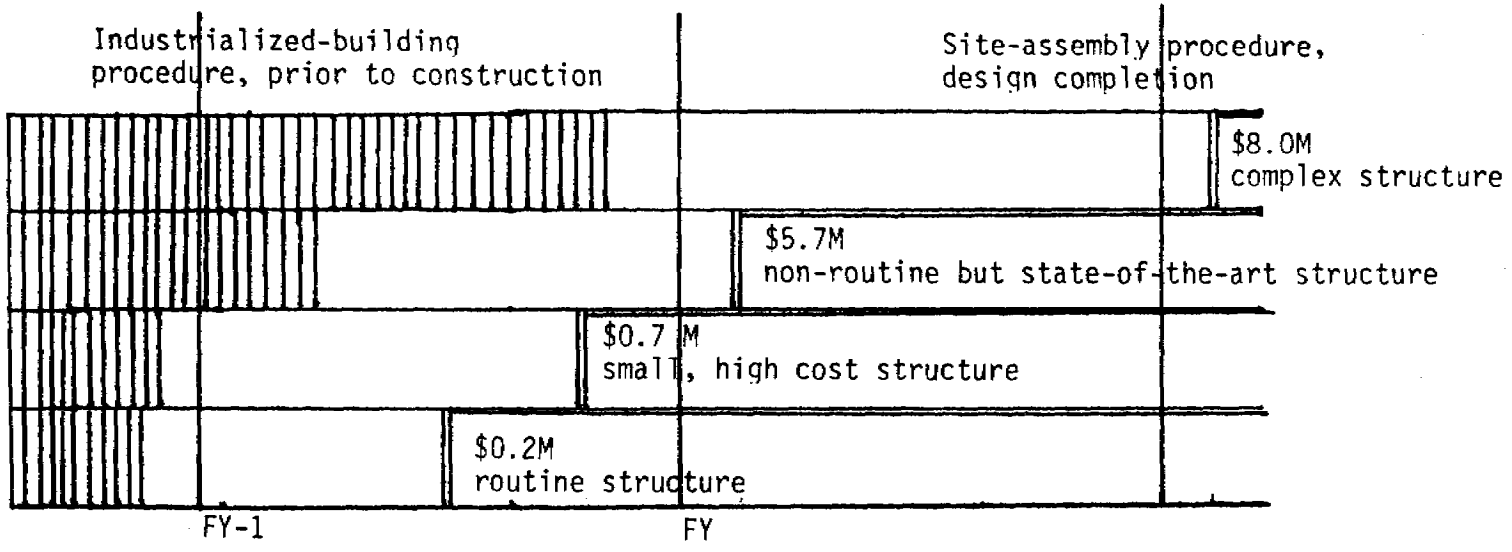


Figure 6. Simulated MCA processing times prior to construction for industrialized-building and site-built facility options (4 projects).

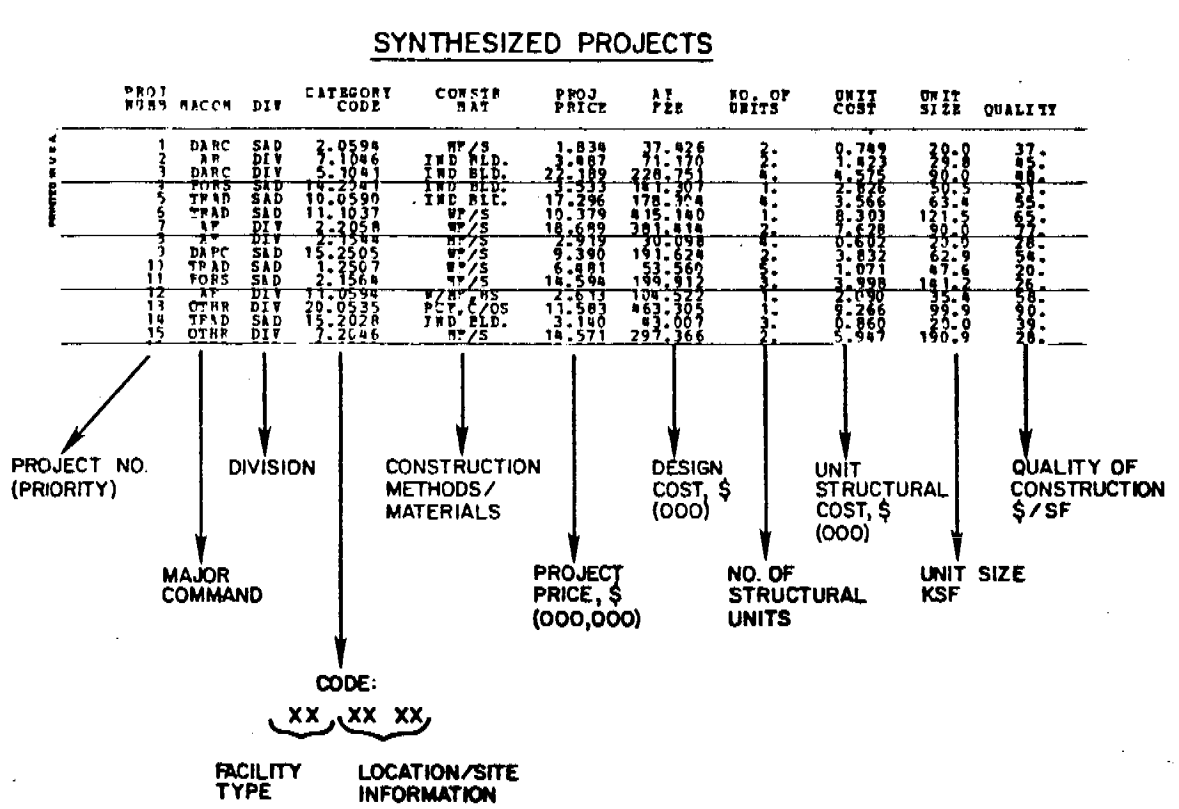


Figure 7. Sample from printout of a list of synthesized projects and their attributes.

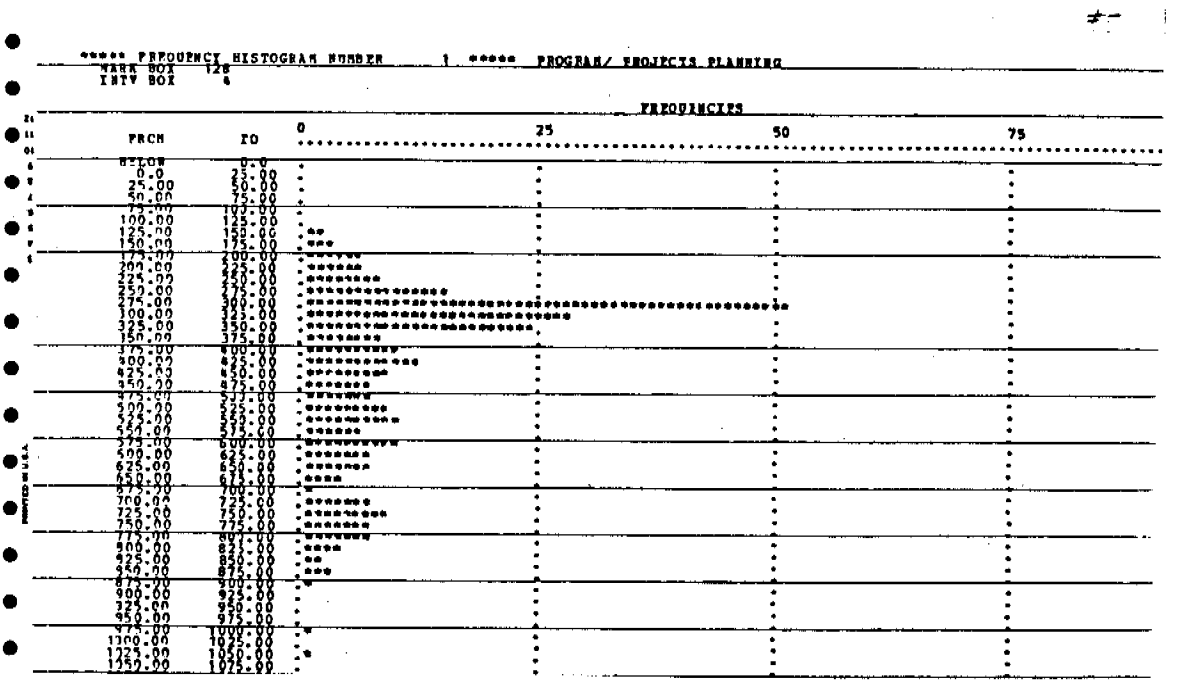


Figure 8. Simulated planning-phase times for 500 MCA projects.

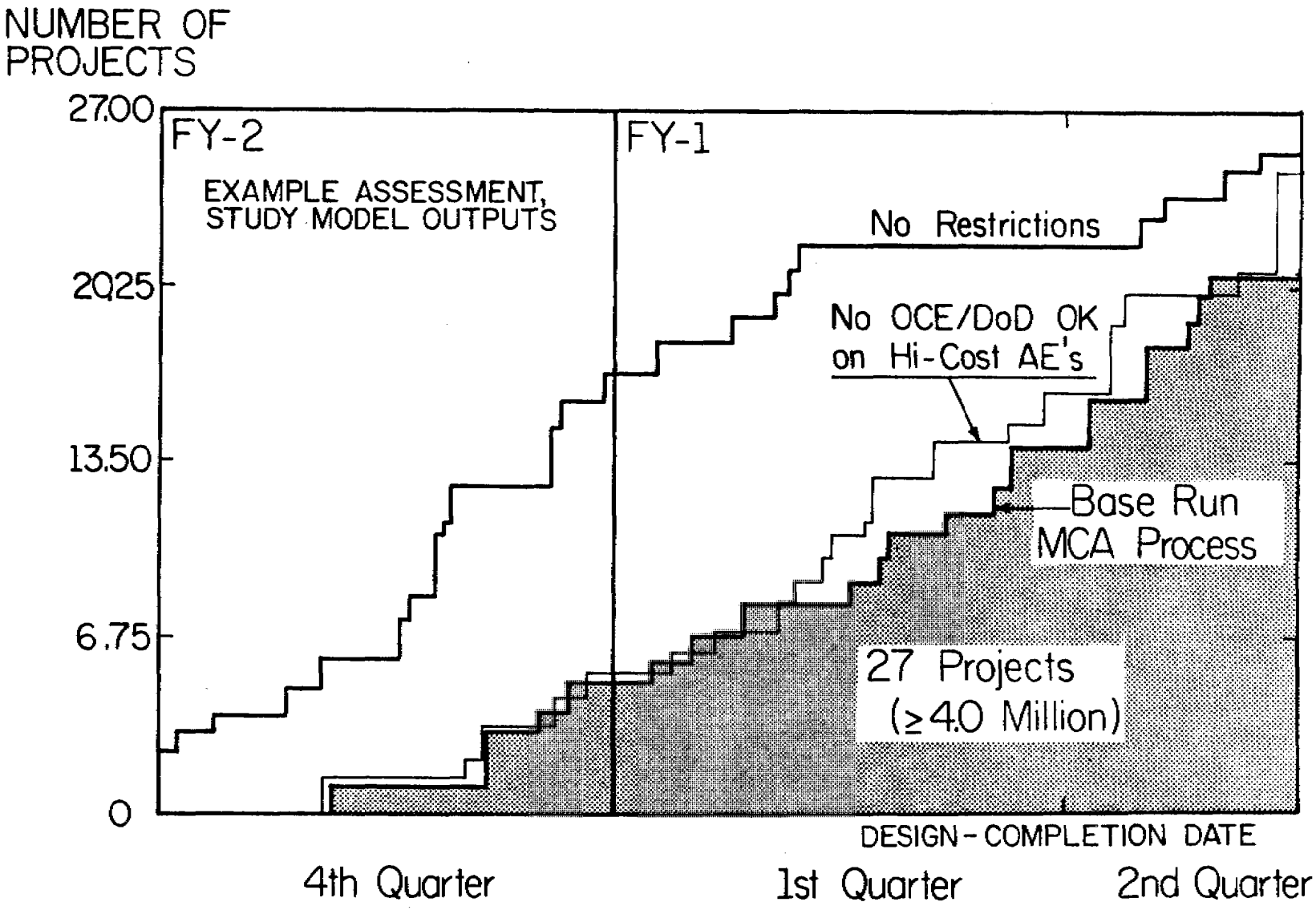


Figure 9. Simulated processing-times for MCA projects under 3 levels of assumed regulations.

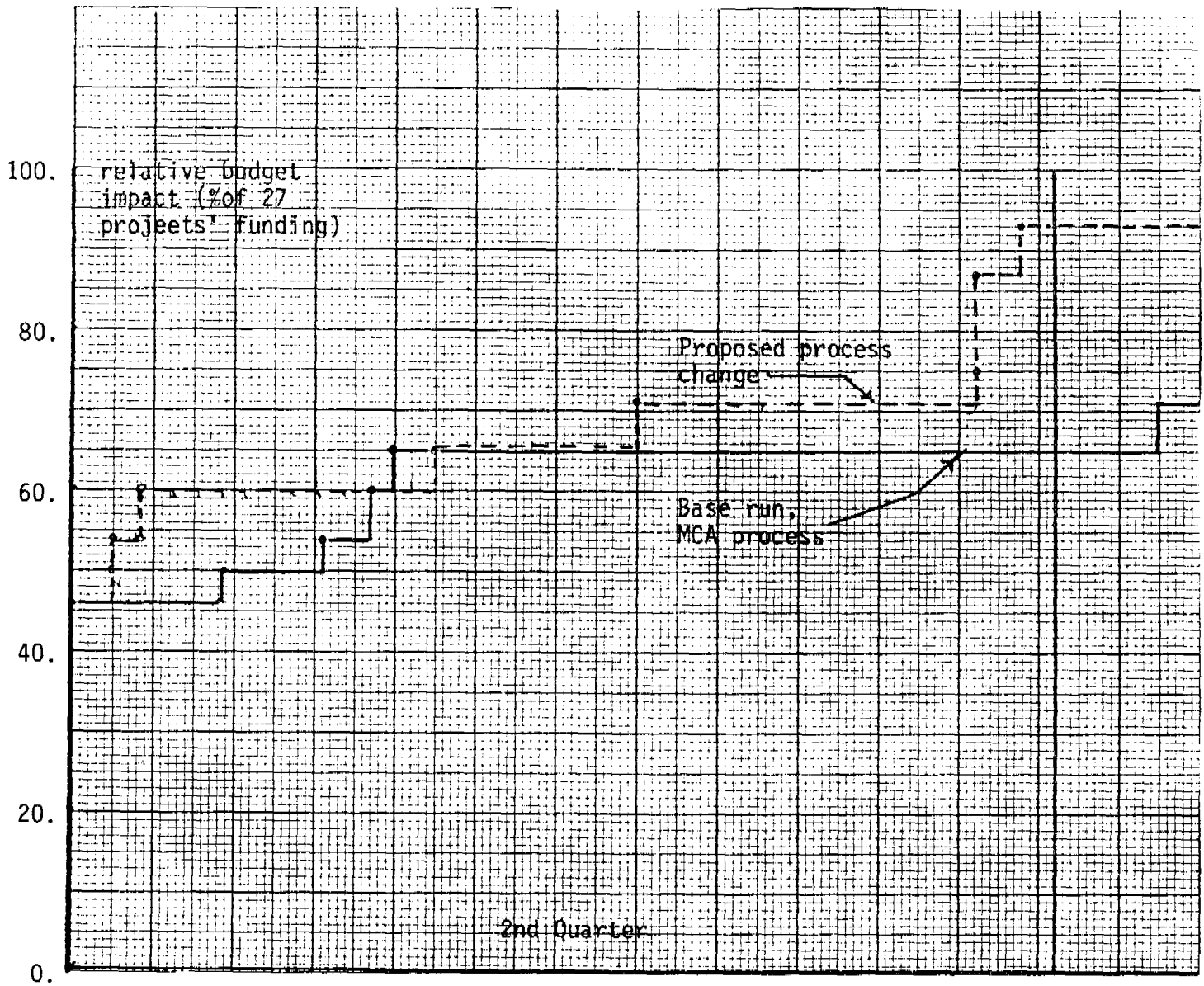


Figure 10. Processing-times vs. accumulated project $ values.

REFERENCES

1. Army Regulation, "Project Development and Design Approval," AR 415-20, DA, Revision Draft (1979).

2. Fishman, G.S., Concepts and Methods in Discrete Event Digitial Simulation, Wiley, NY, 1973.

3. Ghiassi, M., "An Evaluation of the CEGNS Modeling System for the MCA Process," Letter Report, Presented to CERL 30 April 1981.

4. Hogg, G. L., M. I. Dessouky and K. Tonegawa, "Generalized Network Simulator," Presented at the Joint ORSA/TIMS/AIEE National Meeting, Atlantic City, NJ, 70-9 Nov 77.

5. Johnson, J. H., "Information Flow for Military Construction," CERL-IR-ADS-2, Construction Engineering Research Laboratory, Champaign, IL, Oct 76.

6. Mesarovic, M. D., D. Macko, and Y. Takahara, Theory of Hierarchical, Multilevel Systems, Vol. 68 in the "Mathematics in Science and Engineering" series, Academic Press, NY, 1970.

7. Naylor, T., Computer Simulation Experiments with Models of Economic Systems, Wiley, NY, 1971. ("Validation", Chapter 5.)

8. Philips, D. T., M. L. Handwerker and P. Piumsomboon, "A Users Manual for GEMS, A Generalized Manufacturing Simulator," NSF/RANN Project, Grant No. APR76-22610, Dept. of Industrial Engineering, Texas A&M University, 1 Sep 79.

NUMERICAL SOLUTION OF THE BOUNDARY LAYER EQUATIONS AT THE SHOCK/SURFACE INTERFACE BEHIND A HEMISPHERICAL BLAST WAVE

Richard J. Pearson
U.S. Army Ballistic Research Laboratory
U.S. Army Armament Research and Development Command
Aberdeen Proving Ground, Maryland 21005

James E. Danberg
Department of Mechanical and Aerospace Engineering
University of Delaware
Newark, Delaware 19711

ABSTRACT. This paper discusses the solution of the boundary layer equations at the intersection of a hemispherical blast wave and a ground plane over which it is moving. A coordinate stretching transformation is used to eliminate the singularity at the shock/surface interface and reduce the governing equations at the interface to a set of ordinary differential equations. The solution of the equations at the interface is presented.

The original governing equations consist of a set of three, coupled, nonlinear partial differential equations. The coordinate transformation produces a set of three, coupled nonlinear ordinary differential equations at the shock/surface interface. Boundary conditions are given at the surface and at an infinite distance from the surface, forming an asymptotic two point boundary value problem. A method developed by Nachtsheim and Swigert is employed to reduce the problem to an initial value problem. An initial estimate for two unknown gradients at the wall is made and an iterative method is used to systematically reduce the mean square error in the solution at the outer edge of the boundary layer by changing the estimates of the gradients at the surface. This method requires the formulation and solution of six additional ordinary differential equations which are coupled to the first three. A Kutta-Merson method is used to solve the nine coupled equations. The result is a solution to the original equations, plus a correction to the two estimates of the gradients at the surface. The iteration procedure is repeated until the solution at the outer edge of the boundary layer agrees with the given conditions to within some specified degree of accuracy.

NOMENCLATURE.

r	radial distance from center of the hemispherical shock
ω	angle measured from the surface upwards
R_s	radius of the shock front
U_s	velocity of the shock front
U_p	particle velocity at the shock front
T_s	temperature at the shock front
P_s	pressure at the shock front
ρ_s	density at the shock front
T_a	ambient temperature
ρ_a	ambient density
ν_a	kinematic viscosity of the ambient air
μ_a	viscosity of the ambient air
P_r	Prandtl number of the ambient air
C_p	Specific heat of the ambient air
C_R	gas constant for air
K	thermal conductivity of air
T_w	temperature at the surface of the ground
η	transformed angular distance given by $\eta = rw\sqrt{\frac{U_p}{(R_s-r)\ \nu_a}}$
$\tilde{U}_r$	transformed radial velocity given by $\tilde{U}_r = U_r/U_p$
$\tilde{V}_\omega$	transformed angular velocity given by $\tilde{V}_\omega = \frac{V_\omega}{U_p}\sqrt{\frac{(R_s-r)\ U_p}{\nu_a}}$

$\tilde{\rho}$ transformed density given by $\tilde{\rho} = \rho/\rho_s$

$\tilde{T}$ transformed temperature given by $\tilde{T} = T/T_s$

ξ normalized radial distance given by $\xi \equiv r/R_s$

τ normalized time given by $\tau \equiv tU_p/R_s$

$\tilde{T}_\omega$ transformed surface temperature $\tilde{T}_\omega = T_\omega/T_s$

$\Psi \equiv \tilde{U}_r$ transformed radial velocity

$\phi \equiv \tilde{V}_\omega$ transformed angular velocity

$\Theta \equiv \tilde{T}$ transformed temperature

$$\Psi_x \equiv \frac{\partial \tilde{U}_r}{\partial X} \quad , \quad \Psi_y \equiv \frac{\partial \tilde{U}_r}{\partial Y}$$

$$\Theta_x \equiv \frac{\partial \tilde{T}}{\partial X} \quad , \quad \Theta_y \equiv \frac{\partial \tilde{T}}{\partial Y}$$

$$\phi_x \equiv \frac{\partial \tilde{V}_\omega}{\partial X} \quad , \quad \phi_y \equiv \frac{\partial \tilde{V}_w}{\partial Y}$$

$$\Psi' \equiv \frac{\partial \tilde{U}_r}{\partial \eta} \quad , \quad \Psi'_x \equiv \frac{\partial^2 \tilde{U}_r}{\partial X \partial \eta} \quad , \quad \Psi'_y \equiv \frac{\partial^2 \tilde{U}_r}{\partial Y \partial \eta}$$

$$\Theta' \equiv \frac{\partial \tilde{T}}{\partial \eta} \quad , \quad \Theta'_x \equiv \frac{\partial^2 \tilde{T}}{\partial X \partial \eta} \quad , \quad \Theta'_y \equiv \frac{\partial^2 \tilde{T}}{\partial Y \partial \eta}$$

$$\phi' \equiv \frac{\partial \tilde{V}_\omega}{\partial \eta} \quad , \quad \phi'_x \equiv \frac{\partial^2 \tilde{V}_\omega}{\partial X \partial \eta} \quad , \quad \phi'_y \equiv \frac{\partial^2 \tilde{V}_\omega}{\partial Y \partial \eta}$$

$$\Psi'' \equiv \frac{\partial^2 \tilde{U}_r}{\partial \eta^2} \quad , \quad \Psi''_x = \frac{\partial^3 \tilde{U}_r}{\partial X \partial \eta^2} \quad , \quad \Psi''_y = \frac{\partial^3 \tilde{U}_r}{\partial Y \partial \eta^2}$$

$$\Theta'' = \frac{\partial^2 \tilde{T}}{\partial \eta^2} \quad , \quad \Theta''_x = \frac{\partial^3 \tilde{T}}{\partial X \partial^2 \eta} \quad , \quad \Theta''_y = \frac{\partial^3 \tilde{T}}{\partial Y \partial \eta^2}$$

$\Theta_w \equiv \tilde{T}_w$ transformed surface temperature

A coefficient given by $A \equiv \frac{U_s}{2U_p}$

B coefficient given by $B \equiv \frac{P_e \nu_a T_a}{C_R T_s^2 \mu_a} = \frac{P_e T_a}{C_R T_s^2 \rho_a}$

C coefficient given by $C \equiv \frac{P_e T_a \nu_a P_r}{C_R T_s^2 \mu_a} = \frac{P_e T_a P_r}{C_R T_s^2 \rho_a}$

D coefficient given by $D \equiv \frac{U^2 \rho P_r}{C_p T_s}$

1. INTRODUCTION. When a nuclear weapon or high explosive charge is detonated in the atmosphere a blast wave is formed. At the front of the blast wave is a shock surface which separates two regions of gas in different states. In front of the expanding shock surface, the gas is at rest and at ambient pressure and temperature. As gas is engulfed by the moving shock, it undergoes a discontinuous jump to higher pressure, temperature and velocity. In the blast wave behind the shock, the pressure and velocity decay with distance from the shock. In strong blast waves near the detonation point, the temperature increases with increasing distance from the shock.

In order to study the strong blast waves which are formed just beyond the fire ball region of a large explosion, a point detonation model is often used. In the point detonation model the physics of the detonation is ignored and a spherical blast wave is considered to be formed by the instantaneous deposition of a large amount of energy at a point in space. The first analytical solution for the motion of a strong blast wave formed by a point detonation was the Taylor-Sedov[1] similarity solution. The Taylor-Sedov solution gives the motion of a spherical shock front and the pressure, temperature and velocity of the gas behind the front. The solution can be modified to solve the problem of a point detonation on flat surfaces. In this case the effective energy is doubled and the shock surface becomes hemispherical. (see Figure 1)

An implicit assumption in the Taylor-Sedov solution for a hemispherical blast wave is that the flow is inviscid. All real gases are viscous, however. To accurately model the blast wave flow near the surface, viscous effects must be taken into account.

When a viscous gas flows past a stationary surface, the gas directly adjacent to the surface sticks and remains at rest. The stationary gas at the surface acts to decelerate the gas adjacent to it, which in turn decelerates more gas. In this manner, a small but finite transition region developes in which the gas goes from zero velocity at the surface to the undisturbed velocity far from the surface. An analogous temperature transition region also developes in the flow. For an isothermal surface, the gas directly adjacent to the surface remains at surface temperature. The temperature of the gas beyond the surface increases through a small transition region until it reaches the undisturbed flow temperature far from the surface. The velocity and temperature transition regions are known as the velocity and thermal boundary layers. Both types of boundary layers are present in blast wave flow.

The boundary layer flow behind the shock surface starts out as a laminar flow, i.e. a flow in which disturbances tended to die out. As the flow moves away from the shock, it very rapidly becomes a turbulent flow, i.e. a flow in which disturbances tend to grow. Becauses this paper is concerned with flows only very near the shock surface the equation governing laminar boundary layer flow will be used.

Two solutions have been developed for the boundary layer flow within a hemispherical blast wave. Both the solution due to Crawford, et.al[2] and the solution due to S. W. Liu and H. Mirels[3] depended on the similarity property of the flow. The similarity assumption limits the validity of the

solution to the high pressure region near the detonation. In order to extend the solution into the low pressure region far from the detonation, a study is now underway to formulate a solution that does not depend on a similarity assumption. Because the solutions of Crawford and Liu do not agree on the temperature profile behind the shock, it is also hoped that the planned solution will help clarify this matter.

The planned solution is based on a finite difference scheme. The scheme requires three initial conditions and seven boundary conditions. The initial conditions will be taken from Crawford et.al. Three of the boundary conditions will come from the known conditions at the surface. Two of the boundary conditions will come from an inviscid outer flow solution. The final two boundary conditions must come from a solution at either the shock/surface interface or detonation point.

Singularities exist at both the detonation point and the shock/surface interface. For a point detonation, the temperature goes to infinity at the center of explosion, which results in the singularity. The singularity at the shock/surface interface results from the requirements of shock theory and boundary layer theory. Shock theory requires that gas engulfed by the shock jump discontinuously from a state of rest and ambient temperature to a state of higher velocity and temperature. Boundary layer theory requires that the gas adjacent to the surface remain at rest and at surface temperature. (see Figure 2) An attempt was made to find a coordinate transformation which would eliminate both singularities allowing a solution at both ends of the blast wave; none was found. A transformation was found, however, which allowed an asymptotic solution at the shock/surface interface alone.

The transformation used to eliminate the singularity at the shock/surface interface is based on the simularity transform from the Blasius flat plate solution.[4] The boundary layer problem behind a hemispherical shock does not meet the necessary conditions to have a similarity solution. The solution can, however, be considered locally similarly near the interface. Near the interface the transformation reduces the governing equation from a set of partial differential equations with three independent variables to a set of ordinary differential equations with the similarity variable η as the independent variable. The asymptotic solution at the interface based on the similarity transform is the topic of the rest of this paper.

2. THE BASIC GOVERNING EQUATIONS. There are three basic equations governing unsteady compressible boundary layer flow. The first equation is the Continuity Equation, based on the conservation of mass. The second equation is the Momentum Equation, based on the conservation of momentum. The third equation is the Energy Equation, based on the conservation of energy. Because of the geometry of the problem, the two dimensional cartesian form of the equations have been transformed to their axisymmetric spherical form (see Figure 3). To simplify the equation ω, shown in Figure 3, has been substituted for θ. Further, it has been assumed that ω is small so that $\sin \omega \simeq \omega$ and $\cos \omega \simeq 1$, The resulting equations are shown below.

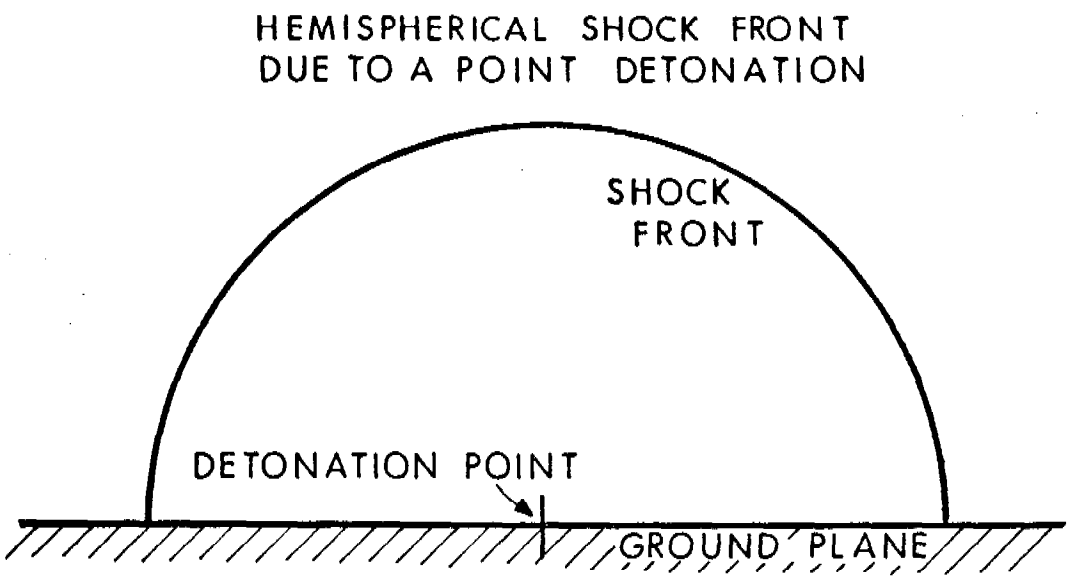


Figure 1. Hemispherical Shock Front.

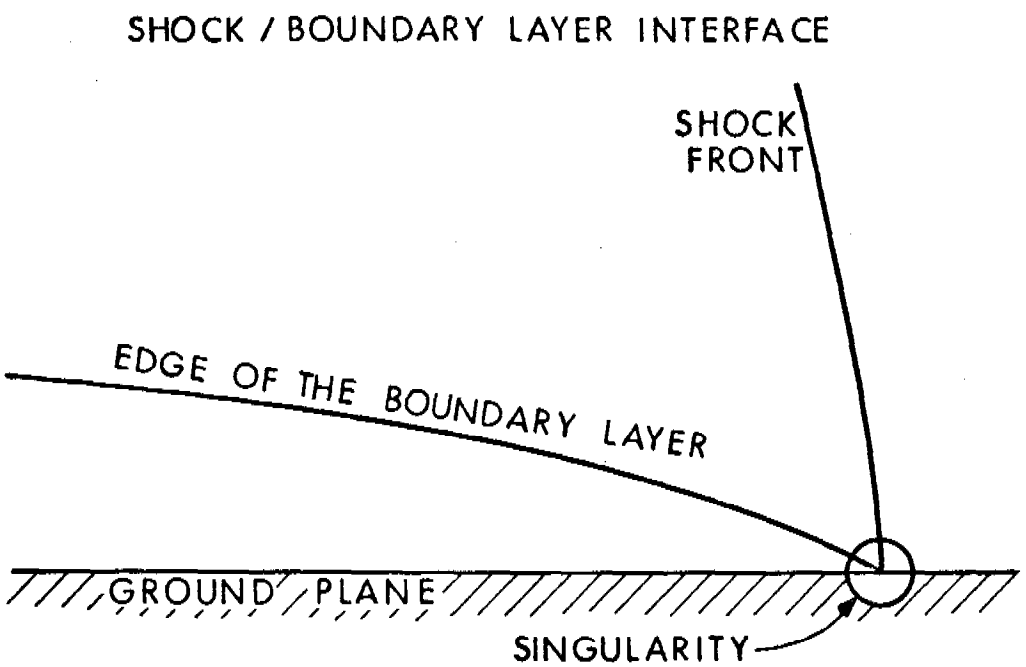


Figure 2. Singularity at the Shock/Boundary Layer Interface.

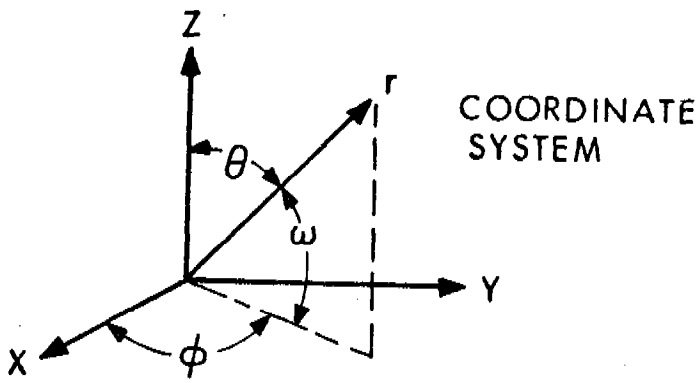


Figure 3. Coordinate System.

Continuity Equation

$$\frac{\partial \rho}{\partial t} + \frac{1}{r^2} \frac{\partial}{\partial r}\left(r^2 \rho U_r\right) + \frac{1}{r} \frac{\partial}{\partial \omega}\left(\rho V_\omega\right) = 0$$

Momentum Equation

$$\rho\left[\frac{\partial U_r}{\partial t} + U_r \frac{\partial U_r}{\partial r} + \frac{V_\omega}{r} \frac{\partial U_r}{\partial \omega}\right] = - \frac{\partial P_e}{\partial r} + \frac{1}{r}\left[\mu\left(\frac{1}{r} \frac{\partial U_r}{\partial \omega}\right)\right]$$

Energy Equation

$$\frac{C_p}{C_R} \frac{1}{\rho}\left[\frac{\partial \rho}{\partial t} + U_r \frac{\partial \rho}{\partial r} + \frac{V_\omega}{r} \frac{\partial \rho}{\partial \omega}\right] = - \frac{1}{P_e}\left\{\left[1 - \frac{C_p}{C_R}\right]\left[\frac{\partial P_e}{\partial t} + U_r \frac{\partial P_e}{\partial r}\right] + \mu\left[\frac{1}{r} \frac{\partial U_r}{\partial \omega}\right]^2 - \frac{P_e}{C_R \rho^2}\left[\frac{1}{r} \frac{\partial}{\partial \omega}\left(\frac{K}{r} \frac{\partial \rho}{\partial \omega}\right) - \frac{2K}{\rho r^2}\left(\frac{\partial \rho}{\partial \omega}\right)^2\right]\right\}$$

In these equation, ρ is the density, t is time, U_r is the radial velocity, r is the radial distance, ω is the angle measured from the surface upwards, V_ω is the angular velocity, P_e is the pressure in the outer flow, μ is the viscosity, C_p is the specific heat, C_R is the ideal gas constants and K is thermal conductivity.

In addition to the three basic equations, there are three assumptions that have been made in formulating the problem. First, it has been assumed that the gas obeys the ideal gas equation of state $\rho = P/C_R T$. Second, that the Prandtl number is constant $P_r \equiv (\mu C_p/K) = \text{constant}$. Finally, that the viscosity is a direct function of temperature $\mu = \mu_a(T/T_a)$.

3. TRANSFORMING AND NORMALIZING THE BASIC GOVERNING EQUATIONS. In order to eliminate the singularity at the shock/surface interface, the basic equations are transformed in terms of the following variables.

$$\tau = \frac{tU_p}{R_s}$$

$$\xi = \frac{r}{R_s}$$

$$\eta = r\omega\sqrt{\frac{U_p}{(R_s-r)\ \nu_o}}$$

$$\tilde{U}_r = U_r/U_p$$

$$\tilde{V}_\omega = \frac{V_\omega}{U_p}\sqrt{\frac{(R_s-r)\ U_p}{\nu}}$$

$$\tilde{\rho} = \rho/\rho_s$$

$$\tilde{T} = T/T_s$$

In the transformation R_s is the shock radius, ν_a is the Kinematic viscosity ahead of the shock, U_p, ρ_s and T_s are the flow velocity, density and temperature at the shock front. The transformation which give τ, ξ, $\tilde{U}_r$, $\tilde{\rho}$ and $\tilde{T}$ serve only to nondimensionalize and normalize the variables. Transformation for η allows us to eliminate the singularity located at the shock radius, R_s.

The three basic equations have been rewritten in terms of the new coordinate system. In addition, the ideal gas equation of state has been used to eliminate $\tilde{\rho}$ in terms of $\tilde{T}$ and P_e. The viscosity μ has been eliminated in terms of $\tilde{T}$ and the constants T_a and μ_a. The thermal conductivity K has been eliminated in terms of $\tilde{T}$ and the constants P_r, C_p, T_a and μ_a. The resulting set of three coupled, partial-differential equations is given below.

Continuity Equation

$$\tilde{T}\left(\frac{\partial \tilde{V}_\omega}{\partial \eta}\right) - \tilde{V}_\omega\left(\frac{\partial \tilde{T}}{\partial \eta}\right) = (1-\xi)\left(\frac{\partial \tilde{T}}{\partial \tau}\right) - \xi(1-\xi)\left(\frac{\partial \tilde{T}}{\partial \xi}\right) +$$

$$(1-\xi)\,\tilde{U}_r\left(\frac{\partial \tilde{T}}{\partial \xi}\right) + \left[\frac{A_p\,R_s\,(1-\xi)\eta}{2U_p^2} - \frac{U_s\eta}{2U_p}\right]\left(\frac{\partial \tilde{T}}{\partial \eta}\right) + \left[\frac{(1-\xi)\eta}{\xi} - \frac{\eta}{2}\right]\tilde{U}_r\left(\frac{\partial T}{\partial \eta}\right) -$$

$$\left[\frac{(1-\xi)\eta}{\xi} - \frac{\eta}{2}\right]\tilde{T}\left(\frac{\partial \tilde{U}_r}{\partial \eta}\right) - \left[\frac{R_s(1-\xi)}{P_e}\left(\frac{\partial P_e}{\partial r}\right) + \frac{2(1-\xi)}{\xi}\right]\tilde{U}_r\tilde{T}$$

$$- \left[\frac{R_s(1-\xi)}{U_p\,P_e}\left(\frac{\partial P_e}{\partial t}\right)\right]\tilde{T}$$

Momentum Equation

$$(1-\xi)\left(\frac{\partial \tilde{U}_r}{\partial \tau}\right) + (1-\xi)\,\tilde{U}_r\left(\frac{\partial \tilde{U}_r}{\partial \xi}\right) - \left[\frac{U_s\xi(1-\xi)}{U_p}\right]\left(\frac{\partial \tilde{U}_r}{\partial \xi}\right) +$$

$$\left[\frac{A_p\,R_s(1-\xi)}{2U_p^2} - \frac{U_s\eta}{2U_p}\right]\left(\frac{2\tilde{U}_r}{\partial \eta}\right) + \left[\frac{(1-\xi)\eta}{\xi} - \frac{\eta}{2}\right]\tilde{U}_r\left(\frac{\partial \tilde{U}_r}{\partial \eta}\right) + \tilde{V}_\omega\left(\frac{\partial \tilde{U}_r}{\partial \eta}\right) -$$

$$\left[\frac{C_R\,\mu_a\,T_s^2}{T_a\,\nu_a\,P_e}\right]\tilde{T}\left(\frac{\partial \tilde{T}}{\partial \eta}\right)\left(\frac{\partial \tilde{U}_r}{\partial \eta}\right) - \left[\frac{C_R\,\mu_a\,T_s^2}{T_a\,\nu_a\,P_e}\right](\tilde{T}^2)\left(\frac{\partial^2 \tilde{U}_r}{\partial \eta^2}\right) +$$

$$\left[\frac{R_s(1-\xi)}{U_p^2}\left(\frac{\partial U_p}{\partial t}\right)\right]\tilde{U}_r + \left[\frac{C_R\,T_s\,R_s(1-\xi)}{U_p^2}\left(\frac{\partial P_e}{\partial r}\right)\right]\tilde{T} = 0$$

Energy Equation

$$(1-\xi)\left(\frac{\partial \tilde{T}}{\partial \tau}\right) + (1-\xi)\,\tilde{U}_r\left(\frac{\partial \tilde{T}}{\partial \xi}\right) - \left[\frac{U_s\xi(1-\xi)}{U_p}\right]\left(\frac{\partial \tilde{T}}{\partial \xi}\right) +$$

$$\left[\frac{A_p\,R_s(1-\xi)\eta}{2U_p^2} - \frac{U_s\eta}{2U_p}\right]\left(\frac{\partial \tilde{T}}{\partial \eta}\right) + \left[\frac{(1-\xi)\eta}{\xi} - \frac{\eta}{2}\right]\tilde{U}_r\left(\frac{\partial \tilde{T}}{\partial \eta}\right) + \tilde{V}_\omega\left(\frac{\partial \tilde{T}}{\partial \eta}\right)$$

$$\left[\frac{C_R\,T_s^2\,\mu_a}{P_e\,\nu_a\,T_a\,P_r}\right]\tilde{T}\left(\frac{\partial \tilde{T}}{\partial \eta}\right)^2 - \left[\frac{C_R\,T_s^2\,\mu_a}{P_e\,\nu_a\,T_a\,P_r}\right](\tilde{T}^2)\left(\frac{\partial^2 \tilde{T}}{\partial \eta^2}\right) -$$

$$\left[\frac{C_R\,R_s(1-\xi)}{C_p\,P_e}\left(\frac{\partial P_e}{\partial r}\right)\right]\tilde{U}\,\tilde{T} - \left[\frac{C_R\,R_s(1-\xi)}{C_p\,U_p\,P_e}\left(\frac{\partial P_e}{\partial t}\right)\right]\tilde{T} - \left[\frac{C_R\,U_p^2\,\mu_a\,T_s}{C_p\,P_e\,\nu_a\,T_a}\right](T^2)\left(\frac{\partial \tilde{U}_r}{\partial \eta}\right)^2 = 0$$

4. TRANSFORMED GOVERNING EQUATION AT THE SHOCK/SURFACE INTERFACE. As the shock/surface interface is approached $r \to R_s$ and therefore $\xi \to 1$. Thus as the interface is approached, terms with $(\xi-1)$ in the numerator drop out. We are left with derivatives with respect to η only. The continuity equation simplifies to η

$$\frac{\partial V_\omega}{\partial \eta} = -\frac{1}{\tilde{T}}\left[\frac{U_s \eta}{2U_p} + \frac{\eta \tilde{U}_r}{2} - V_\omega\right]\left(\frac{\partial \tilde{T}}{\partial \eta}\right) + \frac{\eta}{2}\left(\frac{\partial \tilde{U}_r}{\partial \eta}\right); \qquad (1)$$

the Momentum Equation becomes

$$\frac{\partial^2 \tilde{U}_r}{\partial \eta^2} = -\left[\frac{P_e \gamma_a T_a}{C_R T_s^2 \mu_a}\right]\left[\frac{U_s \eta}{2U_p} + \frac{\eta \tilde{U}_r}{2} - \tilde{V}_\omega\right]\left(\frac{1}{\tilde{T}^2}\right)\left(\frac{\partial \tilde{U}_r}{\partial \eta}\right) - \frac{1}{\tilde{T}}\left(\frac{\partial \tilde{T}}{\partial \eta}\right)\left(\frac{\partial \tilde{U}_r}{\partial \eta}\right); \qquad (2)$$

finally, the Energy Equation becomes

$$\frac{\partial^2 \tilde{T}}{\partial \eta^2} = -\left[\frac{P_e T_a \gamma_a P_r}{C_R T_s^2 \mu_a}\right]\left[\frac{U_s \eta}{2U_p} + \frac{\eta \tilde{U}_r}{2} - \tilde{V}_\omega\right]\left(\frac{1}{\tilde{T}^2}\right)\left(\frac{\partial \tilde{T}}{\partial \eta}\right) - \left[\frac{U_p^2 P_r}{C_p T_s}\right]\left(\frac{\partial \tilde{U}_r}{\partial \eta}\right)^2 - \frac{1}{\tilde{T}}\left(\frac{\partial \tilde{T}}{\partial \eta}\right)^2. \qquad (3)$$

The governing equations require five boundary conditions for solution, three at the surface and two in the outer flow. At the surface the velocity gas must go to zero i.e. $\tilde{U}_r = \tilde{V}_\omega = 0$. In addition for an isothermal surface the gas temperature $\tilde{T}$ must equal the surface or wall temperature $\tilde{T}_w$ i.e. $\tilde{T} = \tilde{T}_w$. Boundary layer theory requires that U_r and T asymptotically approach the values of radial velocity and temperature found in outer flow, which leads to the following conditions $\tilde{U}_r \to 1$ and $\tilde{T} \to 1$. The five conditions are summarized below.

at $\eta = 0$

$\tilde{U}_r = \tilde{V}_\omega = 0$ and $\tilde{T} = \tilde{T}_w$

as $\eta \to \infty$

$\tilde{U}_r \to 1$ and $\tilde{T} \to 1$

With only derivatives of η left in the equation, we can now regard them as a coupled set of ordinary differential equations (O. D. E.'s) rather than a set of coupled P. E. D.'s. Changing to a simplier notation we obtain the following.

Continuity Equation

$$\phi' = -\frac{1}{\Theta}\left[A\eta + \frac{\eta\Psi}{2} - \phi\right]\Theta' + \left(\frac{\eta}{2}\right)\Psi'$$

Momentum Equation

$$\Psi'' = -\frac{B}{\Theta^2}\left[A\eta + \frac{\eta\Psi}{2} - \phi\right]\Psi' - \frac{1}{\Theta}\Theta'\Psi'$$

Energy Equation

$$\Theta'' = -\frac{C}{\Theta^2}\left[A\eta + \frac{\eta\Psi}{2} - \Theta\right]\Theta' - D\Psi'^2 - \frac{\Theta'^2}{\Theta}$$

Boundary Conditions

$$\text{at} \quad \eta = 0 \quad \Psi = \phi = 0 \quad \text{and} \quad \Theta = \Theta_w$$

$$\text{as} \quad \eta \to \infty \quad \Psi \to 1 \quad \text{and} \quad \Theta \to 1$$

In this notation, the primes denote differentiation with respect to η. The coefficients A, B, C and D are independent of η.

5. ASYMPTOTIC SOLUTION AT THE SHOCK/SURFACE INTERFACE. The outer flow boundary conditions make the problem an asymptotic boundary-value problem. In order to make use of conventional algorithms for the integration of O. D. E.'s, the problem must be transformed into a initial-condition problem. A method developed by Nachtsheim and Swigert of the Lewis Research Center[6] is used. Two additional boundary conditions at the surface are substituted for the two conditions in the outer flow. The new conditions at the surface require that $\Psi' = X$ and $\Theta' = Y$ when $\eta = 0$. The correct values of X and Y are unknown initially and must be determined as part of the solution.

The method of solution is iterative. In the initial iteration, the values of X and Y are guessed. X and Y are used to start an integration from the surface to some large value of η: η edge. The results of the integration at η edge are used to estimate the error in the solution in the next iteration after a change in X and Y of ΔX and ΔY. ΔX and ΔY are adjusted to minimize the error. X and Y are changed by the resulting ΔX and ΔY. The new X and Y values serve as the basis for the next integration to η edge. The process is repeated until the estimated error falls below a pre-set limit, at which point the problem is terminated.

The estimated error, which serves as the basis for convergence, is found by expanding the solution about X and Y. The values of Ψ and Θ at η edge are considered to be functions of X and Y. The values of Ψ and Θ in the next iteration are estimated using the first three terms of a Taylor series expansion. The difference between the estimated values of Ψ and Θ at η edge and their known value in the outer flow make up part of the estimated error. Experience has shown that two additional conditions in the outer flow must be imposed on the solution to insure that it is unique. The additional

conditions require that the radial velocity and temperature in the boundary layer approach the outer flow asymptotically, i.e. $\Psi' \to 0$ and $\Theta' \to 0$ as $\eta \to \infty$. The values of Ψ' and Θ' at η edge are estimated in the same manner as Ψ and Θ. The difference between the estimated values of Ψ' and Θ' at η edge and their known values in the outer flow makes up the final part of the estimated error.

The four errors which make up the total estimated error in the solution are designed δ_1, δ_2, δ_3 and δ_4. δ_1 and δ_2 refer to the difference between the known and estimated values of Ψ and Θ respectively. δ_3 and δ_4 refer to the difference between the known and estimated values of Ψ' and Θ'. The equations for δ_1, δ_2, δ_3 and δ_4 are shown below.

$$\delta_1 \equiv \Psi\ (\eta \text{ edge}, X + \Delta X, Y + \Delta Y) - 1 \simeq$$

$$\Psi\ (\eta \text{ edge}, X, Y) + \Psi_x\ (\eta \text{ edge}, X, Y)\ \Delta X + \Psi_y\ (\eta \text{ edge}, X, Y)\ \Delta Y - 1 =$$

$$\Psi + \Psi_x \Delta X + \Psi_y \Delta Y - 1$$

$$\delta_2 \equiv \Theta\ (\eta \text{ edge}, X + \Delta X, \Delta Y) - 1 \simeq$$

$$\Theta\ (\eta \text{ edge}, X, Y) + \Theta_x\ (\eta \text{ edge}, X, Y)\ \Delta X + \Theta_y\ (\eta \text{ edge}, X, Y)\ \Delta Y - 1 =$$

$$\Theta + \Theta_x \Delta X + \Theta_y \Delta Y - 1$$

$$\delta_3 \equiv \Psi'\ (\eta \text{ edge}, X + \Delta X, Y + \Delta Y) - 0 \simeq$$

$$\Psi'\ (\eta \text{ edge}, X, Y) + \Psi'_x\ (\eta \text{ edge}, X, Y)\ \Delta X + \Psi'_y\ (\eta \text{ edge}, X, Y)\ \Delta Y =$$

$$\Psi' + \Psi'_x \Delta X + \Psi'_y\ \Delta Y$$

$$\delta_4 \equiv \Theta'\ (\eta \text{ edge}, X + \Delta X, Y + \Delta X) - 0 \simeq$$

$$\Theta'\ (\eta \text{ edge}, X, Y) + \Theta'_x\ (\eta \text{ edge}, X, Y)\ \Delta X + \Theta'_y\ (\eta \text{ edge}, X, Y)\ \Delta Y =$$

$$\Theta' + \Theta'_x \Delta X + \Theta'_y \Delta Y$$

In these equations, the subscripts x and y denote partial differentiation with respect to X and Y.

The actual convergence criterion is based on the sum of the squares of the errors δ_1, δ_2, δ_3 and δ_4. This sum of the square errors is minimized with respect to ΔX and ΔY resulting in the following equations. Minimizing with respect to ΔX.

$$\frac{\partial \delta_1^2}{\partial \Delta X} + \frac{\partial \delta_2^2}{\partial \Delta X} + \frac{\partial \delta_3^2}{\partial \Delta X} + \frac{\partial \delta_4^2}{\partial \Delta X} =$$

$$\left[\Psi_x^2 + \Theta_x^2 + \Psi_x'^2 + \Theta_x'^2\right]\Delta X + \left[\Psi_x\Psi_y + \Theta_x\Theta_y + \Psi_x'\Psi_y' + \Theta_x'\Theta_y'\right]\Delta Y +$$

$$\left[\Psi_x\ (\Psi - 1) + \Theta_x\ (\Theta - 1) + \Psi'\Psi_x + \Theta'\Theta_x\right] = 0$$

Minimizing with respect to ΔY.

$$\frac{\partial \delta_1^2}{\partial \Delta Y} + \frac{\partial \delta_2^2}{\partial \Delta Y} + \frac{\partial \delta_3^2}{\partial \Delta Y} + \frac{\partial \delta_4^2}{\partial \Delta Y} =$$

$$\left[\Psi_y^2 + \Theta_y^2 + \Psi_y'^2 + \Theta_y'^2\right]\Delta Y + \left[\Psi_y\Psi_x + \Theta_x\Theta_y + \Psi_x'\Psi_y' + \Theta_x'\Theta_y'\right]\Delta X +$$

$$\left[\Psi_y\ (\Psi - 1) + \Theta_y\ (\Theta - 1) + \Psi\Psi_y' + \Theta'\Theta_y'\right] = 0$$

The results are two equations that can be solved simultaneously for ΔX and ΔY.

The Minimization equation contains Ψ, Θ, Ψ' and Θ' which can be evaluated at η edge by integrating Equation 1, Equation 2 and Equation 3. It also contains Ψ_x, Ψ_y, Θ_x, Θ_y, Ψ_x', Ψ_y', Θ_x' and Θ_y' which must be evaluated by integrating another set of equations. The necessary equations are formulated by differentiating Equations 1 through 3 with respect to X and Y. Differentiating with respect to X results in the following

$$\Psi_x'' = \left[A\eta + \frac{\eta\Psi}{2} - \phi\right]\left[\frac{2B\Theta_x\Psi'}{\Theta^3} - \frac{B\Psi_x'}{\Theta^2}\right] - \left[\frac{\eta\Psi_x}{2} - \phi_x\right]\left[\frac{B\Psi'}{\Theta^2}\right] +$$

$$\frac{1}{\Theta}\left[\frac{\Theta_x\Theta'\Psi'}{\Theta} - \Theta_x'\Psi' - \Theta'\Psi_x'\right] \qquad (4)$$

$$\Theta_x'' = \left[A\eta + \frac{\eta\Psi}{2} - \phi\right]\left[\frac{2C\Theta_x\Theta'}{\Theta^3} - \frac{C\Theta_x'}{\Theta^2}\right] - \left[\frac{\eta\Psi_x}{2} - \Theta_x\right]\left[\frac{C\Theta'}{\Theta^2}\right] -$$

$$2D\Psi'\Psi_x' - \frac{2\Theta'\Theta_x'}{\Theta} + \frac{\Theta'^2\Theta_x}{\Theta^2} \qquad (5)$$

$$\phi_x' = \left[A\eta + \frac{\eta\Psi}{2} - \phi\right]\left[\frac{\Theta_x\Theta'}{\Theta^2} - \frac{\Theta_x'}{\Theta}\right] - \left[\frac{\eta\Psi_x}{2} - \Theta_x\right]\frac{\Theta'}{\Theta} + \left[\frac{\eta}{2}\right]\Psi_x' \qquad (6)$$

The boundary conditions for Equations 4, 5 and 6 are the following:

$$\Psi_x = \Theta_x = \phi_x = \Theta_x' = 0 \quad \text{and} \quad \Psi_x' = 1 \quad \text{at} \quad \eta = 0$$

Differentiating with respect to Y results in the following:

$$\Psi''_y = \left[A\eta + \frac{\eta\Psi}{2} - \phi\right]\left[\frac{2B\Theta_y\Psi'}{\Theta^3} - \frac{B\Psi_y}{\Theta^2}\right] - \left[\frac{\eta\Psi_y}{2} - \Theta_y\right]\left[\frac{B\Psi'}{\Theta^2}\right] + \frac{1}{\Theta}\left[\frac{\Theta_y\Theta'\Psi}{\Theta} - \Theta'_y\Psi' - \Theta'\Psi'_y\right] \tag{7}$$

$$\Theta''_y = \left[A\eta + \frac{\eta\Psi}{2} - \phi\right]\left[\frac{2C\Theta_y\Theta'}{\Theta^3} - \frac{C\Theta'_y}{\Theta^2}\right] - \left[\frac{\eta\Psi_y}{2} - \phi\right]\left[\frac{C\Theta'}{\Theta^2}\right] - 2D\Psi'\Psi'_y - \frac{2\Theta'\Theta'_y}{\Theta} + \frac{\Theta'^2\Theta_y}{\Theta^2} \tag{8}$$

$$\phi'_y = \left[A\eta + \frac{\eta\Psi}{2} - \phi\right]\left[\frac{\Theta_y\Theta'}{\Theta^2} - \frac{\Theta_y}{\Theta}\right] - \left[\frac{\eta\Psi_y}{2} - \phi_y\right]\frac{\Theta'}{\Theta} + \left[\frac{\eta}{2}\right]\Psi'_y \tag{9}$$

The boundary condition for Equations 7, 8 and 9 are the following:

$$\Psi_y = \Theta_y = \phi_y = \Psi'_y = 0 \quad \text{and} \quad \Theta'_y = 1 \quad \text{at} \quad \eta = 0$$

Equations 1 through 9 make up a set of coupled O. D. E.'s which can be integrated from $\eta = 0$ to $\eta = \eta$ edge.

The actual integration is carried out using a Kutta-Merson method. The algorithm automatically adjusts step size in η to maintain the absolute truncation error below a specified amount. The algorithm also adjusts the step so that the results of the integration can be printed out at a number of specified locations. The basic computational method is shown in Figure 4.

6. RESULTS. Nachtsheim and Swigert[6] have shown that the range of initial X and Y for which the problem will converge decreases with increasing values of η edge. They have also shown that the accuracy of the solution increases with increasing values of η edge. Therefore it is advisable to start the calculation with small values of η edge and a relatively large acceptable error, then uses the results to move to larger values of η edge and smaller acceptable errors.

A set of four runs were made using different values of η edge. In each run, the step size in η was $\Delta\eta = 0.3$, the truncation error limit was 1.0×10^{-4} and a limit on the sum of the squares of the estimated errors δ_1, δ_2, δ_3 and δ_4 was 1.0×10^{-6}.

The first run was a five step integration with η edge = 1.5. The problem converged rapidly, but failed to match the outer flow to within the error limit. The run was terminated after fifteen iterations. The results of the first six iterations of the five step integration are shown in Figures 5 and 6. The second run was a ten step integration with η edge = 3.0. The problem converged to an acceptable solution in six iterations. The results of the sixth

PROPERTY OF U.S. ARMY
STINFO BRANCH
BRL, APG, MD. 21005

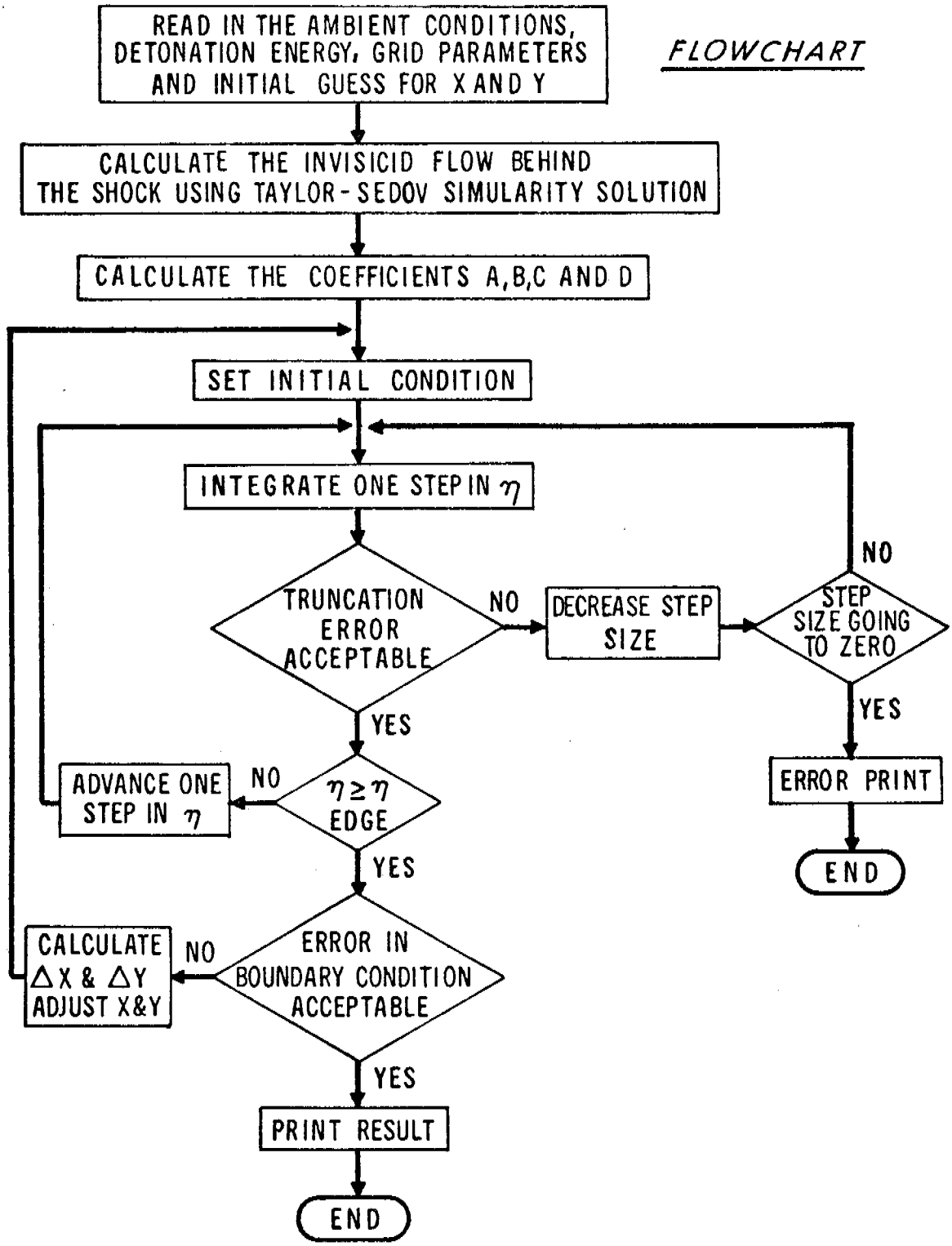


Figure 4. Flowchart of Method of Numerical Solution

iteration are shown in Figures 7 and 8. The third run was a fifteen step integration with η edge = 4.5. The fifteen step integration also converged to an acceptable solution in six iterations. The solution from the fifteen step integration was more accurate than the ten step. The results of the six iterations in the fifteen step run are shown in Figures 9 and 10. A one hundred step run was made for comparison with the other runs. The one hundred step run was also found to converge in six iterations. The results of the sixth iteration of all four runs are shown in Figures 11 and 12.

7. CONCLUSION. The results of the four computer runs indicate that it is possible to find an asymptotic solution to the unsteady, compressible boundary layer equations at the shock/surface interface using the coordinate transforms developed in this study. Further they indicate that a reasonably accurate solution can be achieved using as few as ten integration steps.

The radial velocity (Ψ) profile shown in Figure 11 has the same form as the radial velocity (f') given in References 2 and 3. The temperature (Θ) profile shown in Figure 12 has the same form as the temperature (g) profiles in the same references. A direct one to one comparison with the profiles in the references is not appropriate because of differences in coordinate transforms, however, the similarity does indicate all three methods are yielding the same type of solution in physical space. All three indicate at near the interface, both the velocity and temperature approach their outer flow values asymptotically with no overshoot within the boundary layer.

The asymptotic solution developed in this study is now available for use as a boundary condition in a finite difference solution for the entire boundary layer flow within a hemispherical blast wave. The finite difference scheme will be based on the three transformed, basic governing equations already presented. Because the scheme will not contain a similarity assumption it should be possible to extend the solution in the lower pressure region. The complete solution in the region should provide more accurate estimates of near surface gas velocities, dust pick-up and dust transport, which will in turn allow more accurate estimates of the loading on ground targets during nuclear attacks.

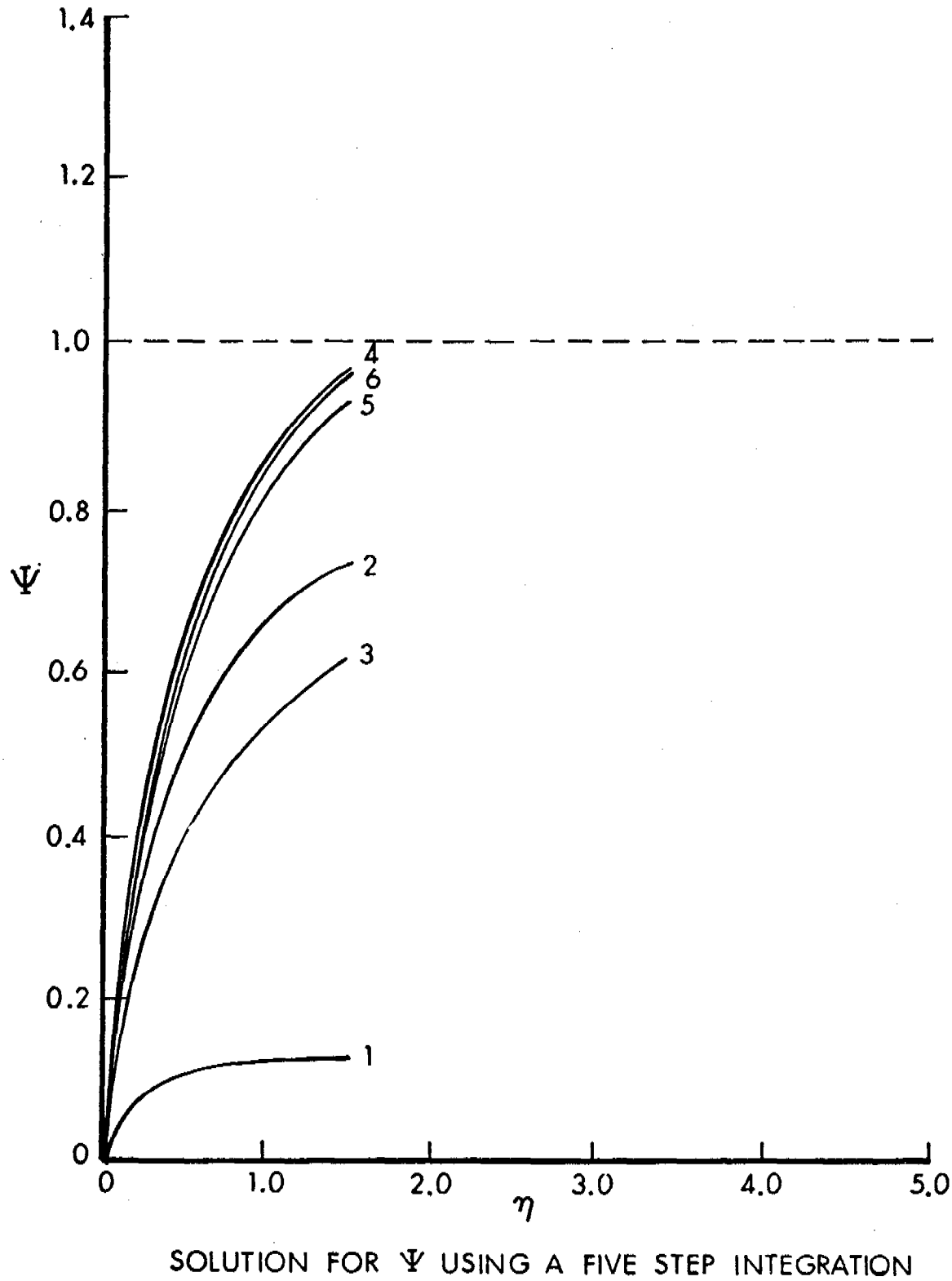


Figure 5. Solution for Transformed Radial Velocity using a Five Step Integration.

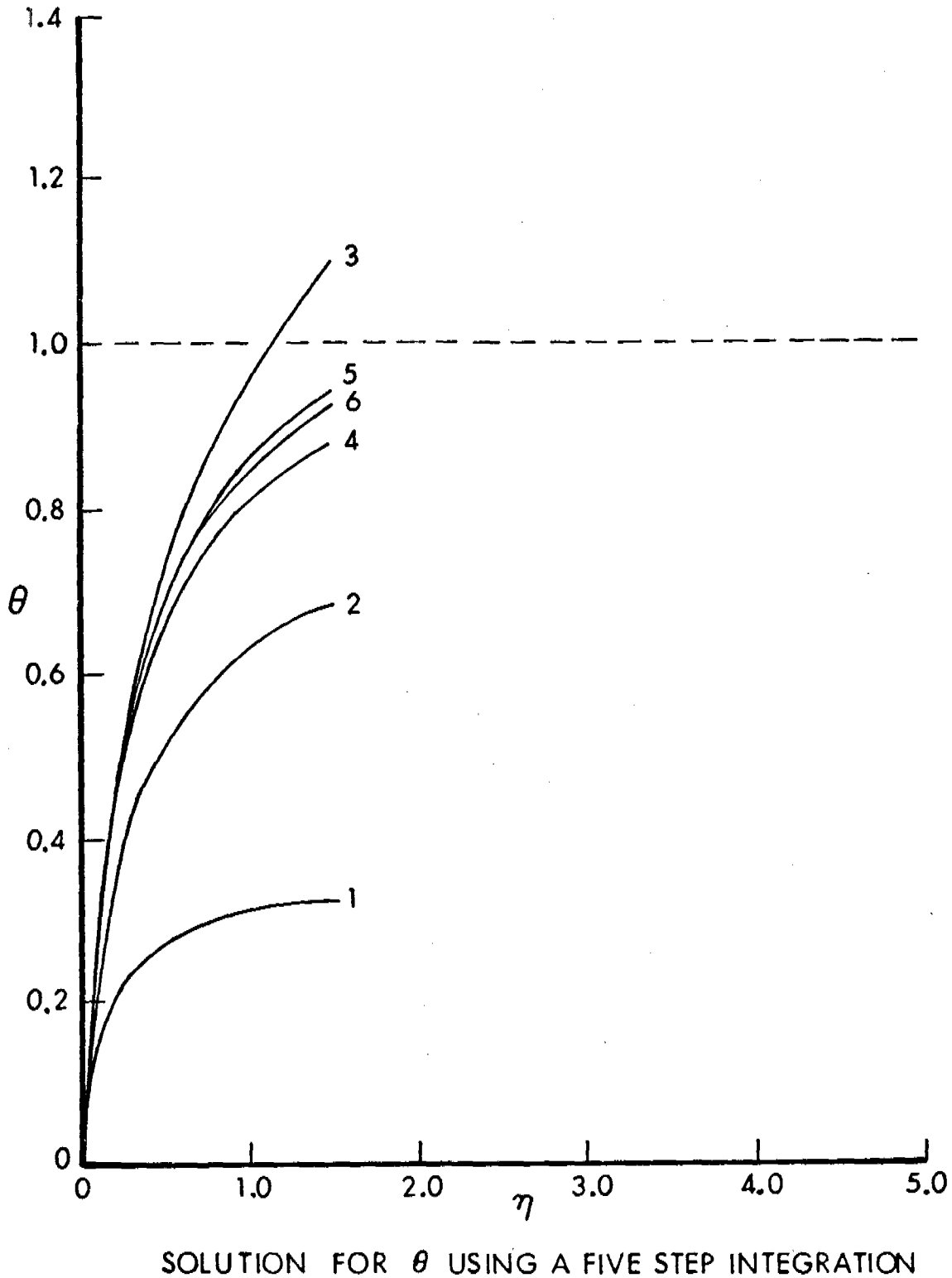


Figure 6. Solution for Transformed Temperature using a Five Step Integration.

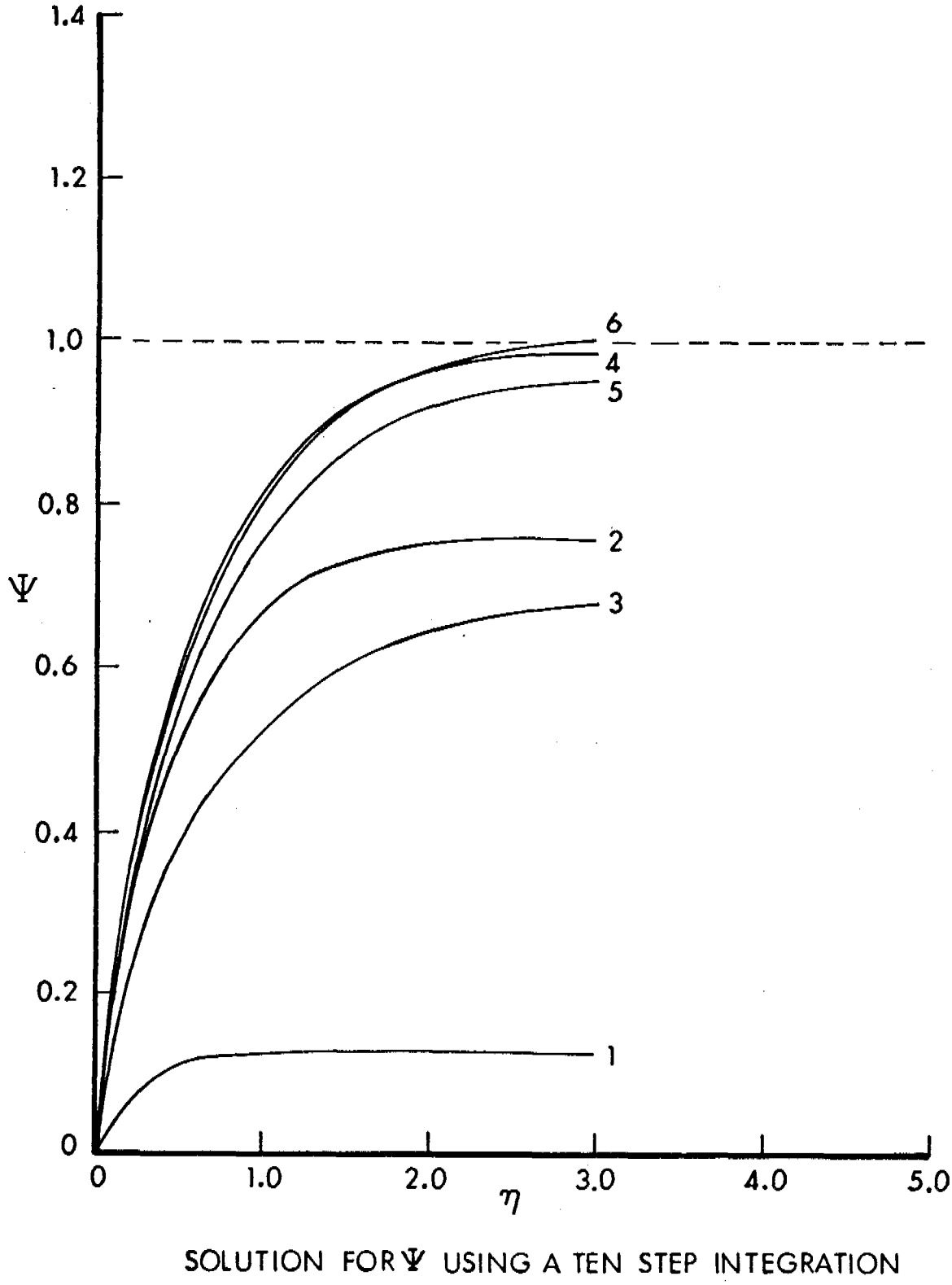


Figure 7. Solution for Transformed Radial Velocity using a Ten Step Integration.

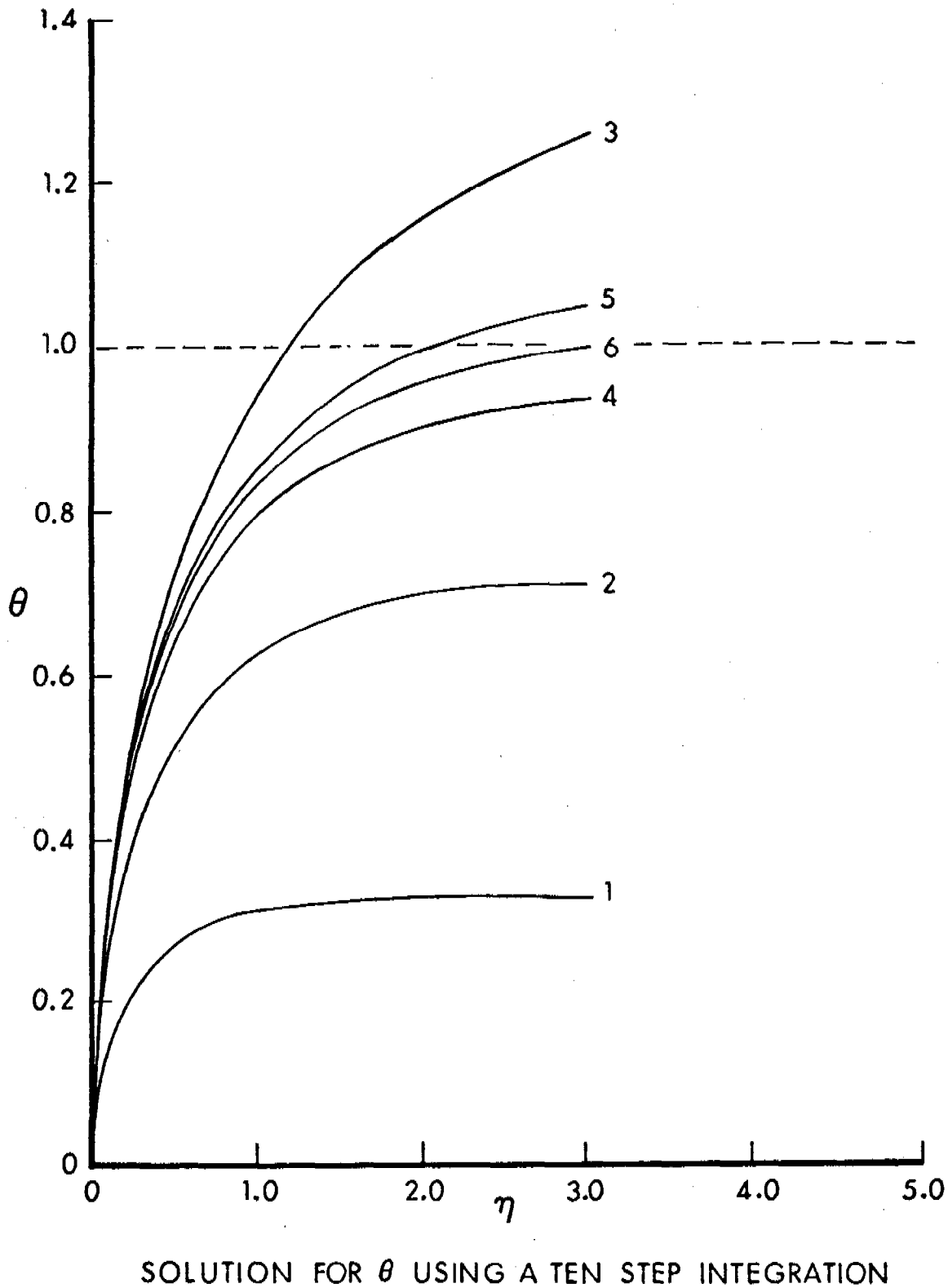


Figure 8. Solution for Transformed Temperature using a Ten Step Integration.

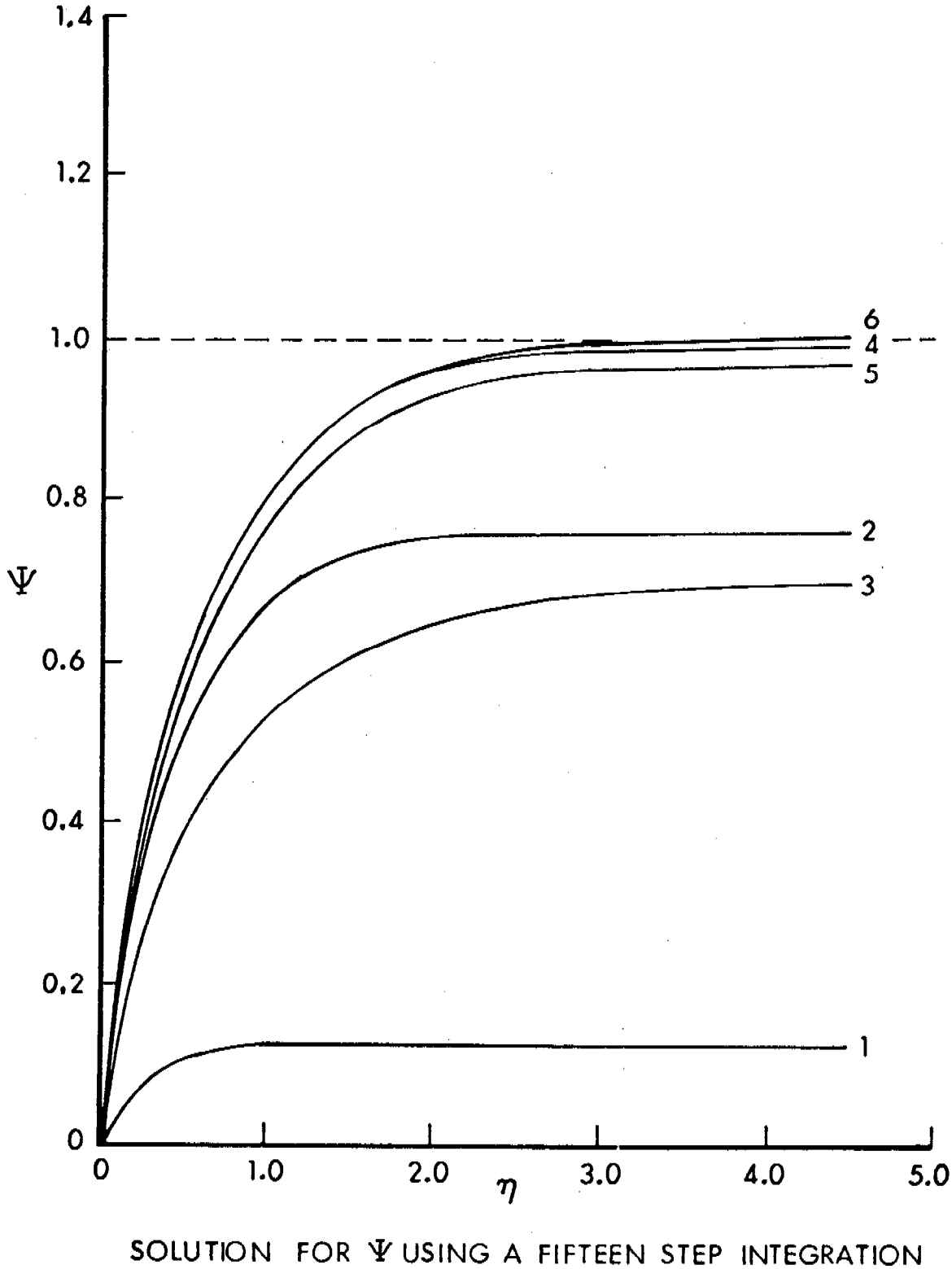


Figure 9. Solution for Transformed Radial Velocity using a Fifteen Step Integration.

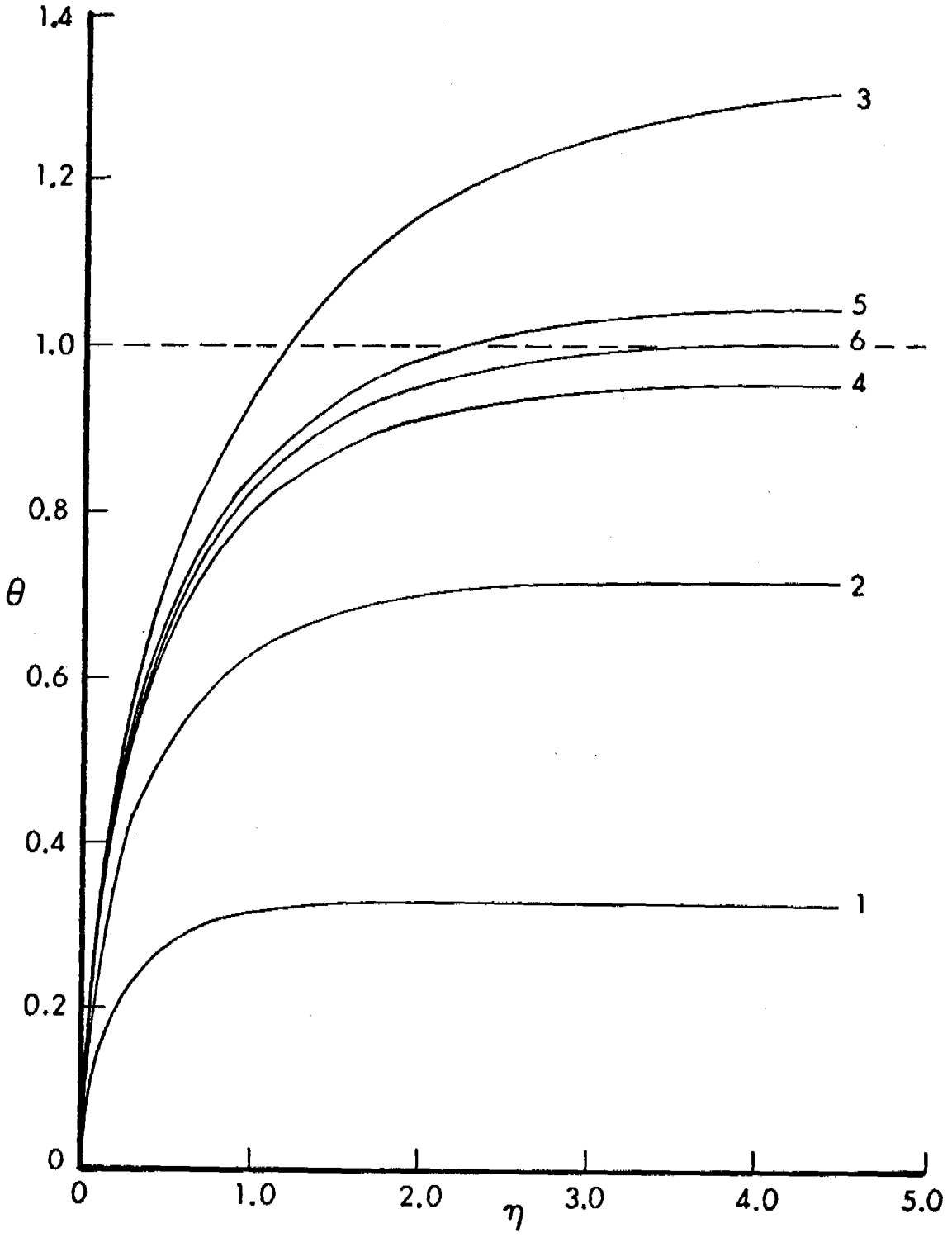


Figure 10. Solution for Transformed Temperature using a Fifteen Step Integration.

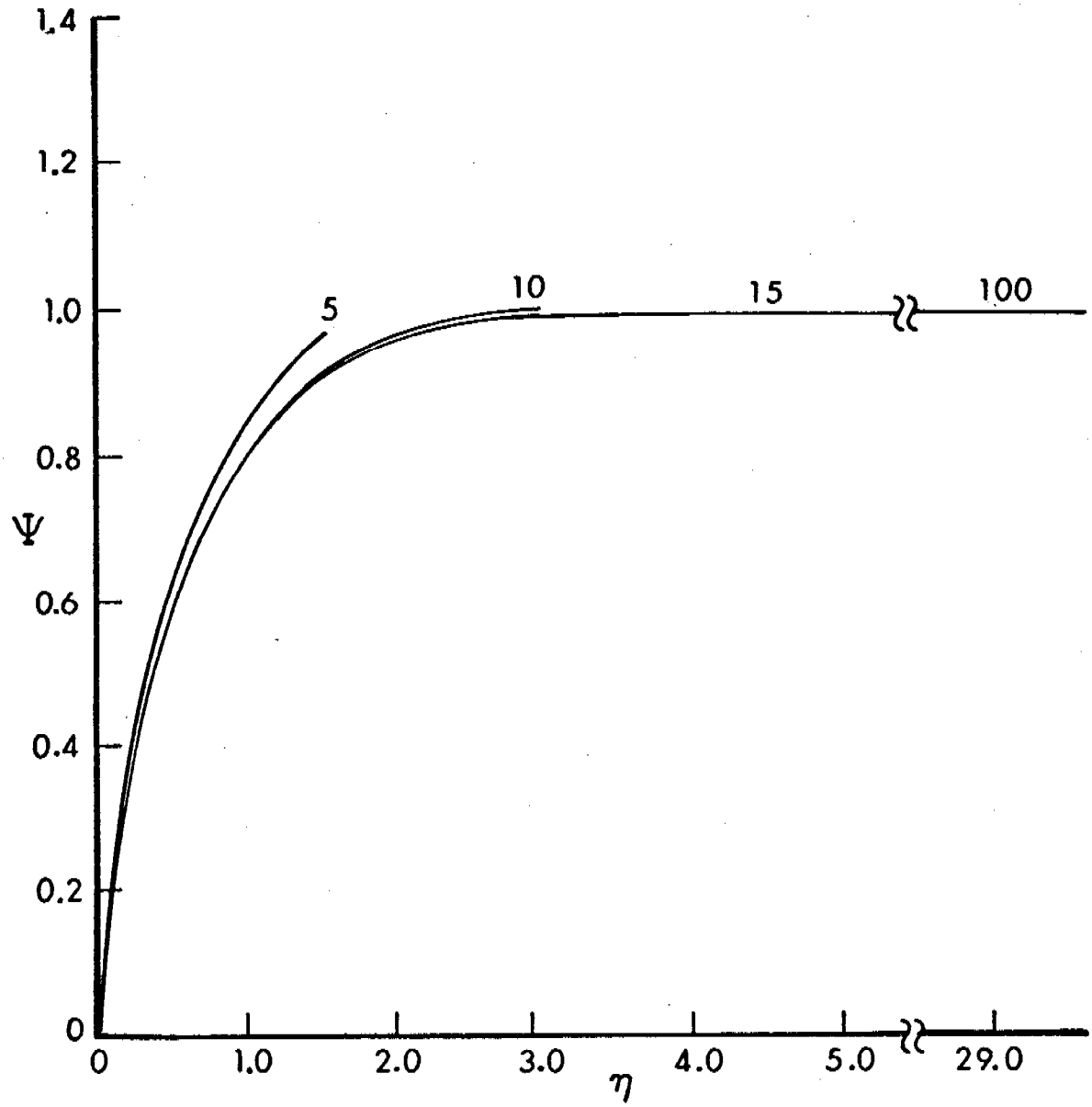


Figure 11. Sixth Iteration in the Solution for the Transformed Radial Velocity.

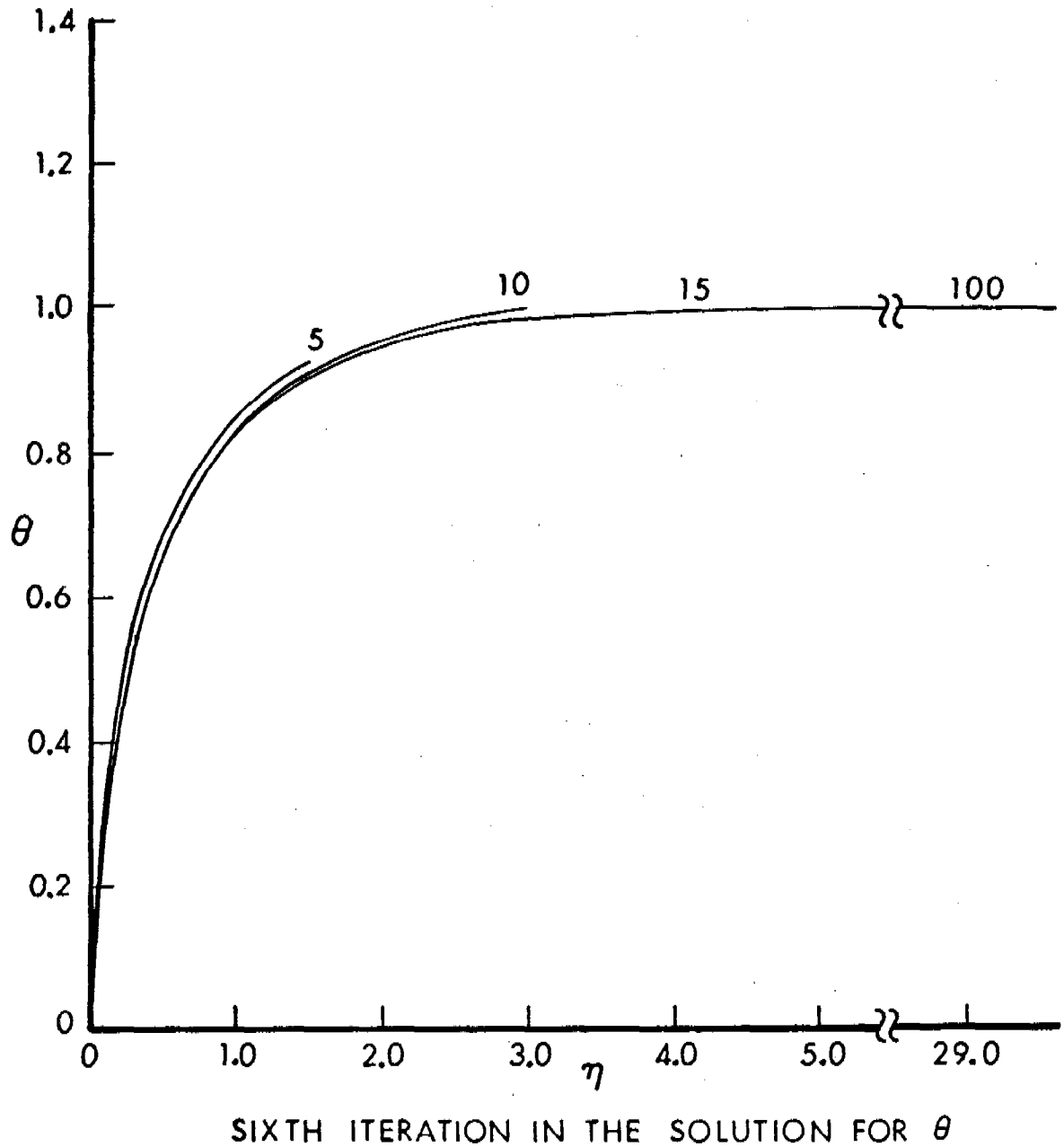


Figure 12. Sixth Iteration in the Solution for the Transformed Temperature.

REFERENCES

1. L. I. Sedov, Simularity and Dimensional Methods in Mechanics, Academic Press, New York, 1959.

2. D. R. Crawford, V. Quan and J. T. Ohnergen, "Blast Wave Turbulent Boundary Layer", DNA 2768F, TRW 18843-6001-RO-00, February 1972.

3. S. W. Liu and H. Mirels, "Numerical Solution for Unsteady Laminar Boundary Layer Behind Blast Waves", The Physics of Fluids, Vol. 23, No. 4, April 80.

4. H. Schlichting, Boundary-Layer Theory, Mc Graw-Hill Book Company, 6th Edition, 1968.

5. P. R. Nachtsheim and P. Swigert, "Satisfaction of Asymptotic Boundary Condition in Numerical Solution of System of Nonlinear Equation of Boundary Layer Type", NASA Technical Note, NASA TN D-3004.

A SIMPLE MODEL FOR PREDICTING THE BLAST LOADS ON BOX-LIKE STRUCTURES

Klaus O. Opalka
Ballistic Research Laboratory
U.S. Army Armament Research and Development Command
Aberdeen Proving Ground, Maryland 21005

ABSTRACT. BLOP is a computer code for predicting the air-blast loads on targets which can be described approximately by a series of rectangular parallelepipeds. The code has been developed at BRL to quickly obtain a prediction of the average loads on the surfaces of a target encountering a blast wave, without having to resort to a hydrocode computation. The empirical model employed in the BLOP code is based on experimental and analytical work done predominantly at BRL. The results compare favorably with available data.

1. INTRODUCTION. This paper describes a simple model for predicting the blast loading on box-like structures, and discusses the results which are compared with available experimental data.

The Blast-Load Prediction (BLOP) code was developed to quickly and inexpensively estimate the blast loading on structures. The only prediction method available prior to the development of this code, is the standard prediction technique[1] which relies on the use of tables and of rules of thumb to estimate the average pressure load on a target surface.

Another method available is the hydrocode. By the use of finite-difference techniques it is possible to describe the flow field around a target in detail. But hydrocodes require considerable set-up time and are expensive to operate. Often a quick estimate of blast loads is needed in engineering and planning situations, e.g. for a proposed high-explosive (H.E.) field test where neither the time, nor the funds, nor the manpower are available to carry out a complex hydrocode, or tedious hand computation. The BLOP code was developed in response to this need.

The BLOP model employs analytical and empirical procedures, the latter of which are based on experimental work done previously at the BRL. A one-dimensional flow scheme is employed assuming head-on collision of the shock with the target. The Rankine-Hugoniot relations are used to define the flow conditions behind the shock. Three different flow situations are considered: (1) The shock-tube situation is characterized by a step shock. (2) The H.E. field test situation is characterized by an exponentially decaying blast wave. (3) The simulated blast-wave situation behind the exit of a shock tube is characterized by a generalized form of the modified Friedlander equation.

The average overpressure functions for the front and back faces of targets are empirical functions developed at BRL. The roof and side faces are treated according to the standard prediction technique. An attempt is made with the BLOP model to apply the simple case of the rectangular parallelepiped to a complex structure as e.g. a truck, or a helicopter tailboom, subdividing it into a convenient number of sections each of which is represented by a rectangular box. Closed, partially open, and open-frame structures are considered.

The code was written in FORTRAN IV for use on the UNIVAC 1108 at the Aberdeen Proving Ground, Edgewood, Maryland. The code contains detailed user instructions and numerous other comments. It will be published as an appendix to a BRL report, which describes the physical phenomena of blast waves and the target loading procedures used in the BLOP model in greater detail. Here, they are reviewed briefly together with a discussion of the results.

2. PHYSICAL PHENOMENA. This chapter offers a brief description of blast-wave phenomena in as much detail as is necessary to introduce and explain the terminology used in this paper.

2.1 Blast Waves. When a high-energy weapon is detonated at some height above the ground, the pressure waves emanating from the center of explosion rapidly form a spherical blast wave, characterized by an abrupt increase of the air pressure across the shock front. Figure 1 illustrates the progress of the blast wave along the ground surface. As the incident blast wave expands, the shock strength at the front decreases. Where the shock front contacts the ground surface, it is reflected.

The reflected shock front moves back into the air already compressed and heated by the incident shock. Behind the reflected shock, a new blast wave forms with properties different from those of the incident blast wave. Because of these conditions, the reflected shock front moves faster than the incident shock front, gradually catches up with it, and combines with it into a reinforced shock front at some distance from ground zero (i.e. the reference point directly under the center of explosion).

This new shock front is called the Mach stem of the blast wave. The Mach stem stands essentially normal, and moves parallel to the ground surface. This phenomenon considerably simplifies the treatment of blast loading of structures located in the region of Mach reflection.

An explosion on the surface results in somewhat different air-blast phenomena. The blast wave forms a hemispherical, reflected shock front over the surface. There is no region of regular reflection in this case, and targets on the ground are subjected to air-blast conditions similar to those in the Mach-reflection region even close to ground zero. The shock front may be assumed to be vertical for most purposes. The wind behind the shock front and near the surface blows horizontally for all practical purposes.

A comprehensive description of blast waves and their effects on man and equipment can be found in Reference 1. Reference 2 contains a comprehensive collection of analytical and experimental studies on the subject of air-blast technology.

2.2 Pressure History. Figure 2 gives a typical overpressure history as it may be recorded at some spatial location in the Mach-reflection region. The origin of the overpressure and time axes is set at the time of explosion, t_o. When the shock front of the blast wave arrives at time t_a, perhaps a few seconds after the explosion, the pressure increases suddenly. The peak value, p_{so}, is called the peak shock overpressure. The temperature and the density of the air suddenly increase, also.

Behind the shock front, the overpressure quickly drops to about one-half of its peak value, and falling steadily returns to zero at time t_+. This time is called the positive-phase duration because of the positive overpressure that prevails. The positive phase is followed by the negative phase during which the overpressure drops below atmospheric pressure. Subsequently it returns to ambient conditions.

During the positive phase, strong winds follow the shock front giving rise to a positive dynamic pressure, q_{so}. This dynamic pressure decays with the static overpressure but at a slower rate, and the wind continues to blow for a short while beyond the positive-phase duration. This means that the positive phase of the dynamic pressure lasts a little longer than the positive phase of the static pressure.

During the negative phase, the wind of the dynamic pressure reverses its direction and blows toward the center of the explosion. Some damage may be expected during the negative phase of the blast wave but it is during the positive phase that most of the damage to structures occurs. Therefore, loading and response studies are restricted to the positive phase of the blast wave.

3. COMPUTATIONAL MODEL. To keep the computational model simple, the following assumptions were made.

(1) The free-field flow is essentially one-dimensional. This entails a shock front which can be considered planar and perpendicular to the direction of propagation.

(2) The shock front will hit the model head on, i.e. the velocity vector will stand normal to the front face. The shock front and the front face of the structure are thus parallel planes.

(3) Empirical equations will be used as pressure-decay functions and as average-load functions for the surfaces.

(4) Target structures can be modelled by a series of rectangular parallelepipeds. This assumption is less restrictive than it may seem at first glance.

The following load cases can be adequately described under these assumptions.

(1) A step shock can simulate the test conditions in a shock tube.

(2) An exponentially decaying wave can simulate the free field conditions in the Mach-reflection region.

(3) A decaying wave as would be generated at the exit of a shock tube can simulate the special test conditions in the field behind the shock tube.

3.1 Shock Relations. The assumption of one-dimensional flow and the restriction to normal shock incidence allow the use of simple analytical equations like the Rankine-Hugoniot relations to define the conditions behind the shock front and behind the reflected shock. The shock-front velocity is defined by

$$U_s = a_o \left(1 + \frac{\gamma+1}{2\gamma} \xi \right)^{\frac{1}{2}} \tag{1}$$

where a_o is the sound velocity in ambient air, and γ is the ratio of specific heats. The shock strength is defined by

$$\xi = \frac{P_1 - P_o}{P_o} , \tag{2}$$

where P_o is the ambient, atmospheric pressure and P_1 the absolute pressure behind the shock-front.

The wind velocity behind the shock-front is given by

$$U_p = a_o \frac{\xi}{\gamma \left(1 + \frac{\gamma+1}{2\gamma} \xi \right)^{\frac{1}{2}}} , \tag{3}$$

and the dynamic pressure is

$$q_{so} = P_o \frac{\xi^2}{2\gamma + (\gamma-1)\xi} \tag{4}$$

For the definitions of other shock relations, the reader is referred to the literature.[3]

3.2 Pressure-Decay Functions. The decay of the blast-wave overpressure at a fixed target location is modelled by the Friedlander equation

$$p(\tau) = p_{so} (1 - \tau) e^{-c\tau}, \tag{5}$$

where $p_{so} = P_1 - P_o$ is the shock overpressure, c is a time coefficient, and τ is the non-dimensional time, defined by

$$0 \leqslant \tau = \frac{t - t_a}{t_+} \leqslant 1. \tag{6}$$

The time coefficient, c, is a function of the peak shock overpressure and time, and the model assumes a linear variation of c with τ from an initial maximum value to a final minimum value. These values are empirical and form part of the required input.

The dynamic pressure decays in a similar fashion as the static pressure and analogous to the Friedlander equation

$$q(\tau) = q_{so} (1 - \tau_q)^2 e^{-2c\tau_q}, \tag{7}$$

where the non-dimensional time is defined by

$$0 \leq \tau_q = \frac{t - t_a}{t_{+_q}} \leq 1. \tag{8}$$

The peak shock overpressure, p_{so}, the time of shock arrival, t_a, and the positive-phase durations, t_+ for static, and t_{+_q} for dynamic pressure, are tabulated functions of the range from ground zero and are part of the required input.

3.3 Average Loading Functions. The loading model used in the BLOP code is based on the Standard Prediction Technique as described in Reference 1. However, empirical loading functions, developed at BRL by Ethridge,[4] were used instead of those functions used in the Standard Prediction Technique.

The basic loading function for the front face chosen by Ethridge is

$$p_{FR}(\tau) = p_{stag}(\tau) \left[1 + \left(\frac{p_r}{p_{stag,s}} - 1 \right) e^{-A(N_r \tau)^B} \right], \tag{9}$$

where p_{FR} is the average overpressure on the front face at time τ, p_{stag} is the average stagnation overpressure on the front face at time τ, p_r is the normally reflected shock overpressure, and $p_{stag,s}$ is the stagnation overpressure immediately behind the shock front. $A(\xi)$, N_r, and $B(\xi,\tau)$ are empirical functions determined by fitting Equation (9) to experimental data.

The basic loading function for the back face chosen by Ethridge is

$$p_{BK}(\tau) = E (1 - e^G) p(\tau_b), \tag{10}$$

where τ_b is non-dimensional time based on the arrival of the shock front at the back face:

$$0 \leq \tau_b = \tau - \frac{\ell}{U_s t_+} \leq 1, \tag{11}$$

with ℓ = length of target in flow direction

U_s = shock-front velocity, given by Equation (1), and

t_+ = positive-phase duration.

$E(\xi)$ and $G(\xi,\tau_b)$ are empirical functions determined by fitting Equation (10) to experimental data.

The average-pressure functions used for the sides and the top of the target are those given by the Standard Prediction Technique. They are considered to give an adequate engineering estimate. Since one cannot predict the direction from which a blast wave may approach the target, all surfaces must be examined and designed for a head-on collision with the shock front.

3.4 Modelling of Targets. Existing methods for calculating the airblast loading on targets cover only a few, idealized, simple shapes. These are (A) rectangular parallelepipeds, and (B) cylinders. The first group can be further divided into

(1) Closed Structures: Structures with a flat roof and bearing walls having either no, or only small openings (amounting to less than 5% of the surface area) fall into this category, e.g. shelters.

(2) Partially Open Structures: Structures which have large openings, or window areas in excess of 5% of the wall area are classified as partially open structures, e.g. houses. Because the blast wave can enter these structures, the net loading of any wall of the structure is the difference between the interior and the exterior load.

(3) Open Frame Structures are those which have a supporting steel, or concrete frame and nonbearing walls, as e.g. modern office buildings or truss structures. The more significant contribution to the loading of these structures is made by the wind behind the shock front which creates a considerable drag loading.

An attempt is made with the BLOP code to apply the simple load case of a rectangular parallelepiped to targets which may be approximately described as an assembly of several rectangular boxes. Figure 3 illustrates the application of this concept to a helicopter tailboom. The target is subdivided into a convenient number of boxes rigidly attached to each other such that they together resemble the shape of the target. The purpose of this subdivision is to accommodate variations of the incident shock overpressure, time of shock arrival, and positive-phase duration along the major target axis.

4. DISCUSSION OF RESULTS. To evaluate the BLOP code, let us compare some blast-loading predictions with available experimental data and a hydrocode computation.

4.1 Shock-Tube Test. The empirical equations used in the BLOP code to determine the average pressure on the front and back faces of a target are based on data obtained from an experimental investigation of diffraction blast

loading on two- and three-dimensional blocks.[5] The data shown in Figure 4 are representative of Taylor's test results and bracket the pressure range for which the empirical equations were derived.

Figure 4a shows the comparison of the BLOP computation with the 34.5 kPa (5 psi) test results. The agreement is good, even though the computation cannot simulate the drop below the stagnation pressure on the front face which the experimental data show. A slight difference between experiment and computation stems from the fact that the shock overpressure in the test did not equal the nominal value. Figure 4b shows the same comparison at the 138 kPa (20 psi) level. Here, the experimental data follow the prediction very closely on both the front and back faces.

4.2 Helicopter Tailboom Test. Open-ended shock tubes are blast-wave generators. It was found that the BRL shock tubes may be used to generate blast waves with peak shock overpressures from 2-20 kPa (0.3-3 psi) in the field behind the shock tube exit. Targets too big to fit into the shock tube can be mounted some distance beyond the exit, and off-axis to avoid the gas jet. This technique was successfully used at BRL to investigate the dynamic response to blast loading of a helicopter tailboom[6] using the 2.4 m (8 ft) shock tube as a blast-wave generator. Figure 5 illustrates the test set-up.

The blast-field parameters needed for input in the BLOP code were determined from a survey of the blast field behind the shock-tube exit. The computed blast-wave history is compared with the experimentally measured overpressure history for a 13.4 kPa (1.9 psi) shock in Figure 6. The shock-tube generated blast wave does not have the typical, classical shape of a high-explosive blast wave shown in Figure 2. After an initial exponential decay, the overpressure reaches a plateau, the height of which appears to depend on the distance from the shock-tube exit. In the final phase of the blast wave the overpressure decays rapidly. This decay, limiting the positive-phase duration of the simulated blast wave, apparently is caused by the action of rarefaction waves at the shock-tube exit which quickly equalize the overpressure in the exiting gas jet.

In the experiment, overpressures were measured along the symmetry line on the front and back surfaces of the helicopter tailboom. These data are compared with the predicted average overpressure on the front face (Figure 7a) and on the back face (Figure 7b) of the tailboom resulting from the 13.4 kPa blast wave described in Figure 6. The predicted average front-face load (Figure 7a) is too high, particularly during the diffraction loading. This overestimation is most likely due to the modelling of the tailboom into box-like sections with plane surfaces and sharp corners while the real tailboom has curved surfaces with rounded corners that accellerate the pressure relief from the sides.

The experimentally measured pressure rise on the back face (Figure 7b) is slightly steeper, and the peak value of the overpressure higher than the predicted load curve indicates. These differences are consistent with those observed on the front face and are also due to the modelling of the tailboom. Two other physical phenomena, vortex formation on, and dynamic response of the tailboom may be influencing the experimental curves. But on the whole, the predicted curve matches the experimental data well.

4.3 Equipment Shelter on MISER'S BLUFF. The S-280 Equipment Shelter was subjected to airblast during the MISER'S BLUFF test series.[7] In Figure 8 the free-field, blast-wave history recorded during the test is compared with the computed prediction. The comparison shows (a) that the Friedlander equation very adequately describes the pressure decay in a blast wave, and (b) that the experimental blast wave deviates in some way from the ideal blast wave.

The front- and back-face load histories of the S-280 equipment shelter recorded during MISER'S BLUFF are compared with the BLOP-code computation in Figures 9a and 9b, respectively. The prediction agrees well with the experiment during the diffraction phase, which lasts about 15 milliseconds. The discrepancy between prediction and experiment during the drag phase can be explained by the dynamic response of the shelter wall during the test. The BLOP model assumes a rigid wall. There, too, exists the possibility that air leaked into the shelter under load, increasing the inside pressure which the differential pressure gages mounted in the shelter walls used as a reference, thus decreasing the pressure difference to the outside.

4.4 HULL Code Prediction. A 3-D HULL computation was performed for an S-280 shelter model[8] exposed to a 34.5 kPa (5 psi) step shock in a shock tube, and the results are compared with the BLOP-code computation. The front- and back-face load histories are shown in Figure 10a, and the side-face load history is shown in Figure 10b.

The BLOP prediction appears to "average" the HULL data points quite well during the diffraction phase (Figure 10a), and the agreement between the results of the two codes is generally good. The "ringing" of the HULL data on the front face is typical for the hydrocode computation when artificial viscosity is not used. A computation with artificial viscosity was not available as of this writing.

On the side face (Figure 10b), the HULL code results come closer to reality because the pressure drop due to the vortex generated at the front edge is accounted for in the loading history. Recent, as yet unpublished experiments at BRL have validated this hydrocode computation.

5. CONCLUSION. From the foregoing discussion the following conclusions can be drawn.

(1) Within the limitations imposed by the model, i.e.

- a simplistic, 1-D flow scheme,
- normal shock incidence only,
- empirical, average-load functions,
- crude modelling of structures

it is possible to obtain a satisfactory estimate of the blast loading on a variety of structures and load situations without resorting to complicated numerical methods.

(2) The BLOP code provides such estimates over a reasonable shock-overpressure range (1-400 kPa) with short set-up time at minimal expense. The cost involved in running BLOP on a digital computer (e.g. UNIVAC 1108) is less than 1% of the cost of a hydrocode run, and therefore very suitable for parametric studies.

(3) The computational model is expandable to improve existing loading functions and include loading functions for other generic (e.g. axisymmetric) shapes and for oblique shock impact and reflection.

REFERENCES

1. The Effects of Nuclear Weapons, S. Glasstone, Editor, D. A. Pamphlet No. 39-3, Headquarters, Department of the Army, Washington, DC, April 1962.

2. Engineering Design Handbook Explosions in Air, Part One, W. E. Baker, Editor, AMC Pamphlet No. 706-181, Headquarters, U.S. Army Materiel Command, Alexandria, VA, July 1974.

3. Fundamentals of Gasdynamics, H. W. Emmons, Editor, Princeton University Press, 1958.

4. Ethridge, N. H., "Blast Diffraction Loading on the Front and Rear Surfaces of a Rectangular Parallelepiped", BRL-MR-2784, U.S. Armament Research and Development Command, Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland 21005, September 1977.

5. Taylor, W. J., "A Method of Predicting Blast Loads During the Diffraction Phase," in: The Shock and Vibration Bulletin. The Shock and Vibration Information Center, Naval Research Laboratory, Washington, DC, January 1972.

6. Bertrand, B. P., Quigley, E. F., "Structural Response of a Helicopter Tailboom to Combined Thermal and Blast Loading", U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland, in Proceedings II, Fifth International Symposium of Military Applications of Blast Simulation (MABS-5), Royal Swedish Fortifications Administration, S-104 50 Stockholm, Sweden, May 23-26, 1977.

7. Schuman, W. J., "C^3 Shelters and Tac Antennas", U.S. Army Ballistic Research Laboratory, in "Proceedings of the MISER'S BLUFF Phase II Results Symposium", 27-29 March 1979, Vol. II, POR 7013-2, Field Command, Defense Nuclear Agency, Kirtland AFB, New Mexico 87115, 26 September 1979.

8. Lottero, R. E., "Comparison of 3-D Hydrocode Computations for Shock Diffraction Loading on an S-280 Electrical Equipment Shelter", U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland 21005, in: Proceedings of the 1980 Army Numerical Analysis and Computers Conference, NASA-AMES Research Center, Moffett Field, CA, 20-21 February 1980.

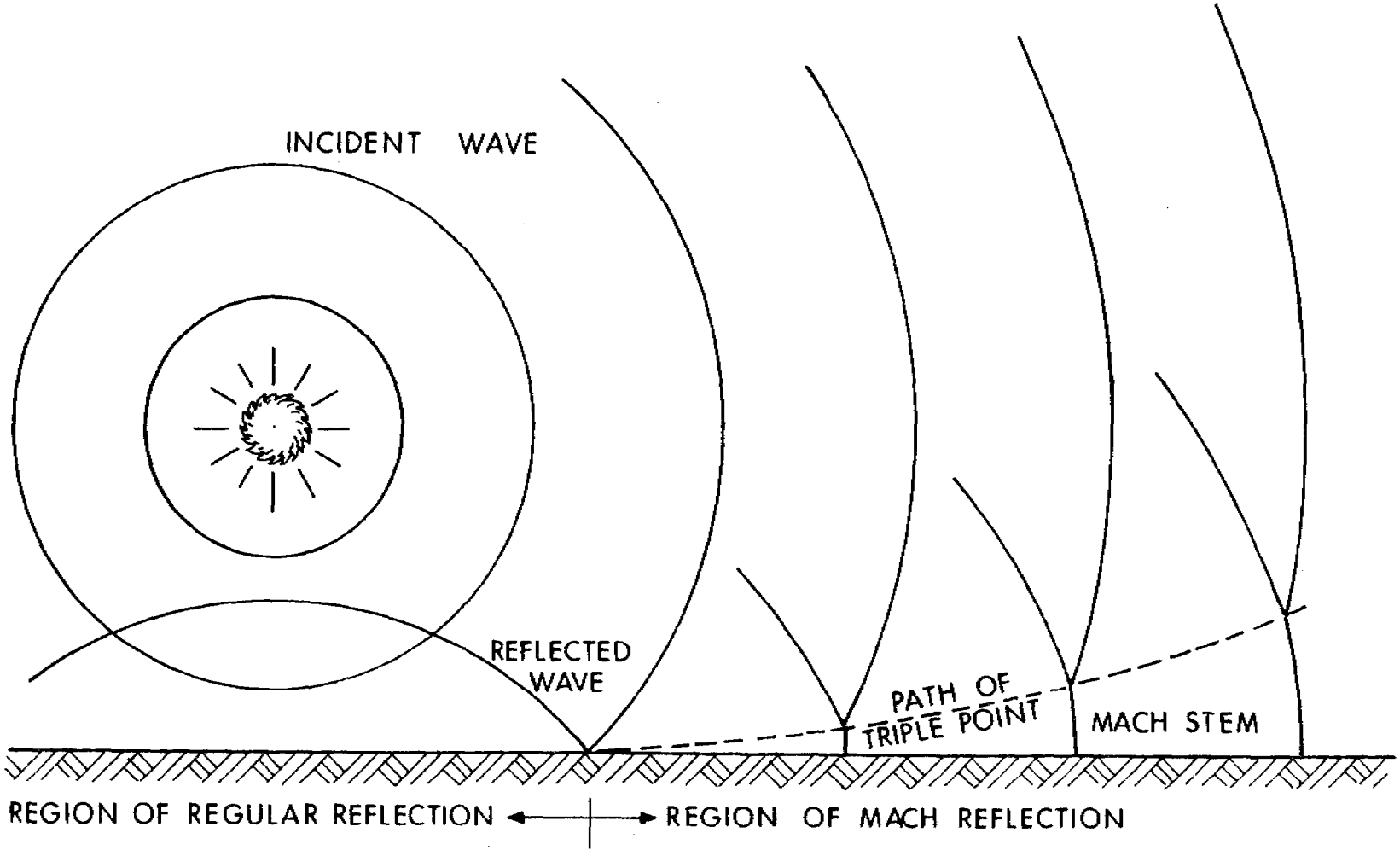


Figure 1. Progress of Incident and Reflected shock Waves Along the Ground Surface.

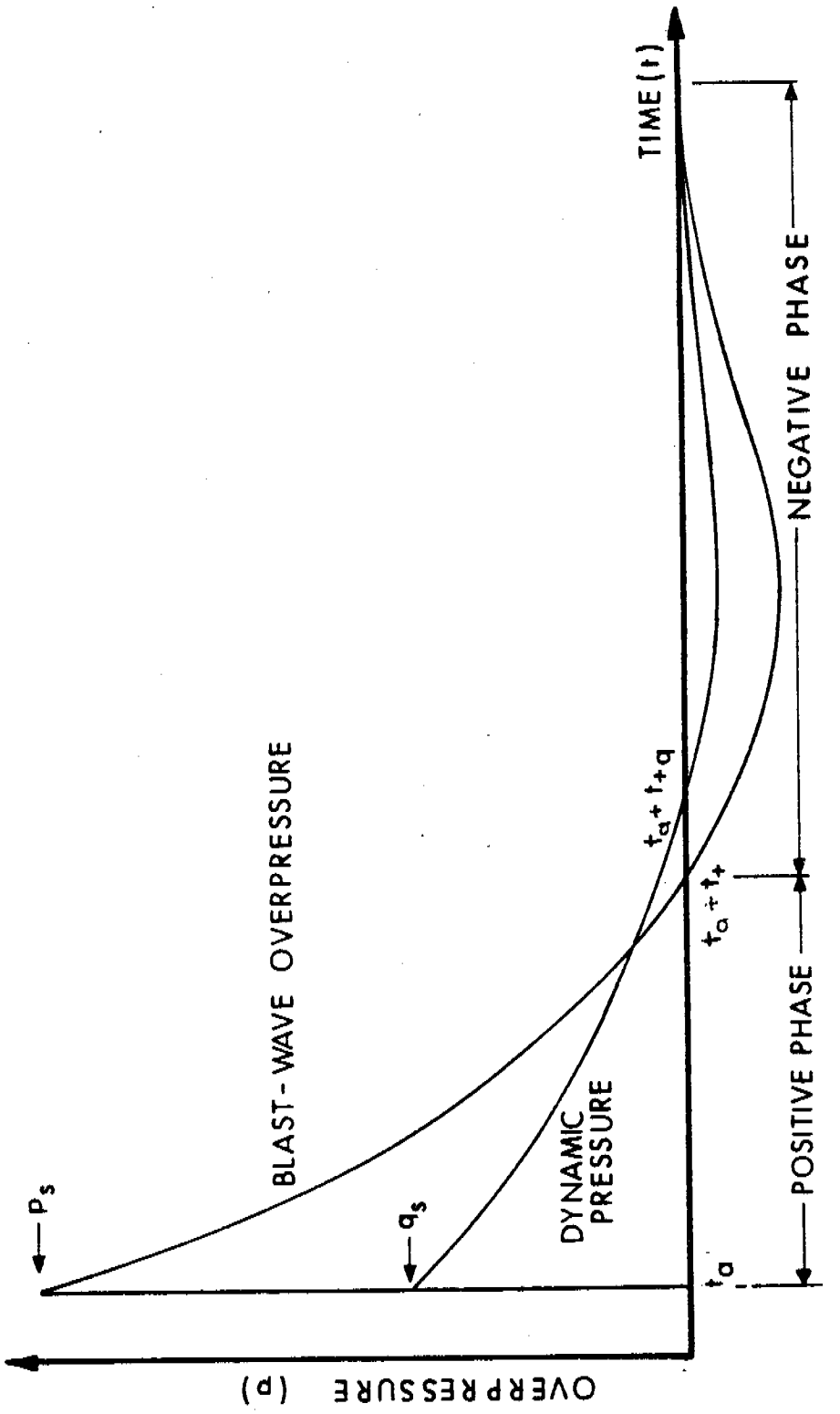


Figure 2. Pressure History of a Blast Wave.

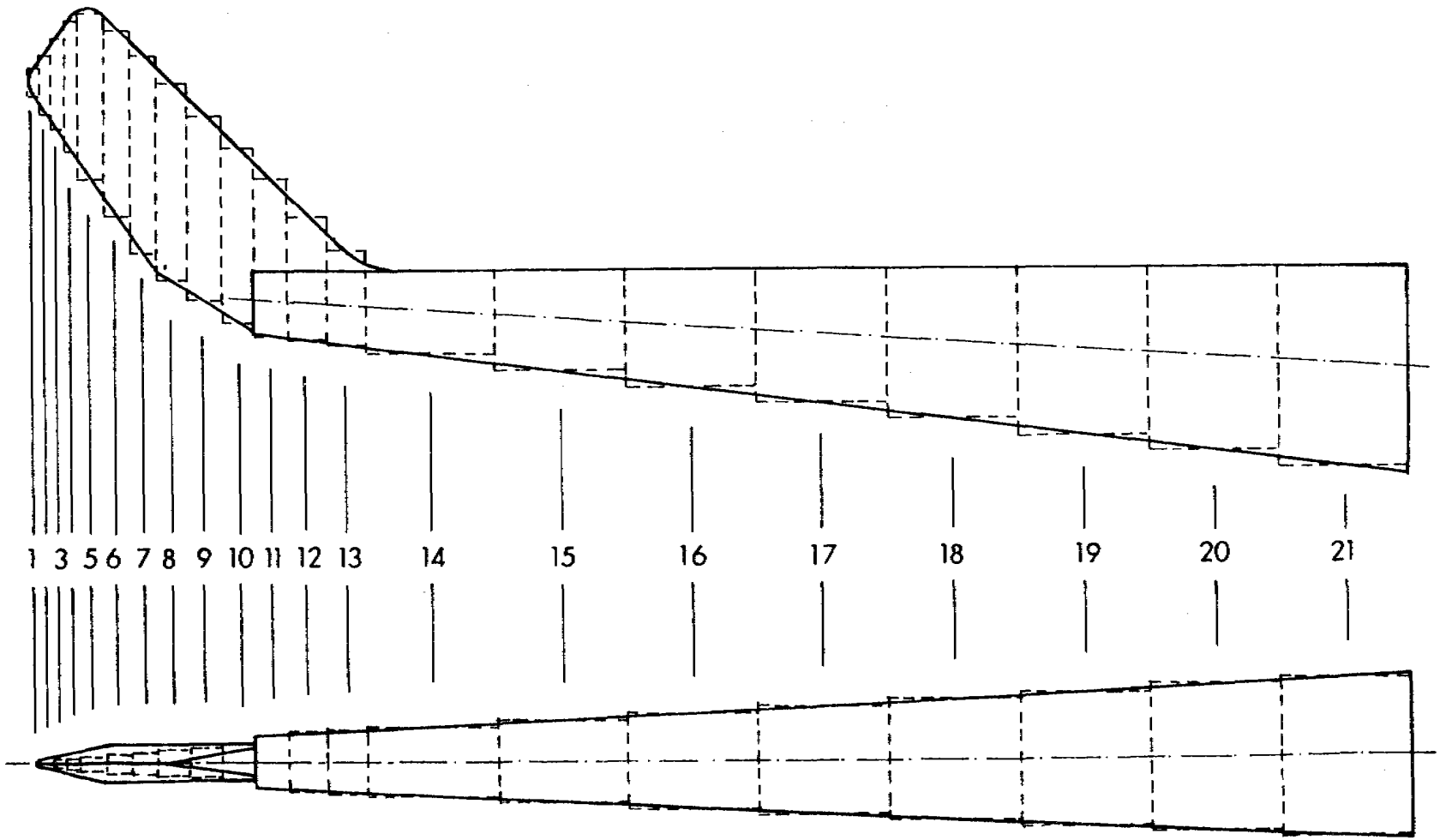
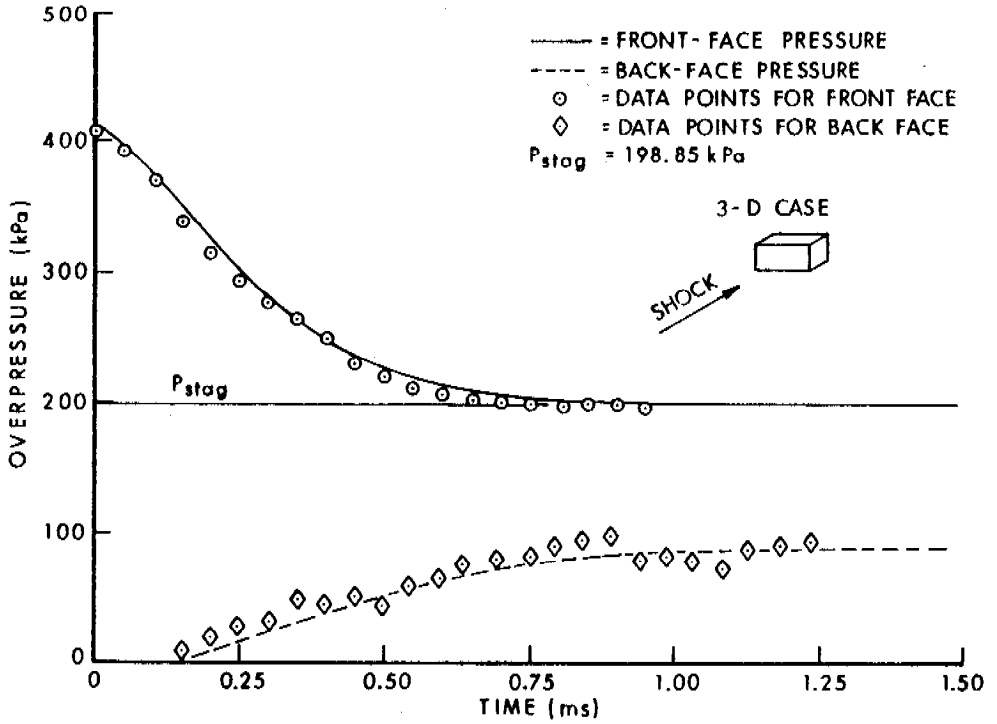
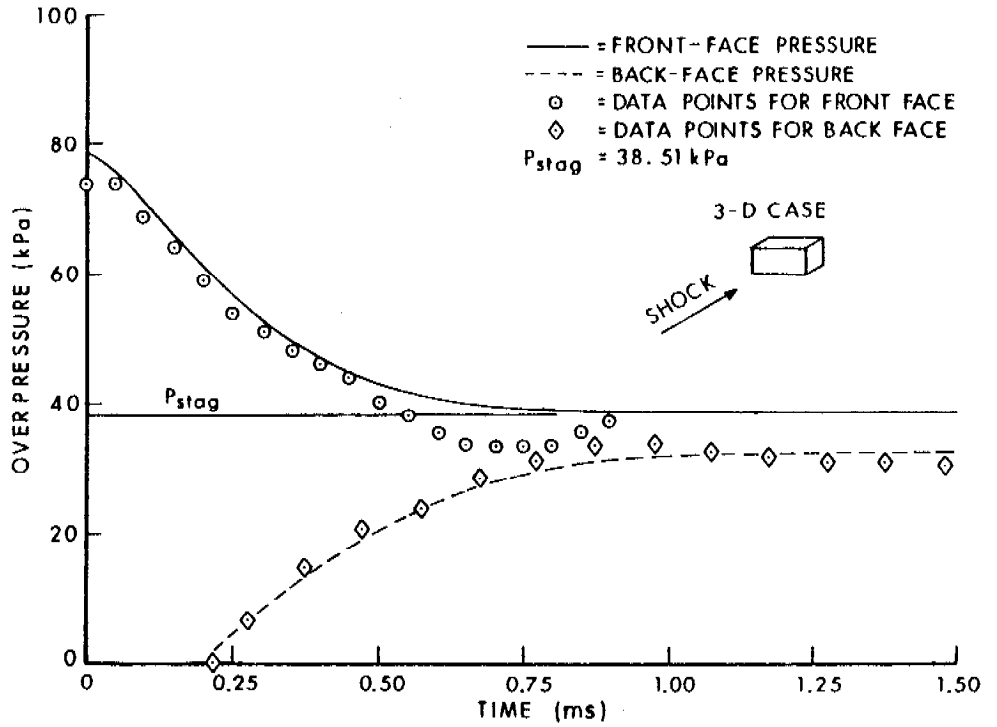


Figure 3. Modelling of a Helicopter Tailboom for Use in the BLOP Code.



b) Peak Shock Overpressure, p_s = 138 kPa (20 psi)



a) Peak Shock Overpressure, p_s = 34.5 kPa (5 psi)

Figure 4. Comparison of BLOP-Code Prediction with Shock-Tube Data.

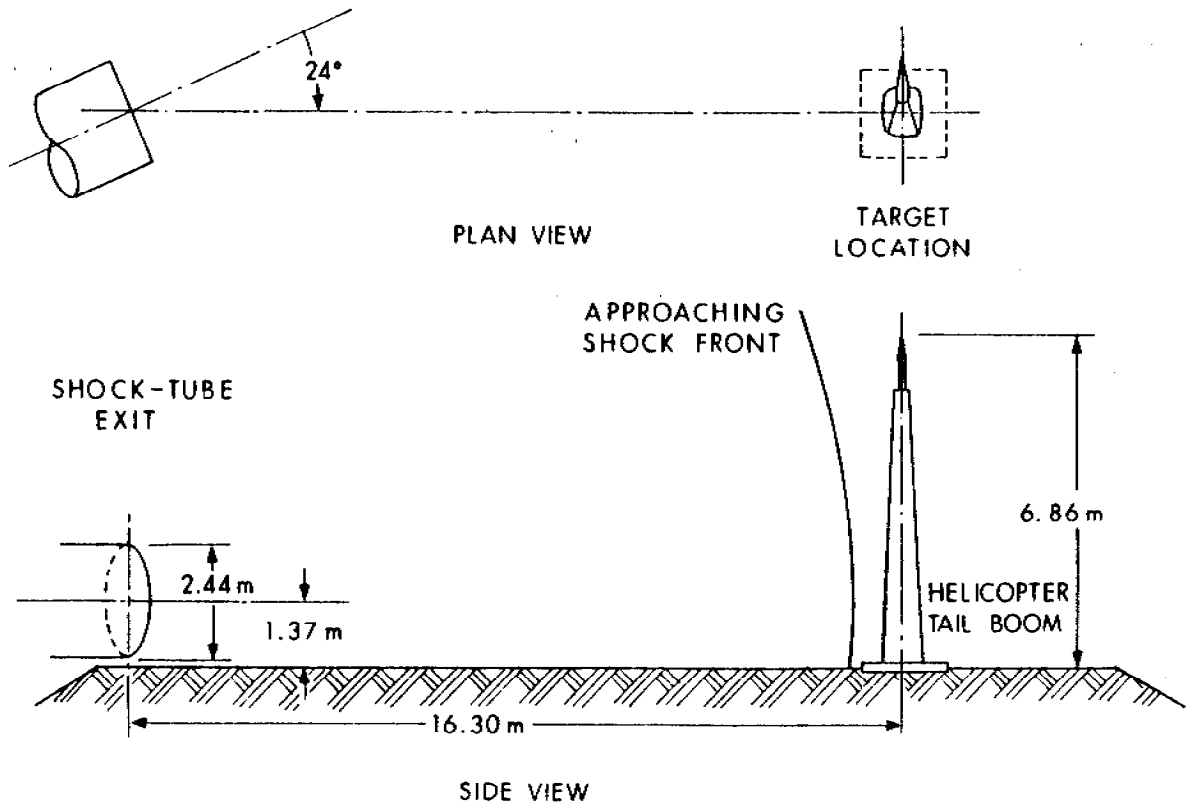


Figure 5. Blast-Wave Simulation Technique Using the BRL 2.4 m Shock Tube.

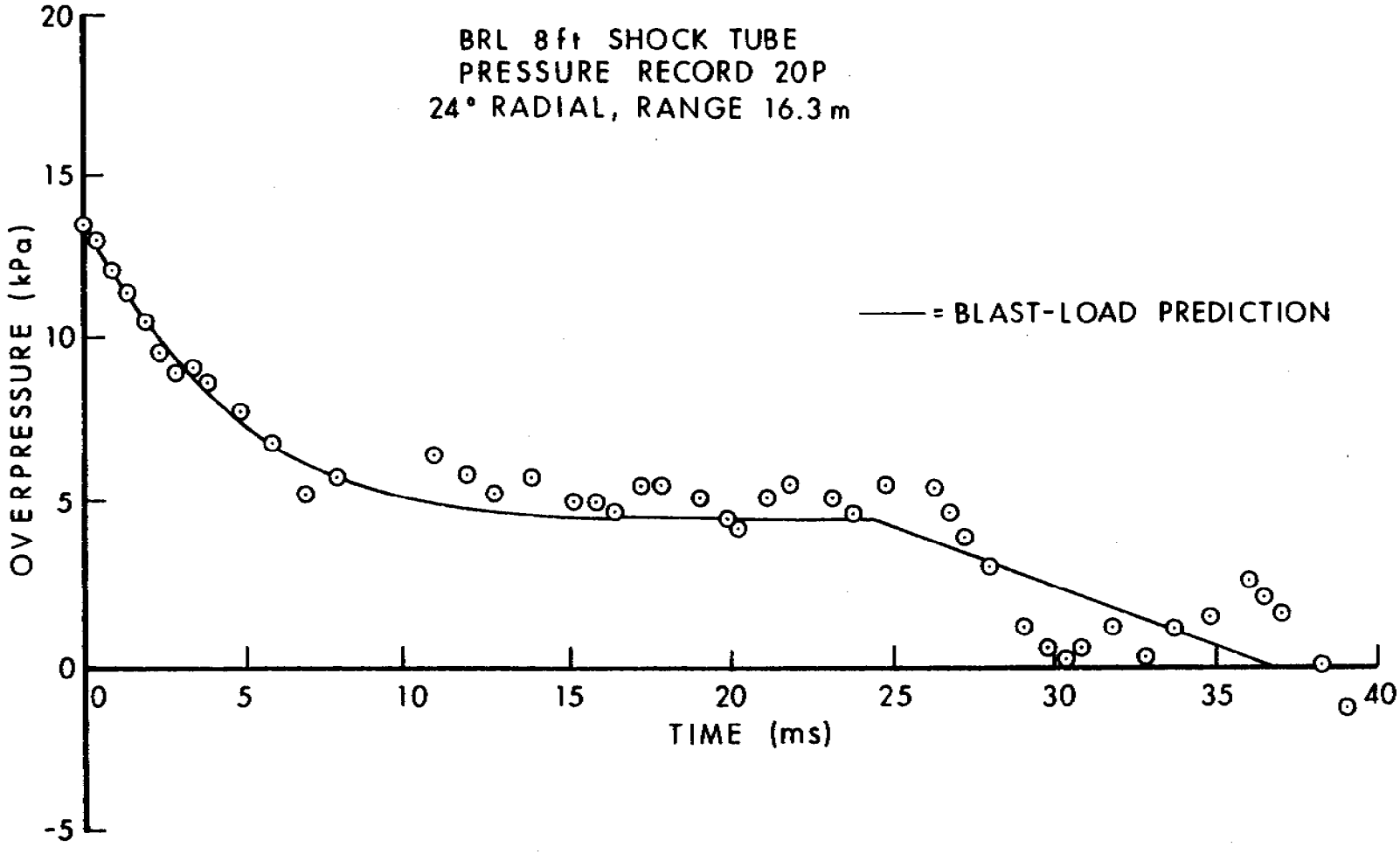
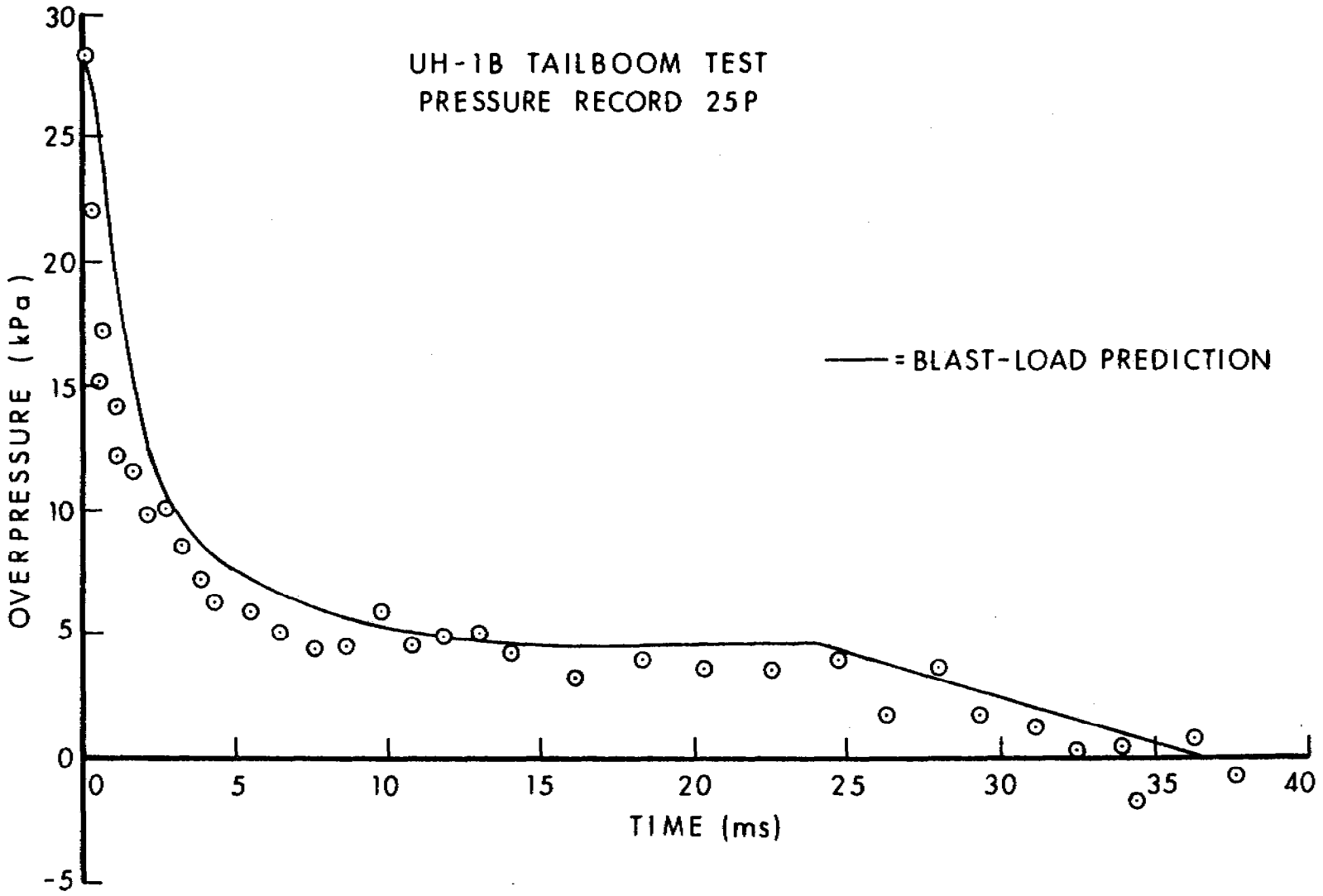
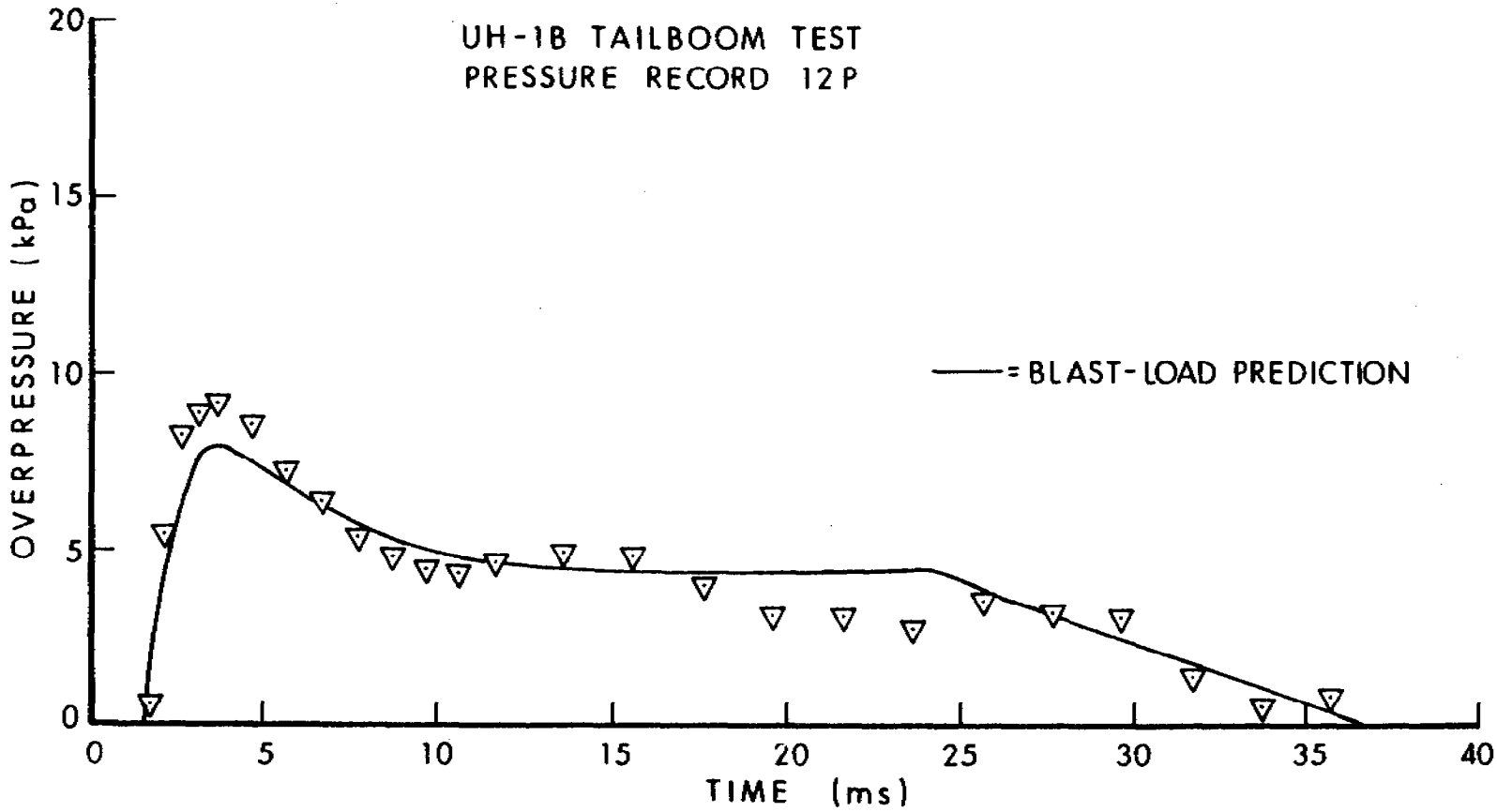


Figure 6. Comparison of the Predicted and Measured Simulated-Blast-Wave History for a Peak Overpressure of 13.4 kPa (1.9 psi).



a) Front-Face Load History

Figure 7. Comparison of the Predicted and Measured Blast-Loading History for a 13.4 kPa (1.9 psi) Simulated Blast Wave.



b) Back-Face Load History

Figure 7. Comparison of the Predicted and Measured Blast-Loading History for a 13.4 kPa (1.9 psi) Simulated Blast Wave.

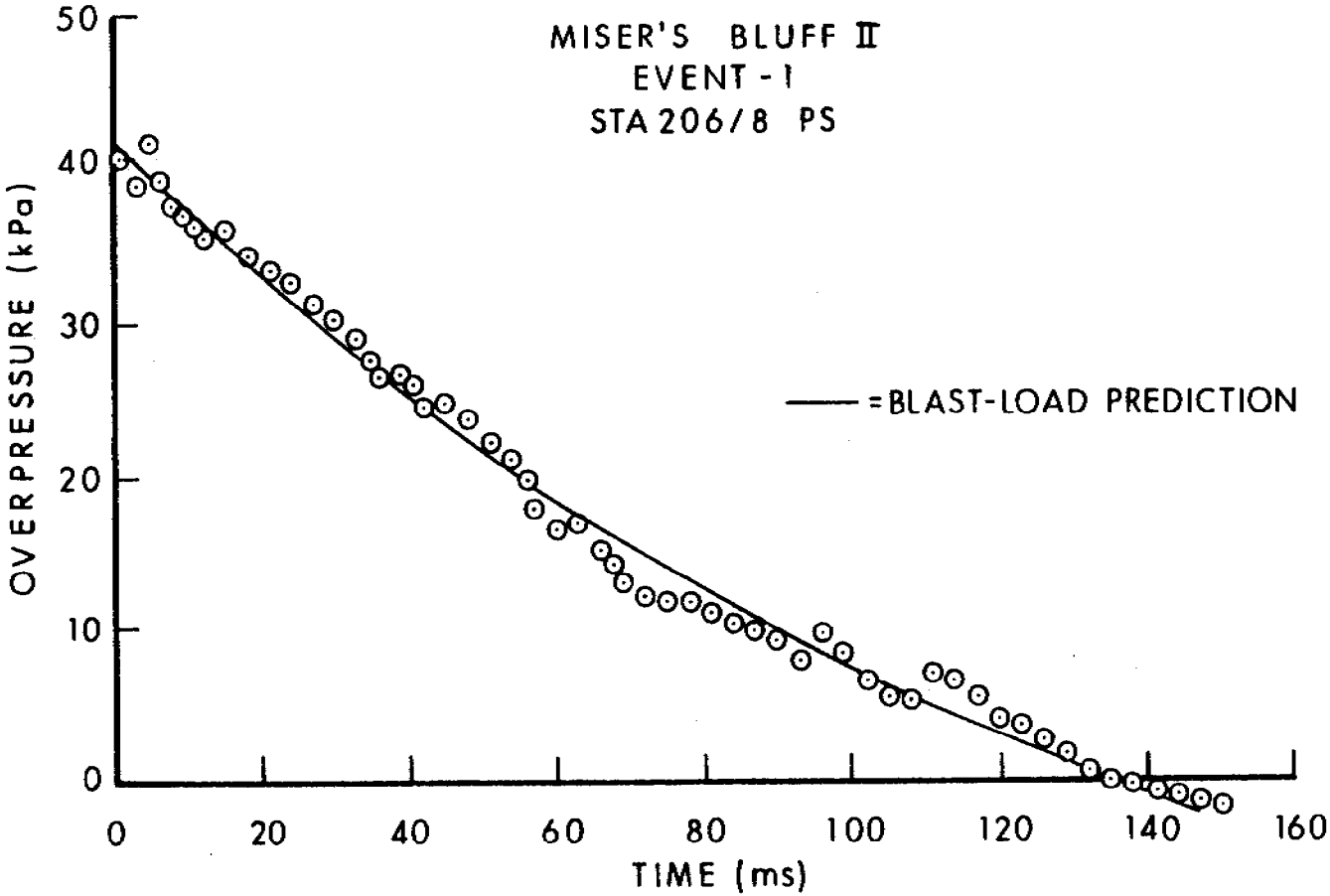
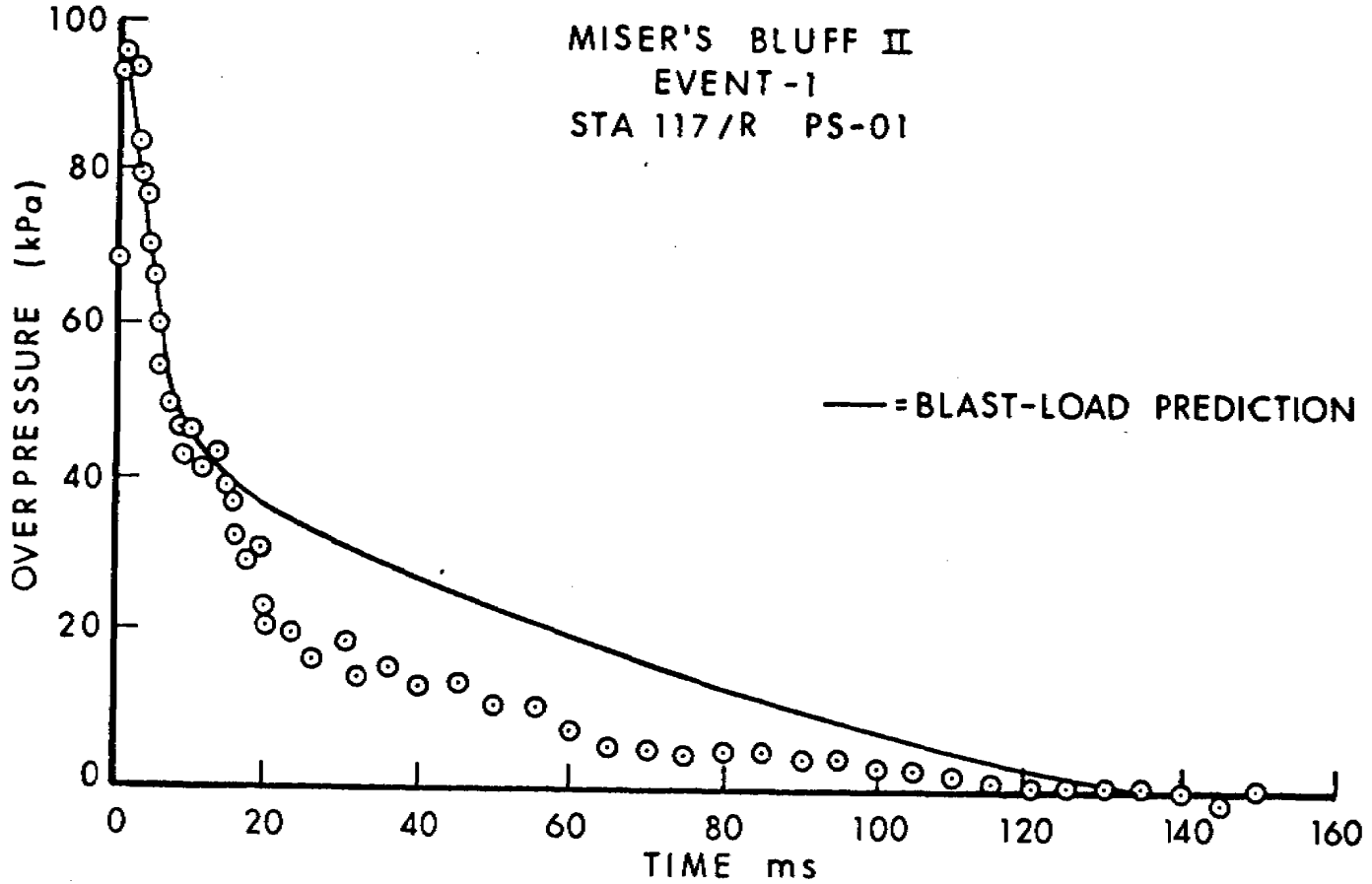
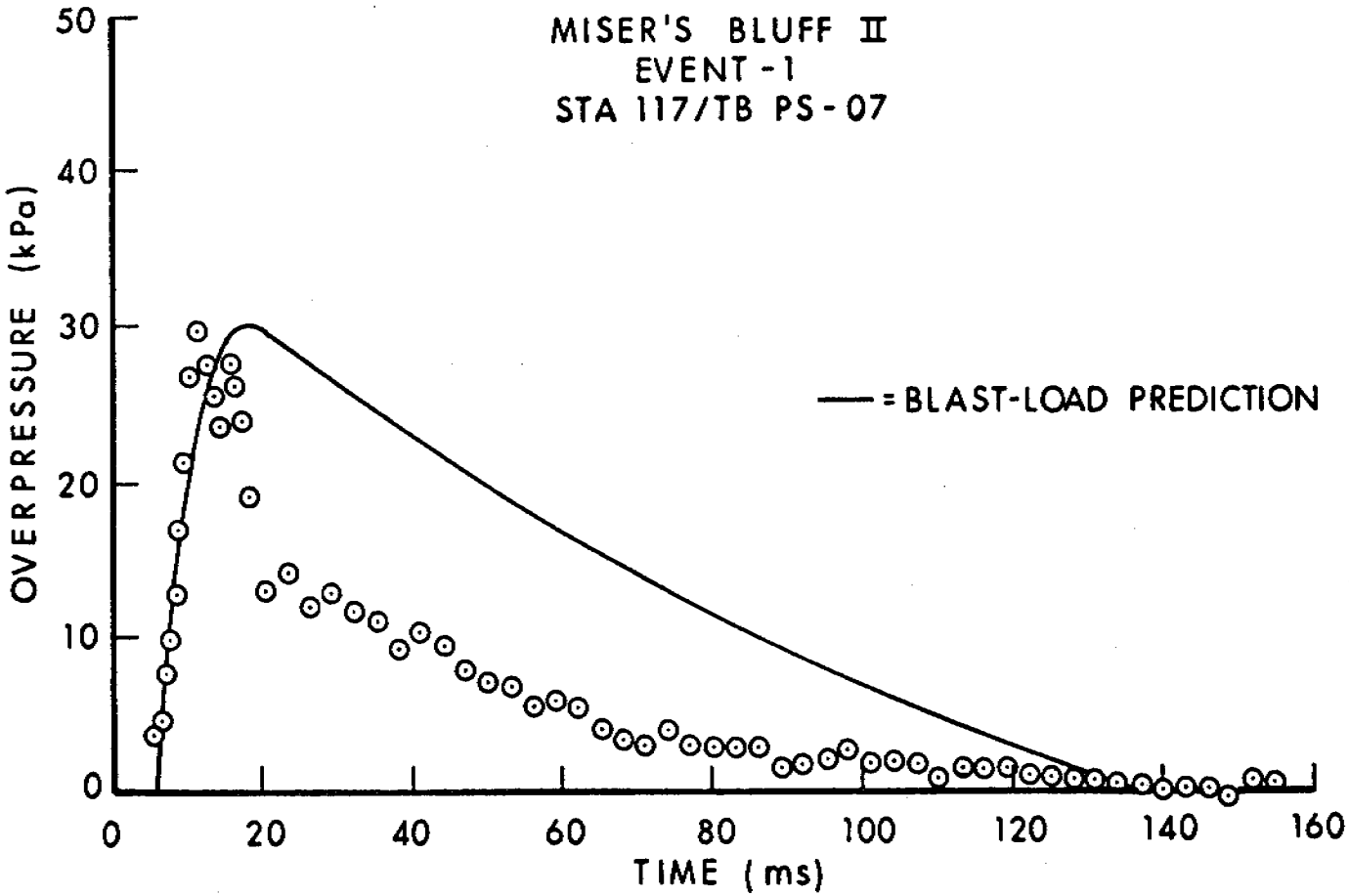


Figure 8. Free-Field Blast-Wave History Compared With BLOP-Code Computation for a Peak Shock Overpressure of 42 kPa (6.1 psi).



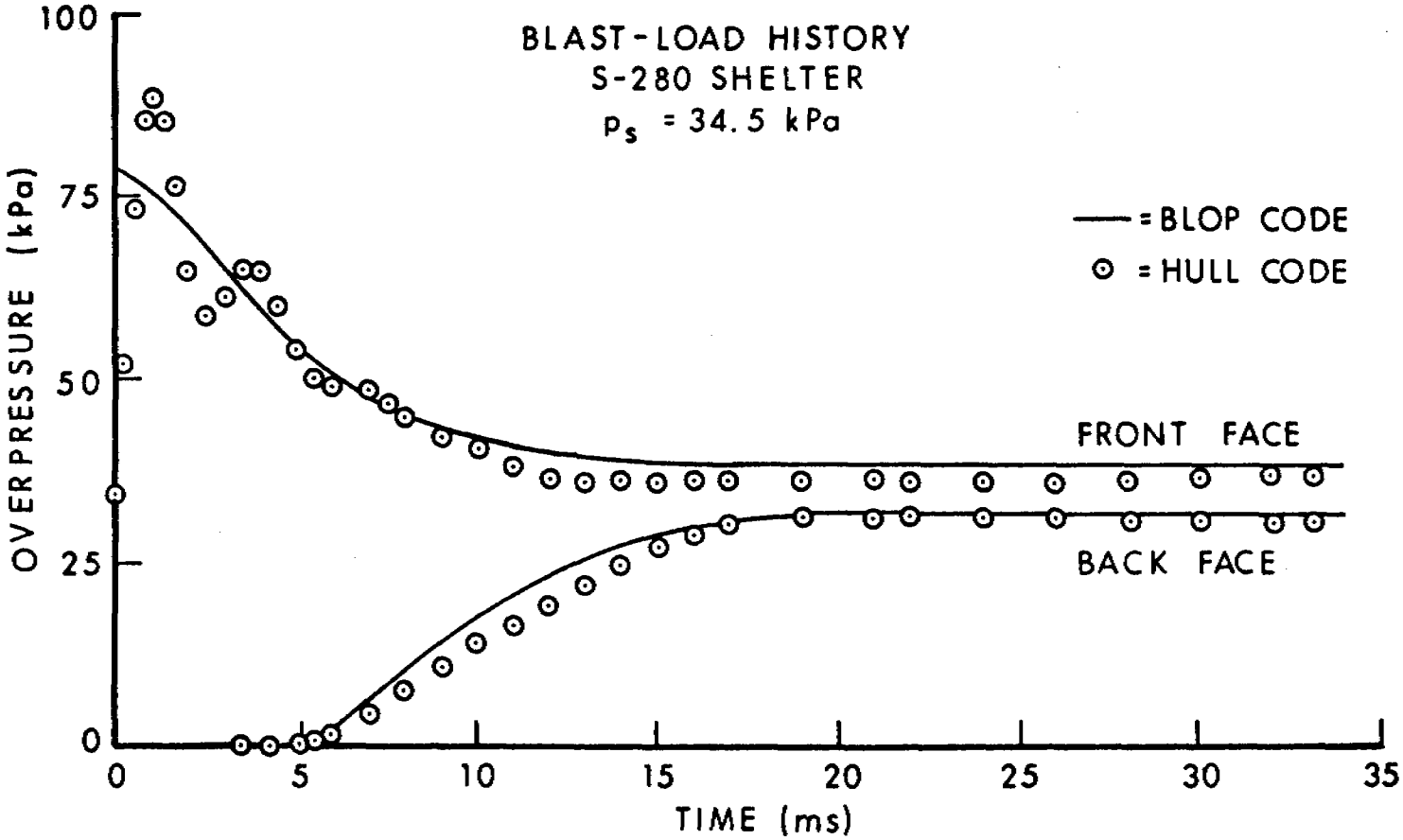
a) Front-Face Load History

Figure 9. S-280 Shelter Test Record Compared with BLOP Prediction for a 42 kPa (6.1 psi) Blast Wave.



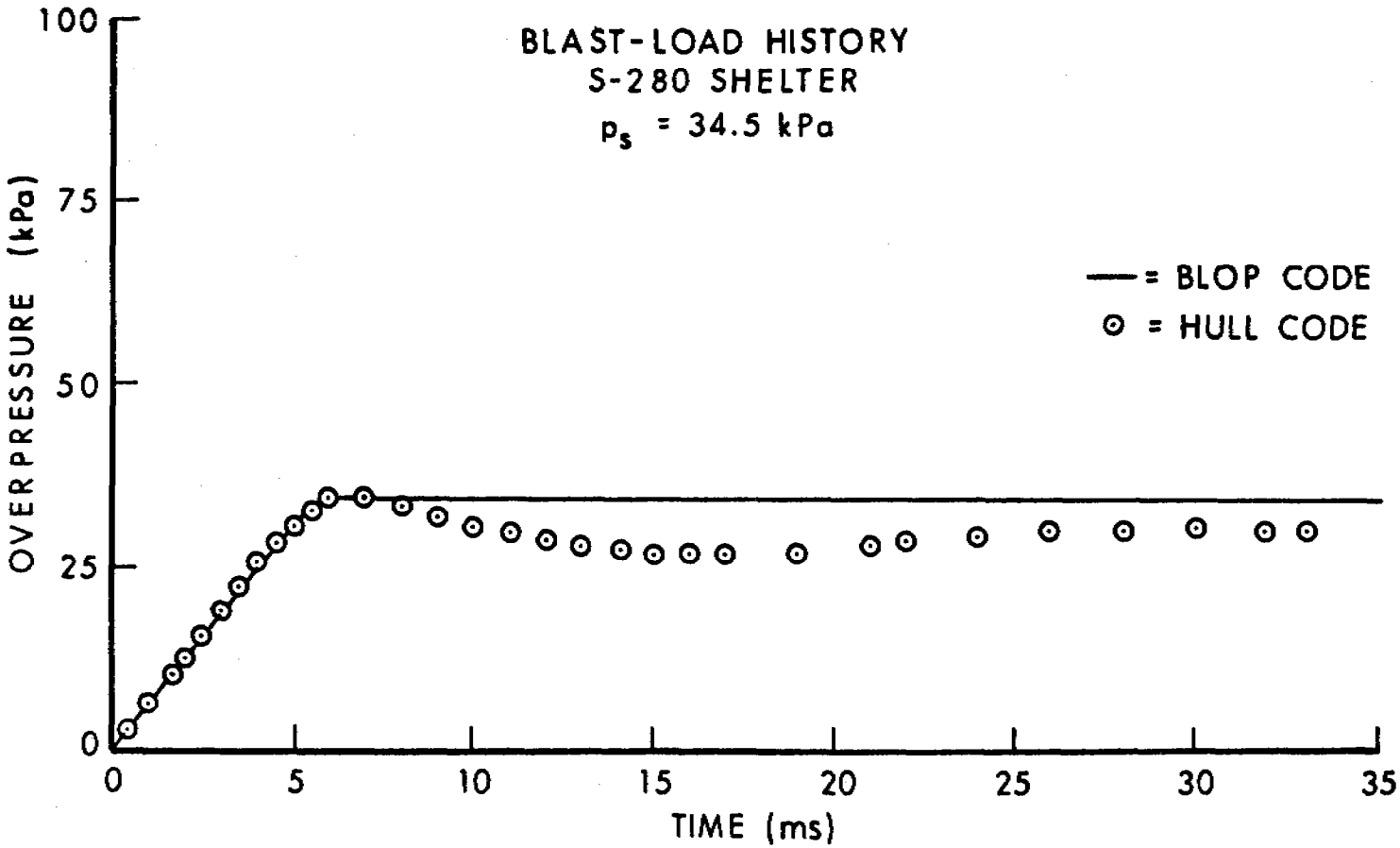
b) Back-Face Load History

Figure 9. S-280 Shelter Test Record Compared with BLOP Prediction for a 42 kPa (6.1 psi) Blast Wave.



a) Front- and Back-Face Load Histories

Figure 10. Comparison of Blast-Loading Computations by BLOP and HULL codes.



b) Side-Face Load History

Figure 10. Comparison of Blast-Loading Computations by BLOP and HULL codes.

Front Tracking for Hyperbolic Conservation Laws: A Progress Report

James Glimm
The Rockefeller University
New York 10021 NY

Oliver McBryan
Courant Institute
New York University
New York 10012 NY

ABSTRACT. Front tracking allows greatly increased resolution and accuracy for fluid flow problems dominated by discontinuities. Progress is reported here on the upgrading of previous calculations [2,3]. The long range goal is a conveniently useable package which is coherently structured and applicable to a broad range of problems.

1. INTRODUCTION. In our previous report [2,3], calculations using front tracking methods were reported. The calculations were performed in the context of petroleum reservoirs, for which the relevent equations are a coupled system of elliptic and hyperbolic equations:

$$v = -k(s)\nabla p \tag{1}$$

$$\nabla \cdot v = \text{source terms} \tag{2}$$

$$s_t + \vec{v}\cdot\nabla f(s) = \text{source terms} \quad . \tag{3}$$

Here $p = p(x,y,t)$ is the pressure and $s = s(x,y,t)$ is the saturation. The calculations tested the concepts of front tracking in a region of parameters for which the problem is unstable and very difficult to compute. The calculations were checked internally for numerical consistency (for example, by testing for grid orientation effects and for convergence under mesh refinement). They were also checked against experimental data. The calculations were performed on a coarse grid, and appear to represent a new capability within computational fluid dynamics, which may be helpful for a broad range of problems.

Recently, progress has centered on upgrading the capability of the calculations in several respects.

2. NEW PHYSICS. Previous one dimensional calculations in gas dynamics [1] are the starting point for a two dimensional gas dynamics front tracking calculation. The main constructive step is the solution of the Riemann problem. This has now been installed in the two dimensional code and is undergoing preliminary tests. Special code (e.g. for reflection of waves at boundaries) has yet to be added. One problem on which this method will be tested is the transient flow past an object (wing foil), or through a tube of variable cross section.

3. NEW GEOMETRY. Arbitrary fronts in two dimensions, including disconnected components and self intersections are allowed within the framework of the calculation's data structure. This is important because self intersections may occur dynamically within a problem which originally may have had a very simple front. Also bifurcations can lead to changes of topology at the self intersection points. Examples are droplet formation and mach stem reflections. Thus it is important to have a computational data structure which allows these events to occur with a minimum of special coding.

4. ELLIPTIC PROBLEMS IN DISCONTINUOUS MATERIALS. Elliptic problems with discontinuous coefficients occur in a wide range of physical problems - for example in incompressible fluid flows. If the location of the discontinuity curve is known accurately, then it is possible to attempt a more accurate solution than would normally be possible. An elliptic solver has been developed recently for media with an irregular material interface, and uses a mesh alignment algorithm to fit the known discontinuity curve, O.McBryan [5]. The main idea is to construct a grid by triangulation of the domain in such a way that each triangle lies entirely on one side of the interface. The grid is a deformation of a regular rectangular grid and is in fact rectangular away from the interface. The equations are then solved using finite elements on this triangulation. The resulting linear equations can be solved efficiently because the matrix is very similar to a regular finite difference operator.

5. INTERFACE PACKAGE. Complex material topologies and interfaces occur in a wide range of problems. Work has begun on developing a subroutine package for manipulating such interfaces. The package allows for arbitrarily complex topologies and geometries and is designed to minimize the programming effort involved in coding interfaces. High level primitive operations such as adding curves to an interface or making a copy of an interface hide the underlying data structures which have been designed to provide efficient access to topological information - such as which component of a domain a given point lies in. This code will be used in both the shock-tracking codes and the elliptic codes referred to previously. Eventually the package will be extended to handle three-dimensional interface surfaces.

6. STRUCTURED DESIGN. The front tracking and mesh alignment codes described previously are large and complex pieces of software. A major effort is underway to ensure that these codes can be applied to new problems with a minimum of programming effort. Principles of structured programming are used throughout and all physics or geometry dependant routines have been isolated. Thus the tracking code can be used as a package and easily applied to other problems. All that is required is a main driver routine and the provision of a set of physics dependant routines - for example a Riemann solver for a hyperbolic conservation law. Similarly the elliptic code is modularized and requires only a few problem-specific routines such as those to define the coefficient functions and boundary data. Lower-level modules such as a general purpose storage allocator and a

debugging package are also of more general use. Supporting graphics programs have been designed with a device and system independance. Thus the same program can generate a Tektronix plot on a Vax 11/780 or a movie on a CDC6600. Further developments, such as three-dimensional and colour graphics, will be needed for the effective interpretation of more complex codes.

7. BIBLIOGRAPHY.

(1) J. Glimm, D. Marchesin and O. McBryan. Subgrid resolution of fluid discontinuities 11. J. Comp. Phys. 37, 1-13 (1980).

(2) J. Glimm, E. Isaacson, D. Marchesin and O. McBryan. A shock tracking method for hyperbolic systems, ARO report 80-3 Sept. 1980.

(3) J. Glimm, E. Isaacson, D. Marchesin and O. McBryan. Front tracking for hyperbolic systems. Adv. Appl. Math. to appear.

(4) D. Marchesin and P. Paes-Leme. Shocks in gas pipelines, Preprint.

(5) O. McBryan. Elliptic and hyperbolic interface refinement. In: Boundary layers and internal layers -- computational and asymptotic methods. J. Miller, Ed, Boole Press.

DISCUSSION OF A VERTICALLY AVERAGED HYDRODYNAMIC MODEL USING BOUNDARY FITTED COORDINATES

Billy H. Johnson
U. S. Army Engineer Waterways Experiment Station
Vicksburg, Mississippi 39180

Joe F. Thompson
Mississippi State University
Starkville, Mississippi 39762

ABSTRACT. A numerical model for computing the vertically averaged hydrodynamics of a water body, including salinity effects, has been developed. The model employs the concept of boundary fitted coordinates to allow for an accurate representation of the boundary of the region being modeled while retaining the simplicity of the finite difference method of solution. Although a general curvilinear coordinate system covers the physical domain, all computations to solve the governing fluid dynamic equations, as well as the computation of the boundary fitted coordinate system, are performed in a transformed rectangular plane with square grid spacing.

A combination implicit-explicit finite difference scheme has been employed to numerically solve the governing equations. With such a scheme, the water surface elevation is computed implicitly using the Accelerated Gauss-Seidel solution technique; whereas, the velocity and salinity fields are solved in an explicit manner. The major advantage of such a scheme is that the speed of a surface gravity wave is removed from the stability criteria while many desirable features of an explicit scheme are retained.

Although additional work remains to be completed before the model can be considered fully operational, preliminary results demonstrate that the basic model behaves properly.

1. INTRODUCTION. Since the equations governing the motion of fluids are nonlinear, analytic solutions in general cannot be found and one is forced to resort to numerical techniques to obtain solutions. The two most common such techniques are the finite difference method (FDM) and the finite element method (FEM). There are, of course, both advantages and disadvantages to each of these approaches.

Perhaps the most often quoted advantage of the finite element method is that with this approach physical boundaries coincide with computational net points. Therefore, the modeling of flow within an irregular domain can be more accurately handled than with the normal finite difference method where the approach is to construct a rectangular grid over the domain, which forces the boundaries to be represented in a "stair stepped" fashion. However, a disadvantage of finite element methods is that they involve dense matrices rather than the sparse matrices involved in finite difference methods. This results in more computational time being required in a finite element model having the same number of mesh points as a finite difference model. An additional disadvantage is that the finite element method is more cumbersome to code into a computer model than the finite difference method. This can be a problem not only during the development of the model but can also increase the level of effort required during later model modifications.

Accepting that the finite difference method possesses an advantage in simplicity and perhaps computational costs, a logical question is whether or not one can develop ways to circumvent the major disadvantage of having to represent irregular boundaries in a "stair stepped" fashion. One such technique which has been developed by Thompson, et al.[1,2,3] involves the use of boundary-fitted coordinates. Thompson's method generates curvilinear coordinates as the solution of two elliptic partial differential equations with Dirichlet boundary conditions, one coordinate being specified to be constant on the boundaries, and a distribution of the other specified along the boundaries. However, the numerical computations to solve the governing flow equations, as well as computations for the solution of the coordinate system, are not made in the physical curvilinear coordinate system but rather are made on a rectangular grid with square mesh spacing.

The mathematical modeling of the hydrodynamics of a body of water plus the transport and dispersion of a conservative constituent within that body involves the solution of a set of partial differential equations expressing the conservation of mass, momentum, and energy of the flow field along with a transport equation for the constituent. These equations involve derivatives with respect to time as well as three spatial dimensions. However, a simplification that is often made in treating relatively shallow bodies of water that are well mixed over the depth is to vertically average the three-dimensional (3D) equations to yield a two-dimensional (2D) set for nearly horizontal flows.

Since the early to mid 1960's, many finite difference, plus a few finite element, computational models for vertically averaged flows have been developed.[4,5,6,7] The purpose of this paper is to describe the development of a new vertically averaged hydrodynamic model which is fully coupled with the water salinity through its influence on the water density. The finite difference method of solution is employed but, unlike the previously developed models, solutions are obtained on a boundary-fitted coordinate system to provide an accurate representation of boundary geometry.

2. BASIC HYDRODYNAMIC EQUATIONS. The Navier Stokes equations are the basic governing equations for the solution of fluid dynamic problems and express the conservation of mass and momentum of the flow field. In addition, for problems in which salinity effect are important, a separate conservation of mass equation must also be written for the salinity along with an equation of state relating the water density to the salt concentration and the water temperature. With the closure of such a system, there exist six equations to be solved for the six unknowns; density - ρ, three velocity components - u, v, w, pressure - p, and salinity - s.

After temporally as well as vertically averaging the equations discussed above, the final form of the governing equations in Cartesian coordinates is:

Continuity: $$\frac{\partial \phi}{\partial t} + \frac{\partial (uh)}{\partial x} + \frac{\partial (vh)}{\partial y} = 0 \qquad (1)$$

x-momentum: $$\frac{\partial (hu)}{\partial t} + \frac{\partial (hu^2)}{\partial x} + \frac{\partial (huv)}{\partial y} = -\frac{h}{\rho_o}\left(\frac{\partial P_a}{\partial x} + g\rho\frac{\partial \phi}{\partial x} + \frac{hg}{2}\frac{\partial \rho}{\partial x}\right)$$

$$+ \frac{\partial\left(hD_{xx}\frac{\partial u}{\partial x}\right)}{\partial x} + \frac{\partial\left(hD_{xy}\frac{\partial u}{\partial y}\right)}{\partial y} + \frac{W_c}{\rho_o}\rho_a v_w^2 \cos\alpha$$

$$- gu\sqrt{u^2 + v^2}/C^2 + fhv \qquad (2)$$

y-momentum: $$\frac{\partial (hv)}{\partial t} + \frac{\partial (huv)}{\partial x} + \frac{\partial (hv^2)}{\partial y} = -\frac{h}{\rho_o}\left(\frac{\partial P_a}{\partial x} + g\rho\frac{\partial \phi}{\partial y} + \frac{hg}{2}\frac{\partial \rho}{\partial y}\right)$$

$$+ \frac{\partial\left(hD_{yx}\frac{\partial v}{\partial x}\right)}{\partial x} + \frac{\partial\left(hD_{yy}\frac{\partial v}{\partial y}\right)}{\partial y} + \frac{W_c}{\rho_o}\rho_a v_w^2 \sin\alpha$$

$$- gv\sqrt{u^2 + v^2}/C^2 - fhu \qquad (3)$$

Salinity: $$\frac{\partial (hs)}{\partial t} + \frac{\partial (hus)}{\partial x} + \frac{\partial (hvs)}{\partial y} = \frac{\partial\left(hE_x\frac{\partial s}{\partial x}\right)}{\partial x} + \frac{\partial\left(hE_y\frac{\partial s}{\partial y}\right)}{\partial y} \qquad (4)$$

Equation of State: $\rho = \rho(s,T)$

where ϕ = water surface elevation,
h = water depth
u,v = velocity components
P_a = atmospheric pressure
ρ = water density
D_{xx}, D_{xy}, D_{yx}, D_{yy} = eddy viscosity coefficients
v_w = wind speed

α = wind direction
f = Coriolis parameter
g = acceleration of gravity
s = salt concentration
T = water temperature
E_x, E_y = eddy diffusivity coefficients

A discussion of the development of these equations can be found in reference 8.

The above set of equations must now be transformed into a (ξ,η) boundary-fitted coordinate system such that (ξ,η) are the independent variables. The resulting set of equations will then be solved in a transformed rectangular plane as previously discussed. In order to accomplish the transformation, the following expressions are utilized.

$$f_x = \frac{1}{J}\left[(fy_\eta)_\xi - (fy_\xi)_\eta\right]$$

$$f_y = \frac{1}{J}\left[-(fx_\eta)_\xi + (fx_\xi)_\eta\right] \qquad (6)$$

It should be noted that these expressions are written in a fully conservative form which should result in a more accurate solution in highly irregular coordinate systems. For brevity, the transformed set of equations are not presented. For the more interested reader, they are presented in reference 8. Obviously, the transformed equations are more complicated than the Cartesian form presented as equations 1-5; however, the advantage of being able to make computations on a rectangular grid far outweighs any disadvantage resulting from the more complicated set of equations.

3. NUMERICAL ASPECTS. In order to obtain a solution of the governing set of transformed equations, the method of finite differences is employed. There are many different types of finite difference schemes that have been employed in numerical solutions of partial differential equations. These schemes range from fully explicit to fully implicit, with a combination of an explicit-implicit scheme being employed in some cases, e.g., Edinger and Buchak.[9] A similar scheme is employed here. Basically, the computational cycle will consist of the following steps.

a. Solve for the water surface from the continuity equation in a fully implicit fashion using the Accelerated Gauss-Seidel technique.

b. Using the most recent values of the water surface elevations, solve for the u and v velocity components from the x and y momentum equations in an explicit fashion.

c. Solve for the salinity from the salt transport equation in an explicit fashion.

d. Compute the density from the equation of state, using the most recently computed salinity field.

e. Step forward in time and repeat the sequence.

Such a scheme as outlined above will have the stability criterion associated with the speed of a free surface gravity wave removed; although, diffusive criteria as well as the Torrence condition associated with the speed of a water particle remain. However, these criteria are not normally over restrictive.

The grid upon which the governing equations are solved is rectangular with a grid spacing of $\Delta\xi = \Delta\eta = 1$. The u and v velocity components are computed at the corners of each cell with the water surface elevation, salinity, and density computed at the center of a cell. The (x,y) coordinates are specified at the corners, the center, and also at the midpoint of each side of a cell.

The basic difference equations are developed using forward differences for all time derivatives. Centered differences are used in all spatial derivatives except in the convective terms where one has the option in the computer model (called VAHM for Vertically Averaged Hydrodynamic Model) of requesting the use of either centered or a form of Roache's second upwind differencing.

4. BOUNDARY CONDITIONS. Three types of boundaries are allowed in VAHM; walls, oceans, and rivers. Wall boundaries are characterized by the specification of a no-slip condition, i.e., the velocity components u and v are set to be zero at walls. Although, physically, the flow must be zero at a solid boundary, slip conditions on the velocity at a wall often give more realistic results if the grid spacing is too large near the wall. Slip conditions would be implemented by setting the normal component of the velocity equal to zero with the tangential component computed from the expression for zero vorticity. At the present time, only the no-slip condition is allowed in VAHM.

Ocean boundaries are characterized by the specification of a time varying water surface elevation at the boundary. Velocities on the ocean boundary are then computed from a simplified form of the momentum equation where the diffusive terms have been neglected. One-sided differences are used to replace derivatives that need points outside the field.

When the flow is directed into the computational field, the boundary condition on the salinity is prescribed as that of the ocean. However, when the flow is moving out of the computational field, the salinity at an ocean boundary is set to be equal to its value at the next point inside.

River boundaries are characterized by the specification of the velocity. The salinity is set to be zero and the water surface elevation at the center of a river boundary cell is computed as in any interior cell.

5. MODEL APPLICATION. In order to demonstrate the versatility of VAHM in its ability to model flows in rather general multiply-connected regions containing both river and ocean boundaries, an application has been made using the physical geometry in Figure 1.

The first step in the application of VAHM is the generation of the boundary-fitted coordinates. This is accomplished through a coordinate generation code developed by Thompson. Output from the coordinate code is saved on a file for subsequent use by VAHM. The basic input to the coordinate code is the specification of the (x,y) coordinates of the boundary points noted on Figure 1. Although various degrees of coordinate control can be exercised, the boundary-fitted coordinates shown in Figure 1 were computed using no control. The coordinate system plotted was the the third attempt at generating a useful grid system. Through the movement of boundary points and/or coordinate control one attempts to compute boundary-fitted coordinates such that the grid spacing does not vary rapidly and such that (ξ,η) lines never approach being parallel to each other. The coordinate system presented satisfies both of these criteria and thus is considered to be adequate.

For the geometry shown in Figure 1, an application, in which a river boundary is assumed at the top with an ocean on the bottom, has been made. A constant velocity of 0.4 m/s and a zero salinity concentration were assumed at the river boundary while the tide curve presented in Figure 2, and an ocean salinity concentration of 30 ppt was prescribed at the ocean boundary. The initial depth was set to be 11.0 m throughout the system with the initial velocity and salinity fields set to zero. Values of various parameters were prescribed by setting the diagonal components of the eddy viscosity tensor to 10 m^2/s , setting the Chezy coefficient for bottom friction to 35 $m^{1/2}/s$ and employing a computational time step of 600 sec.

Figures 3-9 present "snap shots" of the computed flow field at various times. The influence of first the flood and then the ebb portion of the tide can be clearly seen. In addition to velocity vector plots, one can also consider the time history of the water surface elevation as well as the salinity at particular points in the systems. Figures 10 and 11 are examples.

6. SUMMARY. A numerical model for computing vertically averaged velocities and salinity plus water surface elevations has been developed. By employing the concept of boundary-fitted coordinates, irregular boundaries can be accurately modeled in either simply or multiply-connected regions. Even though the numerical grid is a nonorthogonal curvilinear grid in the physical region being modeled, all numerical computations are carried out in a transformed rectangular grid with square grid spacing.

A feature of the model is the particular solution technique employed to numerically solve the governing equations. A combination implicit-explicit finite difference scheme, patterned after work by Edinger and Buchak[9] in their development of a laterally averaged reservoir hydrodynamic model, has been developed to remove the speed of a gravity wave from stability restrictions on the computation time step while still retaining some of the advantages of explicit schemes. With such a scheme, the water surface elevation is computed implicitly using the Accelerated Gauss-Seidel solution technique while the velocities and salinity are computed in an explicit fashion.

The model has been developed for general applications. Any number of river and/or ocean boudaries can be arbitrarily located on the transformed rectangular plane, as can the placement of islands in the interior of the computation field. Even though a great deal of generality exist, there are restrictions. For example, only no-slip boundary conditions are currently treated at solid boundaries and no flooding of those boundaries is allowed; however, work on removing these restrictions is ongoing.

Although VAHM has been developed to the point where results from the test application presented are encouraging, additional work is needed before VAHM can be considered fully operational.

REFERENCES

1. Thompson, Joe F., et al., "Automatic Numerical Generation of Body-Fitted Curvilinear Coordinate System for Field Containing Any Number of Arbitrary Two-Dimensional Bodies," Journal of Computational Physics, Vol 15, No. 3, July 1974.

2. Thompson, Joe F., et al., "TOMCAT - A Code for Numerical Generation of Boundary-Fitted Curvilinear Systems on Fields Containing Any Number of Arbitrary Two-Dimensional Bodies," Journal of Computational Physics, Vol 24, No. 3, July 1977.

3. Thompson, Joe F., et al., Boundary-Fitted Curvilinear Coordinate Systems for Solution of Partial Differential Equations on Fields Containing Any Number of Arbitrary Two-Dimensonal Bodies, NASA CR-2729, National Aeronautics and Space Administration, Washington, D. C., July 1979.

4. Reid, R. O. and Bodine, B. R., "Numerical Model for Storm Surges in Galveston Bay," Journal of Waterways of Harbors Division, Proceedings, ASCE, Vol 94, No. WW1, 1968.

5. Leendertse, J. J., "Aspects of a Computational Model for Long-Period Water Wave Propagation," RM 5294-PR, the Rand Corporation, Santa Monica, California, 1967.

6. Masch, F. D., et al., "A Numerical Model for the Simulation of Tidal Hydrodynamics in Shallow Irregular Estuaries, "TR-HYD-12-6901, Hydraulic Engineering Laboratory, University of Texas, Austin, Texas, 1969.

7. Norton, W. R., et al., "A Finite Element Model for Lower Granite Reservoir," prepared for Walla Walla District, U. S. Army Corps of Engineers, 1973.

8. Johnson, Billy H., "VAHM - A Vertically Averaged Hydrodynamic Model Using Boundary-Fitted Coordinates, MP-HL-80-3, U. S. Army Engineer Waterways Experiment Station, Vicksburg, Mississippi, September 1980.

9. Edinger, J. R. and Buchak, E. M., "A Hydrodynamic Two-Dimensional Reservoir Model: Development and Test Application to Sutton Reservoir, Elk River, West Virginia," prepared for U. S. Army Engineer Division, Ohio River, 1979.

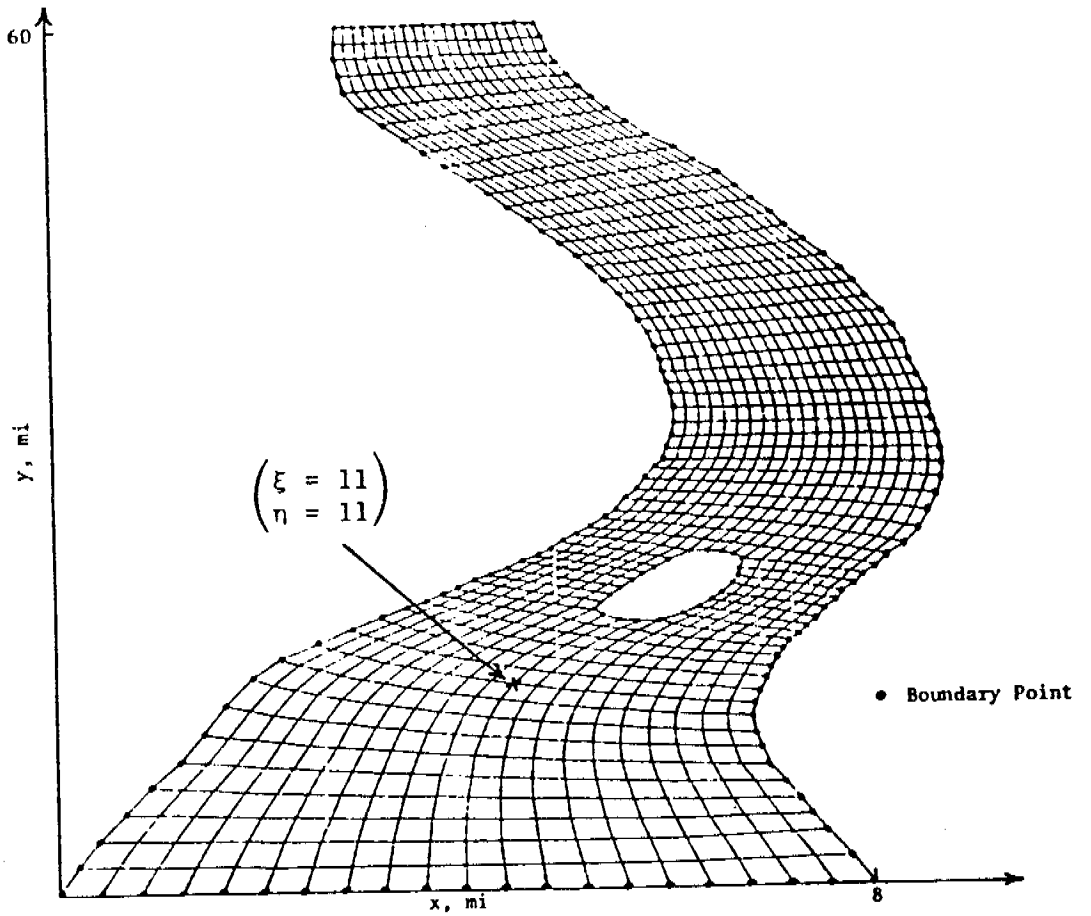


Figure 1. Boundary fitted coordinate system for a multiple connected region

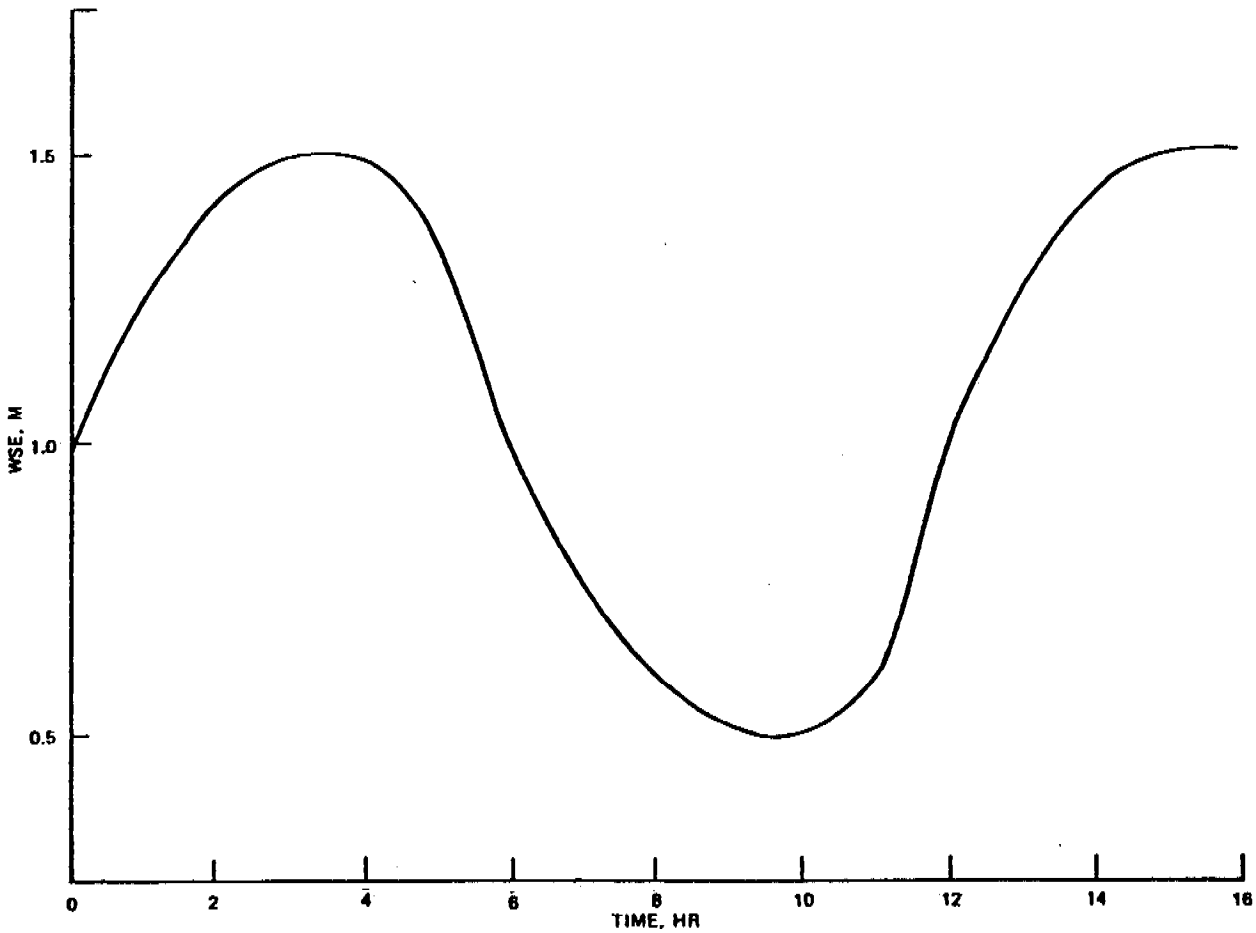


Figure 2. Water surface elevation at ocean boundary

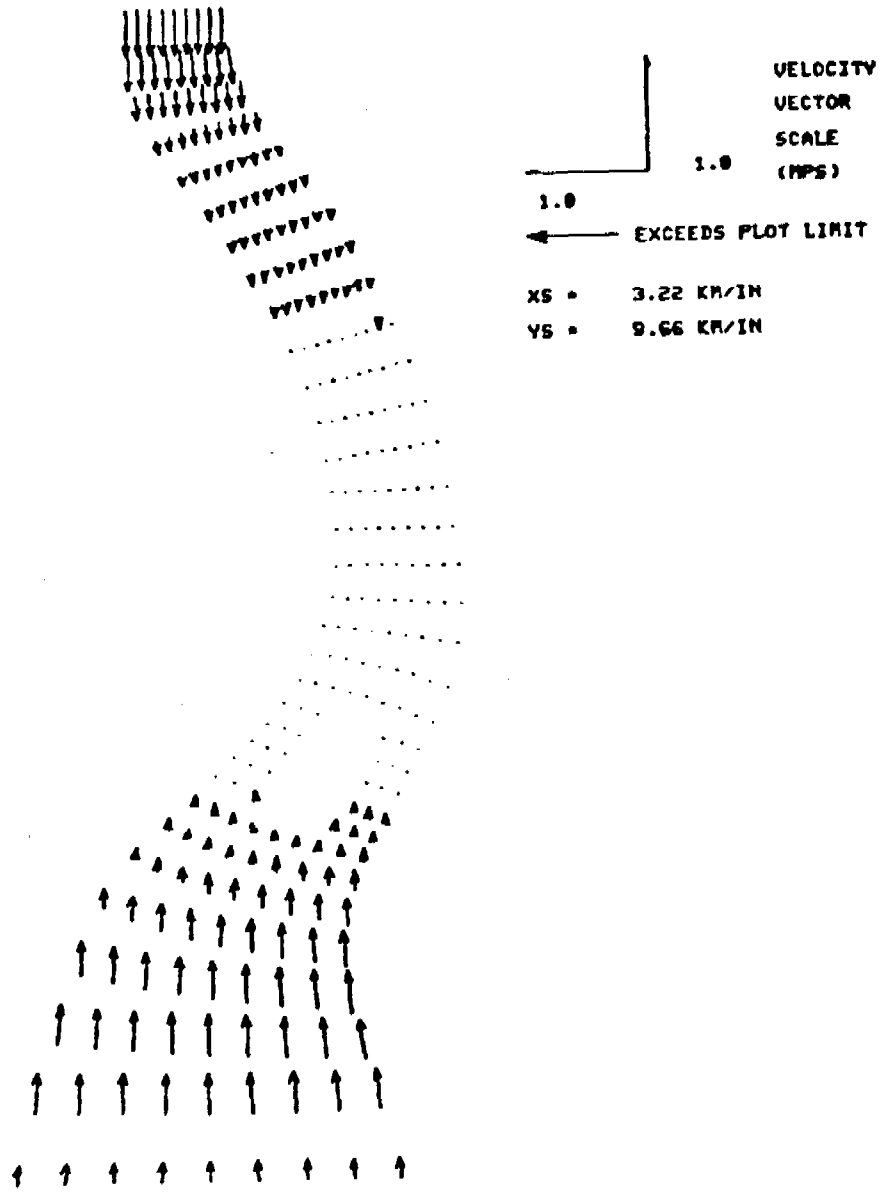


Figure 3 . Velocity field after 1 hour with an ocean and a river boundary

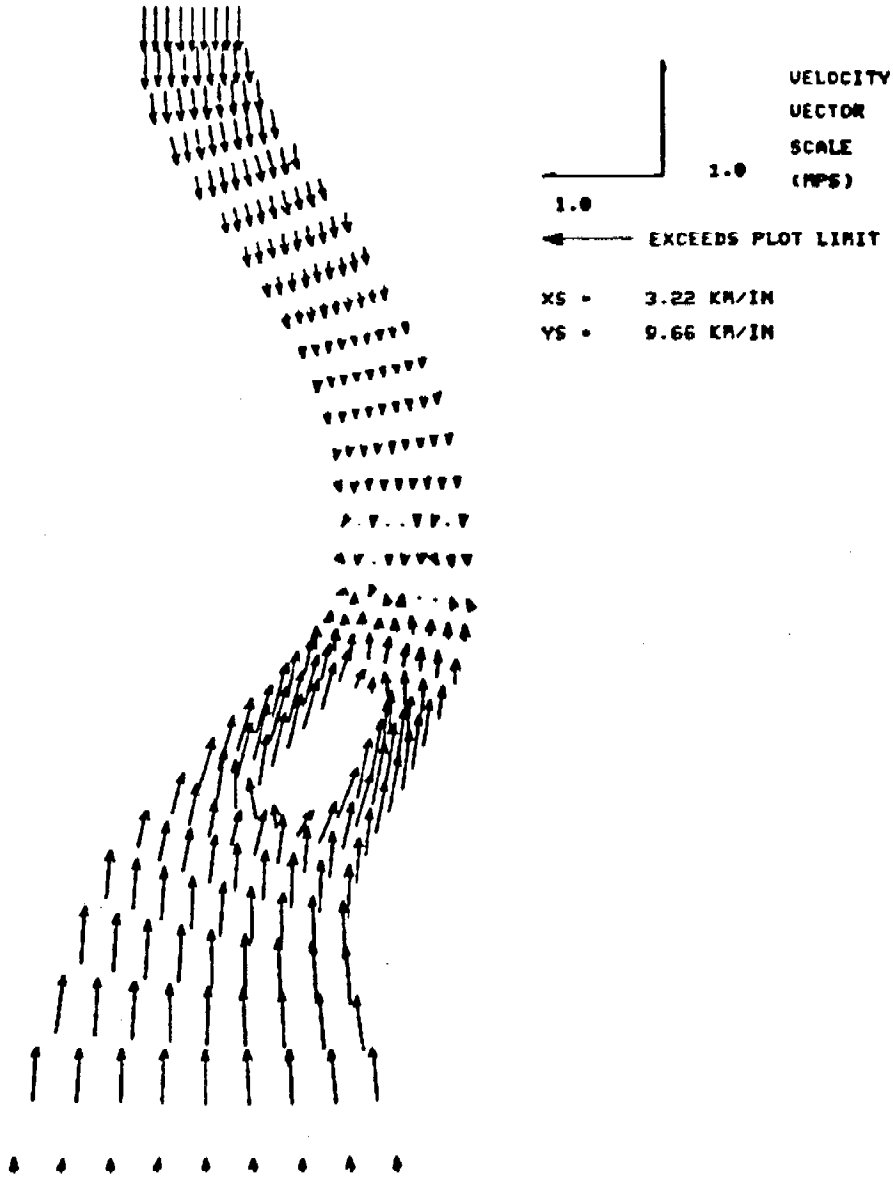


Figure 4. Velocity field after 3 hours with an ocean and a river boundary

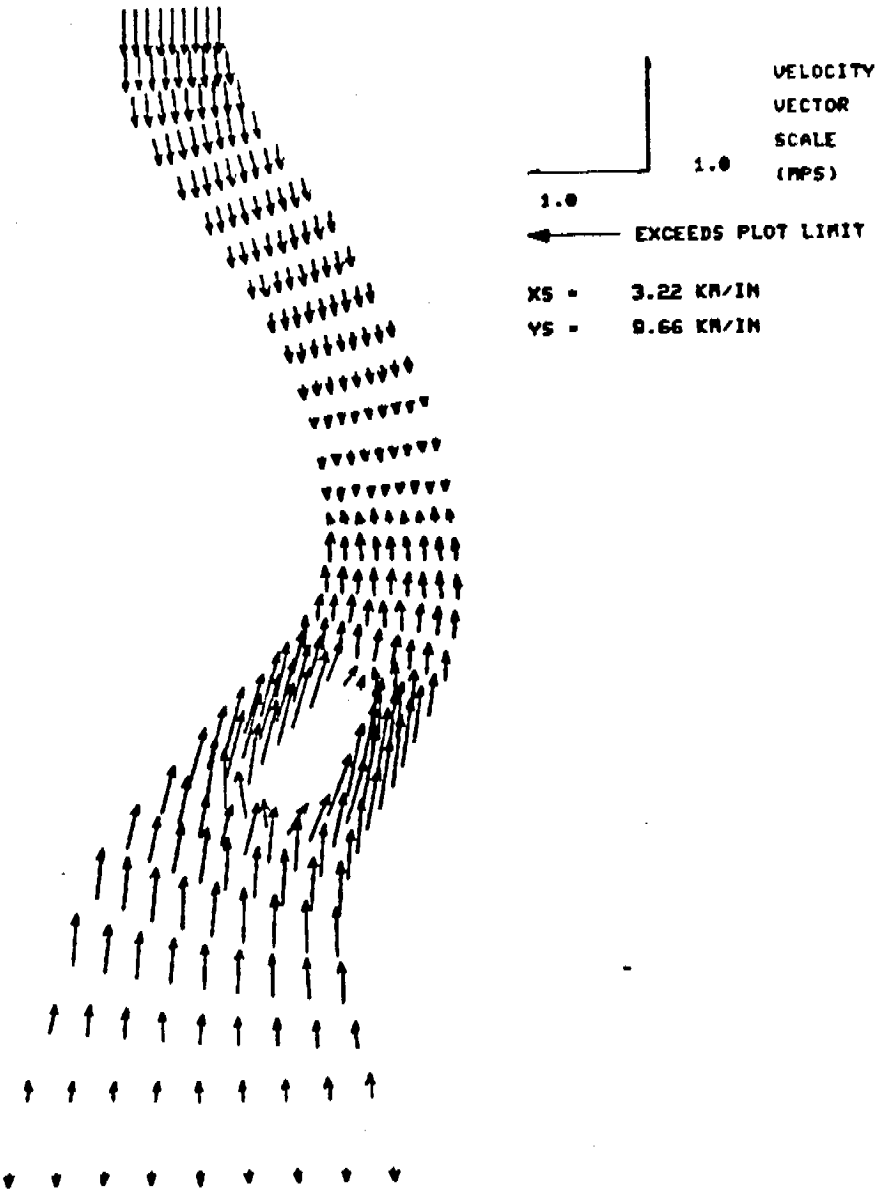


Figure 5. Velocity field after 5 hours with an ocean and a river boundary

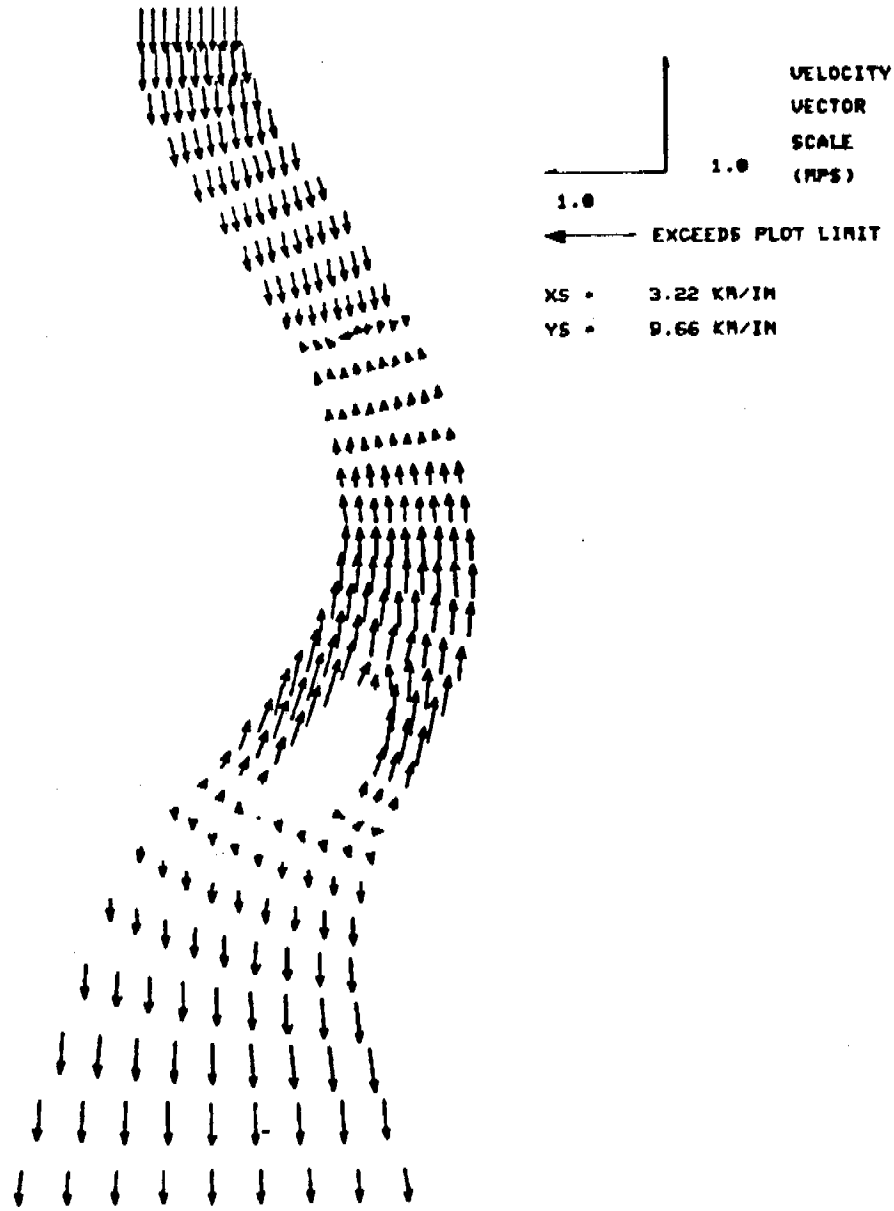


Figure 6. Velocity field after 7 hours with an ocean and a river boundary

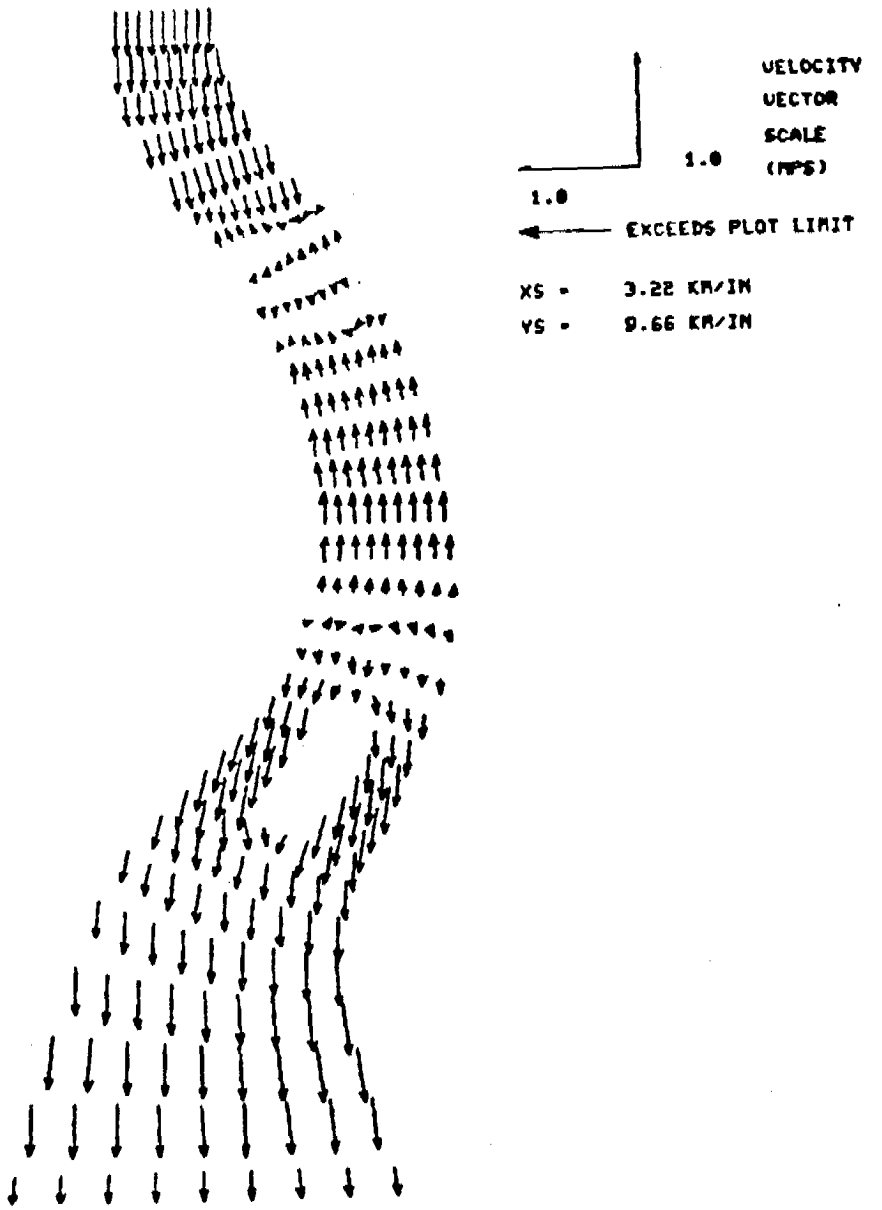


Figure 7. Velocity field after 9 hours with an ocean and a river boundary

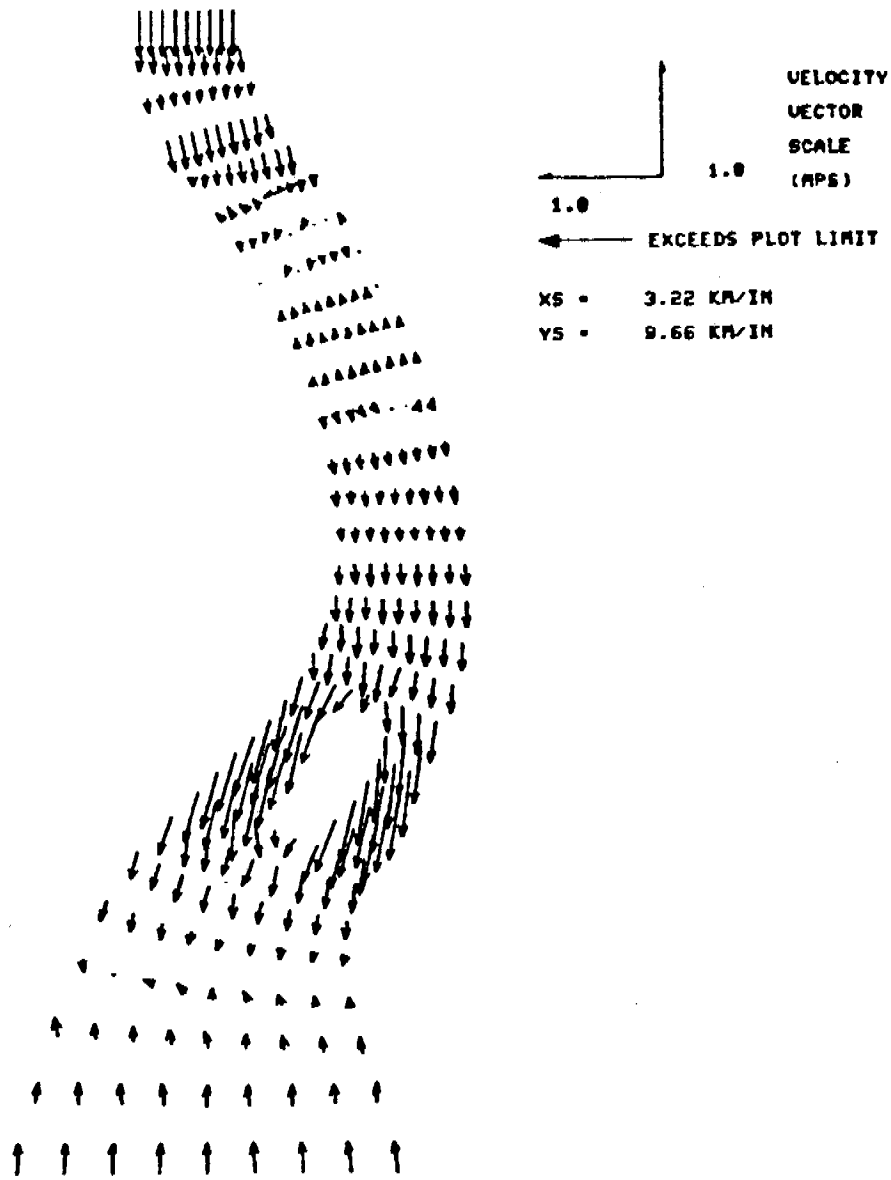


Figure 8. Velocity field after 12 hours with an ocean and a river boundary

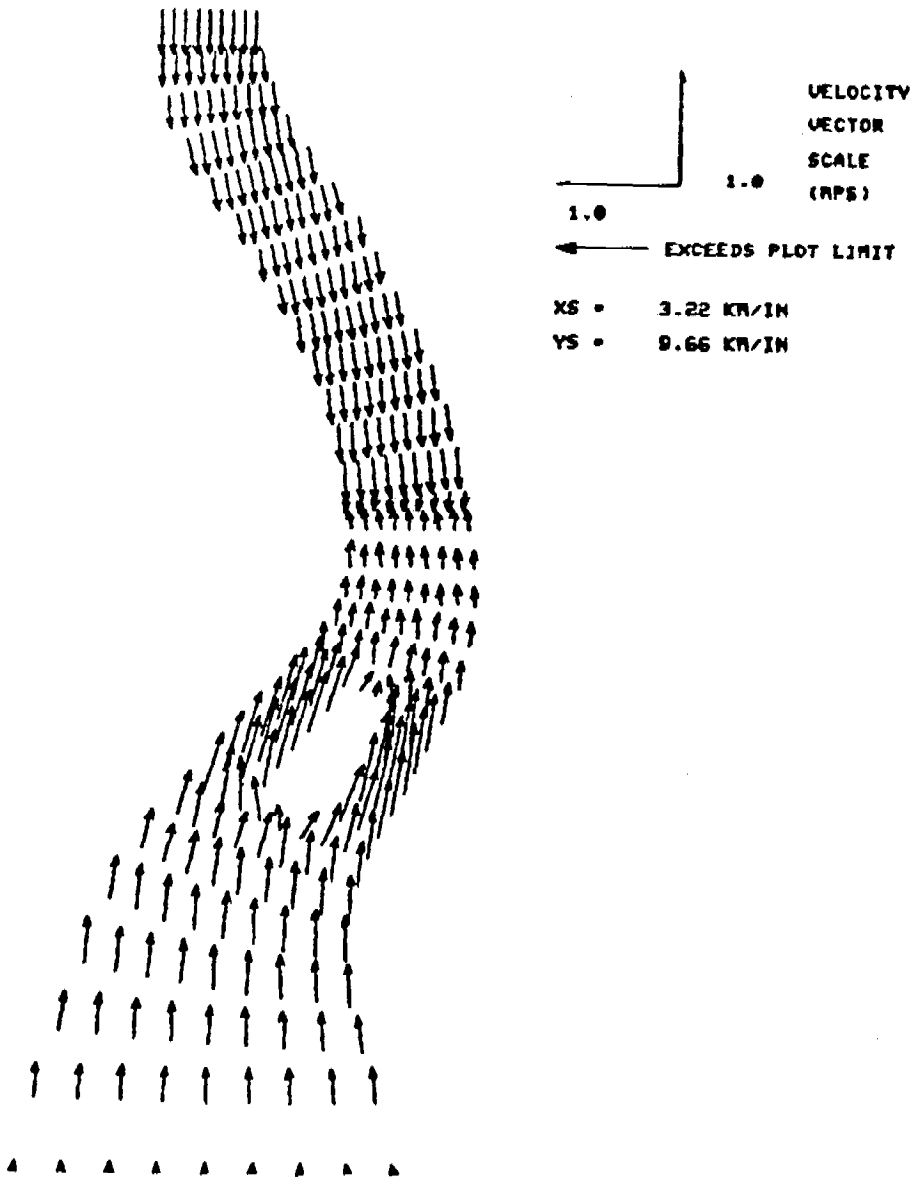


Figure 9. Velocity field after 16 hours with an ocean and a river boundary

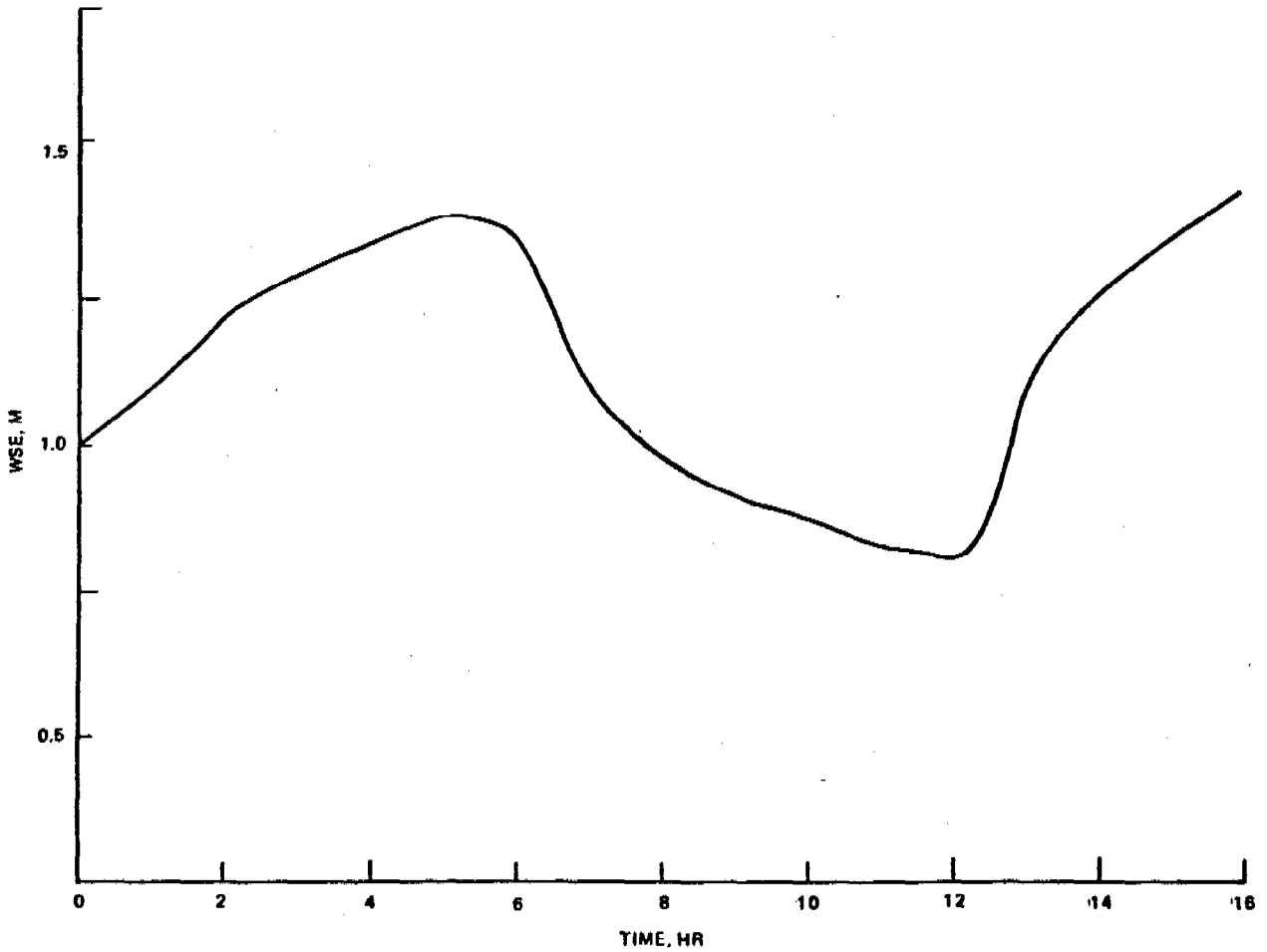


Figure 10. Water surface elevation at $\xi = 11$, $\eta = 11$ with ocean and river boundaries

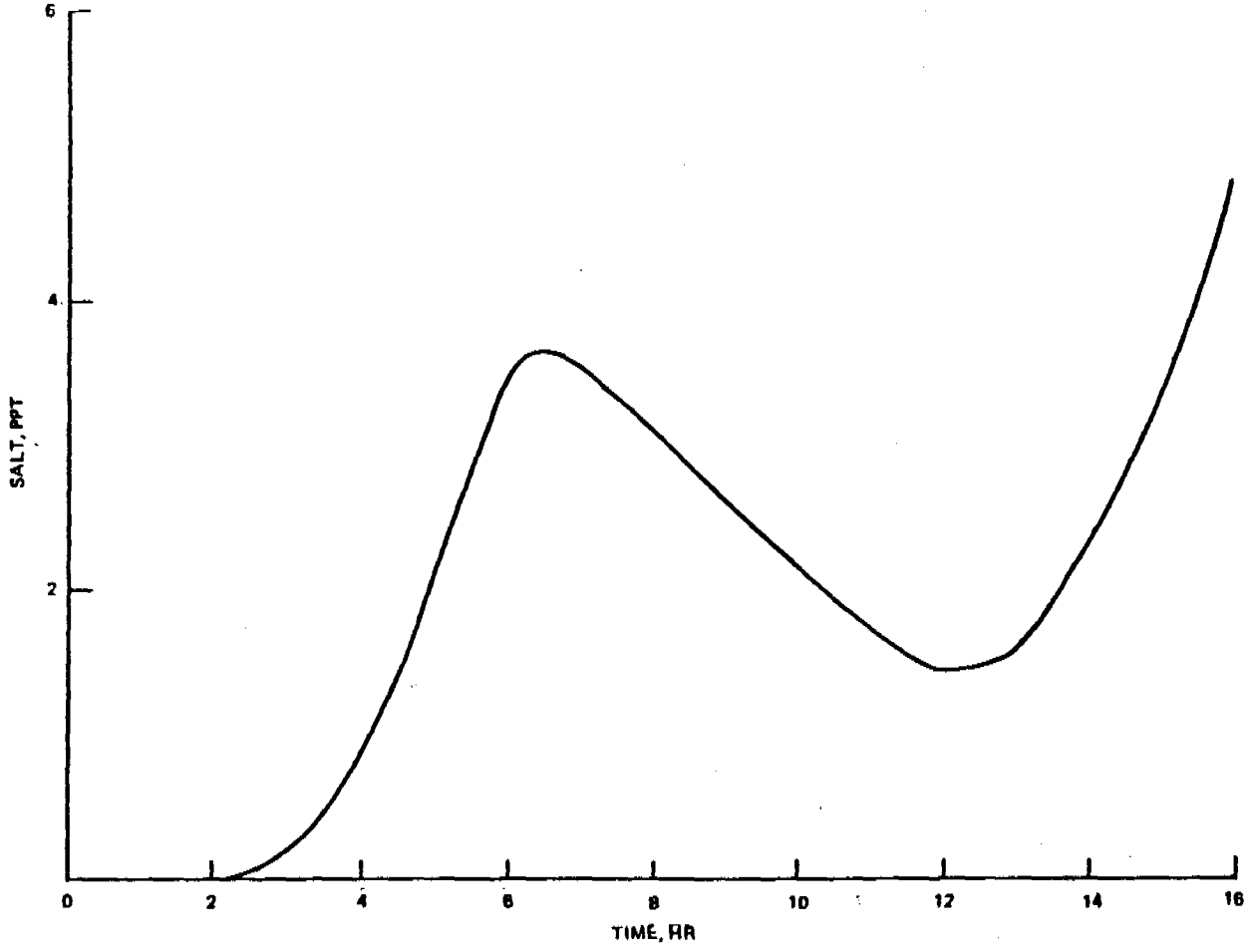


Figure 11. Salinity at $\xi = 11$, $\eta = 11$ with ocean and river boundaries

THE GEM CODE: DIRECT SOLUTIONS OF ELLIPTIC AND MIXED PROBLEMS WITH NON-SEPARABLE 5- AND 9-POINT OPERATORS[1]

Patrick J. Roache
Ecodynamics Research Associates, Inc.
P. O. Box 8172
Albuquerque, New Mexico 87198

ABSTRACT. Timing and accuracy tests of the GEM (General Elliptic Marching) Code are described. The GEM Code solves elliptic and mixed discretized two-dimensional partial differential equations by direct (non-iterative) spatial marching methods. Both 5-point and 9-point stencils may be solved, with no requirement that the coefficients be separable. Repeat solutions of 5-point operators are solved in a CPU time equivalent of 2 SOR iterations. The basic GEM depends on problem parameters (primarily a large cell aspect ratio $\Delta x/\Delta y$) to control the instability incurred in marching elliptic equations. A stabilized version uses the basic GEM in a multiple patching scheme to solve larger problems.

1. INTRODUCTION. Elliptic equations with non-separable coefficients arise in a variety of applications. Even the simple Poisson equation becomes non-separable when written in general non-orthogonal coordinates. The simplest second-order finite difference discretization then leads to a 9-point non-separable stencil.

Such problems are not solvable by fast direct methods such as odd-even reduction, Hackney's method, etc. Direct solution by brute-force banded Gaussian elimination is very expensive and limited in problem size by round-off error and storage.

Iterative methods are most often used for such problems, and multigrid methods in particular can be very effective. However, any iterative method depends on the effectiveness of the smoothing operator which depends on diagonal dominance. This deteriorates with the addition of first-derivative terms, either from the physical laws (e.g. convective terms) or from a non-orthogonal coordinate transformation. Some iterative methods (ADI) fail completely on even a simple problem like the Poisson equation in cartesian coordinates with a large cell aspect ratio $\Delta x/\Delta y$.

Marching methods are at present the only fast direct method of solving such problems. The GEM Code is a user-oriented package of subroutines which implement the marching methods for a fairly wide class of two-dimensional problems. This paper describes the results of timing and accuracy tests on the GEM (General Elliptic Marching) Code.

---

[1] Research sponsored by the U.S. Army Research Office.

2. THE GEM CODE. The GEM Code solves elliptic and mixed discretized two-dimensional partial differential equations by direct (non-iterative) spatial marching methods. Both 5-point and 9-point stencils may be solved, with no requirement that the coefficients be separable. For example, it solves the usual second-order accurate discretization of

$$aF_{xx} + bF_{yy} + cF_x + dF_y + eF_{xy} + fF = g \tag{1}$$

where a,b,...g are all functions of x and y. The methods used in the GEM Code are described in detail in (1). The basic code is based on "simple marching" and depends on problem parameters to control the instability incurred in marching elliptic equations; for realistic physical problems, this primarily depends on a large cell aspect ratio $\Delta x/\Delta y$.

Operation counts θ are given in some detail in (1). For a simple Poisson equation (5-point operator) without making use of symmetry, in a square array, this gives

$$\theta_{\text{initiation}} = 4M^3 + \frac{3}{2}M^2 \ , \ \theta_{\text{repeat}} = 14M^2 \tag{2}$$

The initiation count is less than that required to establish a single solution by point SOR, and the repeat count is less than 2 point SOR iterations. Since operation counts like these neglect many overhead and subscripting operations, it is necessary to validate them with actual timing tests, especially since Equation (2) indicates such remarkable efficiency.

When a 9-point operator is used, the marching solution proceeds a line at a time (like line SOR) and requires a tri-diagonal solution at j + 1 at each step in the march. This of course increases the operation counts, but not their order (i.e. repeat solutions are still optimal, with $\theta \propto M^2$). For other aspects of the method, see (1).

3. PROBLEM DESCRIPTION IN THE GEM CODE. The code is written with a "smart user" in mind, i.e. one who knows both finite differences and FORTRAN. The discretization of the continuum partial differential equation is left to the user. The code is written in Fortran IV, and the subroutine GEM solves the stencil

$$\begin{bmatrix} C_1 & C_2 & C_3 \\ C_4 & C_5 & C_6 \\ C_7 & C_8 & C_9 \end{bmatrix} \quad F_{ij} = C_{10} \tag{3}$$

All the coefficients $C_1, C_2 \ldots C_{10}$ are arrays stored in the labeled COMMON block GEMCOM. (The smart user could change some or all of these to BLANK COMMON for storage efficiency.)

The Subroutine Call is of the following form.

```
CALL GEM(INIT,F,IL,JL,ILD,N59,IPER,ICOR,
   NRC,RCOND,JMAR,JBOT,JTOP,NDBC,FDBC,                    (4)
      IPVT,CI,KLD,NC10,EMX)
```

INIT = 0 initiates only, = 1 initiates and solves, >1 backsolves only. The solution is stored in F. The problem size is ILxJL, with the actual first DIMENSION of the arrays being ILD. N59 = 5 or 9 gives the 5-point or 9-point operator solution. (If N59 = 5, the corner coefficients C_1,C_3,C_7 and C_9 are ignored.) For IPER = 1, periodic boundary conditions are used in the x-direction (normal to the marching direction y). ICOR is the number of corrective clean-up iterations used to reduce round-off error accumulation; usually, ICOR = 0 is used, but in some cases of marginal stability, ICOR = 1 or 2 may be used.

The LU decomposition and back-solve of the influence coefficient matrix is done through LINPACK subroutines (2) which are selected by the option indicator NRCOND. For NRCOND = 1, LINPACK routines are used which give an estimate of the inverse of the condition number RCOND. The time penalty is small, and in the author's experience, RCOND has been valuable as a debugging aid.

JMAR is an option indicator for the march direction, with ±1 giving a march in the +J or -J direction, respectively. This is a significant option because the stability of the marching method is directional. For an expanding coordinate system (typical of turbulent boundary layer calculations, for example) the stability is improved if the march proceeds from the coarse mesh to the fine mesh.

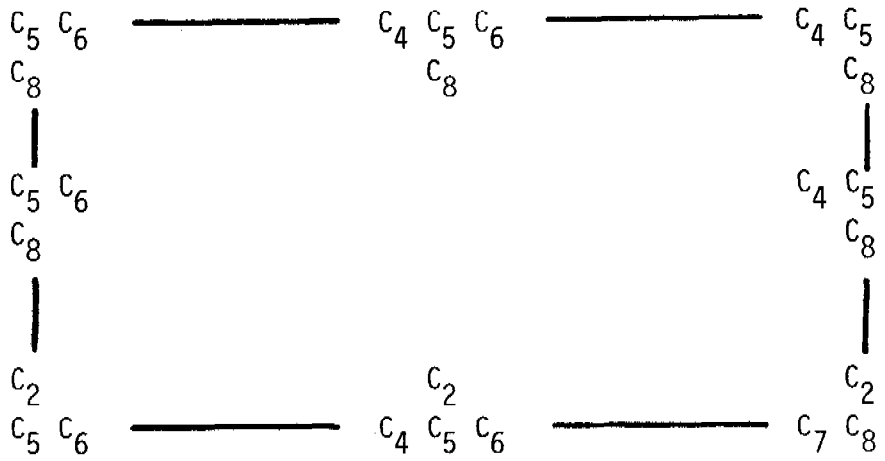
The next four arguments are primarily of use when GEM is driven by another code GEMPAT2 which stabilizes the solution by patching subregions together. Without stabilizing, JBOT = 1, JTOP = JL, NDBC = 0 and FDBC is ignored. In the stabilizing code, JBOT and JTOP define the extent of the subregion being solved, and NDBC = 1 indicates that the solution along the patching line has Dirichlet boundary conditions defined in the vector FDBC(IL).

IPVT and CI are work arrays, dimensioned IPVT(ILD) and CI(KLD,KLD) where KLD $\geq$ IL - 2.

NC10 is another option indicator primarily used when patching subroutines together. For NC10 = 0, the homogeneous problem C10(I,J) = 0 is solved, regardless of the values stored in C10.

Finally, the variable EMX is the output value of the maximum error in the solution, which occurs at the end of the march. A significant advantage of the marching method is that it will not lie to the user. The finite difference stencil is satisfied virtually to single precision everywhere except at the end of the march. The solution obtained can be viewed as a virtually exact solution of a problem with a boundary condition perturbed by EMX.

Boundary conditions are also specified by the coefficients C_1 - C_{10}, as indicated by the following stencils.



$$F = C_{10} \qquad (5)$$

For example, this stencil indicates that in the lower left-hand corner, at i = 1 and j = 1, the boundary condition is

$$\begin{aligned} &C2(1,1)*F(1,2) + C5(1,1)*F(1,1) \\ &+ C6(1,1)*F(2,1) \;=\; C10(1,1) \end{aligned} \qquad (6)$$

The general form of Equation (5) allows for all linear combinations of boundary conditions such as Dirichlet, Neuman, mixed, ratio of derivatives ($af_x + bf_y = c$), etc. However, the requirement for separability of boundary conditions in the marching y-direction dictates that C2 cannot be used at the side boundaries. In x, the periodic option indicator IPER = 1 overrides the matrix specification in Equation (5). For the 9-point operator, the periodic tridiagonal solution is obtained by the method of Reference (3).

The code is written so that none of the arrays C1 - C10 are passed to other subroutines in argument lists, and the unused portions of the arrays (e.g. C1 at J = JL) are not used for temporary storage. The idea here is to allow the user the option of saving the storage space by regenerating some or all of the coefficients as external or statement FUNCTION's in FORTRAN. It is only required of the user that he define the coefficient and remove that name from COMMON GEMCOM. It should be noted that the significant storage problem of the ten arrays C1 - C10 is not an aspect of the marching method, but simply follows from the problem description. The only significant storage penalty of the method is CI, giving essentially a x2 penalty compared to iterative methods.

4. TEST PROBLEMS AND RESULTS ON THE BASIC CODE. One set of test problems used pseudo-random number generation for all coefficients, which was useful in debugging all the options. A second set used a simple Poisson equation modified by a cross-derivative term $VC*f_{xy}$, formulated with centered second-order differences.

The test problems were run on a CDC 6600. A sampling of the results is presented in Table 1. Note that the 81 x 81 mesh problem for the 5-point

operator initiates in ~ 1 ms/cell, the equivalent of 64 Point SOR iterations, and solves repeat solutions in the equivalent of 2 Point SOR iterations. For the simple Poisson equation with $\Delta x/\Delta y = 10$, the maximum residual error is 3.9×10^{-6}.

The 9-point operator with non-periodic boundary condition requires about 67% more initiation time and about 31% more repeat time. With periodic boundary conditions, the 9-point operator requires about 3.2 x as long for initiation and about 2.6 x as long for repeat solutions.

5. THE STABILIZING CODE GEMPAT2. The method of stabilizing selected from several available alternatives (1) is the multiple patching method. The problem size in J (i.e., maximum JL for given cell aspect ratio, etc.) is doubled by breaking the solution into two subregions separated at J = JPATCH. With guessed Dirichlet boundary conditions at JPATCH, each subregion is solved directly using basic GEM. This solution gives non-zero residuals along JPATCH. The new Dirichlet conditions along JPATCH are then solved directly so as to zero these residuals. The technique is a capacity matrix or influence coefficient matrix method which is not essentially connected to marching methods. The patching matrix is established in an initiation procedure which requires IL-2 homogeneous solutions with unit-perturbed Dirichlet conditions along JPATCH; hence, the homogeneous over-ride option NC10 in GEM.

This procedure for a 2-patch solution is incorporated into the subroutine GEMPAT2, which then calls GEM. Although several of the options in GEM are not of interest except for use with GEMPAT2, it was decided to have only one version of GEM available. The possible confusion arising from the unused options seems outweighed by the advantage of having only one version of GEM to document and maintain. Similarly, GEMPAT4, under development, is a code to implement the patching procedure for a 4-patch solution, and it will call the only version of GEMPAT2.

6. TIMING TESTS OF GEMPAT2. The patching method for a 2-patch solution requires two of the CI matrices (one for each subregion) and an additional storage penalty for the patching matrix, and so the storage penalty is 3xILxJL, compared to ILxJL for the single-region solution by the basic GEM. The theoretical operation count given in (1) may be expected to deteriorate in accuracy, especially for initiation, as the number of patches increases.

The GEMPAT2 code is still being refined, but the timing tests on the initial version are very encouraging. For the 5-point operator with non-periodic boundary conditions on a 71 x 71 grid, the GEMPAT2 code initializes in the equivalent of 265 SOR iterations, a factor of 3.1 over the single region solution. This is significantly better than the value of 4.3 predicted by the operation count (1). Repeat solutions are obtained in 5.2 SOR iterations, a factor of 2.1 over the single region solution, in agreement with the operation count.

For the 4-patch solution, operation counts indicate penalty factors of ~10 for initiation and 4.3 for repeat solutions. For further patching, the growing initiation and storage penalties make the method unattractive, and we do not plan a code above GEMPAT4.

Unlike the single-region solution, in which the error is virtually confined to the boundary at the end of the march, the patched solutions also have errors (non-zero residuals) along the patching lines. However, the patching matrix is usually well-conditioned and this error is acceptable in the problems tested to date. A more complete investigation of the errors and timing tests of the codes GEMPAT2 and GEMPAT4 is forthcoming.

Table 1. Timing tests of the GEM Code on a CDC 6600 with Level 2 Optimization of the FORTRAN IV Code. "init" refers to initiation times, "rep" refers to repeat solution times, "SOR" refers to times for a single iteration of a Point SOR method including a convergence test but without boundary calculations, θ refers to predictions based on theoretical operation counts. Total times are in seconds, times/cell are in milliseconds, based on the minimum of three consecutive runs which included one initiation and five repeats.

problem grid (operator)	31 x 31 (5 pt.)	51 x 51 (5 pt.)	81 x 81 (5 pt.)	81 x 81 (9 pt.)	Periodic 81 x 81 (9 pt.)
init time	0.42	1.74	6.61	15.75	30.24
init time/cell	0.47	0.70	1.03	2.46	4.73
init time/SOR	29.7	42.8	64.3	107.1	205.7
rep time	0.035	0.089	0.206	0.384	0.750
rep time/cell	0.039	0.036	0.032	0.060	0.117
rep time/SOR	2.48	2.19	2.00	2.61	5.10
init time/rep time	12.0	19.6	32.1	41.0	40.3
% error, θ_{rep}	-32	-22	-21	-28	-32
% error, $\theta_{init}/\theta_{rep}$	-22	-19	-15	-5	-3

REFERENCES

(1) Roache, P. J., "Marching Methods for Elliptic Problems: Part 1", Numerical Heat Transfer, Vol. 1, No. 1, 1978, pp. 1-25. "Part 2", Vol. 1, No. 2, pp. 163-181. "Part 3", Vol. 1, No. 2, pp. 183-201.

(2) Dongarra, J. J., Moler, C. B., Bunch, J. R., and Stewart, G. W., "LINPACK User's Guide", SIAM, Philadelphia, 1979.

(3) Roache, P. J. and Temperton, C., "Algorithms for the Solution of Cyclic Tridiagonal Systems with General Coefficients", J. Comp. Phys., to appear.

A NEW VARIATIONAL METHOD FOR INITIAL VALUE PROBLEMS, USING PIECEWISE HERMITE POLYNOMIAL SPLINE FUNCTIONS

C. N. Shen* and Julian J. Wu
U.S. Army Armament Research and Development Command
Large Caliber Weapon Systems Laboratory
Benet Weapons Laboratory
Watervliet, NY 12189

ABSTRACT. A variational principle for a functional can be found which satisfies both the original system and its adjoint system. The variations of this functional give no boundary terms if the bilinear concomitant of the systems vanishes. For a second order time varying initial value problem, one can adjust the boundary conditions of the adjoint system in terms of the boundary conditions of the original system so that the bilinear concomitant is identically zero. An expression for the variation of the functional is derived which contains only the terms involving the variations of the adjoint variable and its derivative, but no variation of its second derivative. The variations of the adjoint variable and its derivative are found to be zeroes at the final conditions, just as the variations of the original variable and its derivative are zero at the starting (initial) conditions. This implies that we are able to solve the problem in one direction without worrying about the conditions at the other end as the initial value problem should be. The algorithm is much more simplified than in the past. An example is given to show the procedures of this new variational method.

I. INTRODUCTION. Variational principles apply mostly to boundary problems where eigenvalues are sought. It is seldom used for initial value problems alone where the far end conditions are neither known nor specified. If we use discrete methods to solve an initial value problem, such as finite difference method, only the initial conditions should be given. In the same way, if we employ variational method with spline functions, we should not be concerned with the far end conditions. This paper gives a procedure to find a recursive solution of an initial value problem by variational methods using the cubic hermite polynomial spline functions.

Let us consider a dynamical system governed by the following equation:

$$L(t)y_a(t) = -Q(t) \tag{1}$$

with appropriate boundary conditions. In the above equation L is a linear operator, y_a is the dependent variable, Q is a forcing function, and t is the independent variable.

*Also Professor at Rensselaer Polytechnic Institute, Troy, NY.

Some integral property in the form of a linear functional of the variable [1], such as the inner product of an adjoint forcing function $\bar{Q}$ and the solution of Eq. (1) can be used for estimation.

$$G[y_a] = \int_{t_o}^{t_b} \bar{Q} y_a dt \tag{2}$$

The estimate y which differs from the solution y_a of Eq. (1) by an increment δy can be written as

$$\delta y = y - y_a \tag{3}$$

Then the estimate y becomes

$$G[y] = \int_{t_o}^{t_b} \bar{Q} y dt = \int_{t_o}^{t_b} \bar{Q} y_a dt + \int_{t_o}^{t_b} \bar{Q} \delta y dt$$

$$= G[ya] + \int_{t_o}^{t_b} \bar{Q} \delta y dt \tag{4}$$

which is in error to first order in δy and $\bar{Q}$.

II. THE VARIATION PRINCIPLE. A more accurate estimate can be made by constructing a variational principle [1] for Eq. (2). By using the adjoint variable y as a Lagrange multiple and Eq. (1) as an addition to G[y] we have

$$J[y,\bar{y}] = G[y] + \int_{t_o}^{t_b} \bar{y} (Q+Ly) dt$$

$$= \int_{t_o}^{t_b} \bar{Q} y dt + \int_{t_o}^{t_b} \bar{y} Q dt + \int_{t_o}^{t_b} \bar{y} L y dt \tag{5}$$

In order that J be a variational principle for G the following requirements must be satisfied.

(a) J is stationary about the function y_s which satisfy the relation in Eq. (1).

$$L(t) y_s = -Q(t) \tag{6}$$

(b) The stationary value of J deduced from Eqs. (2) through (5) is

$$J[\bar{y},y] = G[y_s] + G[y_a] \tag{7}$$

Consider first the stationarity of J by taking the variation

$$\delta J = \delta\{\int_{t_o}^{t_b} \bar{Q}ydt + \int_{t_o}^{t_b} \bar{y}Qdt + \int_{t_o}^{t_b} \bar{y}Lydt\}$$

$$\delta J = \int_{t_o}^{t_b} \bar{\delta y}(Ly+Q)dt + \int_{t_o}^{t_b} [\bar{Q}\delta y + \bar{y}L\delta y]dt \tag{8}$$

We will make an effort later to impose certain conditions in order that the following equality holds:

$$\int_{t_o}^{t_b} \bar{y}L\delta ydt = \int_{t_o}^{t_b} \delta y\bar{\bar{L}}ydt \tag{9}$$

where $\bar{L}(t)$ is an adjoint operator.

By combining Eqs. (8) and (9) one obtains

$$\delta J = \int_{t_o}^{t_b} \bar{\delta y}(Ly+Q)dt + \int_{t_o}^{t_b} \delta y[\bar{\bar{L}}\bar{y} + \bar{Q}]dt = 0 \tag{10}$$

Since the variations $\bar{\delta y}$ and δy are arbitrary it leads to the requirement that the stationary values y_s and $\bar{y}_s$ must satisfy.

$$Ly_s = -Q \tag{11}$$

$$\bar{\bar{L}}\bar{y}s = -\bar{Q} \tag{12}$$

Since Eq. (11) is the same as Eq. (6), therefore J is stationary about the function y_s. Equation (12) is the adjoint equation in terms of the adjoint operator $\bar{L}$, the adjoint variable $\bar{y}$, and the adjoint forcing function $\bar{Q}$.

Using the relation in Eq. (11) for the stationary value of J from Eq. (5) we have

$$J[\bar{y}_s\ y_s] = \int_{t_o}^{t_b} \bar{Q}y_s dt + \int_{t_o}^{t_b} \bar{y}_s(Q+Ly_s)dt = G[y_s] \tag{13}$$

Since J is stationary and $\delta J \to 0$ then

$$G[y_s] \to G[y_a] \tag{14}$$

which is the requirement given in Eq. (7).

It is noted that Eq. (10) contains no boundary terms to be satisfied. This bears an important point in the future discussion of the initial value problems.

III. BILINEAR CONCOMITANT. The assumed equality in Eq. (9) is discussed here by considering the following bilinear concomitant [1]:

$$D = \int_{t_o}^{t_b} \bar{y}Ly\,dt - \int_{t_o}^{t_b} y\bar{L}\bar{y}\,dt \tag{15}$$

The above expression can also be written in terms of boundary conditions at $t = t_o$ and $t = t_b$. It is assumed that these boundary conditions are assigned in such a way that the above bilinear concomitant is identically zero, i.e.,

$$D \equiv 0 \tag{16}$$

Then the first variations of D also vanish.

$$\delta D = \delta D(\delta\bar{y}) + \delta D(\delta y) = 0 \tag{17}$$

Since $\delta\bar{y}$ and δy are independent of each other, then

$$\delta D(\delta\bar{y}) = \int_{t_o}^{t_b} \delta\bar{y}Ly\,dt - \int_{t_o}^{t_b} y\bar{L}\delta\bar{y}\,dt = 0 \tag{18}$$

and

$$\delta D(\delta y) = \int_{t_o}^{t_b} \bar{y}L\delta y\,dt - \int_{t_o}^{t_b} \delta y\bar{L}\bar{y}\,dt = 0 \tag{19}$$

Equation (19) is identical to Eq. (9), which is the assumed equality previously. This implies that if Eq. (16) is true then Eq. (9) or (19) is automatically true.

IV. INTEGRAL OF BILINEAR EXPRESSION. The integral of a function is given as

$$I = \int_{t_o}^{t_b} \psi(y\bar{y})\,dt \tag{20}$$

where $\psi(y\bar{y})$ is an arbitrary bilinear expression [2] in the form

$$\psi(y\bar{y}) = \alpha y'\bar{y}' + \beta y'\bar{y} + \gamma y\bar{y}' + \varepsilon y\bar{y} \tag{21}$$

The prime (') in the above expression denotes (d/dt).

Equation (20) can be integrated by parts. Two different forms of integration and end conditions may be obtained as follows.

$$I = -\int_{t_o}^{t_b} \bar{y}Ly\,dt + (\alpha y'+\gamma y)\bar{y}\Big|_{t_o}^{t_b} \tag{22}$$

or

$$I = -\int_{t_o}^{t_b} y\bar{\bar{L}}\bar{y}\,dt + (\alpha\bar{y}'+\beta\bar{y})y\Big|_{t_o}^{t_b} \tag{23}$$

where the differential expressions are

$$Ly = (\alpha y')' - \beta y' + (\gamma y)' - \varepsilon y \tag{24}$$

$$\bar{\bar{L}}\bar{y} = (\alpha\bar{y}')' + (\beta\bar{y})' - \gamma\bar{y}' - \varepsilon\bar{y} \tag{25}$$

The bilinear concomitant given in Eq. (15) can now be expressed in terms of the function values and their derivatives at the end points by equating Eqs. (22) and (23).

$$D = [\alpha(y'\bar{y}-y'\bar{y}) - (\gamma-\beta)y\bar{y}]\Big|_{t_o}^{t_b} \tag{26}$$

V. END CONDITIONS FOR THE ADJOINT SYSTEM. In order to satisfy the expression $D \equiv 0$ in Eqs. (15) and (16) the end terms in Eq. (26) must vanish. Thus it requires

$$\alpha_b(y_b'\bar{y}_b-\bar{y}_b'y_b) - \alpha_o(y_o'\bar{y}_o-\bar{y}_o'y_o) - (\gamma_b-\beta_b)y_b\bar{y}_b + (\gamma_o-\beta_o)y_o\bar{y}_o \equiv 0 \tag{27}$$

Equation (27) can be satisfied identically if the end conditions of the adjoint system are proportional to the end conditions of the original system as follows:

$$\bar{y}_b = (\gamma_o-\beta_o)ky_o \tag{28a}$$

$$\bar{y}_o = (\gamma_b-\beta_b)ky_b \tag{28b}$$

$$\bar{y}_b' = -\alpha_b^{-1}\alpha_o(\gamma_b-\beta_b)ky_o' \tag{28c}$$

$$\bar{y}_b' = -\alpha_o^{-1}\alpha_b(\gamma_o-\beta_o)ky_b' \tag{28d}$$

where k is a constant.

The above expressions give the required end conditions for the adjoint system in terms of that of the original system. Thus from Eqs. (15) and (16):

$$D = \int_{t_o}^{t_b} \bar{y}Ly\,dt - \int_{t_o}^{t_b} y\bar{\bar{L}}\bar{y}\,dt \equiv 0 \tag{29}$$

To summarize, if one can make the end conditions of the adjoint system satisfy the relationship in Eq. (28), the bilinear concomitent D vanishes. The variation in Eq. (10) is then valid.

It is also noted that the variation in Eq. (10) has no far end terms which simplify the computation. This is because the far end terms may cause certain difficulties in many computational schemes on a number of variational methods.

VI. THE FIRST VARIATION. Since the variations $\delta\bar{y}$ and δy are independent to each other, we take the first half of Eq. (10) as

$$\delta J(\delta\bar{y}) = \int_{t_o}^{t_b} \delta\bar{y} L y dt + \int_{t_o}^{t_b} \delta\bar{y} Q dt = 0 \tag{30}$$

Equation (30) is not in a ready form for estimation. We prefer to use δI which can be obtained from the bilinear expression I given in Eqs. (20) and (21). Let

$$\delta I = \delta I(\delta\bar{y}) + \delta I(\delta y) \tag{31}$$

The first part of the above expression can be derived from Eqs. (20) and (21) as

$$\delta I(\delta\bar{y}) = \int_{t_o}^{t_b} [(\alpha y'+\gamma y)\delta\bar{y}' + (\beta y'+\varepsilon y)\delta\bar{y}]dt \tag{32}$$

Integrating by parts one obtains

$$\delta I(\delta\bar{y}) = (\alpha y'+\gamma y)\delta\bar{y}\Big|_{t_o}^{t_b} - \int_{t_o}^{t_b} \delta\bar{y}[(\alpha y'+\gamma y)' - (\beta y'+\varepsilon y)]dt \tag{33}$$

It is recognized that the integrand in the last term of the above formulae is Ly. Solving for the last term we have

$$\int_{t_o}^{t_b} \delta\bar{y} L y dt = (\alpha y'+\gamma y)\delta\bar{y}\Big|_{t_o}^{t_b} - \delta I(\delta\bar{y}) \tag{34}$$

Substituting Eq. (32) into (34) and then Eq. (34) into (30) one obtains

$$\begin{aligned}\delta J(\delta\bar{y}) = {} & (\alpha_b y_b'+\gamma_b y_o)\delta\bar{y}_b - (\alpha_o y_o'+\gamma_o y_o)\delta\bar{y}_o \\ & - \int_{t_o}^{t_b} [(\alpha y'+\gamma y)\delta\bar{y}' + (\beta y'+\varepsilon y)\delta\bar{y}]dt \\ & + \int_{t_o}^{t_b} \delta\bar{y} Q dt = 0\end{aligned} \tag{35}$$

The above equation contains only $\bar{\delta y}$ and $\bar{\delta y}'$ and none of the variation of the higher derivative such as $\bar{\delta y}''$ for a second order system. The dependent variable also contains only y and y' and none of the higher derivative such as y" for a second order system.

VII. ADJOINT VARIABLE FAR END VALUE FOR INITIAL VALUE PROBLEMS. For a second order system the initial values of the function and its first derivative are given, i.e., y_o and y_o' are known in Eq. (28). The far end values for the adjoint system y_b and y_b' are found from Eqs. (28a) and (28c). Since the variation of a constant is zero, then

$$\delta y_o = \delta y_o' = 0 \tag{36}$$

and

$$\bar{\delta y}_b = \bar{\delta y}_b' = 0 \tag{37}$$

The conclusion $\bar{\delta y}_b = 0$ in Eq. (37) is important in that the first term at the right side of Eq. (35) vanishes. Thus the coefficient of $\bar{\delta y}_b$ is not necessarily zero. This implies that the function y_b and its derivative y_b' at the far end are not related as such. By not using any local boundary conditions at the far end, the computation can start at the near end and carry on in one direction.

Thus Eq. (35) is simplified to

$$\delta J(\bar{\delta y}) = -(\gamma_o y_o+\alpha_o y_o')\bar{\delta y}_o + \int_{t_o}^{t_b} \bar{\delta y} Q dt$$

$$-\int_{t_o}^{t_b} [(\varepsilon y+\beta y')\bar{\delta y} + (\gamma y+\alpha y')\bar{\delta y}']dt \tag{38}$$

It is noted that the above equation does not have boundary terms to be satisfied at the far end at time t_b. This is consistent with the notion of "initial value problem" physically.

VIII. TRANSFORMATION OF COORDINATES. The integral sign in Eq. (38) can be converted into a summation sign if discrete intervals for integration are used. Since the analysis is an initial value problem, without losing any generality we may let

$$t_o = 0 \quad \text{and} \quad t_b = 1 \quad , \tag{39}$$

that is the independent variable is within the interval

$$0 \leq t \leq 1 \tag{40}$$

Equation (38) can be discretized by letting

$$\xi = Kt - m+1 \tag{41}$$

$$0 \leq \xi \leq 1, \quad 0 \leq t \leq 1, \quad m = 1,2,\ldots K \tag{42}$$

where K is the number of intervals.

Thus

$$d\xi = Kdt \qquad dt = K^{-1}d\xi \tag{43}$$

The differential relationship is

$$\frac{dy}{dt} = K\frac{dy}{d\xi} \tag{44}$$

or

$$y' = K\dot{y} \tag{45}$$

where

$$(\dot{}) = \frac{d}{d\xi}(\ ) \tag{46}$$

Then Eq. (38) becomes

$$\delta J(\delta y) = 0$$

$$= -(\gamma_o y_o+\alpha_o K\dot{y}_o)\delta\bar{y}_o + \sum_{m=1}^{K}\int_0^1 \delta\bar{y}^{(m)}QK^{-1}d\xi$$

$$-\sum_{m=1}^{K}\int_0^1 [(\varepsilon y^{(m)}+\beta K\dot{y}^{(m)})\delta\bar{y}^{(m)} + \{(\gamma y^{(m)}+\alpha K\dot{y}^{(m)}\}K\delta\dot{\bar{y}}^{(m)}]K^{-1}d\xi \tag{47}$$

IX. PIECEWISE SPLINE FUNCTIONS. We may express the variables $y^{(m)}$ and $y^{(m)}(\xi)$ in terms of piecewise spline function $a^T(\xi)$ and the node point functions $Y^{(m)}$ and $Y^{(m)}$ as follows.

$$y^{(m)}(\xi) = a^T(\xi)\ Y^{(m)} \qquad \delta y^{(m)} = [\delta Y^{(m)}]^T a(\xi) \tag{48}$$

$$\dot{y}^{(m)}(\xi) = \dot{a}^T(\xi)\ Y^{(m)} \qquad \delta\dot{y}^{(m)} = [\delta Y^{(m)}]^T\dot{a}(\xi) \tag{49}$$

$$\bar{y}^{(m)}(\xi) = a^T(\xi)\bar{Y}^{(m)} \qquad \delta\bar{y}^{(m)} = [\delta\bar{Y}^{(m)}]^T a(\xi) \tag{50}$$

$$\dot{\bar{y}}^{(m)}(\xi) = \dot{a}^T(\xi)\bar{Y}^{(m)} \qquad \delta\dot{\bar{y}}^{(m)} = [\delta\bar{Y}^{(m)}]^T\dot{a}(\xi) \tag{51}$$

$$m = 1,2,\ldots K$$

$$y_o \overset{\Delta}{=} a^T(1)Y^{(o)} \tag{52}$$

$$\dot{y}_o \overset{\Delta}{=} \dot{a}^T(1)Y^{(o)} \tag{53}$$

$$\delta\bar{y}_o \overset{\Delta}{=} \delta\bar{Y}^{(o)}a(1) \tag{54}$$

If Eqs. (48) through (54) are substituted into Eq. (47) one obtains

$$0 = -[\delta\bar{Y}^{(o)}]^T a(1)[\gamma_o a^T(1) + \alpha_o K\dot{a}^T(1)]Y^{(o)}$$

$$+ \sum_{m=1}^{K} [\delta\bar{Y}^{(m)}]^T K^{-1} \int_0^1 a(\xi)Q d\xi$$

$$- \sum_{m=1}^{K} [\delta Y^{(m)}]^T \int_0^1 a(\xi)[\varepsilon K^{-1} a^T(\xi) + \beta\dot{a}^T(\xi)]d\xi \; Y^{(m)}$$

$$- \sum_{m=1}^{K} [\delta\bar{Y}^{(m)}]^T \int_0^1 \dot{a}(\xi)[\gamma a^T(\xi) + \alpha K\dot{a}^T(\xi)]d\xi \; Y^{(m)} \qquad (55)$$

This simplifies to

$$0 = -[\delta\bar{Y}^{(o)}]^T a(1)[\gamma_o a^T(1) + \alpha_o K\dot{a}^T(1)]Y^{(o)}$$

$$+ \sum_{m=1}^{K} [\delta\bar{Y}^{(m)}]^T q^{(m)} - \sum_{m=1}^{K} [\delta\bar{Y}^{(m)}]^T p^{(m)} \; Y^{(m)} \qquad (56)$$

where

$$q^{(m)} = K^{-1} \int_0^1 a(\xi)Q(\xi)d\xi$$

$$= [q_1^{(m)}, \; q_2^{(m)}, \; q_3^{(m)}, \; q_4^{(m)}]^T \qquad (57)$$

and

$$p^{(m)} = \int_0^1 \{a(\xi)[\varepsilon^{(m)} K^{-1} a^T(\xi) + \beta^{(m)}\dot{a}^T(\xi)] + \dot{a}(\xi)[\gamma^{(m)} a^T(\xi) + \alpha^{(m)} K\dot{a}^T(\xi)]\}d\xi$$

$$= \varepsilon^{(m)} K^{-1} B + \beta^{(m)} C + \gamma^{(m)} D + \alpha^{(m)} KE \qquad (58)$$

or

$$[p_{ij}^{(m)}] = \varepsilon^{(m)} K^{-1} [b_{ij}] + \beta^{(m)} [c_{ij}] + \gamma^{(m)} [d_{ij}] + \alpha^{(m)} K[e_{ij}] \qquad (59)$$

where

$$B = [b_{ij}] = \int_0^1 a(\xi)a^T(\xi)d\xi \qquad (60)$$

$$C = [c_{ij}] = \int_0^1 a(\xi)\dot{a}^T(\xi)d\xi \qquad (61)$$

$$D = [d_{ij}] = \int_0^1 \dot{a}(\xi)a^T(\xi)d\xi \qquad (62)$$

$$E = [e_{ij}] = \int_0^1 \dot{a}(\xi)\dot{a}^T(\xi)d\xi \qquad (63)$$

X. CUBIC HERMITE POLYNOMIAL SPLINE. The cubic Hermite polynomial spline is continuous in the functional values and its first derivatives across the nodes. Since we have no second derivatives for $a(\xi)$ in Eqs. (58) to (63), no higher order spline is necessary for this problem.

The cubic Hermite polynomial gives

$$a(\xi) = \begin{bmatrix} a_1(\xi) = 1-3\xi^2+2\xi^3 \\ a_2(\xi) = \xi-2\xi^2+\xi^3 \\ a_3(\xi) = 3\xi^2-2\xi^3 \\ a4(\xi) = -\xi^2+\xi^3 \end{bmatrix} \tag{64}$$

whose derivatives are

$$\dot{a}(\xi) = \begin{bmatrix} \dot{a}_1(\xi) = -6\xi+6\xi^2 \\ \dot{a}_2(\xi) = 1-4\xi+3\xi^2 \\ \dot{a}_3(\xi) = 6\xi-6\xi^2 \\ \dot{a}4(\xi) = -2\xi+3\xi^2 \end{bmatrix} \tag{65}$$

It is obvious from the above equations that the node point values are

$$a(0) = [a_1(0)\ a_2(0)\ a_3(0)\ a_4(0)]^T$$

$$= [\ 1\ \ 0\ \ 0\ \ 0\ ]^T \tag{66a}$$

$$\dot{a}(0) = [\dot{a}_1(0)\ \dot{a}_2(0)\ \dot{a}_3(0)\ \dot{a}_4(0)]^T$$

$$= [\ 0\ \ 1\ \ 0\ \ 0\ ]^T \tag{66b}$$

$$a(1) = [a_1(1)\ a_2(1)\ a_3(1)\ a_4(1)]^T$$

$$= [\ 0\ \ 0\ \ 1\ \ 0\ ]^T \tag{66c}$$

$$\dot{a}(1) = [\dot{a}_1(1)\ \dot{a}_2(1)\ \dot{a}_3(1)\ \dot{a}_4(1)]^T$$

$$= [\ 0\ \ 0\ \ 0\ \ 1\ ]^T \tag{66d}$$

We wish to form a vector whose components are taken from the function and its derivative at the left node and then the same at the right node. From Eqs. (48), (49), and (66) we have

$$\begin{bmatrix} y^{(m)}(0) \\ \dot{y}^{(m)}(0) \\ y^{(m)}(1) \\ \dot{y}^{(m)}(1) \end{bmatrix} = \begin{bmatrix} a^T(0)Y^{(m)} \\ \dot{a}^T(0)Y^{(m)} \\ a^T(1)Y^{(m)} \\ \dot{a}^T(1)Y^{(m)} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad Y^{(m)} = Y^{(m)} \tag{67}$$

If we define

$$Y^{(m)} = [Y_1^{(m)} \; Y_2^{(m)} \; Y_3^{(m)} \; Y_4^{(m)}]^T \tag{68}$$

Then

$$Y_1^{(m)} = y^{(m)}(0) \tag{69a}$$

$$Y_2^{(m)} = \dot{y}^{(m)}(0) \tag{69b}$$

$$Y_3^{(m)} = y^{(m)}(1) \tag{69c}$$

$$Y_4^{(m)} = \dot{y}^{(m)}(1) \tag{69d}$$

The above implies that the same node point has been represented by two notations as follows

$$y^{(m+1)}(0) = y^{(m)}(1) \tag{70a}$$

$$\dot{y}^{(m+1)}(0) = \dot{y}^{(m)}(1) \tag{70b}$$

By expanding Eq. (68) for different m one obtains

$$Y^{(0)} = [0 \; 0 \; Y_3^{(0)} \; Y_4^{(0)}]^T = [0 \; 0 \; y^{(0)}(1) \; \dot{y}^{(0)}(1)]^T \tag{71a}$$

$$Y^{(1)} = [Y_1^{(1)} \; Y_2^{(1)} \; Y_3^{(1)} \; Y_4^{(1)}]^T = [y^{(1)}(0) \; \dot{y}^{(1)}(0) \; y^{(1)}(1) \; \dot{y}^{(1)}(1)]^T \tag{71b}$$

$$Y^{(m)} = [Y_1^{(m)} \; Y_2^{(m)} \; Y_3^{(m)} \; Y_4^{(m)}]^T = [y^{(m)}(0) \; \dot{y}^{(m)}(0) \; y^{(m)}(1) \; \dot{y}^{(m)}(1)]^T \tag{71c}$$

$$Y^{(m+1)} = [Y_1^{(m+1)} \; Y_2^{(m+1)} \; Y_3^{(m+1)} \; Y_4^{(m+1)}]^T = [y^{(m+1)}(0) \; \dot{y}^{(m+1)}(0) \; y^{(m+1)}(1) \; \dot{y}^{(m+1)}(1)]^T \tag{71d}$$

Thus we have

$$Y_1^{(m+1)} = Y_3^{(m)} \tag{72a}$$

and

$$Y_2^{(m+1)} = Y_4^{(m)} \tag{72b}$$

for m = 0,1,2,...K.

Similar to the above equation one can prove from Eqs. (50) and (51) that the adjoint variations are

$$\delta\bar{Y}_1^{(m+1)} = \delta\bar{Y}_3^{(m)} \tag{73a}$$

and

$$\delta\bar{Y}_2^{(m+1)} = \delta\bar{Y}_4^{(m)} \tag{73b}$$

XI. METHOD OF SOLUTIONS. First we take the last term of Eq. (56) which is

$$R_3 = -\sum_{m=1}^{K} [\delta Y_1^{(m)}\ \delta Y_2^{(m)}\ \delta Y_3^{(m)}\ \delta Y_4^{(m)}][p_{ij}^{(m)}][Y_1^{(m)}\ Y_2^{(m)}\ Y_3^{(m)}\ Y_4^{(m)}]^T \tag{74}$$

Using the relationship from Eqs. (72) and (73) gives

$$\begin{aligned} R_3 = -\sum_{m=1}^{K} \{ & [p_{11}^{(m)}Y_3^{(m+1)} + p_{12}^{(m)}Y_4^{(m-1)} + p_{13}^{(m)}Y_3^{(m)} + p_{14}^{(m)}Y_4^{(m)}]\delta\bar{Y}_3^{(m-1)} \\ & + [p_{21}^{(m)}Y_3^{(m-1)} + p_{22}^{(m)}Y_4^{(m-1)} + p_{23}^{(m)}Y_3^{(m)} + p_{24}^{(m)}Y_4^{(m)}]\delta\bar{Y}_4^{(m-1)} \\ & + [p_{31}^{(m)}Y_3^{(m-1)} + p_{32}^{(m)}Y_4^{(m-1)} + p_{33}^{(m)}Y_3^{(m)} + p_{34}^{(m)}Y_4^{(m)}]\delta\bar{Y}_3^{(m)} \\ & + [p_{41}^{(m)}Y_3^{(m-1)} + p_{42}^{(m)}Y_4^{(m-1)} + p_{43}^{(m)}Y_3^{(m)} + p_{44}^{(m)}Y_4^{(m)}]\delta\bar{Y}_4^{(m)}\} \end{aligned} \tag{75}$$

$$\begin{aligned} R_3 = & -[p_{11}^{(1)}Y_3^{(0)} + p_{12}^{(1)}Y_4^{(0)} + p_{13}^{(1)}Y_3^{(1)} + p_{14}^{(1)}Y_4^{(1)}]\delta\bar{Y}_3^{(0)} \\ & -[p_{21}^{(1)}Y_3^{(0)} + p_{22}^{(1)}Y_4^{(0)} + p_{23}^{(1)}Y_3^{(1)} + p_{24}^{(1)}Y_4^{(1)}]\delta\bar{Y}_4^{(0)} \\ & -\sum_{m=1}^{K-1} \{ [p_{11}^{(m+1)}Y_3^{(m)} + p_{12}^{(m+1)}Y_4^{(m)} + p_{13}^{(m+1)}Y_3^{(m+1)} + p_{14}^{(m+1)}Y_4^{(m+1)}] \\ & + [p_{31}^{(m)}Y_3^{(m-1)} + p_{32}^{(m)}Y_4^{(m-1)} + p_{33}^{(m)}Y_3^{(m)} + p_{34}^{(m)}Y_4^{(m)}]\}\delta\bar{Y}_3^{(m)} \\ & -\sum_{m=1}^{K-1} \{ [p_{21}^{(m+1)}Y_3^{(m)} + p_{22}^{(m+1)}Y_4^{(m)} + p_{23}^{(m+1)}Y_3^{(m+1)} + p_{24}^{(m+1)}Y_4^{(m+1)}] \\ & + [p_{41}^{(m)}Y_3^{(m-1)} + p_{42}^{(m)}Y_4^{(m-1)} + p_{43}^{(m)}Y_3^{(m)} + p_{44}^{(m)}Y_4^{(m)}]\}\delta\bar{Y}_4^{(m)} \\ & -[p_{31}^{(K)}Y_3^{(K-1)} + p_{32}^{(K)}Y_4^{(K-1)} + p_{33}^{(K)}Y_3^{(K)} + p_{34}^{(K)}Y_4^{(K)}]\delta\bar{Y}_3^{(K)} \\ & -[p_{41}^{(K)}Y_3^{(K-1)} + p_{42}Y_4^{(K-1)} + p_{43}^{(K)}Y_3^{(K)} + p_{44}^{(K)}Y_4^{(K)}]\delta\bar{Y}_4^{(K)} \end{aligned} \tag{76}$$

It is noted here that the variations at the far end are

$$\delta\bar{Y}_3(K) = \delta\bar{y}_b = 0 \tag{77}$$

$$\delta\bar{Y}_4(K) = \delta\bar{y}_b' = 0 \tag{78}$$

by virtue of Eqs. (36) and (37). Thus the last two terms of Eq. (76) drop out.

It is again important to emphasize here that the computation does not contain the condition placed at the far end boundary. The calculation starts with the initial conditions and carries through in one direction.

The second term on the right side of Eq. (56) gives

$$\begin{aligned} R_2 &= \sum_{m=1}^{K} [q_1(m)\ q_2(m)\ q_3(m)\ q_4(m)][\delta\bar{Y}_3(m-1)\ \delta\bar{Y}_4(m-1)\ \delta\bar{Y}_3(m)\ \delta\bar{Y}_4(m)]^T \\ &= q_1(1)\delta\bar{Y}_3(0) + q_2(1)\delta\bar{Y}_4(0) \\ &+ \sum_{m=1}^{K-1} [q_1(m+1) + q_3(m)]\delta\bar{Y}_3(m) + \sum_{m=1}^{K-1} [q_2(m+1) + q_4(m)]\delta\bar{Y}_4(m) \\ &+ q_3(K)\delta\bar{Y}_3(K) + q_4(K)\delta\bar{Y}_4(K) \end{aligned} \tag{79}$$

The last two terms drop out again by virtue of Eqs. (77) and (78) .

The quantity $q^{(m)}$ is again expressed as

$$q_\ell(m) = K^{-1} \int_0^1 a_\ell(\xi) Q^{(m)}(\xi) d\xi \qquad \ell = 1,2,3,4 \tag{80}$$

The first term on the right of Eq. (56) is

$$\begin{aligned} R_1 &= -[0\ 0\ \delta\bar{Y}_3(0)\ \delta\bar{Y}_4(0)][0\ 0\ 1\ 0]^T\{\gamma_o[0\ 0\ 1\ 0] + \alpha_o K[0\ 0\ 01]\}[0\ 0\ Y_3(0) Y_4(0)]^T \\ &= -\delta Y_3(0)\{\gamma_o Y_3(0) + \alpha_o K Y_4(0)\} \end{aligned} \tag{81}$$

Combining all the above results and substituting into Eq. (56) we have

$$0 = R_1 + R_2 + R_3 \tag{82}$$

$$
\begin{aligned}
0 = {} & \{-\{\gamma_o Y_3^{(0)} + \alpha_o K Y_4^{(0)}\} \\
& + q_1^{(1)} - [p_{11}^{(1)} Y_3^{(0)} + p_{12}^{(1)} Y_4^{(0)} + p_{13}^{(1)} Y_3^{(1)} + p_{14}^{(1)} Y_4^{(1)}]\}\delta\bar{Y}_3^{(0)} \\
& + \{q_2^{(1)} - [p_{21}^{(1)} Y_3^{(0)} + p_{22}^{(1)} Y_4^{(0)} + p_{23} Y_3^{(1)} + p_{24}^{(1)} Y_4^{(1)}]\}\delta\bar{Y}_4^{(0)} \\
& + \sum_{m=1}^{K-1} \{[q_1^{(m+1)} + q_3^{(m)}] \\
& - [p_{11}^{(m+1)} Y_3^{(m)} + p_{12}^{(m+1)} Y_4^{(m)} + p_{13}^{(m+1)} Y_3^{(m+1)} + p_{14}^{(m+1)} Y_4^{(m+1)}] \\
& - [p_{31}^{(m)} Y_3^{(m-1)} + p_{32}^{(m)} Y_4^{(m-1)} + p_{33}^{(m)} Y_3^{(m)} + p_{34}^{(m)} Y_4^{(m)}]\}\delta\bar{Y}_3^{(m)} \\
& + \sum_{m=1}^{K-1} \{q_2^{(m+1)} + q_4^{(m)}] \\
& - [p_{21}^{(m+1)} Y_3^{(m)} + p_{22}^{(m+1)} Y_4^{(m)} + p_{23}^{(m+1)} Y_3^{(m+1)} + p_{24}^{(m+1)} Y_4^{(m+1)}] \\
& - [p_{41}^{(m)} Y_3^{(m-1)} + p_{42}^{(m)} Y_4^{(m-1)} + p_{43}^{(m)} Y_3^{(m)} + p_{44}^{(m)} Y_4^{(m)}]\}\delta\bar{Y}_4^{(m)} \qquad (83)
\end{aligned}
$$

XII. RECURSIVE SOLUTIONS. Since the variations $\delta\bar{Y}_3^{(0)}$, $\delta\bar{Y}_4^{(0)}$, $\delta\bar{Y}_3^{(m)}$, and $\delta\bar{Y}_4^{(m)}$ in Eq. (83) are all arbitrary, the coefficients of all these variations must vanish. We first take the coefficients of the variations $\delta Y_3^{(0)}$ and $\delta Y_4^{(0)}$.

$$
\begin{vmatrix} \bar{p}_{13}^{(1)} & \bar{p}_{14}^{(1)} \\ \bar{p}_{23}^{(1)} & \bar{p}_{24}^{(1)} \end{vmatrix} \begin{vmatrix} \bar{Y}_3^{(1)} \\ \bar{Y}_4^{(1)} \end{vmatrix} = \begin{vmatrix} (-p_{11}^{(1)} - \gamma_o) & (-p_{12}^{(1)} - \alpha_o K) \\ (-p_{21}^{(1)} & (-p_{22}^{(1)} \end{vmatrix} \begin{vmatrix} \bar{Y}_3^{(0)} \\ \bar{Y}_4^{(0)} \end{vmatrix} + \begin{vmatrix} \bar{q}_1^{(1)} \\ \bar{q}_2^{(1)} \end{vmatrix} \qquad (84)
$$

It is noted that $Y_3^{(0)}$ and $Y_4^{(0)}$ are the initial conditions of the problem that is from Eq. (67) and (46).

$$
Y_3^{(0)} = y_o \qquad (85)
$$

$$
Y_4^{(0)} = \dot{y}_o = K^{-1} y_o' = K^{-1} \frac{dy}{dt} \qquad (86)
$$

We can solve for Y3(1) and Y4(1) in terms of these initial conditions by inverting the two by two matrix in Eq. (84).

$$\begin{bmatrix} Y_3^{(1)} \\ Y_4^{(1)} \end{bmatrix} = \begin{bmatrix} p_{13}^{(1)} & p_{14}^{(1)} \\ p_{23}^{(1)} & p_{24}^{(1)} \end{bmatrix}^{-1} \begin{bmatrix} (-p_{11}^{(1)}-\gamma_o) & (-p_{12}-\alpha_o K) \\ (-p_{21}^{(1)}) & (-p_{22}^{(1)}) \end{bmatrix} \begin{bmatrix} y_o \\ K^{-1}y'_o \end{bmatrix} + \begin{bmatrix} q_1^{(1)} \\ q_2^{(1)} \end{bmatrix} \tag{87}$$

For a general case where m = 1,2,...K-1, we have by setting the coefficients of $\delta Y_3^{(m)}$ and $\delta Y_4^{(m)}$ in Eq. (83) to zero.

$$\begin{bmatrix} Y_3^{(m+1)} \\ Y_4^{(m+1)} \end{bmatrix} = \begin{bmatrix} p_{13}^{(m+1)} & p_{14}^{(m+1)} \\ p_{23}^{(m+1)} & p_{24}^{(m+1)} \end{bmatrix}^{-1}$$

$$- \begin{bmatrix} (p_{11}^{(m+1)} + p_{33}^{(m)}) & (p_{12}^{(m+1)} + p_{34}^{(m)}) \\ (p_{21}^{(m+1)} + p_{43}^{(m)}) & (p_{22}^{(m+1)} + p_{24}^{(m)}) \end{bmatrix} \begin{bmatrix} Y_3^{(m)} \\ Y_4^{(m)} \end{bmatrix}$$

$$- \begin{bmatrix} p_{31}^{(m)} & p_{32}^{(m)} \\ p_{41}^{(m)} & p_{42}^{(m)} \end{bmatrix} \begin{bmatrix} Y_3^{(m-1)} \\ Y_4^{(m-1)} \end{bmatrix} + \begin{bmatrix} q_1^{(m+1)} + q_3^{(m)} \\ q_2^{(m+1)} + q_4^{(m)} \end{bmatrix} \tag{88}$$

We solve the above equation recursively for $Y_3^{(m+1)}$ and $Y_4^{(m+1)}$. Starting with m = 1 we have the initial conditions $Y_3^{(0)}$ and $Y_4^{(0)}$ and the solutions from Eq. (87) for $Y_3^{(1)}$ and $Y_4^{(1)}$. These values are substituted into Eq. (88) to obtain $Y_3^{(2)}$ and $Y_4^{(2)}$. From the values of $Y_3^{(1)}$, $Y_4^{(1)}$, $Y_3^{(2)}$, and $Y_4^{(2)}$ one can determine $Y_3^{(3)}$ and $Y_4^{(3)}$. This procedure continues until we obtain $Y3^{(K)}$ and $Y4^{(K)}$ which are the final values of the problem.

XIII. NUMERICAL RESULTS AND DISCUSSION. The analysis presented in previous sections will now be tested by way of some numerical examples. Let us consider a simple oscillator subjected to a harmonic force. The differential equation can be written as

$$m\ddot{y} + ky = f_o \cos \omega_f t \qquad 0 \leq t \leq T \tag{89a}$$

where T is some finite time of interest and a dot ($\cdot$) denotes differentiation with respect to time. The initial conditions are

$$y(0) = y_o \quad \text{and} \quad \dot{y}(0) = y_o \tag{89b}$$

The system of Eqs. (89a) and (89b) is normalized with respect to T and it becomes

$$y^{*''} + k^* y^* = f^* \cos \omega_f^* t^* \qquad 0 \leq t^* \leq 1 \tag{90a}$$

and

$$y^*(0) = y_0^* \quad \text{and} \quad y^*(0) = y_0^* \tag{90b}$$

Through the following change of parameters

$$t^* = \frac{t}{T} \quad , \quad dt^* = \frac{dt}{T}$$

$$y^*(t^*) = y(t) \quad , \quad y^{*\prime}(t^*) = T \frac{dy}{dt} \tag{91}$$

$$k^* = kT^2/m \quad , \quad f^* = f_o T^2/m \quad , \quad \omega_f^* = \omega_f T$$

$$y_0^* = y_0 \quad , \quad y_1^* = T y_1$$

Comparing Eq. (90a) with Eqs. (24) and (1), one has

$$\alpha = \text{constant} = 1 \quad , \quad \varepsilon = -1$$

$$\beta = 0 \quad , \quad \gamma = 0 \quad \text{and} \quad Q = -f^* \cos \omega_f^* t^* \tag{92}$$

From the data presented here, we further set

$$m = 1.0 \quad , \quad k = 1.0 \quad , \quad f_o = 1.0 \quad , \quad \omega_f = 0.5$$

The parameter T is given for each set of sample calculations.

First, Eq. (84) can be used exclusively to obtain all the solutions. This is demonstrated in Tables I through III. In these tables T has taken to be ten, five, and two, respectively and the number of steps for all cases is taken to be ten. Both $y(t)$ and $\dot{y}(t)$ are shown and the exact solutions are given in parentheses for comparison. It is clear that the results are convergent, i.e., they are improved as the interval of time is decreased.

TABLE I. SOLUTION TO A FORCED VIBRATION PROBLEM OF A SIMPLE OSCILLATOR

($0 \leq t \leq 10$, Ten Steps. Exact Solution Shown in Parenthesis)

t	y(t)		$\dot{y}(t)$	
0	1.0000	(Given)	1.000	(Given)
2.0	1.7590	(1.7684)	-0.711	(-0.674)
4.0	-1.1495	(-1.0938)	-1.450	(-1.512)
6.0	-1.8534	(-1.9195)	0.867	(0.773)
8.0	0.2261	(0.1663)	0.564	(0.689)
10.0	-0.0531	(0.1139)	-0.404	(-0.381)

TABLE II. SOLUTIONS TO A FORCED VIBRATION PROBLEM OF A SIMPLE OSCILLATOR

($0 \leq t \leq 5$, Ten Steps. Exact Solutions Shown in Parenthesis)

t	y(t)		$\dot{y}(t)$	
0	1.0000	(Given)	1.0000	(Given)
1.0	1.8314	(1.8315)	0.4991	(.5012)
2.0	1.7646	(1.7684)	-0.6828	(-0.6740)
3.0	0.5536	(0.5654)	-1.6161	(-1.6079)
4.0	-1.1074	(-1.0938)	-1.5060	(-1.5121)
5.0	-2.1221	(-2.1217)	-0.4129	(-0.4350)

TABLE III. SOLUTIONS TO A FORCED VIBRATION PROBLEM OF A SIMPLE OSCILLATOR

($0 \leq t \leq 2.0$, Ten Steps. Exact Solutions Shown in Parenthesis)

t	y(t)		$\dot{y}(t)$	
0.0	1.0000	(Given)	1.0000	(Given)
0.4	1.3892	(1.3892)	0.9184	(.9184)
0.8	1.7132	(1.7132)	0.6760	(-0.6760)
1.2	1.9116	(1.9117)	0.2961	(0.2966)
1.6	1.9379	(1.9382)	-0.1752	(-0.1742)
2.0	1.7676	(1.7684)	-0.6754	(-0.6740)

Some discussion on the present formulation compared with previous work [3,4] is in order here. In previous work on unconstrained, adjoint variational formulation, the point of emphasis was to free the requirements of satisfying any of the initial conditions and to let the approximate solution converge to them. In the present analysis it is shown that the far end conditions need not be considered in a variational formulation of approximate solutions. A more detailed comparison in terms of numerical convergence, competency, efficiency, etc. is planned.

REFERENCES

1. W. M. Stacy, Jr., "Variational Methods in Nuclear Reactor Physics," Academic Press, 1974, p. 7.

2. R. Courant and D. Hilbert, "Methods of Mathematical Physics, Vol. I," Interscience Publishers Inc., 1953, p. 278.

3. J. J. Wu, "Solutions to Initial Value Problems By Use of Finite Element-Unconstrained Variational Formulations," Journal of Sound and Vibration, 53, 1977, pp. 344-356.

4. J. J. Wu and T. E. Simkins, "A Numerical Comparison Between Two Unconstrained Variational Formulations," Journal of Sound and Vibration, 72, 1980, pp. 491-506.

TIME-STEPPING METHODS FOR SECOND-ORDER EVOLUTION EQUATIONS

Vassilios A. Dougalis and Steven M. Serbin
Department of Mathematics
University of Tennessee
Knoxville, Tennessee 37916

ABSTRACT. Some recently developed fully discrete methods for the numerical solution of linear, second-order systems of ordinary differential equations, arising e.g. from finite difference or finite element semidiscretizations of hyperbolic equations, are reviewed. These methods are of high order of accuracy, have desirable stability properties and are computationally efficient. Extensions to problems with first-order time derivative terms (arising e.g. from the equations of structural dynamics) and nonlinear problems are also considered.

1. INTRODUCTION. This paper traces several investigations which the authors have pursued in the study of second-order systems of differential equations. We first consider a linear problem in an abstract setting and recall the two-step "cosine" schemes studied first in [3] for a homogeneous problem and extended in [6] to the nonhomogeneous problem. Next, we introduce a first-order time-derivative into the problem; in [7] we establish that several methods in the literature can be explained as particular examples of a general class of implicit Runge-Kutta methods when specified to these problems. Finally, we make some preliminary remarks on a class of "Rosenbrock"-like methods which we are currently proposing for some nonlinear second-order problems [12].

2. COSINE METHODS. We consider a second-order evolution equation in an abstract setting. Let H be a complex Hilbert space endowed with an inner product $(\cdot, \cdot)$ and corresponding norm $\|\cdot\|$. Let A be a linear operator with domain $\mathcal{D}(A)$, dense in H. A is assumed to be positive definite and self-adjoint on $\mathcal{D}(A)$; it may be unbounded. For u^0, u_t^0 given in H, the problem to be solved is

$$\begin{aligned} &u_{tt} + Au = 0, \quad 0 < t \leq t^* \\ &u(0) = u^0, \quad u_t(0) = u_t^0 \end{aligned} \tag{1}$$

For example, with $H = \mathbb{C}^N$, A a hermitian, positive definite $N \times N$ matrix, (1) is just a system of ordinary differential equations. Such systems can also be realized as semidiscretizations (finite difference or Galerkin) of certain second-order hyperbolic partial differential equations.

It is well-known that the solution to (1), under the assumptions $u^0 \in \mathcal{D}(A)$, $u_t^0 \in \mathcal{D}(A^{1/2})$, can be obtained uniquely as

$$u(t) = \cos(t\, A^{1/2})u^0 + A^{-1/2} \sin(t\, A^{1/2})u_t^0 \tag{2}$$

Further, the solution (2) satisfies the recursion (for $h > 0$ constant)

$$u(t + 2h) - 2 \cos(h\, A^{1/2})\, u(t + h) + u(t) = 0 \tag{3}$$

The approximation scheme is based on (3) . We suppose that $R(\tau)$ is a rational approximation to $\cos \tau$, $\tau \geq 0$ which satisfies

$$|R(\tau) - \cos \tau| \leq C\, \tau^{\nu+2} \ , \ 0 \leq \tau \leq \sigma \tag{4}$$

$$|R(\tau)| \leq 1 \ , \ 0 \leq \tau \leq \eta \tag{5}$$

for certain constants C, σ, η, ν . (4) represents an accuracy requirement, while (5) is for stability of the scheme; when $\eta = +\infty$, we shall obtain unconditionally stable approximation schemes. Then, we determine $\omega^n \in \mathcal{D}(A)$ to approximate $u(nh)$ by the three-term recurrence

$$\omega^{n+2} - 2\, R(h\, A^{1/2})\omega^{n+1} + \omega^n = 0 \tag{6}$$

We prove in [3] the error estimate

$$\max_n \|\omega^n - u(nh)\| = \mathcal{O}(h^\nu) \tag{7}$$

In order for the approximation scheme (6) to be effective, we propose the approximations $R(\tau)$ determined from the generating relation

$$(1 + x^2\, z^2)^s \cos z = \sum_{n=0}^{\infty} \phi_n^{(s)}(x)\, z^{2n} \tag{8}$$

$$\phi_n^{(s)}(x) = \sum_{j=0}^{n} \frac{(-1)^j}{(2j)!} \binom{s}{n-j} x^{2(n-j)} \tag{9}$$

By setting $z = \tau$ and truncating at $n = s$, we obtain the rational approximation

$$R_s(x;\, \tau) = \frac{\sum_{n=0}^{s} \phi_n^{(s)}(x)\, \tau^{2n}}{(1 + x^2\, \tau^2)^s} \tag{10}$$

which satisfies (4) with $\nu = 2s$ and for which there exists a parameter $x^{(s)} > 0$ such that (5) holds with $\eta = +\infty$ for $x \geq x^{(s)}$. The computational advantages of (10) are two-fold: i) $R(\tau)$ is, in fact, a function of τ^2 , so $A^{1/2}$ is never computed, and ii) the denominator of $R(h\, A^{1/2})$ is of

the form $(I + x^2 h^2 A)^s$, hence one matrix decomposition and s back solves are required for each time step.

The cosine approximation (10) is related to a class of rational approximations to the exponential introduced by Baker and Bramble [2]. This connection enables us in [8] to employ the "real pole sandwich" results of Nørsett and Wanner [10] and the "order stars" of Wanner, Hairer, and Nørsett [14] to further study stability and accuracy dependence on the parameters x and s.

The extension of the cosine schemes to nonhomogeneous equations

$$u_{tt} + Au = g(t) , \tag{11}$$

is considered in [6]. The recurrence relation becomes

$$u^{n+2} - 2\cos(h A^{1/2})\, u^{n+1} + u^n = \gamma^n \tag{12}$$

$$\gamma^n = h \int_0^1 A^{-1/2} \sin(h A^{1/2} s)[g((n + 2 - s)h) + g((n + s) h]ds$$

Approximating (12) by

$$\omega^{n+2} - 2 R(h A^{1/2})\omega^{n+1} + \omega^n = \delta^n \tag{13}$$

we show that if we select δ^n so that $\|\gamma^n - \delta^n\| = \mathcal{O}(h^{\nu+2})$, then the error estimate $\|\omega^n - u^n\| = \mathcal{O}(h^{\nu})$ maintains.

In order for the scheme (13) to be viable, the choice of δ^n as a quadrature for γ^n is nonstandard. For example, a fourth-order scheme ($\nu = 4$) uses (with $\beta = x^2$ a parameter)

$$R(h A^{1/2}) = (I + \beta h^2 A)^{-2}[I + (2\beta - 1/2) h^2 A + (\beta^2 - \beta + \tfrac{1}{24})h^4 A^2]$$

$$\delta^n = (I + \beta h^2 A)^{-2}[\tfrac{h^2}{12}\{(g^{n+2} + 10\, g^{n+1} + g^n) + h^2(24\beta - 1)A\, g^{n+1}\}] . \tag{14}$$

The result of (14) is an algorithm implemented as follows. Define $B = I + \beta h^2 A$. Then we determine ρ^n via $A\rho^n = g^{n+2} + 10\, g^{n+1} + g^n$, define $\psi^n = \frac{h^2}{12}[\rho^n + h^2(24\beta - 1)\, g^{n+1}]$, solve $B\, q^{(1)} = -h^2[I + (2\beta - \frac{1}{12})h^2 A]\, \omega^{n+1} + \psi^n$, solve $B\, q^{(2)} = A\, q^{(1)}$, and set $\omega^{n+2} = 2\,\omega^{n+1} - \omega^n + q^{(2)}$.

Finally, we observe that if $A = M^{-1}K$ with M and K sparse (e.g. the Galerkin semidiscretization of the wave equation), each of the equations to be solved in the above algorithm can be multiplied through by M and hence all matrix operations are sparse.

3. DIAGONALLY IMPLICIT RUNGE-KUTTA FOR DAMPED PROBLEMS. We next seek to extend our considerations to second-order systems with first-order time derivatives also present. While a class of two-step schemes extending (6) to certain of these evaluations has been studied by one of us in [13], we find that a more efficient treatment is afforded by the application of certain implicit Runge-Kutta methods studied by Crouziex [5] and Alexander [1] on an equivalent first-order system. We specifically study the linear structural dynamics equation

$$M\, y_{tt} + C\, y_t + K\, y = f(t) \tag{15}$$

with M, C, K sparse, positive definite $N \times N$ matrices. Brusa and Nigro [4] propose a (globally) third-order, computationally efficient single-step scheme for (15); our goal in [7] is to identify their method as equivalent to a specific choice of implicit Runge-Kutta method, which then allows us to generalize their scheme.

A first-order system equivalent to (15) is

$$\begin{bmatrix} I & 0 \\ 0 & M \end{bmatrix} \frac{d}{dt} \begin{bmatrix} y \\ y_t \end{bmatrix} = \begin{bmatrix} 0 & I \\ -K & -C \end{bmatrix} \begin{bmatrix} y \\ y_t \end{bmatrix} + \begin{bmatrix} 0 \\ f \end{bmatrix} \tag{16}$$

which, with obvious notation, is

$$\mathcal{M}\, \mathcal{Y}_t = -\mathcal{K}\, \mathcal{Y} + \mathcal{F} \tag{17}$$

or

$$\mathcal{Y}_t = \mathcal{A}\, \mathcal{Y} + \mathcal{G}\ , \quad \mathcal{A} = -\mathcal{M}^{-1}\, \mathcal{K}\ , \quad \mathcal{G} = \mathcal{M}^{-1}\, \mathcal{F} \tag{18}$$

To (18), we employ Crouzeix's form of implicit Runge-Kutta scheme, defined for an equation of the form $\mathcal{Y}_t = \hat{\mathcal{F}}(t, \mathcal{Y})$ by a tableau of coefficients

$$\begin{array}{c|c} \begin{matrix} a_{11} & \cdots & a_{1q} \\ \vdots & & \vdots \\ a_{q1} & \cdots & a_{qq} \end{matrix} & \begin{matrix} \tau_1 \\ \vdots \\ \tau_q \end{matrix} \\ \hline \begin{matrix} b_1 & \cdots & b_q \end{matrix} & \end{array} \tag{19}$$

such that if $Q_{n,i} = \hat{\mathcal{F}}(t_{n,i}, \mathcal{Y}_{n,i})$ with $t_{n,i} = t_n + \tau_i\, h$, we solve

$$Q_{n,i} = \hat{\mathcal{F}}(t_{n,i}\ , \mathcal{Y}_n + h \sum_{j=1}^{q} a_{ij}\, Q_{n,j}) \tag{20}$$

and set

$$\mathcal{Y}_{n+1} = \mathcal{Y}_n + h \sum_{i=1}^{q} b_i\, Q_{n,i} \tag{21}$$

In particular, Alexander discusses the DIRK (diagonally-implicit Runge-Kutta) schemes, wherein $a_{ij} = 0$, $i < j$ and all $a_{ii} = \beta$. It is easy to see that for (16) (hence (15)), a DIRK scheme is accomplished by the following algorithm:

For $i = 1, \ldots, q$,

(i) Let $r_i \equiv u_n + h \sum_{j=1}^{i-1} a_{ij} u_{n,j}$

$$s_i \equiv v_n + h \sum_{j=1}^{i=1} a_{ij} v_{n,j} \qquad (22)$$

$$f_{n,i} \equiv f(t_{n,i})$$

(ii) Solve

$$(M + \beta h C + \beta^2 h^2 K) v_{n,i} = -K(r_i + \beta h s_i) - C s_i + f_{n,i}$$

(iii) Set $u_{n,i} = \beta h v_{n,i} + s_i$

Then, form $u_{n+1} = u_n + h \sum_{i=1}^{q} b_i u_{n,i}$

and $v_{n+1} = v_n + h \sum_{i=1}^{q} b_i v_{n,i}$

The study of the algorithm (22) is facilitated when one notes the correspondence with certain rational exponential approximations discussed by Nørsett [9] . Of particular note, though, is the information conveyed in the τ_i , where loads are to be evaluated, which was not found in earlier work specifically for the linear problem. Alexander [1] tabulates the coefficients for several specific A-stable and strongly S-stable DIRK schemes, to which we apply (22) for a model problem in [7] .

4. A SCHEME FOR A NONLINEAR PROBLEM. Our current interest encompasses certain nonlinear second-order problems of the form $u_{tt} = F(t, u, u_t)$, but we shall limit our consideration here to the simpler problem $u_{tt} = G(u)$. Of course, we could apply the implicit Runge-Kutta methods indicated above (or some similar methods suggested first by Rosenbrock [11]); again, we shall defer these considerations to a future paper. Our goal here is to produce a Rosenbrock-type method which is $4^{\underline{th}}$ order accurate, two-stage, and reduces to a scheme (mentioned above) introduced by Baker and Bramble when applied with $G(u) = -Au$. In particular, each in this class of schemes is stable on a strip containing the imaginary axis for certain choice of parameter.

Again, we convert to a first-order system of the form

$$Y_t = F(Y) \ , \quad Y = [u, v]^T \ , \quad v = u_t \ .$$

The idea of Rosenbrock was to introduce the Jacobian F_Y directly into the scheme, rather than introducing it later in a Newton-like effort to solve nonlinear algebraic equations. For the Baker-Bramble analogue, though, we introduce the *square* of the Jacobian F_Y . Specifically, we propose a two-stage computation of the form

$$\begin{aligned}
&[I - \beta h^2 F_Y^2(Y_n)]K_1 = [I + \alpha h F_Y(Y_n)]F(Y_n) \\
&[I - \beta h^2 F_Y^2(Y_n)]K_2 = [I + \theta h F_Y(Y_n)]F(Y_n + \gamma h K_1) \\
&\qquad + \delta h F_Y(Y_n + \eta h K_1)F(Y_n + \nu h K_1) - \mu K_1 \\
&Y_{n+1} = Y_n + h(b_1 K_1 + b_2 K_2)
\end{aligned} \tag{23}$$

Applied to a linear problem, i.e. $F(Y) = BY$, it can be seen easily that (23) reduces to the difference equation $Y_{n+1} = r(hB)Yn$, where

$$r(z) = \frac{1 + z + (\frac{1}{2} - 2\beta)z^2 + (\frac{1}{6} - 2\beta)z^3 + (\frac{1}{24} - \beta + \beta^2)z^4}{(1 - \beta z^2)^2} \tag{24}$$

is indeed the fourth-order Baker-Bramble exponential approximation.

When converted over to the notation of the second-order problem, letting $K_i = [P_i \ Q_i]^T$, we find that (23) requires that we solve at each time step <u>four</u> systems with the same $N \times N$ system matrix

$$\begin{aligned}
&[I - \beta h^2 G_u(u_n)]P_1 = v_n + \alpha h G(u_n) \\
&[I - \beta h^2 G_u(u_n)]Q_1 = \alpha h G_u(u_n)v_n \\
&[I - \beta h^2 G_u(u_n)]P_2 = v_n + \gamma h Q_1 \\
&\qquad + (\theta + \gamma) h G(u_n + \gamma h P_1) - \mu P_1 \\
&[I - \beta h^2 G_u(u_n)]Q_2 = \theta h G_u(u_n) \cdot [v_n + \gamma h Q_1] + G(u_n + \gamma h P_1) \\
&\qquad + \delta h G_u(u_n + \eta h P_1) \cdot [v_n + \nu h Q_1] - \mu Q_1
\end{aligned} \tag{25}$$

The parameters of the scheme are determined so that the local truncation error (determined here for the scalar problem $Y_t = F(Y)$) is $O(h^5)$. This requires that

$$b_1 K_1 + b_2 K_2 = [F + \frac{h}{2} F_Y F + \frac{h^2}{6}\{F_{YY} F^2 + F_Y^2 F\} + \frac{h^3}{24}\{F_{YYY} F^3 + 4 F_{YY} F_Y F^2 + F_Y^3 F\}] + O(h^4) \tag{26}$$

By requiring that the parameter β remain free for stability assignment, (26) eventuates a set of nine nonlinear algebraic equations in the nine remaining parameters (c.f. [12] for details). We have ascertained the existence of many possible solutions of this system; it remains for us to determine which of these solutions provides the best (e.g. smallest error constant) of the schemes (25) and to report the results of ongoing numerical experiments.

REFERENCES

[1] R. Alexander, "Diagonally implicit Runge-Kutta methods for stiff O.D.E.'s," SIAM J. Numer. Anal., 14, 1006 - 1021 (1976).

[2] G.A. Baker, J.H. Bramble, "Semidiscrete and single step fully discrete approximations for second-order hyperbolic equations," RAIRO Analyse Numérique, 13, 79 - 100 (1979).

[3] G.A. Baker, V.A. Dougalis, S.M. Serbin, "An approximation theorem for second-order evolution equations," Numer. Math., 35, 127 - 142 (1980).

[4] L. Brusa, L. Nigro, "A one-step method for direct integration of structural dynamic equations," Int'l J. Numer. Meth. Eng'g, 15, 685 - 699 (1980).

[5] M. Crouzeix, Sur l'approximation des équations différentielles opérationelles linéaires par des méthodes de Runge-Kutta, Thèse, Univ. Paris VI, 1975.

[6] V.A. Dougalis, S.M. Serbin, "Two-step high-order accurate full discretizations of second-order hyperbolic equations", in "Advances in Computer Methods for Partial Differential Equations - III", ed. by R. Vichnevetsky and R.S. Stepleman, 214 - 220 (1979).

[7] V.A. Dougalis, S.M. Serbin, "On some unconditionally stable, higher-order methods for the numerical solution of the structural dynamics equations", to appear.

[8] V.A. Dougalis, S.M. Serbin, "Some remarks on a class of rational approximations to the cosine," BIT, 20, 204 - 211 (1980).

[9] S.P. Nørsett, "Restricted Padé approximations to the exponential function," SIAM J. Numer. Anal., 15, 1008 - 1029 (1978).

[10] S.P. Nørsett, G. Wanner, "The real-pole sandwich for rational approximations and oscillation equations", BIT, 19, 79 - 94 (1979).

[11] H.H. Rosenbrock, "Some general implicit processes for the numerical solution of differential equations," Comput. J., 13, 329 - 330 (1963).

[12] S.M. Serbin, "A nonlinear analogue of the fourth-order Baker-Bramble scheme for second-order systems", to appear.

[13] S.M. Serbin, "On the construction and analysis of approximations of arbitrarily high order for proportionally-damped second-order systems," Comp. and Maths. with Appls., 6, 251 - 263 (1980).

[14] G. Wanner, E. Hairer, S.P. Nørsett, "Order stars and stability theorems," BIT, 18, 475 - 489 (1978).

ACKNOWLEDGEMENT: The work reported here was supported by USARO Grant DAAG 29-80-K-0056.

ON NUMERICAL BOUNDARY CONDITIONS FOR HYPERBOLIC SYSTEMS

Max D. Gunzburger
Department of Mathematics
University of Tennessee
Knoxville, Tennessee 37916

and

William J. Layton
School of Mathematics
Georgia Institute of Technology
Atlanta, Georgia 30332

ABSTRACT. It is well known that numerical algorithms for the approximate solution of first order hyperbolic partial differential equations which are stable for the Cauchy problem and for scalar initial-boundary value problems are often unstable when used for initial-boundary value problems for systems. These instabilities arise from the particular boundary conditions used to close the discrete system. Two methods of generating stable boundary treatments are presented. The first is applicable to finite difference and Galerkin finite element schemes and is based on the theory of characteristics. The second scheme is based on "energy" estimates of the solution in norms equivalent to L_2, but which lead to different discretizations. The latter scheme is used in conjunction with Galerkin finite element methods.

1. INTRODUCTION. It is often observed that instabilities in solving hyperbolic equations are caused by the incorrect treatment of the boundary conditions. Algorithms which are stable for the Cauchy problem can be unstable when used in conjunction with particular boundary treatments. What clouds the issue further is the observation that boundary treatments which are stable for scalar hyperbolic equations may be unstable when applied to systems.

Consider the system

$$\begin{pmatrix} u \\ v \end{pmatrix}_t = \begin{pmatrix} 1/2 & 1 \\ 1 & 1/2 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}_x \qquad \begin{matrix} 0 \le x \le 1 \\ t > 0 \end{matrix} \tag{1}$$

with the initial conditions $u(x, 0)$, $v(x, 0)$ and the boundary conditions $u(0, t)$, $u(1, t)$ given. It can be shown that this problem is well posed. In solving the equations numerically one generally needs special equations to find v at both boundaries, even though analytically it is determined. A natural procedure is to do something special for the variable v by itself since u is determined at both boundaries by the given boundary conditions. It is shown in [1] that if one uses the Lax-Wendroff finite difference method to solve (1) in the interior and uses quadratic extrapolation to determine v at both boundaries, the resulting scheme is unstable. This is in spite of the fact that in [2] it is shown that this scheme is stable for scalar hyperbolic equations. Similarly, in [3] it is shown that the obvious use of the Galerkin finite element method to solve (1) is unstable, although the scalar case is well behaved.

In this paper we present two boundary treatments which are stable for systems of hyperbolic equations. The first scheme is discussed in Sections 2 and 3 and is based on the use of characteristics to carry correct information to and from boundaries. The second scheme, discussed in Section 4, is based on measuring the solution of the differential equation in a norm which, although equivalent to the L_2-norm, leads to a different discretization.

2. THE CHARACTERISTIC SCHEME FOR FINITE DIFFERENCES. For finite differences we will use the theory developed in [2]. Without loss of generality we assume that the system is in characteristic form, i.e.

$$u_t = \Lambda u_x \qquad 0 \le x \le 1 \tag{2}$$

where Λ is a diagonal $q \times q$ matrix with elements $\lambda_\ell \neq 0$. This system can be partitioned into the systems

$$u_t^I = \Lambda^I u_x^I \quad , \quad \Lambda^I < 0 \ , \ \Lambda^I : \ p \times p \ , \ p + r = q \tag{3}$$

and

$$u_t^{II} = \Lambda^{II} u_x^{II} \quad , \ \Lambda^{II} > 0 \ , \quad \Lambda^{II} : \ r \times r \tag{4}$$

together with the initial conditions

$$u(x, 0) = f(x) \tag{5}$$

and boundary conditions

$$u^I(0, t) = S_0 u^{II}(0, t) + g_0(t) \tag{6}$$

$$u^{II}(1, t) = S_1 u^I(1, t) + g_1(t) \ . \tag{7}$$

In the interior we use a scheme

$$Q_{-1} u_j^{n+1} = \sum_{k=0}^{s} Q_k u_j^{n-k} \tag{8}$$

At the boundary $x = 0$ we have two types of conditions. The first uses the given boundary condition (6), i.e.

$$(u_0^I)^n = S_0 (u_0^{II})^n + g_0^n \ . \tag{9}$$

The second kind of boundary treatment is a numerical scheme for the remaining unknowns u_0^{II}. By construction this operation involves only these unknowns, e.g.

$$(u_0^{II})^{n+1} = \sum_{k=0}^{s} T_k (u^{II})^{n-k} \qquad (10)$$

We note that in general the operators Q_k and T_k appearing in (8) and (10) are all block diagonal. Only S_0 in (6) need not be diagonal. In addition to the boundary treatment (9) - (10), there is an analogous set at $x = 1$.

We assume that the scheme (8) is stable for the Cauchy problem and that the scheme (8) - (10) is stable for the scalar semi-infinite problem, i.e. we have $0 < x < \infty$, u a scalar with $u = u^I$ or $u = u^{II}$. Using these assumptions we can prove the following. (The proofs are supplied in [4].)

Proposition 1 - The scheme (8) - (10) for the semi-infinite vector problem is stable.

Theorem 2 - The scheme (8) - (10), along with the corresponding boundary treatment $x = 1$ is stable for the vector initial-boundary value problem on $0 \leq x \leq 1$.

In practice one may deal directly with the non-diagonal form of the equations and, at least for explicit schemes, we may implement the above boundary treatment as a post-correction to an existing and perhaps unstable algorithm. We illustrate these points by the use of an example.

Consider the system (1) with initial conditions

$$u(x, 0) = u_0(x) , \quad v(x, 0) = v_0(x) , \qquad (11)$$

and boundary conditions

$$u(0, t) = g_0(t) , \quad u(1, t) = g_1(t) . \qquad (12)$$

We then find that

$$\begin{aligned} (u + v)_t &= \tfrac{3}{2}(u + v)_x \\ (u - v)_t &= -\tfrac{1}{2}(u - v)_x \end{aligned} \qquad (13)$$

so that at $x = 0$, $(u + v)$ is the characteristic variable coming into the boundary while $(u - v)$ is the characteristic variable moving away from the boundary. We denote by U_0, V_0 the values of u and v at the boundary $x = 0$ calculated by some scheme. We then set

$$\begin{aligned} u(0, t) &= g_0(t) \\ u(0, t) + v(0, t) &= U_0 + V_0 \end{aligned} \qquad (14)$$

or, solving,

$$u(0, t) = g_0(t)$$
$$v(0, t) = V_0 + [U_0 - g_0(t)] . \tag{15}$$

Similarly

$$u(1, t) = g_1(t)$$
$$v(1, t) = V_1 + [g_1(t) - U_1] . \tag{16}$$

Thus, the bracketed terms in (15) and (16) can be considered as correction terms to a given algorithm and as such, one may keep all coding in existing programs and one only need calculate the correction terms. For explicit schemes, the method suggested by (14) is exactly equivalent to doing the boundary treatment on the characteristic variables.

3. THE CHARACTERISTIC SCHEME FOR FINITE ELEMENTS. In this section we describe the characteristic scheme in conjunction with Galerkin methods. Without loss of generality, we may consider the system

$$u_t = \Lambda u_x + F \quad \text{for} \quad 0 < x < 1 \tag{17}$$

where Λ is a diagonal matrix. Again, we can partition Λ, F, and u so that

$$u_t^I = \Lambda^I u_x^I + F^I \quad \text{and} \quad u_t^{II} = \Lambda^{II} u_x^{II} + F^{II} . \tag{18}$$

We impose the initial condition

$$u^I(x, 0) = u_0^I(x) \quad \text{and} \quad u^{II}(x, 0) = u_0^{II}(x) \tag{19}$$

and the boundary conditions

$$u^I = S_0 u^{II} \quad \text{at} \quad x = 0 \tag{20}$$

$$u^{II} = S_1 u^I \quad \text{at} \quad x = 1 . \tag{21}$$

Problems with inhomogeneous boundary conditions may easily be converted into a problem of the type (18) - (21).

The L_2-Galerkin formulation of the problem (18) - (21) is given by: find u^I and u^{II} such that

$$(v^I,\ u_t^I - \Lambda^I u_x^I - F^I) = 0$$
$$(v^{II},\ u_t^{II} - \Lambda^{II} u_x^{II} - F^{II}) = 0 \qquad (22)$$

for all v^I and v^{II} in suitable vector spaces. Here

$$(v,\ u) = \int_0^1 v^T u\ dx\ . \qquad (23)$$

At any time t, an approximation to the solution of (22) is found by seeking $(u^I,\ u^{II}) \in U_1 \times U_2$ such that (22) holds for all $(v^I,\ v^{II}) \in V_1 \times V_2$ where U_1, U_2, V_1 and V_2 are finite dimensional spaces. Here we assume that $V_j = U_j$ except perhaps for boundary effects. Furthermore, we shall assume that, e.g. $V_1 = H \times H \times H \times \cdots \times H$ where H is a space of scalar functions and where the number of products is determined by the dimension of u^I.

Let $\{\hat{\psi}_0, \cdots, \hat{\psi}_J\}$ be a basis for H. We may construct another basis for H by setting (assuming that $\hat{\psi}_0(0) \neq 0$ and $\hat{\psi}_J(1) \neq 0$)

$$\psi_0(x) = \hat{\psi}_0(x) - \frac{\hat{\psi}_0(1)}{\hat{\psi}_J(1)}\ \hat{\psi}_J(1)$$

$$\psi_J(x) = \hat{\psi}_J(x) - \frac{\hat{\psi}_J(0)}{\hat{\psi}_0(0)}\ \hat{\psi}_0(0)$$

and

$$\psi_j(x) = \hat{\psi}_j(x) - \frac{\hat{\psi}_j(0)}{\psi_0(0)}\ \psi_0(x) - \frac{\hat{\psi}_j(1)}{\psi_J(1)}\ \psi_J(x) \quad \text{for} \quad 1 \leq j \leq J-1\ .$$

Then the new basis $\{\psi_0, \cdots, \psi_J\}$ has the properties that

$$\psi_j(0) = 0 \quad \text{for} \quad j > 0 \quad \text{and} \quad \psi_j(1) = 0 \quad \text{for} \quad j < J\ . \qquad (24)$$

If p and r are the dimensions of u^I and u^{II}, respectively, then we choose the bases

$$\{e_k^p\ \psi_j\} \quad k = 1, \ldots, p \quad \text{and} \quad j = 0, \ldots, J$$

$$\{e_k^r\ \psi_j\} \quad k = 1, \ldots, r \quad \text{and} \quad j = 0, \ldots, J$$

for the space U_1 and U_2, respectively, where e_k^p and e_k^r are the k-th unit vectors of dimension p and r, respectively. Then since $u^I \in U^I$ and $u^{II} \in U^{II}$, i.e. each component of u^I and u^{II} is in H, we may write

$$u^I(x, t) = \sum_{j=0}^{J} C_j^I(t)\, \psi_j(x)$$

and (25)

$$u^{II}(x, t) = \sum_{j=0}^{J} C_j^{II}(t)\, \psi_j(x) \ .$$

where C_j^I and C_j^{II} are vectors of dimension p and r, respectively. The values of $C_j^I(0)$ and $C_j^{II}(0)$ for $j = 0, \ldots, J$ are determined by solving an interpolation problem using the initial data.

The bases for V_1 and V_2 are similarly chosen. Substitution of (25) into (22) then yields

$$\Sigma\{(\psi_i, \psi_j) \frac{d\, c_j^{k\ell}}{dt} - \lambda_k^{\ell} (\psi_i, \frac{d\psi_j}{dx})\, c_j^{k\ell}\} = (\psi_i, F^{k\ell}) \qquad (26)$$

for $\ell = I, II$ and $i = 0, \ldots, J$ and $k = 1, \ldots, p$ if $\ell = 1$ and $k = 1, \ldots, r$ if $\ell = II$.

Here $c_j^{k\ell}$ represents the k-th component of the vector C_j^{ℓ} and similarly for λ and F. If we let

$$(M)_{ij} = (\psi_i, \psi_j) \ , \quad (Q)_{ij} = (\psi_i, \frac{d\psi_j}{dx}) \quad \text{for} \quad i, j = 0, \ldots, J$$

and

$$(\underline{f}_k^{\ell})_i = (\psi_i, F^{k\ell}) \ , \quad (\underline{c}_k^{\ell})_i = c_i^{k\ell} \quad \text{for} \quad i = 0, \ldots, J \ ,$$

then (26) may be expressed as

$$M \frac{d\, \underline{c}_k^{\ell}}{dt} = \lambda_k\, Q\, \underline{c}_k^{\ell} + \underline{f}_k^{\ell} \ .$$

The matrix M is the Gram matrix for the basis $\{\psi_0, \ldots, \psi_J\}$ under the inner product (23) and therefore M is symmetric and positive definite.

As a consequence of (24) we have that

$$(\psi_i, \frac{d\psi_j}{dx}) = -(\frac{d\psi_i}{dx}, \psi_j)$$

whenever either i or j is different from 0 or J. Therefore the matrix Q is skew-symmetric except for the $(0, 0)$ and (J, J) elements. These elements are given by

$$(Q)_{00} = -\frac{1}{2}[\psi_0(0)]^2 < 0 \quad \text{and} \quad (Q)_{JJ} = \frac{1}{2}[\psi_J(1)]^2 > 0 .$$

The boundary conditions (20) and (21) may be applied by constraining the coefficients $c_j^{k\ell}$ appropriately. We stress that

$$(v, u_t - A\, u_x) = 0 \quad \text{and} \quad (v, u_t - \Lambda\, u_x) = 0$$

for non-diagonal A are not equivalent at the boundaries. In the latter case, a characteristic variable is left unconstrained at the boundary, while in the former case some linear combination of characteristic variables is left unconstrainted at the boundary. As indicated by the results of [3], using non-diagonal boundary treatments can yield instabilities. However, the diagonal boundary treatment yields stable schemes. Of course, in practice, we may implement the stable boundary treatment on the non-diagonal system in much the same manner as that employed in Section 2 for finite difference methods.

We may prove the following concerning the above Galerkin method. (Again, see [4] for details.)

Proposition 3 - The above Galerkin method for a scalar initial-boundary value problem is stable.

Proposition 4 - The above Galerkin method is stable for vector semi-infinite problems, i.e. problems posed on $0 < x < \infty$.

Theorem 5 - The above Galerkin method is stable for the vector initial-boundary value problem, i.e. posed on $0 < x < 1$.

Thus, as in the finite difference case, boundary treatments based on the use of characteristic variables yield stable approximations.

3. THE CHANGE OF NORM SCHEME. We consider the general hyperbolic initial-boundary value problem (Λ may be assumed diagonal)

$$u_t = \Lambda(x,t)u_x + B(x,t)u + F(x,t) \quad \text{for} \quad 0 < x < 1 , \quad t > 0 \qquad (23)$$

along with the initial conditions

$$u(x, 0) = u_0(x) \quad \text{for} \quad 0 \le x \le 1 \tag{24}$$

and boundary conditions

$$u^I(0, t) = S_0\, u^{II}(0, t)\ , \quad u^{II}(1, t) = S_1\, u^I(1, t) \quad \text{for} \quad t > 0\ , \tag{25}$$

where u, Λ, B and F are partitioned in the usual manner. We immediately may prove the following (see [5] for details)

Proposition 6 - There is an inner product $(\cdot, \cdot)'$ on L^2 with associated norm $\|\cdot\|'$ equivalent to the standard L^2-norm $\|\cdot\|$ with respect to which the operator

$$L = A \frac{\partial}{\partial x}$$

is semibounded, i.e. there is an $\alpha < \infty$ such that

$$(Lu, u)' \le \alpha \|u\|^2\ .$$

The inner product $(\cdot, \cdot)'$ of Proposition 6 is defined by

$$(v, u)' = \int_0^1 v^T G(x)\, u\, dx \tag{26}$$

where G is a positive definite symmetric matrix (diagonal when Λ is diagonal). The matrix G is chosen so that

$$S_1^T\, \Lambda^{II}(1)\, G^{II}(1)\, S_1 \le \Lambda^I(1)\, G^I(1) \tag{27}$$

and

$$S_0^T\, \Lambda^I(0)\, G^I(0)\, S_0 \le \Lambda^{II}(0)\, G^{II}(0) \tag{28}$$

where G has been partitioned corresponding to Λ. For non-diagonal systems, the matrix G may be chosen by first diagonalizing the system, choosing the corresponding G, and then transforming back to non-diagonal form.

Using this G matrix, one can easily show that the exact solution of our differential problem satisfies the estimate

$$\|u(t)\| \le C[e^{\alpha t}\|u(0)\| + \int_0^t e^{\alpha(t-s)}\, \|F(s)\| ds] \tag{29}$$

where C and α are constants (see [5]).

The modified Galerkin method we propose is at each time t to seek a $U \in S^h$ such that

$$(\frac{\partial U}{\partial t}, V)' = (A\frac{\partial U}{\partial x}, V)' + (BU, V)' + (F, V)' \qquad (30)$$

for all $V \in S^k$. Here S^k is a subspace of H^1. Note that if we choose $G = I$ so that $(\cdot, \cdot)' = (\cdot, \cdot)$, the ordinary L^2 inner product, the method (30) is in general unstable (see [3]).

If the inner product $(\cdot, \cdot)'$ is chosen as in (26) with G satisfying (27) - (28), then analogously to (29) we may show (see [5]) that

$$\|U(t)\|' \leq e^{\alpha t}\|U(0)\|' + \int_0^t e^{\alpha(t-s)} \|F(s)\|' \, ds ,$$

i.e. the semi-discrete Galerkin method (30) is stable. Furthermore, if the ordinary differential equations (30) are approximately solved by the Crank-Nicolson method, we can prove that

$$\|U^n\|' \leq e^{\alpha_k t}\|U^0\|' + C\, e^{\alpha_k t} \max_{0\leq s\leq t} \|F(s)\|'$$

where $\alpha_k = \alpha(1 - \frac{\alpha k}{2})^{-1}$, k is the time step and U^n is the approximation to $U(kn)$.

We summarize these results in the following theorem (see [5] for details).

Theorem 7 - The Galerkin scheme based on (30), with the inner product $(\cdot, \cdot)'$ and matrix G chosen according to (26) - (28) is stable. Furthermore, the fully discrete scheme based on (30) in conjunction with a Crank-Nicolson method for approximating the time derivative is also stable.

In [5] are also to be found results concerning the rate of convergence of both the semi-discrete and fully discrete schemes based on (30). Some remarks about extensions of this approach to problems in two space dimensions are also given in [5].

We conclude by giving two examples of the construction of the matrices G appearing in the inner product (26). First consider the system (1) with initial conditions (11) and boundary conditions

$$u(0, t) = u(1, t) = 0 . \qquad (31)$$

Here, the system matrix is non-diagonal so that the matrix G will be non-diagonal. We note that

$$A = \begin{pmatrix} \frac{1}{2} & 1 \\ 1 & \frac{1}{2} \end{pmatrix} = T^{-1} \Lambda T$$

where

$$T = T^{-1} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad \text{and} \quad \Lambda = \begin{pmatrix} \frac{3}{2} & 0 \\ 0 & -\frac{1}{2} \end{pmatrix} .$$

Thus the diagonal form of our problem is

$$\begin{pmatrix} u^I \\ u^{II} \end{pmatrix}_t = \begin{pmatrix} \frac{3}{2} & 0 \\ 0 & -\frac{1}{2} \end{pmatrix} \begin{pmatrix} u^I \\ u^{II} \end{pmatrix}_t$$

$$u^I = -u^{II} \quad \text{at} \quad x = 0 , \quad u^{II} = -u^I \quad \text{at} \quad x = 1$$

plus initial conditions. Thus, in the notation of (26), (27) we have

$$\Lambda^I = \frac{3}{2} , \quad \Lambda^{II} = \frac{1}{2} , \quad S_0 = -1 \quad S_1 = -1$$

so that the scalars G^I and G^{II} satisfy, by (26), (27)

$$G^{II}(1) \leq 3\, G^I(1) \quad \text{and} \quad 3G^I(0) \leq G^{II}(0) .$$

These inequalities are satisfied by

$$G^{II}(x) = 3 \quad \text{and} \quad G^I(x) = 1 .$$

Then

$$G = T \begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix} T^{-1} = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} .$$

We note that with the boundary conditions (31),

$$(A \frac{\partial u}{\partial x} , u)' = (A \frac{\partial u}{\partial x} , Gu) = 0$$

so that

$$\|u(t)\|' = \|u(0)\|' \quad \text{for} \quad t \geq 0 \,, \tag{32}$$

i.e. the "G-energy" is conserved.

In our second example a constant G matrix will not work to simultaneously satisfy (26), (27). Consider the problem

$$\frac{\partial u}{\partial t} = \Lambda \frac{\partial u}{\partial x} \quad \text{where} \quad u = \begin{pmatrix} u^I \\ u^{II} \end{pmatrix} \text{ and } \Lambda = \begin{pmatrix} -1 & 0 \\ 0 & 10 \end{pmatrix} ,$$

$$u^I(0, t) = 5\, u^{II}(0, t) \quad \text{and} \quad u^{II}(1, t) = u^I(1, t)$$

plus initial conditions. In the notation of (26) - (27), we have

$$\Lambda^I = 1 \,, \quad \Lambda^{II} = 10 \,, \quad S_0 = 5 \quad \text{and} \quad S_1 = 1$$

so that if

$$G = \begin{pmatrix} G^I(x) & 0 \\ 0 & G^{II}(x) \end{pmatrix}$$

(26) - (27) require that

$$10\, G^{II}(1) \leq G^I(1) \quad \text{and} \quad 25\, G^I(0) \leq 10\, G^{II}(0) \tag{33}$$

which cannot be simultaneously satisfied by constant matrix G. However, if we choose

$$G^I(x) = 8x + 2 \quad \text{and} \quad G^{II}(x) = -4x + 5$$

then (33) is satisfied. Thus, we may choose

$$G = \begin{pmatrix} 8x + 2 & 0 \\ 0 & -4x + 5 \end{pmatrix}.$$

It can be shown that

$$(A \frac{\partial u}{\partial x} \,, u)' = (A \frac{\partial u}{\partial x} \,, Gu) \leq 40 \|u\|^2 \,.$$

ACKNOWLEDGEMENT. This work was supported by the Army Research Office under Contracts Nos. DAAG 29-78-C-0024 and DAAG 29-80-K-0056.

REFERENCES.

1. D. Gottlieb and E. Turkel, "Boundary conditions for multi-step finite difference methods for time-dependent equations", J. Comp. Phys., 26(1978), pp. 181 - 196.

2. B. Gustafsson, H.O. Kreiss and A. Sundström, "Stability theory of difference approximations for mixed boundary value problems, II", Math. Comp., 27(1972), pp. 649 - 686.

3. M. Gunzburger, "On the stability of Galerkin methods for initial-boundary value problems for hyperbolic systems", Math. Comp., 31(1977), pp. 661 - 675.

4. D. Gottlieb, M. Gunzburger, and E. Turkel, "On numerical boundary treatment of hyperbolic systems", SIAM J. Num. Anal., to appear.

5. W. Layton, "The Galerkin method for first order hyperbolic equations", Ph.D. thesis, U. of Tennessee, (1980).

EXTRAPOLATING METEOROLOGICAL DATA FOR ARTILLERY APPLICATIONS

Abel J. Blanco
US Army Atmospheric Sciences Laboratory
White Sands Missile Range, NM 88002

ABSTRACT. Preliminary results derived from a mathematical algorithm for calculating impact dispersion due to meteorological factors are presented. The report presents a comparison of three techniques for extending the maximum ordinate of the Artillery Computer Meteorological Message from 20 to 23 km, for application to projectiles traversing higher altitudes. The three techniques, called the default, the extrapolation, and the modified extrapolation (or climatological), are analyzed against data from 69 rocketsonde flights that were conducted over White Sands Missile Range, New Mexico, during 1979. The measured and estimated data are used to ballistically simulate 552 impact displacements for a trajectory of a proposed rocket system. The findings show that the extrapolated meteorological correction yields a significant improvement over the current default method of using a standard meteorological message. Impact dispersion error analyses illustrate that a software addition to the current meteorological message procedure predicts all impacts within the current one probable error when the meteorological message is extended 3 km in altitude.

1. INTRODUCTION. With advanced technology in artillery ballistics, projectile delivery at ranges greater than 50 km can be expected. Under certain conditions these projectiles will traverse altitudes higher than 20 km above ground level (AGL). The expected meteorological effects on the target displacement error need to be investigated for projectile traversals beyond 20 km AGL because the current computer meteorological message reports information only to 20 km AGL. This paper presents preliminary results from a comparison of three techniques that extend the maximum ordinate of the artillery meteorological message for application to projectiles traversing higher than the 20 km meteorological message limit. The comparison really reduces to the question of how well the actual meteorological profile can be estimated from available information at the lower altitudes.

The techniques investigated include the current default method of using a standard meteorological message, the method of extrapolating available data from lower levels, and the method of using climatological values. The paper illustrates the effect of the default method in assuming zero wind and using temperature and density and pressure profiles representative for global applications. The method of extrapolating wind, temperature, and density and pressure provided the smallest expected (meteorological) impact displacement for the sample considered. For extrapolations extended up to 3 km beyond the 20 km current maximum altitude, the extrapolated values proved to be good estimates of the actual ballistic parameter values effecting the projectile impact. The climatological method which required adjusted corrections from available information at the lower altitudes also showed a significant improvement over the default method. The meteorological impact errors are smaller than those allowed from the default method but larger than those allowed from the extrapolated method. Climatological input is also required. The method is included in this study because it may prove

advantageous when extrapolated values are needed at ranges which cause the ballistic trajectory to exceed the extended 3 km height.

The development of the extended meteorological message techniques and ballistic simulation programs was tested by using a single rocket configuration. The selected trajectory reaches 65 km range and traverses 23 km AGL in altitude. Data needed to describe this trajectory (for example, ballistic wind and temperature and density coefficients including weighting factors and unit effects) were obtained from the Project Manager of the Multiple Launched Rocket Systems (MLRS).* To attain this altitude, the projectile had to be launched at 3048 m above sea level; consequently, the meteorological extending techniques could be evaluated at the 23 to 26 km level of the lower stratosphere. As is the case for the artillery techniques for aiming a gun (ref 1) on a target, this paper uses the launcher surface as the zero level.

2. EXTENDING METEOROLOGICAL APPLICATION. Available techniques for extending the meteorological data for projectiles reaching higher than 20 km AGL vary from hardware and software or a combination of these. In this paper only software techniques will be discussed. The rocketsonde data are assumed to represent the actual atmospheric parameters; then the extending technique comparison reduces to how well the actual meteorological profile can be estimated from available information below the 20 km AGL limit. Also, the implication is that if these measured meteorological data are used for aiming an artillery piece, then the displacement due to meteorology on the target is zero. When the true meteorological data are known, the simulated fire provides a hit every time.

The first technique examined--one which the Artillery currently uses--will be called the default method. Whenever a meteorological message or climatological tables are unavailable, the artillery pieces are aimed by using a meteorological message which contains standard temperature and pressure and density data. The standard wind is a constant zero speed for all (line numbers) layers. In cases where the meteorological messages are unavailable or are not complete to the 20 km AGL limit, the current procedure defaults to the standard meteorological conditions for the missing data.

The second technique is extrapolation. The missing data are defined from the last available layer and are used to estimate the remainder of the meteorological message for application up to the maximum ordinate of the artillery projectiles. A persistent wind is used which is the wind direction and windspeed at the 20 km layer held constant up to the apogee of the trajectory. The extended values for temperature are computed by adding the standard gradient of the temperature default method to the last known temperature value. Finally, for the last parameters, the hydrostatic extrapolation of the density and pressure is computed by using the extrapolated temperature values and available density and pressure value. The detailed extrapolation, assuming the hydrostatic equation and the perfect gas law, yields the following expressions:

---

*Personal communication between Mr. Henry Oldham, Missile Command, and Dr. Donald M. Swingle, Atmospheric Sciences Laboratory, January-February 1980

Gravity $g_0 = 9.80665 \text{ m s}^{-2}$

Air molecular weight $M = 28.966 \text{ g mol}^{-1}$

Gas constant $R = 8314.32 \text{ J } (^\circ\text{K})^{-1} \text{ mol}^{-1}$

Geopotential layer $$\Delta H(I) = \left(\frac{9.7376}{g_0}\right)(I)(1000)\text{m}$$
where I = 1, 2, 3

Extended temperature $$T(I) = T_0 + T_S(I) - T_{S_0}$$
where T_0 = 20 km value
T_S = standard temperature
T_{S_0} = standard temperature at 20 km
T → °C

Lapse rate $$L(I) = [T(I) - T_0]/_\Delta H(I)$$
L → (°K) km^{-1}

Extended density $$\rho(I) = \rho_0 \frac{[T_0 + 273.16]}{[T(I) + 273.16]}^{\left(1 + \frac{gM}{RL}\right)}$$
where ρ_0 = 20 km value
$\rho \rightarrow \text{g/m}^3$

The extrapolated values for the layers of 1 km thickness are extended by iterating the above relationships with respect to I until the maximum altitude desired is reached.

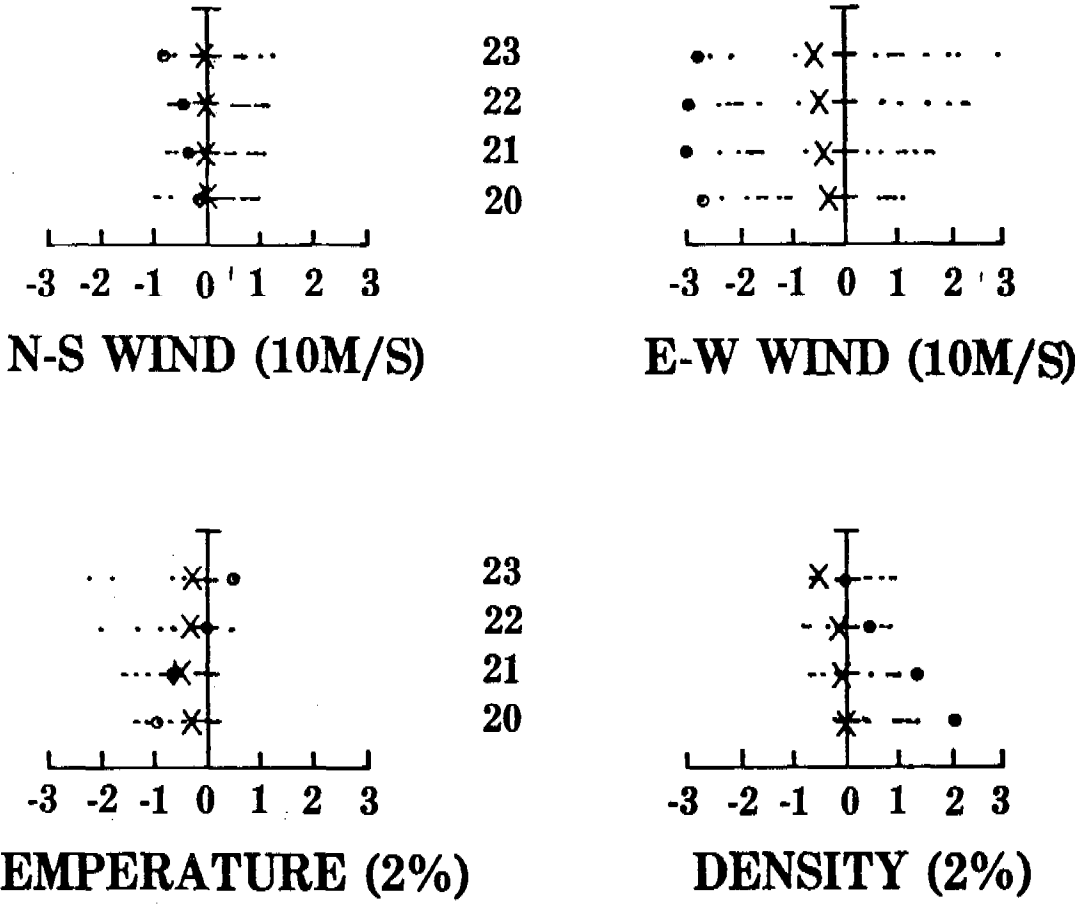
The third technique is defined by a modification to the extrapolated technique. This method uses climatological data to estimate values of the unavailable data. The difference between the data at the 20 km AGL layer and the data of the climatological values for the time of year and location of actual meteorological application is used to adjust the climatological estimate. Even though the Field Artillery does not have climatological tables available for these extended heights, this technique was included to develop the concept of translating the meteorological trend from climatological or fallout meteorological messages to continue the extended meteorological message from the 20 km AGL values.

The US Army Field Artillery needs an estimate of the meteorological impact displacement for proposed high trajectory weapons. Therefore, the emphasis of this paper is to estimate the ballistic meteorological effects and not the

actual value of the missing meteorological data at the extended altitudes. The three methods for extending meteorological data above the altitude actually measured are then transformed into a departure from a selected meteorological standard, and the error in failing to estimate the ballistic atmospheric effect will be illustrated by a displacement about the target. In summary, figure 1 illustrates the percent departures, plotted as (.), from the United States Standard Atmosphere (USSA) 1962 (ref 2) for 16 rocketsonde data flights collected during January 1979. This is the standard atmosphere the Ballistic Research Laboratory uses for trajectory computations (ref 3). The departures for the month's climatological data are also plotted (x). Extending technique 1 uses the default value of the USSA (no departure). Technique 2 uses the wind components measured at the 20 km layer and also uses this value at the 21, 22, and 23 km layers. The extended values for the departure temperatures and density and pressure values are the normalized deviations between the extended values and corresponding values of the USSA. Technique 3 adds the climatological data with respect to the corresponding heights and the difference between the last available data and the climatology at 20 km to compute the data at the missing layers. By superimposing the climatological departure value (x) on a particular value of the 20 km level, one computes the difference that will be arithmetically added to the remaining climatological profile levels.

3. BALLISTIC SIMULATION. A comparison of the impact dispersions (realized by three techniques) was reviewed to evaluate the extending techniques and to gain some insight on the effect of the extended meteorological message. This report assumes that the actual meteorology is defined as the measured parameters deduced from the rocketsonde data (ref 4). These data were then represented in the Artillery computer meteorological message format (ref 5) with new layers of 1 km thickness added to complete an extended message to 23 km AGL. The investigated techniques used measured data below 20 km AGL and extrapolated or climatological data for each layer up to the maximum ordinate of 23 km AGL. Using the same data, each extending method yields a dispersion about an assumed target. The meteorological technique that yields the smallest dispersion about the simulated target is selected as the best of those tested.

The corresponding dispersions are defined as the group of displacements calculated by the ballistic weighting technique. Here an algorithm is introduced that utilizes the extended messages and ballistically computes a displacement about a fixed target. This algorithm can be used to compute the deviation between the extended and a standard (USSA) method. This deviation is then normalized with respect to the standard (USSA) condition. The deviation is calculated for the averaged parameter (wind, temperature, density and pressure) $\overline{P}(Z)$ at each layer through 23 km. Finally, in the ballistic technique, the normalized parameter is multiplied by the weighted response function $[\delta w'(Z)]$. This function contains the ballistic characteristics of the high trajectory weapon system. The required information is the weighting factors and the unit effect for each of the meteorological parameters at the identical layer structure of the extended messages under evaluation. The sum of these products through the maximum altitude of the proposed trajectory yields the effective displacement (D) from the standard conditions. In reality, by knowing this displacement, an artilleryman can compensate for the meteorological deviations from the standard by appropriately adjusting his weapon aim and firing for effect. This displacement is formulated as follows:



(.) January rocketsonde data

(X) January climatological data

(0) January 4, 1979, 1900 hours, data

Figure 1. Percent departures from the 1962 United States Standard Atmosphere 16 Rocketsonde data flights.

$$D = \int_{Z_o}^{Z} \delta\omega'(Z) \left[\frac{\bar{P}(Z) - \bar{P}_s(Z)}{\bar{P}_s(Z)} \right] dZ \ , \qquad (1)$$

where δ = unit effect; $\omega'(Z)$ = ballistic weighting; dZ is the increment of height; and the parameter $\bar{P}(Z)$ is temperature, density, or wind. In the case of wind there is no standard, and the $\bar{P}(Z)$ is not normalized.

A sample of rocketsonde flights containing different atmospheric conditions yields a set of impact displacements describing the dispersion of the analyzed weapon system. This dispersion is mathematically represented with a bias and a variance for each component (cross and range) about the target. The conventional artillery practice is to describe the dispersion of a weapon in terms of a circular error probable (CEP) (ref 1). This criterion is defined as the circular radius of the smallest circle about the target that contains one-half of the total impact displacements. This procedure is used even though the actual dispersion of a gun is elliptical. In demonstrating the differences between the evaluated extending techniques, this report uses elliptical probable error rather than the CEP. There are cases when a small dispersion is biased too far from the target, thereby yielding artillery fire ineffective. One is cautioned that when converting to CEP about the target the comparison of results will produce a different interpretation of the evaluated meteorological messages. The bias due to meteorological parameters is a major contributor to the impact displacement. In practice, through observed fire the Fire Direction Center would correct for this bias which is caused from the unavailability of a meteorological message update or lack of a procedure to obtain data above 20 km AGL.

The results show that the dispersion is a function of the atmospheric condition. Wind, temperature, and density and pressure effect the range impact displacement, while only wind effects the cross component (ref 5). Since the azimuth of fire determines the wind bias, calculation of a mathematical composite of eight single azimuth (θ_i) dispersions was considered to be more appropriate. The weapon system was therefore launched at targets on a circle of radius of 65 km at increments of 45 degrees. Figure 2 illustrates the one-probable-error dispersions produced from 16 rocketsonde flights collected during the same month and at the same location. All impacts were computed without an extended meteorological correction between 20 to 23 km AGL. The effectiveness of fire is different for the particular target. This paper groups the 128 impact displacements and defines the composite dispersion plotted in the center of figure 2. Notice that the range and cross bias due to the wind are cancelled in the composite dispersion. This cancellation would also be true for a single azimuth target if the sample rocketsonde data included winds from all directions. The temperature and density and pressure bias are not cancelled because of the nature of the ballistic computation. If the sample contained data with the temperature and density and pressure above and below the standard, then the bias would be effected in the composite. The next section will present results for the rocketsonde data collected during different months illustrating the variation of temperature and density and pressure effects. The composite results can be interpreted as results of a large sample containing 128 rocketsonde flights collected on the same month. With the inclusion of several months of data

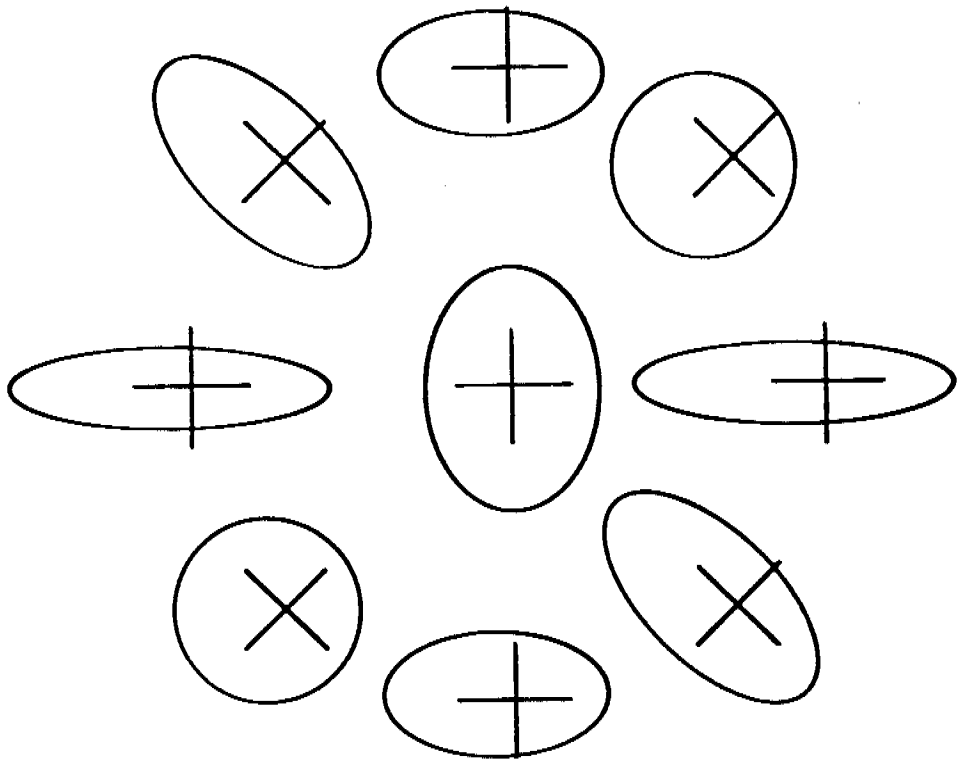


Figure 2. One probable error elliptical dispersions from 16 impact displacements computed without extended meteorological correction between 24 and 26 km above mean sea level. The weapon system is fired at targets on a circle of radius 65 km at 45-degree increments. The center dispersion is the mathematical composite of the 128 displacements.

collected at one location and following the outlined procedure, the final results can be interpreted for general application.

For each rocketsonde flight, equation (1) is applied to the cross (D_C) and range (D_R) components as follows:

$$D_{C_i}(\theta_j, Z) = \delta_C \sum_{Z_0}^{Z} \omega_C'(Z)\overline{W}_{C_i}(\theta_j, Z) \; ; \tag{2}$$

$$D_{R_i}(\theta_j, Z) = \delta_R \sum_{Z_0}^{Z} \omega_R'(Z)\overline{W}_{R_i}(\theta_j, Z) + \delta_T \sum_{Z_0}^{Z} \omega_T'(Z) \frac{\Delta T_i}{T_S} + \delta_\rho \sum_{Z_0}^{Z} \omega_\rho'(Z) \frac{\Delta \rho_i}{\rho_S} . \tag{3}$$

The cross component does not contain the temperature (T) and density (ρ) effects as illustrated in equation (3). The displacement statistics for the error due to the unextended meteorological message are computed as follows:

$$\text{Bias} = \sum_{i}^{n} \sum_{j}^{8} D_i(\theta_j, Z)/8n \; ; \tag{4}$$

$$\text{Variance} = \frac{8n \sum_{i}^{n} \sum_{j}^{8} D_i(\theta_j, Z)^2 - \left[\sum_{i}^{n} \sum_{j}^{8} D_i(\theta_j, Z)\right]^2}{8n(8n - 1)} . \tag{5}$$

Generalizing the results, consider that for each (i) rocketsonde flight there are (j = 8) azimuths providing a total of 8n impact displacements for each month.

4. TECHNIQUE COMPARISON. Measurements from 69 rocketsonde data flights collected at White Sands Missile Range (WSMR), New Mexico, during January through June 1979 were used to compute the meteorological displacement for the high trajectory projectile at the simulated 65 km range target. Since the evaluation of the three proposed extending techniques is based on the comparison of the dispersion from the simulated displacements, the formula in equation (1) is computed for heights of 20 through 23 km AGL. These computations represent the meteorological effects which are not compensated for when the selected projectile is fired. However, use of extending meteorological data techniques will provide meteorological compensation for the missing data and should improve the accuracy.

The meteorological effect from surface through 20 km AGL is not computed in this report since the first 20 km of data are the same for each of the three extended meteorological messages. The extended meteorological message that best estimates the rocketsonde data will yield the smallest dispersion

about the target at 65 km range. Only the displacement due to meteorology above 20 km is analyzed. The largest displacement is 164 m and the smallest is 19 m. Note that this study assumes that there is no time and space difference between the point of measurement and application. For an actual firing, these errors are further increased by a factor determined from the time and space variability.

The results indicate that the presently used default values for the meteorological message above 20 km AGL yield large displacement variations that the Field Artillery should be correcting.

The miss distance is computed for an extrapolation defined from the last available data estimating the missing three meteorological layers. Another miss distance computed is that represented by persistent meteorology modified by climatological gradients. The following interpretation can be made: If the high trajectory projectile were fired on a cross-road target located 65 km in range on 4 January 1979, 1900 hours, using the current artillery default method, it would miss the target by 164 m. The smallest miss for the month is 50 m (17 January) and this assumes that there are no other time and space associated meteorological contributions. This unacceptable error can be improved significantly by any of the proposed extending techniques. By the simple extrapolated technique, the 164-m miss is reduced to 37 m and the 50-m miss to 17 m. A statistical extrapolation technique may provide further improvement. This improvement is expected from the better estimate of the wind and density effect. The temperature related errors are small because the variations at 23 to 26 km (above mean sea level) were small; and when normalized with the standard (in degrees Kelvin), the ballistic effect is a minimum.

In this study, a procedure is automated to compute the expected meteorological errors associated with the high trajectory profile. The algorithm compares the statistics from the evaluated techniques. In summary, the no-correction or the default displacement is computed first by setting J = 0. This error is the total effect of the extended layers as computed from the actual rocketsonde data. For each flight, this displacement is saved for comparison with the other evaluated techniques. The difference and square of difference are saved to compute statistics leading to description of the one probable error, elliptical dispersion. In detail the miss distance is defined, using no-correction or default standard, as J0 (C0, R0). E(C1, R1) is the miss distance computed by using the extrapolated correction. The miss distance provided from the climatology method is labled as E(C2, R2). The differences J1 and J2, where

$$\begin{aligned} J1 &= E(C1, R1) - J0(C0, R0), \\ J2 &= E(C2, R2) - J0(C0, R0), \end{aligned} \tag{6}$$

provide the comparative values for the evaluated methods. A difference equal to zero indicates that the extended method has fully compensated for the actual extended values. The value of the difference is the error that remains uncompensated. By grouping the corresponding displacements, one can then compare the evaluated technique dispersions.

Table 1 presents statistics partitioned into the January, February, March, April, May, and June subsets of 16, 13, 12, 12, 8, and 8 rocketsonde data flights. An analysis of the total sample shows that there is a 64 percent improvement afforded by the extrapolated method over the current default method. Figure 3 presents a graphic demonstration of improved accuracy.

To assure the reader that this sample provides representative results, a test of significance was performed. The chi square distribution test involves the comparison of the computed displacements versus the expected displacements. A desired risk is selected, and a test statistic is compared with the chi square table value (ref 6). This test statistic is defined as follows:

$$\chi^2 = \sum_{i}^{n} \frac{(O_i - E_i)^2}{E_i}, \qquad (7)$$

where O_i is the observed frequency of occurrence of the computed displacements, E_i is the expected frequency of displacement for the different technique, and χ^2 is the computed chi square value.

For ease in organizing the results, a contingency table is arranged in table 2. The expected number of less than 30 m displacement is computed as follows: If there were no difference in the effect of the three techniques, the fraction of displacement with better than 30 m would be expected to be the same ratio as the totals in the last column of table 2. The number of the sample displacements is multiplied by this ratio to define the expected results. The computed value of χ^2 is greater than 34. Since the calculated value exceeds the table value (10), the conclusion is that the data indicate a difference from the expected value with a risk less than 0.005.

5. CONCLUSIONS. There is a large variation in the displacement effect due to the measured rocketsonde data collected at 23 through 26 km above mean sea level. For this theoretical study, the largest meteorological displacement in the sample size of 69 is 164 m and the smallest is 19 m. Note that for an actual firing these errors are further increased by a factor determined from the time and space differences between the point of measurement and application of the meteorological data. Under the assumption of no time and space variability, extrapolated meteorological data above 20 km AGL yielded a significant improvement over the current default method of using a standard meteorological message. The total rms 78-m displacement error was reduced to 28 m. The comparison reduces to how well the actual meteorological profile can be estimated from available information. If the estimate is poor, then actual measurements become important. Preliminary results for the high trajectory projectile considered indicate that a software addition to the current message procedure may be sufficient. This indication appears to be true when the meteorological message is extended 3 km in altitude for compensating meteorological effects on a 65 km range trajectory.

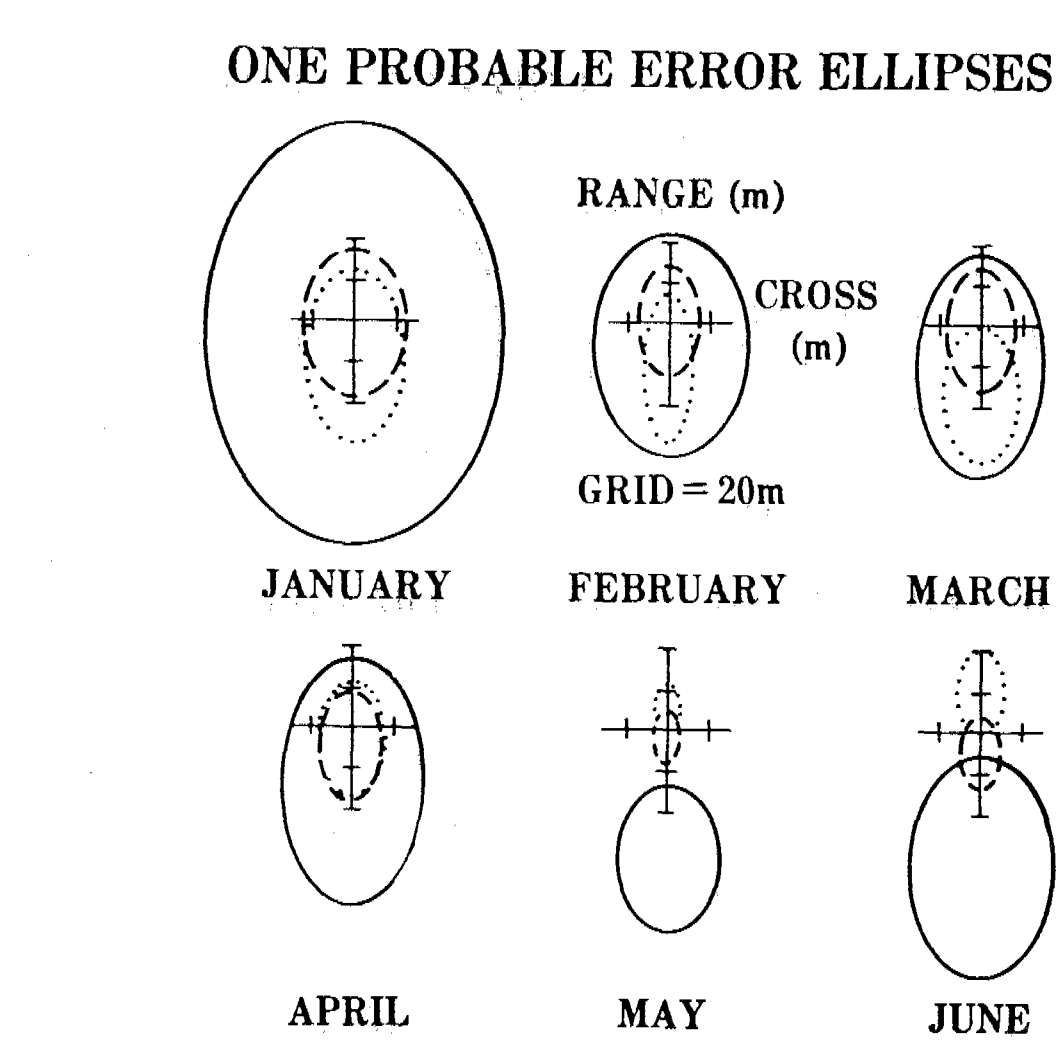


Figure 3. Graphic display of improved one probable error afforded by the extrapolated and climatological messages.

TABLE 1. COMPARISON OF RMS MISS FOR THREE EXTRAPOLATED MET MESSAGES USED AS INPUT FOR SIMULATED TRAJECTORY

$(\overline{M}^2 + \sigma^2_{\overline{M}})^{1/2}$ in meters

Month	Jan	Feb	Mar	Apr	May	Jun	Total
Sample size	16	13	12	12	8	8	69
Rocketsonde	———	———	———	Actual impact	———	———	———
Techniques (20-23 km)							
Default standard	106	55	58	64	74	89	78
Extrapolated	36	25	30	27	12	20	28
Climatology	43	39	46	27	18	27	37

TABLE 2. CONTINGENCY TABLE BASED ON RESULTS OF TEST OF THREE EXTENDING METEOROLOGICAL TECHNIQUES

Technique J =	0	1	2	Total
Total displacement	69	69	69	207
$\leq$ 30 m criteria	5	50	37	92
Expected improvement	30.7	30.7	30.7	

The next report to the United States Field Artillery School will present the status on the accuracy and dispersion effects on target impact displacement provided by using statistical extrapolation techniques. The improvement expected originates from bounded physical estimates of density and temperature effects and modified persistent winds with expected wind gradient effects. Instead of climatology, the last available fallout message can be used to provide the trend of the missing data. A more representative case of a high trajectory traversing to the middle stratosphere will be investigated. Under this condition, the default method of using the standard meteorological message is expected to yield increasingly larger errors.

6. SUMMARY. The United States Army Field Artillery School needs to know the expected meteorological impact displacement for new weapons traversing the atmosphere to altitudes where measurements are not available from the meteorological field units. The preliminary status is that simple persistence for extrapolating the wind, extending temperature by adding the standard gradient to the last known temperature value, and using the hydrostatic extrapolation of density and pressure significantly reduces the meteorological impact error. The improvement is summarized as allowing all impacts to locate within the current one probable error dispersion. A software addition to the current meteorological message procedure reduces the error when the message is extended 3 km in altitude.

REFERENCES

1. Field Manual 6-40, June 1974, Field Artillery Cannon Gunnery, Department of the Army Field Manual, Headquarters, Department of the Army, Washington, DC.

2. "US Standard Atmosphere, 1962," December 1962, National Aeronautics and Space Administration, United States Air Force, United States Weather Bureau.

3. Barnhart, William, June 1966, "The Standard Atmosphere Used by BRL for Trajectory Computations," BRL MR 1766, US Army Materiel Command, Ballistic Research Laboratories, Aberdeen Proving Ground, MD.

4. Federal Meteorological Handbook No. 10, July 1975, Meteorological Rocket Observations, National Aeronautics and Space Administration, US Department of Commerce, US Department of Defense.

5. Field Manual 6-15, August 1978, Artillery Meteorology, Department of the Army Field Manual, Headquarters, Department of the Army, Washington, DC.

6. Rickmers, Albert D., and Hollis N. Todd, 1967, Statistics An Introduction, McGraw-Hill Book Company, New York.

COMPUTER-AIDED SOLUTION OF THE BACTERIAL SURVIVAL EQUATIONS IN MICROBIOLOGY II. - EIGENVALUES FOR HEAT DIFFUSION EQUATION FOR FINITE CYLINDERS AND PLATES BY COMPUTER

Chia Ping Wang and Ari Brynjolfsson

US Army Natick Research and Development Laboratories
Natick, Massachusetts 01760

ABSTRACT

The temperature distribution in a specimen during heating and cooling is determined by the heat diffusion equation with appropriate boundary conditions. In convective heating, the eigenvalues for the heat diffusion equation, for cylinders and plates, are given by the roots of either or both of the following equations:

$$x_n J_1(x_n) = \nu \cdot J_0(x_n)$$

$$C \cos y_j = y_j \sin y_j$$

The present paper describes the solution of these equations with computer to their 36th roots for parameter ν extending from 0 to 20,000, and C, from 0 to 7,000, with rapid convergence. The parameter ν corresponds to the "conductive Nusselt number", or the Biot number, in convective heating.

The very rapid and accurate calculations of the eigenvalues for the very wide ranges of ν and C covered, have great applicability in many fields of thermal engineering. In the present case, these very rapid calculations have made the evaluation of the microbial kill practicable for food samples of various sizes under different heating conditions.

COMPUTER-AIDED SOLUTION OF THE BACTERIAL SURVIVAL EQUATIONS IN MICROBIOLOGY II. - EIGENVALUES FOR HEAT DIFFUSION EQUATION FOR FINITE-CYLINDERS AND PLATES BY COMPUTER

Chia Ping Wang and Ari Brynjolfsson

I. INTRODUCTION

The task of finding changes with time of the heat distribution in samples heated from outside is an important engineering problem in the industry. It is important in the thermal food canning industry, it is important in the nuclear power generation plants and many other fields. The thermal diffusion equation, with given boundary condition, can be solved and the results are usually reported in tables. Graphical methods are then used to interpolate and extrapolate the tabulated values. When the effect of the time temperature relationship on different systems is to be estimated, the calculations become very tedious and time-consuming. For instance, when the time temperature relation is to be used for estimating the microbial kill in the canning industry, the standard procedures are very tedious, and for all practical purposes, make study of the effect of small variation in the different experimental parameters too difficult to estimate accurately. In previous papers [1,3,4] we have shown methods for computer calculation of the diffusion equation and we have shown, as an example, how these could be used directly to calculate the microbial kill in the canning industry. We now report on a method which shortens the computer time for these calculations from one hour to one minute for each sample, and still retains high degree of accuracy.

II. BASIC EQUATIONS

The temperature distribution $T(x,y,z,t)$ in a specimen during heating and cooling of its outer surface is determined by the heat diffusion equation

$$\frac{dT}{dt} = \kappa \nabla^2 T \qquad (1)$$

with appropriate boundary conditions. In paper I [1], it was shown

that for an infinitely long cylinder in convective heating, the boundary condition is given by the equation

$$-\frac{\partial \theta}{\partial r} = \frac{h}{\lambda}\theta, \text{ at } r = a \tag{2}$$

where $\theta = T - T_h$, and T_h is the heating temperature,

h = coefficient of heat transfer at the surface

λ = thermal conductivity of the cylinder material

a = radius of the cylinder

r = radius vector of the cylindrical coordinates used.

Eqs. (1) and (2) lead to

$$x_n J_1(x_n) = \frac{h}{\lambda} a \cdot J_0(x_n)$$
$$\text{or } x_n J_1(x_n) = \nu J_0(x_n) \tag{3}$$

where $(h/\lambda)\, a = \nu$, and J_0 and J_1 are the Bessel functions of order zero and one respectively. The eigenvalues for Eq. (1), with the boundary condition Eq. (2) are

$$\alpha_n = x_n/a \tag{4}$$

Thus, the eigenvalues in this case are the roots of Eq. (3) divided by the radius a.

In Eq. (3),

$$\frac{h}{\lambda} a = \nu \tag{5}$$

is the "conduction Nusselt number" or the Biot number [2].

We note that h/λ in the constant ν comes from the boundary condition Eq. (2) and includes the effects of all surface heat resistances and surface radiation exchanges.

With these notations, the temperature $\theta = T - T_h$ is obtained by solving the heat diffusion equation. For $\theta_i = T_i - T_h$ where T_i is the

initial temperature of the cylinder, we get [1, 3]:

$$\theta = 2\theta_i \sum_{n=1} \frac{J_o(\alpha_n r)}{x_n[(x_n/\nu)^2 + 1] \cdot J_1(x_n)} e^{-\kappa\alpha_n^2 t} \tag{6}$$

which gives the temperature $\theta = T - T_h$ for any point at a distance r from the axis of the infinite cylinder.

We will then consider a finite cylinder of half-length ℓ. In addition to the boundary condition Eq. (2) for the curved surface, we have the following boundary conditions at the ends perpendicular to the z axis [1,4,5]

$$\pm \frac{\partial\theta}{\partial z} + \frac{h}{\lambda}\theta = 0 \text{ at } z = \pm \ell \tag{7}$$

The general solution of Eq. (1), with the boundary conditions, Eqs. (2) and (7), can be obtained by the usual technique of separation of variables [1] and [4]. The conditions, Eq. (7) lead to:

$$\frac{h\ell}{\lambda} \cos \lambda_j\ell - \lambda_j\ell \sin \lambda_j\ell = 0$$

$$\text{or} \quad y_j \sin y_j = C \cos y_j \tag{8}$$

where

$$C = h\ell/\lambda \tag{9}$$

is a constant corresponding to the conduction Nusselt number ν for the curved surface, and

$$\lambda_j = y_j/\ell \tag{10}$$

are the eigenvalues of the separated z equation of the heat diffusion equation.

The methods for solving the heat diffusion equation (1) in case of finite cylinder can be applied directly to solve heat distribution in samples having other forms. For an infinite plate of half-thickness ℓ along the z direction, the eigenvalues are again the λ_j of Eq. (8). For a

rectangular parallelepipeds, we have three separate sets of λ_j's for the three coordinates x,y and z, which have to be calculated separately.

In Section III below, we give a method to solve Eq. (3) to its 36th root by computer, for values of ν extending from 0 to 20,000, with rapid convergence. In Section IV, we used a similar method to solve Eq. (8) to its 36th root by computer for values of C extending from 0 to 7,000. The computation time to their 36th roots for each value of ν or C on UNIVAC 1106 is about 1 to 2 seconds depending on the number of iterations.

III. SOLUTION OF $x_n J_1(x_n) = \nu \cdot J_0(x_n)$

We rewrite Eq. (3) as

$$f(x) = x \cdot J_1(x) - \nu \cdot J_0(x) = 0 \tag{11}$$

and use Newton-Raphson's method of iteration [6] to find the successive roots of f(x). In order to carry out such calculation, one suitable approximate value of x must be assigned to each root. This is accomplished by observing the asymptotic expressions for $J_0(x)$ and $J_1(x)$:

$$J_0(x) \rightarrow \frac{\cos (x - \pi/4)}{\sqrt{\frac{\pi}{2} x}} \tag{12}$$

$$J_1(x) \rightarrow \frac{\sin (x - \pi/4)}{\sqrt{\frac{\pi}{2} x}} \tag{13}$$

From Eqs. (12) and (13) we obtain the following asymptotic expression for Eq. (11),

$$x \tan \left(x - \frac{\pi}{4}\right) - \nu = 0 \tag{11a}$$

Let us introduce δ such that

$$x = n\pi + \frac{\pi}{4} + \delta \tag{14}$$

Eq. (11a) becomes

$$(n\pi + \frac{\pi}{4} + \delta) \cdot \tan\delta = \nu \quad \text{or} \quad \tan\delta = \frac{\nu}{n\pi + \frac{\pi}{4} + \delta} \tag{15}$$

which for large x or n can be approximated by

$$\delta \approx \frac{\nu}{n\pi + \beta\pi} \tag{16}$$

where we have set

$$\frac{\pi}{4} + \delta = \beta\pi \tag{17}$$

From Eq.(14) $x \approx n\pi + \frac{\pi}{4} + \frac{\nu}{n\pi + \beta\pi}$ (18)

It was found by actual computer calculation that x_n given by Eq. (18) can be used as the initial value in the iteration, not only for obtaining the first root x_0 with the assigned accuracy with rapid convergence, but also for obtaining all the first 36 roots within the assigned accuracy with equally rapid convergence. Moreover, the same initial value in the iteration could be used to calculate the 36 roots for a large range of ν-values. We thus express the parameter β in terms of the first root x_0 using Eq. (18)

$$\beta = \frac{\nu}{(x_0 - \frac{\pi}{4})\pi} \tag{19}$$

and treat x_0 as a parameter after substituting Eq. (19) into Eq. (16)

$$\delta \approx \frac{\nu}{n\pi + \nu/(x_0 - \frac{\pi}{4})} \tag{20}$$

For the initial values of x we then use:

$$x = n\pi + \frac{\pi}{4} + \frac{\nu}{n\pi + \nu/(x_0 - \pi/4)} \tag{21}$$

for the first 36 roots, for the given range of ν.

For the different ranges of ν, we let x_0 assume different values (as a parameter), and obtain thus the first 36 roots of x for ν extending from 0 to 20,000 with rapid convergences as mentioned before.

Table I shows, as examples, the 36 roots so computed for $\nu = 5$, 100 and 20,000. The values of ν are given in the upper left corner of each data block. The error in the calculation is less than 5×10^{-6} for $\nu \leq 100$, and 10^{-4} for $10^2 < \nu \leq 10^4$.

IV. SOLUTION OF $y_j \sin y_i = C \cos y_i$

We rewrite Eq. (8) as

$$f(x) = x \tan x - C = 0 \tag{22}$$

For the $(n+1)^{th}$ root we write

$$x = n\pi + \delta \tag{23}$$

then
$$\tan x = \tan \delta \tag{24}$$

and
$$(n\pi + \delta) \tan \delta = C$$

$$\tan \delta = \frac{C}{n\pi + \delta} \tag{25}$$

which for large n approaches

$$\delta = \frac{C}{n\pi + \delta} \tag{25a}$$

The successive roots will be given by different values of n and δ:

$$x = n\pi + \frac{C}{n\pi + \delta} \tag{26}$$

Using Newton-Raphson's method of iteration, we need only an approximate initial value of x. This, in turn, means that we only need an approximate value for δ in the denominator of Eqs. (25) or (26). Obviously, δ is less than $\pi/2$. We thus write for the approximate value of δ as

$$\delta \approx \frac{C}{n\pi + \beta\pi} \tag{27}$$

where β is a parameter which we introduce and whose values are less than 1/2.

The approximate value of the first root x_0 is

$$x_o = \delta \approx \frac{C}{\beta\pi} \tag{28}$$

and we express Eq. (27) as

$$\delta \approx \frac{C}{n\pi + \frac{C}{x_o}} \tag{29}$$

and treat x_o in Eq. (29) as a parameter instead of β.

The initial value of x is given by Eq. (23)

$$x = n\pi + \frac{C}{n\pi + C/x_o} \tag{23}$$

using the approximate value of δ given by Eq. (29).

As in Section III, it is found by actual computer calculations that for a given range of values of C, a single value of x_o for Eq. (29) is sufficient to compute all the first 36 roots of Eq. (22) with rapid convergence. The range of the values of C tested extends from 0 to 7000.

Table II shows, as examples, the 36 roots so computed for C = 5, 100 and 7000. The values of C are given in the upper left corner of each data block. The error in the calculation is set = 10^{-6}, though the roots are printed out only to the 4th decimal place.

V. CONCLUDING REMARKS

The computer programs developed here make use of the NEWTIT subroutine for Newton-Raphson's method of iteration, and the BSSL subroutine for the calculation of the Bessel functions J_0 and J_1. These subroutines, in the present case, are already in the UNIVAC system. Self-checking that the roots calculated fall in the correct quadrant in each iteration is also provided in the programs.

Before concluding, we give the temperature distribution so calculated for a finite cylinder in Tables III and IV. This is expressed in terms of the "relative temperature difference" [5]

$$\psi = \frac{T - T_h}{T_i - T_h}$$

Fig. 1 is the 3 dimensional plot of ψ for such a cylinder given in Tables III and IV.

REFERENCES

[1] Wang, C.P. and Brynjolfsson, A., "Computer-Aided Solution of the Bacterial Survival Equations in Microbiology", Proc. 1979 Army Numerical Analysis and Computers Conference at ElPaso, Feb 14-16, 1979. Hereafter referred to as Paper I.

[2] Schneider, P.J., *Conduction Heat Transfer*, Addison-Welsly Publishing Co., 1955.

[3] Wang, C.P. and Brynjolfsson, A., "Heat Transfer and the Killing of Bacteria during Thermal Sterilization of Meat Rolls," *paper 78-WA/HT-56, Winter Annual Meeting of the American Society of Mechanical Engineers* in San Francisco, Dec. 10-15, 1978.

[4] Wang, C.P. and Brynjolfsson, A., "Heat Conduction in Finite Cylinders and the Computer-Aided Calculation of Bacteria Survival in Heat Sterilization", Vol. 3, 357. Proc., 1980 Army Science Conference at West Point, June 17-20, 1980, Deputy Chief of Staff for Research, Development & Acquisity, Dept. of Army.

[5] Carslow, H.S., and Jagger, J.C., *Conduction of Heat in Solids*, 2nd ed., Oxford University Press, 1959.

[6] "Large Scale Systems MATH-PACK", UP-7542, Rev. 1, Section 4, Sperry Univac, 1980.

TABLE I. THE FIRST 36 ROOTS, x_n, OF

$x_n J_1(x_n) = \nu \cdot J_0(x_n)$

COMPUTED FOR ν = 5, 100, AND 20,000

NU = 5.0000

1.9898	4.7131	7.6177	10.6223
13.6786	16.7630	19.8640	22.9754
26.0937	29.2168	32.3434	35.4726
38.6038	41.7365	44.8704	48.0054
51.1411	54.2775	57.4145	60.5519
63.6898	66.8279	69.9664	73.1052
76.2442	79.3834	82.5228	85.6623
88.8020	91.9418	95.0818	98.2218
101.3620	104.5022	107.6425	110.7829

NU = 100.0000

2.3809	5.4652	8.5678	11.6747
14.7834	17.8931	21.0036	24.1147
27.2264	30.3387	33.4515	36.5649
39.6790	42.7936	45.9089	49.0248
52.1414	55.2586	58.3764	61.4949
64.6140	67.7338	70.8542	73.9752
77.0968	80.2190	83.3418	86.4652
89.5891	92.7136	95.8386	98.9641
102.0901	105.2166	108.3436	111.4710

NU = 20000.0000

2.4047	5.5198	8.6533	11.7909
14.9302	18.0702	21.2106	24.3513
27.4921	30.6331	33.7741	36.9153
40.0564	43.1976	46.3389	49.4801
52.6214	55.7627	58.9040	62.0454
65.1867	68.3281	71.4694	74.6108
77.7521	80.8935	84.0349	87.1763
90.3177	93.4590	96.6004	99.7418
102.8832	106.0246	109.1660	112.3074

TABLE II. THE FIRST 36 ROOTS, x_n, OF

$x_n \sin x_n = C \cdot \cos x_n$

COMPUTED FOR C = 5, 100, AND 7,000

C = 5.0000

1.3138	4.0336	6.9096	9.8928
12.9352	16.0107	19.1055	22.2126
25.3276	28.4463	31.5730	34.7006
37.8305	40.9622	44.0952	47.2294
50.3644	53.5003	56.6367	59.7737
62.9112	66.0490	69.1872	72.3257
75.4644	78.6033	81.7425	84.8818
88.0213	91.1610	94.3008	97.4406
100.5806	103.7207	106.8609	110.0012

C = 100.0000

1.5552	4.6658	7.7764	10.8871
13.9981	17.1093	20.2208	23.3327
26.4450	29.5577	32.6709	35.7847
38.8989	42.0138	45.1292	48.2452
51.3618	54.4790	57.5969	60.7154
63.8345	66.9543	70.0746	73.1956
76.3171	79.4393	82.5620	85.6853
88.8092	91.9336	95.0585	98.1839
101.3099	104.4363	107.5631	110.6904

C = 7000.0000

1.5706	4.7117	7.8529	10.9940
14.1351	17.2763	20.4174	23.5586
26.6997	29.8409	32.9820	36.1232
39.2643	42.4054	45.5466	48.6877
51.8289	54.9700	58.1112	61.2523
64.3935	67.5346	70.6757	73.8169
76.9580	80.0992	83.2403	86.3815
89.5226	92.6637	95.8049	98.9460
102.0872	105.2283	108.3695	111.5106

TABLES III & IV- RELATIVE TEMPERATURE DIFFERENCES, $\psi = (T-T_h)/(T_i-T_h)$ IN TWO CROSS SECTIONS AT Z/ℓ = 0 AND 0.6. THE FIRST COLUMN GIVES THE TIME, THE COLUMNS 2-7 GIVE ψ FOR DIFFERENT DISTANCES r/a = 0, 0.2, 0.4, 0.6, 0.8, 1.0, AND COLUMN 8 GIVES THE SURVIVAL FRACTION FOR Cℓ. BOTULINUM SPORES AT r/a = 0.

TABLE III

A = 5.0 CM L = 2.5 CM Z/L = .0
NU = 6.0 K1 = 1.35-03 CM SQ/S

TIME	DISTANCE FROM AXIS						S F
MIN	0.0	0.2	0.4	0.6	0.8	1.0	AT AXIS
5	.9973	.9973	.9970	.9876	.8853	.4809	1.000+00
10	.9627	.9618	.9528	.9015	.7252	.3632	1.000+00
15	.8972	.8921	.8656	.7814	.5906	.2855	1.000+00
20	.8163	.8059	.7644	.6649	.4833	.2290	1.000+00
25	.7297	.7154	.6652	.5628	.3984	.1863	1.000+00
30	.6436	.6274	.5744	.4761	.3307	.1533	1.000+00
35	.5623	.5457	.4937	.4031	.2763	.1272	1.000+00
40	.4879	.4719	.4233	.3417	.2319	.1062	9.999-01
45	.4212	.4064	.3622	.2900	.1953	.0892	9.991-01
50	.3624	.3490	.3096	.2463	.1650	.0752	9.951-01
55	.3109	.2991	.2643	.2094	.1397	.0635	9.789-01
60	.2663	.2559	.2256	.1781	.1185	.0538	9.268-01
65	.2277	.2187	.1924	.1516	.1006	.0456	7.968-01
70	.1946	.1868	.1641	.1290	.0855	.0387	5.576-01
75	.1661	.1594	.1399	.1099	.0727	.0329	2.665-01
80	.1417	.1360	.1193	.0936	.0619	.0280	6.806-02
85	.1209	.1159	.1016	.0797	.0527	.0238	6.851-03
90	.1031	.0989	.0866	.0679	.0448	.0203	1.938-04
95	.0879	.0843	.0738	.0578	.0382	.0173	1.096-06
100	.0749	.0718	.0629	.0493	.0325	.0147	9.060-10
105	.0639	.0612	.0536	.0420	.0277	.0125	8.467-14
110	.0544	.0522	.0457	.0357	.0236	.0107	7.418-19
115	.0464	.0444	.0389	.0305	.0201	.0091	5.456-25
120	.0395	.0379	.0332	.0259	.0171	.0077	3.249-32
125	.0337	.0323	.0283	.0221	.0146	.0066	0.000
130	.0287	.0275	.0241	.0188	.0124	.0056	0.000
135	.0244	.0234	.0205	.0160	.0106	.0048	0.000
140	.0208	.0200	.0175	.0137	.0090	.0041	0.000
145	.0177	.0170	.0149	.0116	.0077	.0035	0.000
150	.0151	.0145	.0127	.0099	.0065	.0030	0.000
155	.0129	.0123	.0108	.0085	.0056	.0025	0.000
160	.0110	.0105	.0092	.0072	.0048	.0021	0.000
165	.0094	.0090	.0078	.0061	.0040	.0018	0.000
170	.0080	.0076	.0067	.0052	.0034	.0016	0.000
175	.0068	.0065	.0057	.0045	.0029	.0013	0.000
180	.0058	.0055	.0049	.0038	.0025	.0011	0.000
185	.0049	.0047	.0041	.0032	.0021	.0010	0.000
190	.0042	.0040	.0035	.0028	.0018	.0008	0.000
195	.0036	.0034	.0030	.0023	.0015	.0007	0.000
200	.0030	.0029	.0026	.0020	.0013	.0006	0.000
205	.0026	.0025	.0022	.0017	.0011	.0005	0.000
210	.0022	.0021	.0019	.0015	.0010	.0004	0.000
215	.0019	.0018	.0016	.0012	.0008	.0004	0.000
220	.0016	.0015	.0013	.0011	.0007	.0003	0.000
225	.0014	.0013	.0011	.0009	.0006	.0003	0.000
230	.0012	.0011	.0010	.0008	.0005	.0002	0.000
235	.0010	.0010	.0008	.0007	.0004	.0002	0.000
240	.0008	.0008	.0007	.0006	.0004	.0002	0.000

TABLE IV

A = 5.0 CM L = 2.5 CM Z/L = .6
NU = 6.0 K1 = 1.35-03 CM SQ/S

TIME	DISTANCE FROM AXIS						S F
MIN	0.0	0.2	0.4	0.6	0.8	1.0	AT AXIS
5	.9027	.9027	.9024	.8939	.8013	.4353	1.000+00
10	.7881	.7874	.7800	.7380	.5937	.2974	1.000+00
15	.7013	.6973	.6766	.6107	.4616	.2232	1.000+00
20	.6255	.6175	.5856	.5094	.3703	.1754	1.000+00
25	.5543	.5434	.5053	.4275	.3026	.1415	1.000+00
30	.4871	.4748	.4347	.3603	.2503	.1160	9.999-01
35	.4249	.4124	.3731	.3046	.2088	.0961	9.991-01
40	.3685	.3564	.3196	.2580	.1751	.0802	9.957-01
45	.3180	.3068	.2734	.2189	.1475	.0673	9.820-01
50	.2735	.2635	.2337	.1859	.1246	.0567	9.382-01
55	.2347	.2258	.1995	.1581	.1055	.0479	8.264-01
60	.2010	.1932	.1703	.1345	.0894	.0406	6.098-01
65	.1719	.1651	.1453	.1144	.0759	.0344	3.217-01
70	.1469	.1410	.1239	.0974	.0645	.0292	9.662-02
75	.1254	.1203	.1056	.0829	.0549	.0249	1.236-02
80	.1070	.1026	.0900	.0706	.0467	.0211	4.852-04
85	.0913	.0875	.0767	.0601	.0397	.0180	4.101-06
90	.0778	.0746	.0654	.0512	.0338	.0153	5.482-09
95	.0663	.0636	.0557	.0436	.0288	.0130	8.787-13
100	.0565	.0542	.0475	.0372	.0245	.0111	1.376-17
105	.0482	.0462	.0405	.0317	.0209	.0095	1.848-23
110	.0411	.0394	.0345	.0270	.0178	.0080	2.018-30
115	.0350	.0335	.0294	.0230	.0152	.0069	1.817-38
120	.0298	.0286	.0250	.0196	.0129	.0058	0.000
125	.0254	.0244	.0213	.0167	.0110	.0050	0.000
130	.0217	.0208	.0182	.0142	.0094	.0042	0.000
135	.0184	.0177	.0155	.0121	.0080	.0036	0.000
140	.0157	.0151	.0132	.0103	.0068	.0031	0.000
145	.0134	.0128	.0112	.0088	.0058	.0026	0.000
150	.0114	.0109	.0096	.0075	.0049	.0022	0.000
155	.0097	.0093	.0082	.0064	.0042	.0019	0.000
160	.0083	.0079	.0070	.0054	.0036	.0016	0.000
165	.0071	.0068	.0059	.0046	.0031	.0014	0.000
170	.0060	.0058	.0050	.0039	.0026	.0012	0.000
175	.0051	.0049	.0043	.0034	.0022	.0010	0.000
180	.0044	.0042	.0037	.0029	.0019	.0009	0.000
185	.0037	.0036	.0031	.0024	.0016	.0007	0.000
190	.0032	.0030	.0027	.0021	.0014	.0006	0.000
195	.0027	.0026	.0023	.0018	.0012	.0005	0.000
200	.0023	.0022	.0019	.0015	.0010	.0005	0.000
205	.0020	.0019	.0016	.0013	.0008	.0004	0.000
210	.0017	.0016	.0014	.0011	.0007	.0003	0.000
215	.0014	.0014	.0012	.0009	.0006	.0003	0.000
220	.0012	.0012	.0010	.0008	.0005	.0002	0.000
225	.0010	.0010	.0009	.0007	.0004	.0002	0.000
230	.0009	.0008	.0007	.0006	.0004	.0002	0.000
235	.0007	.0007	.0006	.0005	.0003	.0001	0.000
240	.0006	.0006	.0005	.0004	.0003	.0001	0.000

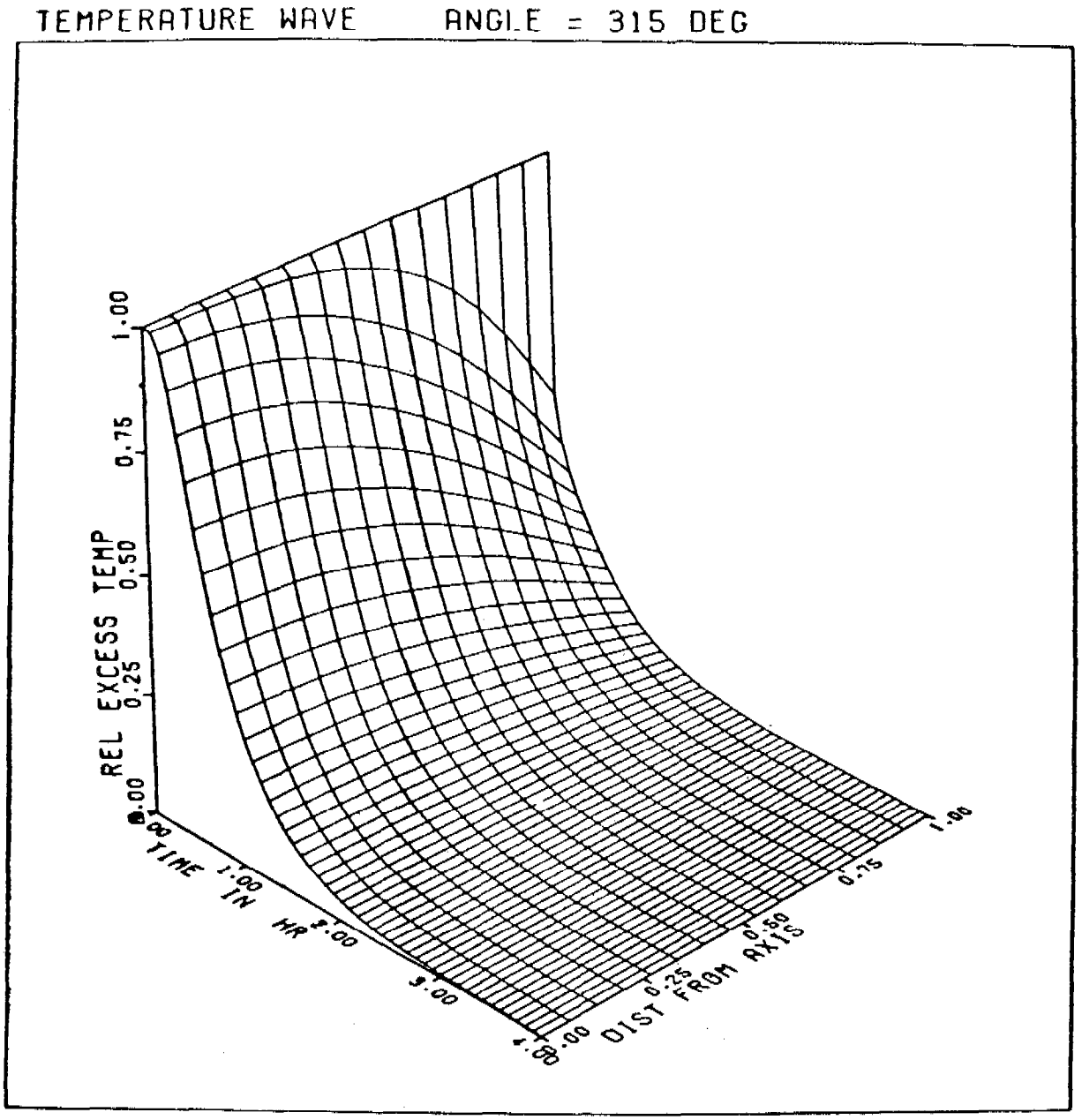


FIG. 1. THREE DIMENSIONAL PLOT OF THE RELATIVE TEMPERATURE DIFFERENCE $\psi = \frac{T - T_h}{T_i - T_h}$, OR THE RELATIVE EXCESS TEMPERATURE, FOR THE CYLINDER GIVEN IN TABLES III AND IV

ALGORITHMS FOR SPARSE, SYMMETRIC, DEFINITE QUADRATIC λ-MATRIX EIGENPROBLEMS

David S. Scott and Robert C. Ward
Computer Sciences Division
Union Carbide Corporation - Nuclear Division
Oak Ridge, Tennessee 37830

ABSTRACT. Methods are presented for computing eigenpairs of the quadratic λ-matrix, $M\lambda^2 + C\lambda + K$, where M, C, and K are large and sparse, and have special symmetry-type properties. These properties are sufficient to insure that all the eigenvalues are real and that theory analogous to the standard symmetric eigenproblem exists. The methods employ some standard techniques such as partial tri-diagonalization via the Lanczos Method and subsequent eigenpair calculation, shift-and-invert strategy and subspace iteration. The methods also employ some new techniques such as Rayleigh-Ritz quadratic roots and the inertia of symmetric, definite, quadratic λ-matrices.

1. INTRODUCTION. Quadratic λ-matrix problems consist of determining scalars λ, called eigenvalues, and corresponding n x 1 nonzero vectors x, called eigenvectors, such that the equation

$$(M\lambda^2 + C\lambda + K)x = 0 \qquad (1)$$

is satisfied, where M, C, and K are given nxn matrices. In addition, we assume that M, C, and K are symmetric or Hermitian, M is definite (either positive or negative definite), and the eigenvalues of (1) are real and can be divided into two disjoint sets P and S with the following properties:

P1) If $\lambda_i \in P$ and $\lambda_j \in S$, then $\lambda_i > \lambda_j$.

P2) If $\lambda_i \in P$ (S) and x_i is its associated eigenvector, then λ_i is the larger (smaller) root of the quadratic equation

$$(x_i^* M x_i)\,\lambda^2 + (x_i^* C x_i)\,\lambda + (x_i^* K x_i) = 0.$$

The eigenvalues in P will be called primary eigenvalues, and those in S will be called secondary. Their eigenvectors will be referenced similarly.

Problems of this nature occur in several application areas; we will briefly discuss two of them. Lancaster [2] states that the determination of sinusoidal solutions to the equations of motion for vibrating systems which are heavily damped results in such a quadratic λ-matrix problem. In these overdamped systems M, C, and K are

symmetric, M and C are positive definite, K is non-negative definite, and the overdamping condition

$$(y^*Cy)^2 - 4(y^*My)(y^*Ky) > 0$$

is satisfied for all vectors $y \neq 0$. Proof that the eigenvalues for overdamped systems are all real and obey properties P1 and P2 above can be found in Lancaster [2]. Problem (1) also arises in the dynamic analysis of rotating structures where the gyroscopic effects cannot be ignored. (See Wildheim [8] and Lancaster [2].) In gyroscopic systems M, C, and K are symmetric (Hermitian), M is negative definite, and K is positive definite. One can determine (Scott and Ward [7]) that all the eigenvalues are real, that P and S are the positive and negative eigenvalues, respectively, and that properties P1 and P2 are satisfied. In both overdamped and gyroscopic systems, the M matrix is usually called the mass matrix and K the stiffness matrix. Thus, we have chosen the notation given in (1) rather than the more standard mathematical notation using A, B, and C for the matrices.

In this paper we present various methods for computing eigenpairs of these quadratic λ-matrices when M, C, and K are also large and sparse. Due to the simplicity of the properties of gyroscopic systems, our model problem for presentation of the methods will be from this application area. That is, we will discuss algorithms for computing eigenpairs of equation (1) where M, C, and K are large, sparse, and symmetric, M is negative definite, and K is positive definite.

In Section 2 we discuss the approach of transforming the quadratic problem into a linear one. Some methods based on the factorization of a nxn matrix are presented in Section 3 with methods not requiring any factorization presented in Section 4. We close the paper by summarizing our results.

2. LINEARIZATION. It may be immediately verified that the eigenpair (λ, x) satisfies the quadratic problem (1) if and only if it also satisfies the 2n x 2n linear problem

$$\left(\begin{bmatrix} 0 & K \\ K & C \end{bmatrix} - \lambda \begin{bmatrix} K & 0 \\ 0 & -M \end{bmatrix} \right) \begin{bmatrix} x \\ \lambda x \end{bmatrix} = 0 , \quad (2)$$

which we denote as $(A-\lambda B)z = 0$. By the hypotheses on M, C, and K, A and B are symmetric and B is positive definite. Thus from well known linear theory, there are 2n real eigenvalues. Applying the Cauchy interlace theorem to the nxn zero block of A leads to the conclusion that exactly n of the eigenvalues are positive and n are negative. Finally, the eigenvectors of the linear problem are B orthogonal so that if (λ_1, x_1) and (λ_2, x_2) are different eigenpairs, then

$$(x_1^* K x_2) - \lambda_1 \lambda_2 (x_1^* M x_2) = 0. \quad (3)$$

Unfortunately, equation (3) involves both λ_1 and λ_2 and does not lead to a useful deflation technique.

Sparse linear eigenvalue problems have been studied in some detail and good solution techniques exist. However, a general linear solver may not be the best choice for solving a quadratic problem in that the linear problem has dimension 2n even though the original problem has dimension n and no advantage will be taken of the special structure of A and B. Also, $A-\sigma B$ is not banded even if M, C, and K are so that factoring $A-\sigma B$, which is an integral part of most linear solvers, will require special care to preserve sparsity.

For these reasons we will investigate solution techniques which take advantage of the underlying quadratic problem.

3. FACTORIZATION TECHNIQUES. In this section we show that the linear problem (2) can be solved using well-known techniques by factoring an nxn matrix only. The Lanczos algorithm and subspace iteration appear to require the factorization of the 2n x 2n matrix $A-\sigma B$. However what is actually needed is the ability to multiply vectors by $(A-\sigma B)^{-1}B$. The special structure of the A and B matrices allows this operator to be realized by factoring only the nxn matrix $W(\sigma) = M\sigma^2 + C\sigma + K$.

Theorem 1. Let A and B be as in equation (2) and let $W(\sigma) = M\sigma^2 + C\sigma + K$. Then

1) The number of negative eigenvalues of W equals the number of eigenvalues of $A-\lambda B$ between σ and 0.

2) $$(A-\sigma B)^{-1}B \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} W(\sigma)^{-1}\,(-Cx - \sigma Mx - My) \\ W(\sigma)^{-1}\,(Kx - \sigma My) \end{bmatrix}$$

The proof is given in Scott [5]. Once the operator $(A-\sigma B)^{-1}B$ has been realized then it is straight forward to implement subspace iteration or the Lanczos algorithm (as described in Scott [4]) to find the eigenvalues of $A-\lambda B$ near σ. The number of negative eigenvalues of W can be easily determined as a byproduct of the factorization and so the index of the computed eigenvalues can be found.

If many eigenvalues are desired then a sequence of shifts σ can be used. The eigenvalue count then gives the number of eigenvalues between two consecutive shifts so that no eigenvalue can be knowingly missed.

4. NONFACTORIZATION TECHNIQUES. In this section we assume that the factorization of M, C, K, or any linear combination of them is either impossible or undesirable. Thus, we are basically limited to algorithms similar to the Lanczos Rayleigh Quotient algorithm presented by Scott [6] for the linear pencil eigenproblem which uses only matrix-vector multiplications.

We have developed an algorithm based on techniques for determining the "best" approximation to an eigenvalue given an approximate eigenvector and the "best" approximation to an eigenvector given an approximate eigenvalue. The algorithm alternates between these approximations until convergence, as the following outline illustrates:

I. Set the vector x_0 to random numbers.

II. For i = 1, 2, ... until convergence, do a and b.

a. Determine "best" σ_i from x_{i-1}.

b. Determine "best" x_i from σ_i.

Step II.a uses a generalization of the Rayleigh quotient different from that of Lancaster's [2] and specifically designed for the quadratic problem. Given any nonzero vector x, potential eigenvectors of the linear pencil (A,B) given by (2) would be linear combinations of the vectors $[x^*, 0]^*$ and $[0, x^*]^*$. Using the Rayleigh-Ritz procedure, the "best" approximations to eigenvectors in this space and corresponding eigenvalues can be determined. Best in this context means minimizing the Frobenius norm of the 2 x 2 scaled residual matrix (see Parlett [3]). The characteristic equation of the reduced linear pencil in the Rayleigh-Ritz procedure is equivalent to the quadratic equation

$$(x^*Mx)\,\theta^2 + (x^*Cx)\,\theta + (x^*Kx) = 0. \qquad (4)$$

Thus, the approximations to two eigenvalues of the quadratic λ-matrix are given by its roots, $\theta^+(x)$ and $\theta^-(x)$, which can be easily determined by the quadratic formula. If we are trying to converge to a positive (primary) eigenvalue, then the larger root $\theta^+(x)$ is chosen for σ_i; conversely, the smaller root $\theta^-(x)$ is chosen when trying to converge to a negative (secondary) eigenvalue. The roots of (4) are identical to the primary and secondary functionals discussed by Duffin [1]. However, Duffin does not present a theoretical basis for how and why these roots along with x most closely approximates an eigenpair of the quadratic λ-matrix. A more thorough discussion of Rayleigh quotient generalizations can be found in Scott and Ward [7].

Step II.b is based on the observation that if σ is an eigenvalue of the quadratic λ-matrix with x as its eigenvector, the matrix $W(\sigma)$ defined in Theorem 1 has the eigenpair (0, x). Theorem 1 relates the

eigenvalues of the symmetric matrix $W(\sigma)$ to the primary and secondary eigenvalues of the quadratic λ-matrix. Thus, to which eigenvalue we are converging can be controlled by the selection of the appropriate eigenvector of $W(\sigma)$ to be used in Step II.b. For example, the following algorithm is used to converge to the m smallest positive eigenvalues:

I. Set the vector x_0 to random numbers.

II. For k = 1, 2, ... m, do 1 and 2.

1. For i = 1, 2, ... until convergence, do a and b.

a. Set $\sigma_i = \theta^+(x_{i-1})$.

b. Set $x_i = y_k$, where (μ_j, y_j) are eigenpairs of $W(\sigma_i)$ with $\mu_1 \leq \mu_2 \leq \ldots \leq \mu_n$ and y_j unit-length.

2. Set x_0 to the y_{k+1} computed in step 1.b above.

From Scott and Ward [7], we know that the sequence $\{\sigma_i\}$ for k = 1 converges monotonically downward to the smallest positive eigenvalue, and the convergence is asymptotically quadratic. Also, the algorithm is expected to quadratically converge to the other m-1 eigenvalues, but convergence is not guaranteed.

A minor modification can be made to the algorithm to guarantee quadratic convergence to interior primary or secondary eigenvalues. This modification requires the solution to a 2k x 2k dense linear pencil eigenproblem in step II.1.a. and the computation of k eigenvectors in step II.1.b. The following algorithm is guaranteed to quadratically convergence to the m smallest positive eigenvalues:

I. Set the vector y_1 to random numbers.

II. For k = 1, 2, ..., m, do 1 and 2.

1. Set the $r^{\underline{th}}$ column of the nxk matrix X to y_r from step I if k = 1 or from step II.2.b otherwise.

2. For i = 1, 2, ... until convergence, do a and b.

a. Set $\sigma_i = \theta_k$ where $\theta_{-k} \leq \theta_{-k+1} \leq \ldots \leq \theta_{-1} < 0 < \theta_1 \leq \ldots \leq \theta_k$ are the eigenvalues of

$$\begin{bmatrix} X_{i-1} & 0 \\ 0 & X_{i-1} \end{bmatrix}^* \left(\begin{bmatrix} 0 & K \\ K & C \end{bmatrix} - \theta \begin{bmatrix} K & 0 \\ 0 & -M \end{bmatrix} \right) \begin{bmatrix} X_{i-1} & 0 \\ 0 & X_{i-1} \end{bmatrix}.$$

b. Set the $r^{\underline{th}}$ column of X_i to y_r, where (μ_j, y_j) are the eigenpairs of $W(\sigma_i)$ with $\mu_1 \leq \mu_2 \leq \ldots \leq \mu_n$ and y_j are unit-length.

Similar algorithms can be developed for computing the m largest positive eigenvalues and the m largest and smallest negative eigenvalues.

5. CONCLUSIONS. In this paper we have presented several techniques for solving symmetric, definite, quadratic λ-matrix problems. These techniques are more efficient, in general, than applying linear techniques to the equivalent 2n x 2n linear problem. The convergence rates of the methods based on factoring $W(\sigma)$ are superior to the convergence rates of the nonfactorization methods presented in Section 4, and so the factorization methods should always be used if the factorization is possible. If the nonfactorization methods must be used, then it is still possible to use preconditioning techniques as in Scott [6] to improve the convergence, if desired. Portable software implementing these algorithms should be available in the near future.

REFERENCES

1. R. J. Duffin, "A Minimax Theory for Overdamped Networks," J. Rational Mech. Anal. 4 (1955), 221-233.

2. Peter Lancaster, Lambda-Matrices and Vibrating Systems, Pergamon Press, New York, 1966.

3. Beresford N. Parlett, The Symmetric Eigenvalue Problem, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1980.

4. D. S. Scott, The Advantages of Inverted Operators in Rayleigh-Ritz Approximations, Technical Report ORNL/CSD-68, Union Carbide Corporation, Nuclear Division, Oak Ridge, Tennessee, 1980.

5. D. S. Scott, Solving Sparse Quadratic λ-Matrix Problems, Technical Report ORNL/CSD-69, Union Carbide Corporation, Nuclear Division, Oak Ridge, Tennessee, 1980.

6. D. S. Scott, "Solving Sparse Symmetric Generalized Eigenvalue Problems Without Factorization," SIAM J. Numer. Anal. 18 (1981), 102-110.

7. D. S. Scott and R. C. Ward, Solving Quadratic λ-Matrix Problems Without Factorization, Technical Report ORNL/CSD-76, Union Carbide Corporation, Nuclear Division, Oak Ridge, Tennessee, 1981.

8. J. Wildheim, "Vibrations of Rotating Circumferentially Periodic Structures," Q. J. Mech. Appl. Math., to appear.

SOFTWARE FOR ORDERING LARGE SPARSE LEAST SQUARES PROBLEMS PRIOR TO GIVENS REDUCTION

David Hume*

James Litsey*

Robert Plemmons**

ABSTRACT

Very large scale least squares problems arise in a variety of applications, including geodetic network adjustments, multiple regression analysis, photogrammetry, earthquake studies, instrumentation planning, and certain types of finite element analysis. For example, the adjustment of a geodetic network with 6,000,000 observations and 400,000 unknowns is being considered. In this paper a new automatic ordering and partitioning scheme for large sparse observation matrices is developed. The method parallels somewhat the concept of block triangularization of square unsymmetric linear systems. Comparisons are made with automatic ordering schemes based upon software from the sparse matrix package SPARSPAK. These comparisons are made by investigating the computational efficiency of solving the resulting least squares problems using orthogonal decompositions by Givens reduction.

April 1981

* Parts of this paper are included in the Masters Degree theses of David Hume in Computer Science and James Litsey in Mathematics at the University of Tennessee.

** This research was supported in part by the U. S. Army Research Office under Contract No. DAAG29-80-K-0025

I. INTRODUCTION AND OVERVIEW

1. Introduction. Let A be an $m \times n$ sparse matrix with $m \geq n$ and consider the system of linear equations

$$Ax = b \tag{1.1}$$

where b is a fixed vector of length m. In general, (1.1) may not have a solution x. In such cases, (1.1) is usually solved in the least squares sense; that is, the solution x is chosen to minimize the Euclidean norm of the residual vector

$$r = b - Ax .$$

We shall assume throughout that A has full column rank n. Under this assumption it is easy to verify that the least squares solution x is unique and satisfies the normal equations

$$A^TAx = A^Tb . \tag{1.2}$$

Since the matrix A^TA is symmetric and positive definite, (1.2) can often be solved efficiently by the Cholesky algorithm. Moreover, in the sparse case, there exists well-developed software for ordering the rows and columns of A^TA to reduce the fill-in during the solution process. Such software is available, for example, in the sparse matrix package SPARSPAK (see George and Liu [1978]). In particular, an ordering is determined in terms of a permutation matrix P so that the Cholesky factor R for

$$PA^TAP^T \tag{1.3}$$

suffers less fill-in than the fill-in for the Choleksy factor of A^TA. Here (1.3) is factored into R^TR, where R is upper triangular, and then x is computed by solving the two triangular systems $R^Ty = PA^Tb$ and $RPx = y$.

Unfortunately, the normal equations method may be numerically unstable. This is due to the potential loss of information in explicitly forming A^TA and A^Tb, and due to the fact that the condition number of A^TA is the square of that of A. In addition, A^TA may no longer be as sparse as the original matrix A.

A well-known stable alternative to the computation of x by solving the normal equations (1.2), is provided by orthogonal factorization (Golub [1965]). The original matrix A is reduced by orthogonal reduction to

$$QA = \begin{bmatrix} R \\ 0 \end{bmatrix}, \qquad Qb = \begin{bmatrix} y \\ z \end{bmatrix}$$

where Q represents an orthogonal matrix and where R is the Cholesky factor of A^TA. The least squares solution x to (1.1) is then obtained by solving the triangular system $Rx = y$. The matrix Q usually results from Gram-Schmidt orthogonalization or from a sequence of Householder or Givens transformations. Both the Gram-Schmidt and Householder algorithms process the unreduced part of A by columns and can cause severe intermediate fill-in. The use of Givens rotations is much more attractive in that the matrix A is processed by rows, gradually building

up R, and intermediate fill-in is confined to the working row. Thus, as indicated in Golub and Plemmons [1980], Givens rotations are generally preferable in the sparse case.

2. Sources of Sparse Problems. Large scale sparse least squares problems of ever increasing size have arisen in recent years. One reason for this is that modern data acquisition technology allows the collection of massive amounts of data. Another factor is the tendency of scientists to formulate more and more complex and comprehensive models in order to obtain finer resolution and more realistic detail in describing physical systems. Particular areas in which such large-scale least squares problems occur include geodetic network adjustments (Golub and Plemmons [1980], Meissl [1980]), photogrammetry (Golub, Luk and Pagano [1980]), earthquake studies (Vanicek, Elliott and Castle [1979]), instrumentation planning (Agee, Turner and Meyer [1976]), and in the natural factor formulation of the finite element problem (Argyris and Brönlund [1975], Argyris et al [1978]). An example of truly spectacular size is the least squares adjustment of coordinates (latitudes and longitudes) of stations comprising the North American Datum, which is to be completed in 1983 by the U.S. National Geodetic Survey (Kolata [1978]). This enormous task requires solving, perhaps several times, a least squares problem having 6,000,000 equations in 400,000 unknowns.

3. A Sparse Least Squares Algorithm. In Golub and Plemmons [1980], an orthogonal decomposition procedure was suggested for solving large scale sparse least squares problems such as those that arise in geodetic adjustment problems. The method has been further developed and coded by George and Heath [1980], where some preliminary tests on geodetic data are reported. This algorithm consists essentially of the following steps:

1. Determine the adjacency structure of the normal equations matrix A^TA.

2. Order the columns of A by a permutation P so that P^TA^TAP has a sparse Cholesky factor R.

3. Symbolically factorize P^TA^TAP, generating a row-oriented data structure for R.

4. Compute R by processing the rows of AP one-by-one using Givens rotations.

Notice that Step 2 produces an indirect ordering of A, by considering the structure of A^TA. The ordering tested by Goerge and Heath [1980] was the quotient minimum degree algorithm (see George and Liu [1981], Chapter 5).

4. Overview. The purpose of this paper is to provide a direct ordering scheme for the observation matrix A as an alternative to Step 2. A scheme based upon permuting A to block upper trapezoidal form is given in Section II. In Section III some comparisons and tests results are given with respect to our block triangularization scheme and two indirect ordering schemes from the package SPARSPAK: the quotient minimum degree algorithm used by George and Heath [1980] and the nested dissection algorithm described in George and Liu [1981], Chapter 8. The results of these tests are reported in Section III. It is found that our direct ordering scheme is quite competitive with the indirect schemes for the examples we tested. This and other observations are summarized in Section IV.

II. A DIRECT ORDERING SCHEME

1. Introduction. We now discuss an algorithm for permuting a rectangular matrix A of dimension $m \times n$, $m \geq n$, and column rank n, into a block upper trapezoidal form given by

$$PAQ = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1k} \\ 0 & A_{22} & \cdots & A_{2k} \\ 0 & 0 & \ddots & \vdots \\ \vdots & \vdots & & \\ 0 & 0 & & A_{kk} \end{bmatrix} .$$

In the above equation P and Q represent permutation matrices, while the A_{ii}, $1 \leq i \leq k$, are rectangular matrices of dimensions $m_i \times n_i$, with $m_i \geq n_i$.

The square case, where $m = n$ and $m_i = n_i$ for $1 \leq i \leq k$, is examined first. An algorithm developed by Duff [1979] for permuting a matrix to obtain a zero-free main diagonal and also an algorithm by Duff and Reid [1978] for permuting a matrix with a zero-free main diagonal to block upper triangular form are reviewed.

The rectangular case, where $m > n$ and $m_i \geq n_i$ for $1 \leq i \leq k$ is then examined, where the algorithms for the square case are modified to accomodate a rectangular matrix. An algorithm is provided to select the rows with fewest nonzero entries when obtaining the zero-free diagonal mentioned above. A heuristic justification is given for this algorithm. Also an algorithm is presented which incorporates the rows not included in the intermediate block upper triangular structure to effect a final block upper trapezoidal structure. Following the presentation of this algorithm is a brief discussion on uniqueness.

2. The Square Case. The problem of permuting a square matrix A to block upper triangular form with square, indecomposable, diagonal blocks is reviewed first. Algorithms for determining certain permutation matrices R and Q such that Q^tRAQ is block upper triangular were developed in Duff and Reid [1978].

The advantages of expressing a square sparse matrix A in a block upper triangular form have to do mainly with the following facts:

1. The eigenvalues of A are those of the diagonal blocks A_{ii} and hence those of A can be computed more easily.

2. When solving $Ax = b$ with A block upper triangular by Gaussian Elimination, the elimination process can be restricted to the diagonal blocks.

3. A matrix in block upper triangular form can be stored by blocks. Depending upon the sizes of the diagonal blocks, storage of only the nonzero upper blocks can reduce storage requirements by nearly one-half in many computational problems.

Advantages also exist for permuting rectangular matrices to block upper trapezoidal form. These advantages involve least squares computations.

The following definitions will facilitate the remaining discussion. A matrix A is said to be decomposable if there exist permutation matrices P and Q such that

$$PAQ = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix},$$

where A_{11} and A_{22} are square. Otherwise A is indecomposable. A matrix A is said to be reducible if there exists a permutation matrix Q such that

$$Q^T AQ = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix},$$

where A_{11} and A_{22} are square. Otherwise A is irreducible. Thus a decomposable matrix can be asymmetrically permuted to block upper triangular form, while a reducible matrix can be symmetrically permuted to such a form. Indecomposability implies irreducibility, but the converse of this statement is not true.

A transversal of a matrix is a set of nonzero elements of the matrix, no two of which are on the same row or column. The length of a transversal refers to the number of elements included in it. A maximal transversal of a matrix is a transversal of greatest length. Of course a matrix can have more than one maximal transversal. A matrix A with full column rank n must have at least one transversal of length n. This fact can be shown in the following way. Suppose A has full column rank n and has a maximal transversal of length $k < n$. Then there exist $n - k$ columns of A that do not include a transversal element. If the rows and columns of A are now permuted by P and Q so that

$$PAQ = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},$$

where A_{11} is a $k \times k$ matrix with main diagonal consisting of the k transversal elements, then the $n - k$ columns of $\begin{bmatrix} A_{12} \\ A_{22} \end{bmatrix}$ do not contain any transversal elements. It follows from the assumption that the maximal transversal has length k, that all entries of the $(n - k) \times (n - k)$ matrix A_{22} are zero. Now form the set V of the k columns of $\begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix}$. There exists a subset S of V consisting of columns which contain the transversal element in a row in which one of the columns of $\begin{bmatrix} A_{12} \\ A_{22} \end{bmatrix}$ has a nonzero entry. The columns in S contain no nonzeroes below their k^{th} entry; otherwise the assumption that the maximum transversal is of length k is violated. The columns of S are linearly independent and form a basis for a vector space of which the columns of $\begin{bmatrix} A_{12} \\ A_{22} \end{bmatrix}$ are vectors. Therefore the columns of PAQ

are not linearly independent and A cannot have column rank n. A contradiction has been reached. Thus every matrix A of full column rank n must have a transversal of length n. It should be noted that this transversal is generally not unique, and also that a matrix of column rank less than n may still possess a transversal of length n. Once a transversal is selected, row permutations are sufficient to bring the transversal elements to the main diagonal.

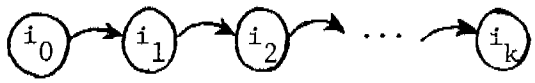
A simple proof of the following well-known fact can be found in Duff and Reid [1978] and in George and Gustavson [1980].

Fact. If a square matrix A has a zero-free main diagonal, then A is irreducible if and only if it is indecomposable.

Thus given that a square matrix in block upper triangular form has a zero-free main diagonal, irreducible diagonal blocks are also indecomposable. The usefulness of this fact will become apparent momentarily.

The Duff-Reid algorithms for transversal selection and block triangularization of square matrices are briefly summarized below. A more complete description can be found in Duff [1978] and Duff and Reid [1978].

The Duff approach to the selection of a transversal consists of two phases. In the cheap assignment phase, each row is examined and within that row the locations of the nonzeroes are determined. As soon as a nonzero is found in a column with no transversal element assigned to it, that nonzero is included in the transversal, and the next row is examined. After completion of the cheap assignment phase, the transversal is generally not of maximum length. The task of selecting a maximal transversal falls to the second phase of the algorithm, the depth first search. In the depth first search phase, each row i_0 not already containing a transversal element is examined and a graph theoretical reassignment chain is constructed as follows. The chain of row indices



implies that element $a_{i_1 j_1}$ is currently a transversal element and $a_{i_0 j_1}$ is nonzero; $a_{i_2 j_2}$ is currently a transversal element and $a_{i_1 j_2}$ is nonzero; etc. If such a chain can be formed such that there exists a nonzero $a_{i_k j_0}$ which is in a column j_0 not currently containing a transversal element, a reassignment is performed. This means that the elements $\{a_{i_1 j_1}, a_{i_2 j_2}, \cdots, a_{i_k j_k}\}$ belonging to the previous transversal are deleted from the transversal and the elements $\{a_{i_0 j_1}, a_{i_1 j_2}, \cdots, a_{i_k j_0}\}$ replace them, thereby increasing the length of the transversal by one. For a matrix of full column rank n, sufficient repetitions of this scheme will produce a transversal of length n. Finally a row permutation is performed, bringing the transversal entries to the main diagonal.

The Duff and Reid [1978] implementation of the Tarjan algorithm for permuting a matrix to block lower triangular form is easily adapted to the task of permuting a matrix to block upper triangular form. The algorithm is best applied to matrices already permuted so that the main diagonal is zero-free. For such matrices, a symmetric (rather than an asymmetric) permutation will be sufficient to put the matrix in block triangular form with indecomposable diagonal blocks. To construct the

permutation matrix, a directed graph of the matrix A is formed with one vertex of the digraph corresponding to each row of the matrix. Edges of the directed graph are defined in the following way. The edge  means $a_{ij} \neq 0$, so that a one-to-one correspondence exists between off-diagonal nonzeroes of the matrix and edges of the graph. A set of vertices any one of which may be reached from any other in the set by travelling along a set of edges is said to be a connected component. If no other vertex may be added to a component without destroying its connectedness, the set is said to be a maximal connected component. Every directed graph has a unique set of maximal connected components and once these are found for the directed graph under consideration, a symmetric permutation is formed to relabel the vertices such that all vertices within a maximal connected component are labelled consecutively and such that if there exists an edge from a vertex in component i to a vertex in component j, then all vertices in component i are labelled before all those in component j.

Although the final block upper triangular form is not unique, it is unique up to symmetric permutations within diagonal blocks and the order of the diagonal blocks along the diagonal. Another important fact is that the final form is independent of which maximum transversal is selected. This is because if rows (or columns) i and j can be interchanged in RA and a zero-free diagonal maintained, then vertices i and j will be in the same connected component and all four elements a_{ii}, a_{ij}, a_{ji}, a_{jj} will end up in one of the diagonal blocks regardless of which transversal is utilized. The scheme discussed here for permuting a square matrix to block upper triangular form is modified next to produce a scheme for permuting rectangular matrices to block upper trapezoidal form.

3. The Rectangular Case. We now present a new algorithm for permuting a matrix A of dimension $m \times n$, $m \geq n$ with full column rank, to block upper trapezoidal form. The algorithm consists of four main steps. The first and third steps are modifications of the two steps of the algorithm described earlier for permuting square matrices to block upper triangular form. The four steps are as follows:

1. A transversal of length n is chosen and a permutation matrix R formed so that

$$RA = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix},$$

where A_1 is a square $n \times n$ matrix with a zero-free main diagonal.

2. In general, matrix A can have more than one transversal of length n. This implies that the matrix A_1 chosen above is not unique. In this step the rows of RA are permuted so that A_1 is replaced by A_1' which contains as few nonzeroes as possible and still maintains a zero-free main diagonal. At this point in the algorithm

$$R'(RA) = R' \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} = \begin{bmatrix} A_1' \\ A_2' \end{bmatrix},$$

where A_1' consists of the n rows of the original matrix A which contain the fewest number of nonzeroes and can still be permuted to yield a zero-free diagonal.

3. In this step a permutation matrix Q is formed to symmetrically permute the square matrix A_1' to block upper triangular form,

$$\begin{bmatrix} Q^t & 0 \\ 0 & I_{m-n} \end{bmatrix} R'RAQ = \begin{bmatrix} A_1^* \\ A_2^* \end{bmatrix},$$

where A_1^* is block upper triangular with indecomposable diagonal blocks.

4. Finally, a block upper trapezoidal form is achieved by forming a permutation matrix P such that

$$P \begin{bmatrix} Q^t & 0 \\ 0 & I_{m-n} \end{bmatrix} R'RAQ = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1p} \\ 0 & A_{22} & \cdots & A_{2p} \\ 0 & 0 & \ddots & \vdots \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & A_{pp} \end{bmatrix},$$

where each A_{ii} is a rectangular block of dimension $m_i \times n_i$, with $m_i \geq n_i$.

Each step is now examined in more detail.

Selecting an A_1

Selection of a transversal for A is achieved by slightly modifying the Harwell subroutine MC21A (Duff [1978]) so that all m rows may be scanned during the cheap assignment and depth first search phases. Selecting a transversal of length n is generally faster for an $m \times n$ matrix of rank n with $m > n$ than for an $n \times n$ matrix of rank n; for, since there are more rows to scan during the cheap assignment phase, one would expect a greater number of transversal elements to be found during cheap assignment in the rectangular matrix, thereby reducing the number which must be found by the more expensive depth first search. Output of the modified MC21A subroutine is an m-vector defining a row permutation of A to $\begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$. The permutation matrix R is, of course, never formed.

Selecting an A_1'

The matrix A_1', as noted earlier, is a member with fewest nonzeroes of the set of all possible $n \times n$ matrices with zero-free diagonal which may be formed by choosing n rows of A. A_1' is not unique, a fact that will be discussed later.

The selection of A_1' is motivated by heuristic considerations. If it is agreed that a final block upper trapezoidal structure with more zeroes below the diagonal blocks is more desirable than an alternate block upper trapezoidal structure with fewer such zeroes, then it is often better to have narrower (and thus more) diagonal blocks. Since the width of the diagonal blocks is completely determined by the sizes of the square diagonal blocks in A_1^* at the end of step three, it follows that one would like to promote, if possible, smallness of these blocks. This may be

done, in a heuristic sense, by choosing a matrix A_1' for block upper triangularization in step three for which the associated graph has few edges and therefore tends to have more, smaller, connected components rather than a few large ones. The size of the connected components determines the size of the diagonal blocks. Using an arbitrary A_1 rather than A_1' can have a drastic effect on the block structure of the final form.

In the example below, the rows of RA in (2.1), are permuted to yield R'RA in (2.2). The effect of this permutation is to replace an A_1 submatrix with the A_1' submatrix.

$$RA = \left[\begin{array}{cccc} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \hline 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{array}\right] = \left[\begin{array}{c} A_1 \\ \hline A_2 \end{array}\right] \tag{2.1}$$

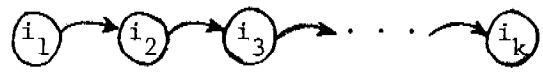
$$R'RA = \left[\begin{array}{cccc} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \hline 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{array}\right] \tag{2.2}$$

Note that the A_1 submatrix in (2.1) is in block upper triangular form, with one 3×3 diagonal block and one 1×1 diagonal block. The A_1' submatrix in (2.2) is not yet in indecomposable block upper triangular form. This form is shown below. Note the four 1×1 diagonal blocks.

$$\left[\begin{array}{cc} Q^t & 0 \\ 0 & I_{m-n} \end{array}\right] R'RAQ = \left[\begin{array}{cccc} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \hline 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{array}\right] \tag{2.3}$$

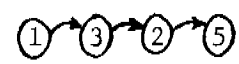
The following algorithm finds an A_1' using a graph theoretical technique. In the discussion below, a path is sought in a directed graph with m vertices, each corresponding to one of the m rows of the matrix RA, and where the edge  corresponds to a nonzero entry a_{ji}. Note that the existence of an edge from i to j means that if row i is currently in A_1, row j could replace row i and A_1 will still have a zero-free main diagonal. If row j had previously been in A_2 and had fewer nonzeroes than row i, then exchanging row i and row j would have the desired effect of reducing the number of nonzeroes in A_1 while preserving the zero-free main diagonal.

In general, a path corresponding to a desirable row permutation is of the form



where all vertices except i_k correspond to rows in A_1 and row i_k has fewer nonzeroes than row i_1. The cyclic row permutation defined by this path would replace each row by the row after it on the path and would replace row i_k by row i_1. This would result in a net exchange of one row between A_1 and A_2. The permutation corresponding to each such path is performed as the path is found.

The path can end without defining a desirable permutation. This can happen either of two ways. The path may reach a vertex corresponding to a row in A_1 from which there are no departing edges, or the path may reach a vertex corresponding to a row in A_2 which does not contain fewer nonzeroes than the row at the beginning of the path. If either event occurs, the last edge on the path is removed and a replacement sought. This process is called backtracking.

The following path could be constructed using the matrix in (2.1):  . The path ends at 5 since $5 > 4 = n$, and the fifth row contains fewer nonzeroes than does the first row. Performing the reassignment indicated by this chain involves replacing row 1 by row 3, row 3 by row 2, row 2 by row 5, and row 5 by row 1. The resulting matrix is shown below.

$$R'RA = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad (2.4)$$

Performing the permutation results in a net gain of one zero for A_1. At each stage in the construction of a path, the algorithm will attempt to add a vertex corresponding to a row in A_2 before adding one corresponding to a row in A_1, thus tending to find the shortest and simplest permissible path. Note that Step 2 is not yet complete, for rows 2 and 6 should be interchanged to minimize the number of nonzeroes in $R'RA$.

Block Triangularizing A_1'

The square submatrix A_1' can be permuted to upper block triangular form by the modified Harwell subroutine MC13D (Duff and Reid [1978]) discussed earlier. No special modifications need be made for the rectangular case since A_1' is square. The user must remember, however, not only to symmetrically permute the rows and columns of A_1', but also the columns of A_2^* as well.

Row Permutation to Final Form

After the matrix has been permuted so that A_1^* is block upper triangular with square indecomposable diagonal blocks, the rows of the lower submatrix A_2^* must be permuted into the A_1^* matrix in such a manner as to obtain the block upper trapezoidal form. This will make some of the diagonal blocks rectangular and will remove all nonzeroes from beneath the diagonal blocks. Each row in A_2^* is examined to determine the column index of its first nonzero. The row is then inserted into A_1^* just above the row containing the transversal element in that column. If another row in A_2^* has its first nonzero in the same column, this row is inserted into A_1^* just above the row previously inserted. For example, this method would permute the matrix in (2.3) in the following manner,

$$P \begin{bmatrix} Q^t & 0 \\ 0 & I_2 \end{bmatrix} R'RAQ = \begin{bmatrix} \boxed{1} & 0 & 1 & 1 \\ 1 & \boxed{1} & 1 & 0 \\ 1 & 1 & \boxed{0} & 0 \\ 0 & 1 & 1 & \boxed{0} \\ 0 & 0 & 1 & \boxed{0} \\ 0 & 0 & 0 & \boxed{1} \end{bmatrix} \qquad (2.5)$$

The final block upper trapezoidal structure of a given matrix is of course not unique. In general there are several possible maximal transversals yielding several different A_1' submatrices, each having the minimum number of nonzeroes possible. Lack of uniqueness of the final block structure is also due to the fact that there may be more than one way in which the diagonal blocks may be permuted among themselves and still preserve the block upper trapezoidal form. Some of these permutations are more desirable than others, as they result in a "dual angular" form as described in Golub and Plemmons [1980].

III. COMPARISONS AND TEST RESULTS

1. Test Problems. The relative performance of four methods for ordering a large sparse system of linear equations prior to solution of the system using Givens rotations was compared. The four ordering options used were: 1) no ordering at all, 2) quotient minimum degree ordering, 3) nested dissection, and 4) block trapezoidal ordering. These four ordering options were compared on four sparse systems of equations, of sizes 75×50, 100×75, 150×100, and 892×261. The first three examples were constructed by the authors with known solutions in order

to check the accuracy of the programs. The fourth example consisted of actual geodetic network data obtained by the U.S. National Geodetic Survey. All programs were written in IBM H Extended Fortran and run on an IBM 370/3031 computer.

2. Software Used. The basic software package used in this project was a double precision version of SPARSPAK, a sparse matrix package of subroutines written and documented by George and Liu [1978, 1981]. This package was extended by George and Heath [1980] to provide for solving a sparse least squares problem using Givens rotations in conjunction with quotient minimum degree ordering. Harwell subroutines MC21A and MC13D, developed by Duff [1978] and Duff and Reid [1978], were adapted and used by Litsey [1980] in his implementation of the block trapezoidal ordering method. In this project, the authors combined and modified all of the above software in order to test and compare the four ordering options in conjunction with the Givens method for solving least squares problems by orthogonal decomposition.

The "no ordering" option was tested in order to provide a benchmark against which to compare the effectiveness of the other three ordering methods. The quotient minimum degree ordering scheme attempts to minimize fill-in by reordering an original $n \times n$ matrix A in the following way: at each stage, if columns $1, \ldots, k$ have been selected already for the reordered matrix, then the column in the remaining $(n - k) \times (n - k)$ submatrix with the fewest number of nonzeroes is selected as the $(k+1)$st column in the reordered matrix. The nested dissection ordering method attempts to permute the matrix A in such a way that it can be broken down recursively into subblocks which are connected in a well-defined way. As a result of this dissection process, zero blocks are formed with remain zero after the reordered matrix is factored, thus reducing the fill-in.

3. Format of Data. Data for sparse matrices is entered according to one of two schemes. In the first scheme, each non-zero entry of the matrix is entered as a triple consisting of its row index, column index, and value. Triples may be entered in any order, subject to the condition that all entries of a given row must be entered together as a group. The matrix is then stored in four arrays: arrays ICN and VALUE contain column indices and corresponding values of all non-zero elements in the matrix; array element IP(I) points to the position in ICN where column indices for the I^{th} row begin, and array element LENR(I) gives the number of non-zero entries in the I^{th} row. The Harwell-block trapezoidal ordering code requires this storage scheme. SPARSPAK routines, as adapted for Givens rotations, utilize a different scheme for storing a sparse matrix. They accept one row of the matrix at a time, according to the format NSUBS, (SUBS(K)), VALUE (K), K = 1, NSUBS) where NSUBS is the number of non-zero entries in the row, SUBS(K) is the column index of the k^{th} non-zero entry in the row, and VALUE(K) is the corresponding value of that entry.

In addition to providing the sparse coefficient matrix according to the correct format, one must input for each equation its right-hand side and a possible weighting factor for the equation (in the context of least squares).

4. Organization of Computer Programs. The basic program used in this project consists of two job steps, as indicated in Figure 1. In job step 1, data for a sparse system of equations is read according to its original format and converted to an appropriate format for the next stage (Step A). For ordering options other than block trapezoidal ordering, the data is stored on a disk according to the SPARSPAK row-by-row format and the data is passed to job step 2 (Step C). For the

block trapezoidal ordering, the original data is converted to the Harwell scheme, ordering is performed (Step B), and the resulting matrix is then converted to SPARSPAK format on the disk before being passed to step 2.

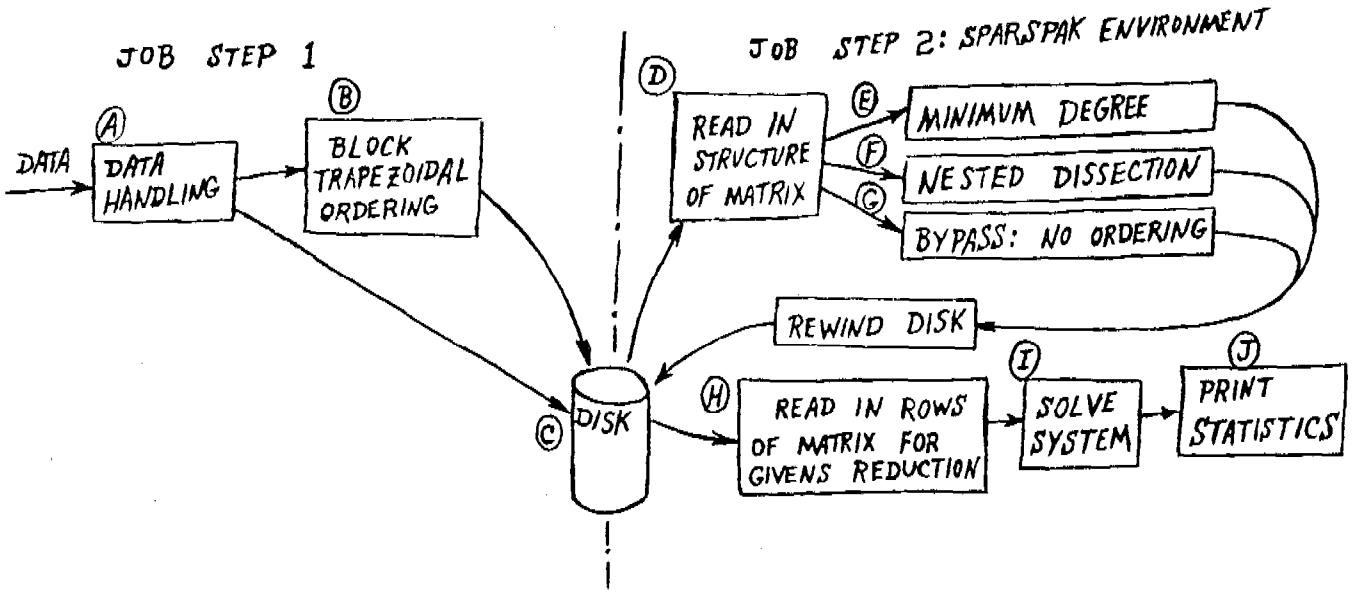
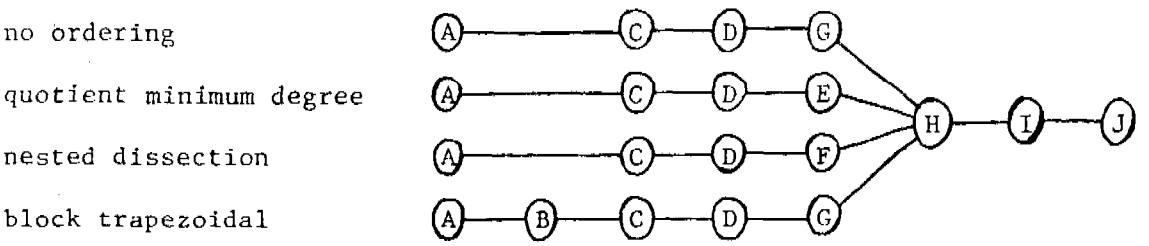


Figure 1. Schematic View of Program

Job step 2 is executed in a SPARSPAK environment under control of a main driver routine. After the zero-nonzero structure of the coefficient matrix is read in (Step D), one of three ordering options is selected: the system is ordered by quotient minimum degree ordering (Step E), nested dissection (Step F), or no ordering is performed (Step G). After rewinding the reordered data set on the disk, numerical values for the system are read one row at a time, each row is reduced by Givens rotations to form the Cholesky factor R (Step H), and the least squares solution to the system is computed (Step I). Finally, statistics provided by SPARSPAK are printed out (Step J).

The four sequences of execution steps listed below correspond to the four ordering options named:



5. Basis for Comparison of Methods. The SPARSPAK package provides statistics which estimate the storage and execution time required for solving a given system of linear equations. Execution times (in seconds) are reported for each of four individual steps: ordering (Finding permutation matrix P to obtain PAP^T), storage allocation (set up data structure for storing the Cholesky factor R of PAP^T),

factorization (numerically factor PAP^T into R^TR), and triangular solution (solve $R^Ty = Pb$, $Rz = y$, and set $x = P^Tz$). Storage requirements are given for the ordering, allocation, and factorization solution steps. Also, operation counts are given for the factorization and triangular solution steps; actual values are manipulated in these two steps. An operation is defined to be a multiplication or division since most arithmetic operations in matrix computations occur as a multiply-add pair. Final fill-in is said to occur whenever the Cholesky factor R has a non-zero element in a position which contains a zero element in the upper triangular portion of the matrix PAP^T. The amount of intermediate fill-in along with the final fill-in are reflected in the number of operations for factorization.

6. Test Results. Table I contains selected results for the four systems of equations $Ax = b$ which were tested. The size of sparse matrix A is given at the top of each column.

IV. OBSERVATIONS

1. Test results reported in Table 1 indicate that the block trapezoidal ordering scheme performs quite well in reducing both intermediate and final fill-in to the observation matrix during the orthogonal decomposition. For these four test problems, the minimum degree algorithm performed slightly better than the other two ordering schemes. For the geodetic network problem, all three ordering methods resulted in a ten-fold or greater reduction in factorization time, operations during factorization, and final fill-in.

2. With regard to ordering time, the block trapezoidal algorithm performed considerably worse than the other schemes. We feel that this is due in large part to the fact that the software for the minimum degree and nested dissection algorithms is well-developed and optimized whereas the software for the block trapezoidal algorithm is still in a rough state; further work in developing this software is underway.

TABLE I

Test Results

	75 × 50	100 × 75	150 × 100	892 × 261
No ordering				
1. Fill-in (approximate)	182	209	893	7884
2. Operations for factorization	100,314	106,186	387,445	37,211,056
3. Time for allocation	.326	.415	.532	1.301
4. Time for factorization and solution	1.084	1.223	3.274	246.728
Minimum Degree				
1. Fill-in (approximate)	0	0	254	0
2. Operations for factorization	46,162	49,288	185,588	3,128,844
3. Time for ordering and allocation	.565	.669	1.112	6.593
4. Time for factorization and solution	.676	.778	1.930	27.187
Nested Dissection				
1. Fill-in (approximate)	64	66	512	429
2. Operations for factorization	65,063	63,597	287,323	3,056,212
3. Time for ordering and allocation	.359	.491	.608	1.459
4. Time for factorization and solution	.824	.881	2.680	26.303
Block Trapezoidal				
1. Fill-in (approximate)	134	185	512	226
2. Operations for factorization	60,032	79,446	284,555	4,386,519
3. Time for ordering and allocation	1.853	4.33	3.150	219.830
4. Time for factorization and solution	.776	.975	2.651	37.720

W. F. Apce, R. H. Turner and J. L. Meyer [1976], "Optimal instrumentation planning using an LDL^T factorization," Proc. 1976 Army Numer. Anal. and Computers Conf.

J. H. Argyris and O. E. Brönlund [1975], "The natural factor formulation of the matrix displacement method," Computer Methods in Applied Mechanics and Engineering, Vol. 15, pp. 365-388.

J. H. Argyris, et al [1978], "Finite element method -- The natural factor approach," Computer Methods in Applied Mechanics and Engineering, Vols. 17, 18, pp. 1-106.

I. S. Duff [1978], "On algorithms for obtaining a maximum transversal," AERE Harwell Report CSS49, Harwell, England.

I. S. Duff and J. K. Reid [1978], "An implementation of Tarjan's algorithm for the block triangularization of a matrix," ACM Transactions on Mathematical Software Vol. 4, pp. 137-147.

A. George and F. G. Gustavson [1980], "A new proof on permuting to block triangular form," Preprint.

A. George and M. T. Heath [1980], "Solution of sparse linear least squares problems using Givens rotations," Linear Algebra and Its Applications, Vol. 34, pp. 69-84.

A. George, M. T. Heath and R. J. Plemmons [1981], "Solution of large-scale sparse least squares problems using auxiliary storage," SIAM J. Scientific and Statistical Computing, to appear.

A. George and J. Liu [1978], "Users guide for SPARSPAK -- Waterloo sparse linear equations package," Report CS-78-30, Dept. of Computer Science, University of Waterloo, Canada.

A George and J. Liu [1981], Computer Solution of Large Sparse Positive Definite Systems, Prentice-Hall, Englewood Cliffs, N.J.

G. H. Golub [1965], "Numerical methods for solving linear least squares problems," Numer. Math., Vol. 7, pp. 206-216.

G. H. Golub, F. T. Luk and M. Pagano [1980], "A sparse least squares problem in photogrammetry," Proc. Computer Science and Statistics-- Twelfth Annual Symposium on the Interface.

G. H. Golub and R. J. Plemmons [1980], "Large scale geodetic least squares adjustments by dissection and orthogonal decomposition," Linear Algebra and Its Applications, Vol. 34, pp. 3-28.

G. B. Kolata [1978], "Geodesy-Dealing with an enormous computer task," Science, Vol. 700, pp. 421-422.

J. Litsey [1980], "Finding a block upper trapezoidal form of a rectangular matrix," Thesis for the M.S. degree, Dept. of Math., University of Tennessee, Knoxville, TN.

P. Meissl [1980], "A priori prediction of roundoff error accumulation during the direct solution of super-large geodetic normal equations, "NOAA Professional Paper 12, National Geodetic Survey, Rockville, MD.

P. Vanicek, M. R. Elliott and R. O. Castle [1979], "Four dimensional modeling of recent vertical movements in the area of the Southern California uplift," Technophysics, Vol. 52, pp. 287-300.

TRANSVERSE CURRENTS AND OHMIC LOSSES OF MICROSTRIP OBTAINED FROM A MATRIX FORMULATION WHICH FACILITATES THEIR NUMERICAL CALCULATION

Peter J. McConnell and Robert L. Brooke
US Army Mobility Equipment R&D Command
Fort Belvoir, Virginia 22060

ABSTRACT: A matrix formulation is developed for calculating the transverse current distribution of microwave transmission lines and applied to solving for frequency dependent impedance and loss of microstrip commonly used for antenna feedlines and in microwave construction. The solution is based on calculating the effective inductance per unit length, and is unique in providing the frequency dependence and ohmic losses for any geometry. Results for specific geonetries are compared favorably with earlier capacitance based solutions. The frequency dependance of current distribution and characteristic impedance will be shown for two commonly employed geometries.

1. INTRODUCTION. The characteristic impedance of Microstrip Transmission Lines has been of interest for twenty-five years. An excellent compilation of selected papers is contained in reference (1) including several early papers specifically addressing Microstrip. The papers by Cohn (2), Wheeler (3), and Bryant and Weiss (4) are especially fundamental and useful for common engineering problems. These papers, and a multitude of others published since, depend on solving for the static capacitance per unit length of the selected line configuration. Most practical geometries do not lend themselves to an exact analytic solution so much effort has been devoted to developing approximate analytic solutions. With the advent of the high speed computer a considerable effort has been devoted to developing numerical techniques for solving Laplace's equation to yield the electric field configuration and capacitance of useful geometries. Reference (5) is a notable example of this approach.

The approach used in this paper, while nonanalytic is quite general and without any geometric limitation in the transverse plane. This approach is unique in that the effects of finite conductor losses and frequency dependence can be included in the analysis. Assumptions made by other authors are also made here. The lines to be analyzed are assumed to be relatively low loss lines supporting Quasi-TEM modes. Capacitance based solutions are static solutions which approach exactness only for lossless lines. The inductance based solution to be developed and applied here can allow for losses but is quasi-static and retarded potentials have not been considered.

2. THEORETICAL DEVELOPMENT. If the line has finite losses, the characteristic impedance and progagation constant, γ, can be calculated from

$$Z_o = \sqrt{\frac{R + j\omega L}{G + j\omega C}} \qquad \gamma = \alpha + j\beta = \sqrt{(R + j\omega L)(G + j\omega C)}$$

Where R and G are the series resistance and shunt conductance per unit length of the line and α and β are the attenuation and phase constant of the line. With the appropriate modifications to account for normal low loss transmission line and other reasonable assumptions given in reference (7), an alternate equation for characteristic impedance results in the form

$$Z_o = vL$$

where L is the inductance per unit length and is directly affected by both the resistance of the line and frequency.

The geometry to be solved is shown in Figure 1.

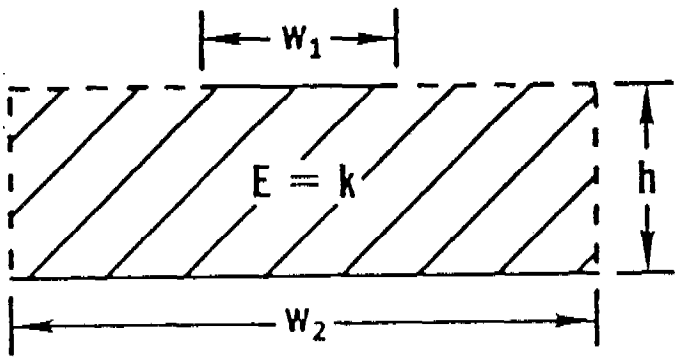


Figure 1
Transverse Geometry

For generality the two conducting tapes are allowed to have differing widths, however they will be maintained parallel and with their mid-points defining a plane normal to both. This is a simplification, and is not required, but will shorten computation time substantially. This configuration can be used to represent microstrip examples given in references (3) and (4), and direct comparisons of results made for limiting cases of high frequency and no loss. In addition, this arrangement allows for calculating the parameters of antenna feedlines where the widths are the same as well as microwave components where the ground plane has a known finite width.

Mathematically subdividing the conductors into smaller parallel sections is accomplished as shown in Figure 2. This method of subdivision is arbitrary and is retained for consistancy with reference (7).

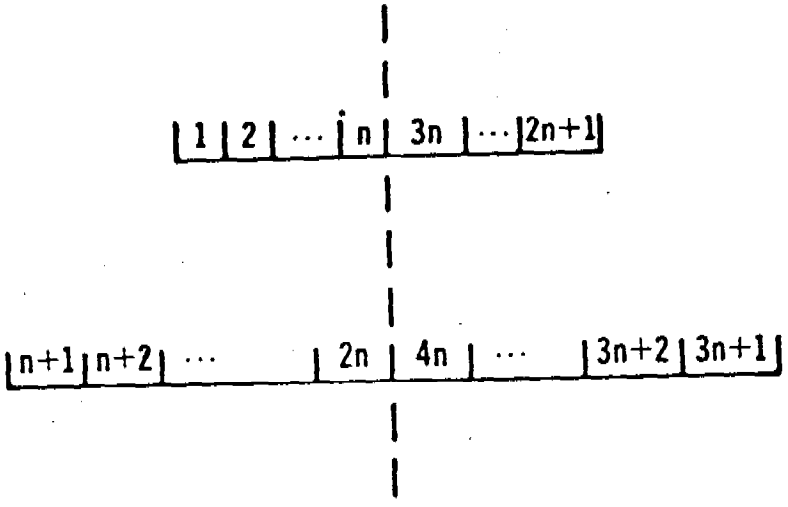


Figure 2
Method of Indexing Subdivisions

The two conducting tapes have now been replaced by 4n thin parallel tapes, each of which may carry a different current. An equivalent circuit of the transmission line then looks like that in Figure 3.

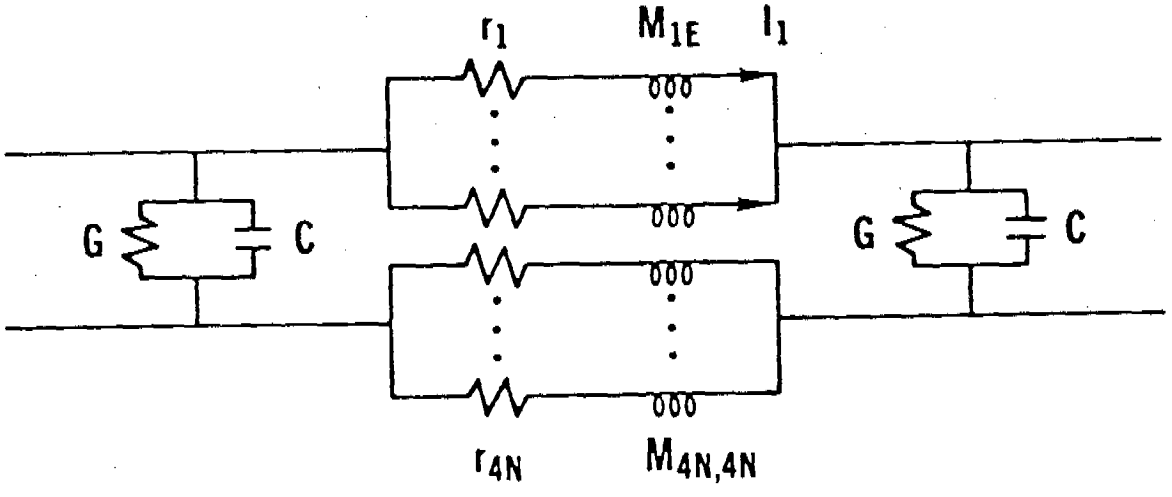


Figure 3
Equivalent Circuit

The width of each subsection will be chosen sufficiently small to consider the current density in each to be uniform. DC inductance equations are available to calculate the mutual inductance between any two subsections and the self inductance of each. The resistance of each section will be the dc resistance calculated from the input parameters of bulk resistivity and incremental area. The incremental area is defined by the smaller of the actual tape thickness or an arbitrary multiple (an input parameter) of the skin depth, and the subsection width. In this paper, the upper and lower tapes are each divided into 2n equal width subsections. References (7) and (8) discuss alternative methods used to speed convergence for different geometries.

The current in each element can be expressed in terms of an arbitrary applied voltage, resistive voltage drops, and induced voltages due to all current elements. This leads to a set of linear equations which can be numerically solved for the current in each subsection. The matrix algebra is tedious and will not be given here. The effective inductance per unit length can then be found as;

$$Leff = \frac{\frac{1}{2w} \Sigma a_k}{(\Sigma a_k)^2 + (\Sigma b_k)^2} ,$$

and the effective resistance as;

$$Reff = \frac{\frac{1}{2} \Sigma a_k}{(\Sigma a_k)^2 + (\Sigma b_k)^2} ,$$

where a_k and b_k are the in phase and quadrature components of current in each substitution.

3. RESULTS & COMPARISON. Two cases were calculated and compared with results produced by other authors. An equal width case was compared directly with the results of reference (3) and an unequal width case with the results of reference (4) providing the larger tape is at least ten times the width of the smaller, as the latter reference assumes an infinite ground plane. Only the high frequency or lossless case will be considered since this is also an assumption of the references. The comparative values obtained from the references required interpolating published response curves. The excellen agreement is more than would have been expected. The slightly high bias of the results in Table 2 are probably the result of the finite dimension of the larger tape. The program calculations have been rounded to the nearest ohm.

The current distributions produced in solving for the results of Table 1 are shown superimposed in Figure 6 for values of w/h from one to thirty. It is clear that widening the two conductors results in a more uniform distribution of current in the transverse plane and better shielding. This of course is what

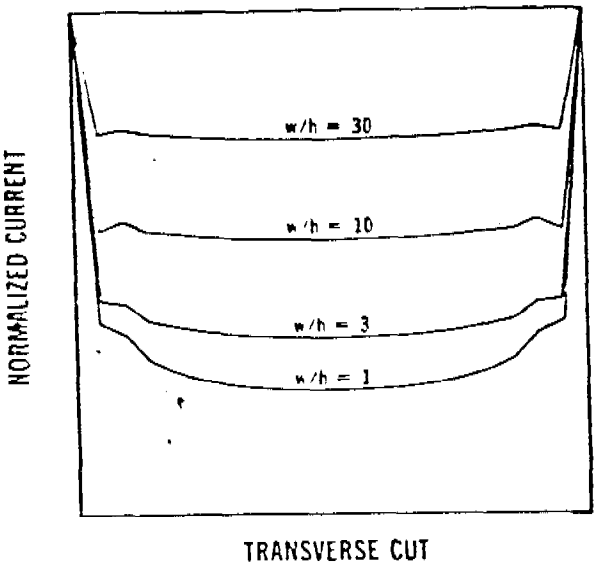
one should expect, and is used as the basis in reference (3) for a wide tape approximation, and to establish a limit for the effective dielectric constant of K dielectric and propagation velocity of $v_c/\sqrt{K_{diel}}$.

TABLE 1
COMPARISON OF RESULTS FOR EQUAL WIDTH (K=1)

w/h	W. HEILER (REF 3)	THIS METHOD
0.3	315	313
0.4	279	280
0.5	252	254
0.6	232	234
0.7	216	217
0.8	202	202
0.9	189	190
1.0	178	179
3.0	87	87
10.0	32	32.5

TABLE 2
COMPARISON OF RESULTS FOR UNEQUAL WIDTH (K=1)

w_1 / h $w_2 / w_1 = 10$	BRYANT & WEISS (REFERENCE 4)	THIS METHOD
0.6	156	160
0.8	140	142
1.0	127	128
2.0	87	90
3.0	67	70



EQUAL TAPE CURRENT DISTRIBUTION
AS A FUNCTION OF WIDTH/SEPARATION

4. CONCLUSION. In this report we have shown a method to calculate the inductance per unit length and ohmic losses of microstrip line with general cross-sectional geometry. A program, developed to apply this technique, was exercised for simple cases and the results found to agree with those of other authors using capacitance based solutions. This approach is unique in that it directly provides the transverse current distribution and ohmic attenuation at all frequencies. It can provide new insights into factors which cause loss in microwave components and help explain the effective behavior of currents on extended antenna structures. A clear understanding of antenna feedlines and radiating elements can only be reached if the current distribution is known.

For a more complete development of this approach and a detailed examination of frequency dependence of current, impedance, and loss, the reader is referred to reference (11).

References

(1) Leo Young, "Parallel coupled lines and directional couplers", Artech House Inc., 1972.

(2) S.B. Cohn, "Shielded coupled-strip transmission line", IEEE Trans. on Microwave Theory and Techniques, Vol MTT-3, No.5 pp 29-38, Oct 1955.

(3) H.A. Wheeler, "Transmission line properties of parallel strips separated by a dielectric sheet", IEEE Trans. on Microwave Theory and Techniques, Vol. MTT-13, No.2, pp 172-185, Mar 1965.

(4) T.G. Bryant and J.A. Weiss, "Parameters of Microstrip transmission lines and of coupled pairs of microstrip lines", IEEE Trans. on Microwave Theory and Techniques, Vol. MTT-16, No. 12, pp 1021-1027, Dec 1968.

(5) O.R. Guizan and R.V. Garver, "Characteristic impedance of rectangular coaxial transmission lines," IEEE Trans. on Microwave Theory and Techniques, MTT-12, pp 489-495, Sep 1964.

(6) W.J. Getsinger, "Microstrip characteristic impedance," IEEE Trans. on Microwave Theory and Techniques, Vol. MTT-27, No. 4, April 1979.

(7) R.L. Brooke, C.A. Hoer, and C.H. Love, "Inductance and characteristic impedance of a strip-transmission line," Journal of Research of NBS, C. Engineering and Instrumentation, Vol 71C, No. 1, Jan-Mar 1967.

(8) R.E. Brooke and J.E. Cruz, "Current distribution and impedance of lossless conductor systems," IEEE Trans. on Microwave Theory and Techniques, Vol. MTT-15, No. 6, June 1967.

(9) R.H. Jansen, "High speed computation of single and coupled microstrip parameters including dispersions, high-order nodes, loss, and finite strip width," IEEE Trans, on Microwave Theory and Techniques, Vol. MTT-26, No. 2, Feb 1978.

(10) C.A. Hoer and C.H. Love, "Exact inductance equations for rectangular conductors with applications to more complicated geometries, Journal of Research NBS 69C2 C. Engineering and Instrumentation, No. 2, April-June 1965.

(11) P. J. McConnell and R. L. Brooke, "Frequency Dependent Characteristics Impedance, Attenuation, and Ohmic Loss of Microstrip, Based on Numerical Calculation of Effective Inductance," to be published as a MERADCOM Technical Report.

ASPECTS OF ALGEBRAIC COMPUTATION

B. F. Caviness[1]
General Electric Research and Development Center
Schenectady, NY 12345

ABSTRACT In this brief paper we give some examples of the current state of algebraic computation plus some references for further reading on applications and the design of algebraic algorithms. The appendix contains a short directory of computer algebra systems.

1. INTRODUCTION Algebraic (or symbolic) computation is a type of scientific computing in which computations are carried out with algebraic and other symbols in addition to numeric entities. Also, typically, the computations are carried out exactly, unlike most numerical calculations where computations are carried out using approximate arithmetic. In this short paper we will give some examples of the capabilities of current computer algebra systems, note some applications, and suggest what the future holds.

2. EXAMPLES OF CURRENT CAPABILITIES Some examples will help to clarify what is meant by algebraic computation, to distinguish it from the more commonly known numeric computation, and to indicate the scope and abilities of current systems. It is possible using algebraic systems to calculate indefinite integrals such as $\int \sin x\, dx$ and obtain the result $-\cos x$. It is also possible to calculate some definite integrals exactly. For example, using Macsyma (see the end of the demonstration given below) [MATH77] one can ask to calculate $\int_{-\infty}^{\infty} \frac{\sin x}{x} dx$ and receive the exact answer, π. This integral can, of course, also be calculated using numerical techniques, but one then obtains an approximate answer.

To further indicate the scope and abilities of modern algebraic computation systems, we give below a copy of a session using Macsyma. The lines labelled (C1), (C2), ···, are inputs typed by the user. The corresponding output lines produced by the computer are labelled (D1), (D2), ···. The CPU time required for each computation is given just before each computer response. The times, given in milliseconds, are for a PDP-KL10 at MIT. Lines enclosed in /* ··· */ brackets are comments typed by the user.

```
(C1) /*  Read pretyped file of commands */
demo(bfc,demo1);

(C2) /* Display CPU time */
showtime:true$
Time= 2 msec.

(C3) /* Type and display function */
```

1. Present address: Department of Computer and Information Sciences, University of Delaware, Newark, Delaware 19711

```
X/(X**3+1);

Time= 9 msec.
                                        X
(D3)                                  ------
                                       3
                                      X  + 1


(C4) /*  Add previous function to itself. % refers to the immediately
         preceding expression, D3 in this case. */

%+%;
Time= 3 msec.
                                      2 X
(D4)                                  ------
                                       3
                                      X  + 1

(C5) (X+3)**20;
Time= 8 msec.
                                             20
(D5)                                 (X + 3)

(C6) /*  Expand the previous expression */

RATSIMP(%);
Time= 100 msec.
      20       19         18          17           16            15
(D6) X   + 60 X   + 1710 X   + 30780 X   + 392445 X   + 3767472 X

             14              13              12              11
 + 28256040 X   + 169536240 X   + 826489170 X   + 3305956680 X

                10                9                8                 7
 + 10909657044 X   + 29753610120 X  + 66945622770 X  + 123591918960 X

                 6                 5                4                 3
 + 185387878440 X  + 222465454128 X  + 208561363245 X  + 147219785820 X

                2
 + 73609892910 X  + 23245229340 X + 3486784401
/* Note the large integers in the above expression that occur without
   any truncation */
(C7) /* Differentiate the previous expression */

DIFF(%,X);
```

```
Time= 198 msec.
          19          18          17          16          15          14
(D7) 20 X   + 1140 X   + 30780 X   + 523260 X   + 6279120 X   + 56512080 X

             13                12                11                10
 + 395584560 X   + 2203971120 X   + 9917870040 X   + 36365523480 X

                9                 8                 7                 6
 + 109096570440 X  + 267782491080 X  + 535564982160 X  + 865143432720 X

                 5                 4                 3                 2
 + 1112327270640 X  + 1112327270640 X  + 834245452980 X  + 441659357460 X

 + 147219785820 X + 23245229340


(C8) /*  Now factor it */

FACTOR(%);
Time= 1186 msec.
                                                19
(D8)                                  20 (X + 3)

(C9) /*  This shows a numerical capability of the system.  %e is the
         constant e. */

%e**x**3;
Time= 10 msec.
                                              3
                                             X
(D9)                                        %E

(C10) ROMBERG(ev(%),X,1,2);
/*  This computation requires some programs to be loaded from disk */

ROMBRG FASL DSK MACSYM being loaded
Loading done

NUMER FASL DSK MACSYM being loaded
Loading done
Time= 1165 msec.
(D10)                                275.51098

(C11) /* Macsyma has several routines for manipulating
   series of various kinds.  The following is a Taylor series. */

TAYLOR(SIN(X),X,0,9);

HAYAT FASL DSK MACSYM being loaded
Loading done
```

```
Time= 71 msec.
                                  3    5      7       9
                                 X    X      X       X
(D11)/T/                  X - -- + --- - ---- + ------ + . . .
                             6    120   5040   362880


(C12) /* Taylor can also compute Laurent expansions */

TAYLOR(1/(COS(X)-SEC(X))**3,X,0,5);

EULBRN FASL DSK MAXOUT being loaded
Loading done
Time= 233 msec.
                                                        2          4
           1     1       11      347    6767 X    15377 X
(D12)/T/ - -- + ---- + ------ - ----- - ------- - -------- + . . .
            6      4        2   15120   604800    7983360
           X    2 X    120 X

(C13) /* Macsyma can solve some systems of non-linear equations.  Here
         we compute exactly the six roots of unity. */

SOLVE(X**6-1);

SOLVE FASL DSK MACSYM being loaded
Loading done
Time= 1295 msec.
           SQRT(3) %I + 1       SQRT(3) %I - 1                    SQRT(3) %I + 1
(D13) [X = --------------, X = --------------, X = - 1, X = - --------------,
                 2                    2                              2

                                                      SQRT(3) %I - 1
                                                X = - --------------, X = 1]
                                                            2

(C14) /* Now solve system of equations */

SOLVE([A*X+B*Y = 0,C*X+D*Y = 1],[X,Y]);
Time= 163 msec.
                                    B                 A
(D14)                  [[X = ----------, Y = - ----------]]
                             B C - A D         B C - A D

(C15) /* Now define a matrix */

MATRIX([A,B,C],[D,E,F],[G,H,I]);
Time= 7 msec.
                                  [ A  B  C ]
                                  [         ]
(D15)                             [ D  E  F ]
                                  [         ]
                                  [ G  H  I ]
```

```
(C15) /* and take its transpose */

TRANSPOSE(%);

Time= 6 msec.
                                  [ A  D  G ]
                                  [         ]
(D16)                             [ B  E  H ]
                                  [         ]
                                  [ C  F  I ]


(C17) /* Now compute the matrix product of it and its transpose.
         %TH(2) refers to the second previous expression, D15 in this
         case. */

% . %TH(2);

MDOT FASL DSK MACSYM being loaded
Loading done
Time= 130 msec.
           [   2    2    2                                          ]
           [  G  + D  + A     G H + D E + A B   G I + D F + A C     ]
           [                                                        ]
(D17)      [                    2    2    2                        ]
           [ G H + D E + A B   H  + E  + B     H I + E F + B C      ]
           [                                                        ]
           [                                     2    2    2       ]
           [ G I + D F + A C   H I + E F + B C   I  + F  + C       ]

(C18) /* Create a Vandermonde matrix */

MATRIX([X**2,X,1],[Y**2,Y,1],[Z**2,Z,1]);
Time= 10 msec.
                                  [  2       ]
                                  [ X   X  1 ]
                                  [          ]
(D18)                             [  2       ]
                                  [ Y   Y  1 ]
                                  [          ]
                                  [  2       ]
                                  [ Z   Z  1 ]

(C19) /* Now compute its determinant */

DETERMINANT(%);
Time= 39 msec.
                        2       2    2     2      2
(D19)            - Y Z  - X (Y  - Z ) + Y  Z + X  (Y - Z)

(C20) /* Factor the determinant */

Factor(%);
```

```
Time= 750 msec.
(D20)                    - (Y - X) (Z - X) (Z - Y)

(C21) /* Symbols can be given mathematical properties */

DECLARE(N,ODD)$
Time= 12 msec.

(C22) /* and then these are used in evaluating subsequent expressions */

COS(N*%PI/2);
Time= 56 msec.
(D22)                                0

(C23) F(X+Y);
Time= 4 msec.
(D23)                             F(Y + X)

(C24) /* Another such example */

DECLARE(F,LINEAR)$
Time= 3 msec.

(C25) F(X+Y);
Time= 9 msec.
(D25)                           F(Y) + F(X)

(C26) /* A dramatic example of "infinite-precision" arithmetic */

100!;
Time= 34 msec.
(D26) 933262154439441526816992388562667004907159682643816214685929638952175999#

9932299156089414639761565182862536979208272237582511852109168640000000000000#

0000000000

(C27) /* Macsyma can also be used as a programming language.  The following
         code defines the factorial function. */

FAC(N):=IF N = 0 THEN 1 ELSE N*FAC(N-1);
Time= 5 msec.
(D27)           FAC(N) := IF N = 0 THEN 1 ELSE N FAC(N - 1)

(C28) /* Now use our newly defined function.  Compare the execution time
         to the built-in factorial function used in C26. */
FAC(5);
Time= 54 msec.
(D28)                               120

(C29) /* There are also facilities for large floating-point precision.
         The following instruction sets the floating-point precision
         to 50 decimal places. */
```

```
FPPREC:50$

FLOAT FASL DSK MACSYM being loaded
Loading done
Time= 41 msec.

(C30) /* Now print pi with this precision. */
BFLOAT(%PI);
Time= 53 msec.
(D30)       3.14159265358979323846264338327950288419716939937510
```

```
(C32) demo(bfc,demo2);

(C33) SHOWTIME:TRUE$
Time= 3 msec.

(C34) X/(X**3-1);
Time= 10 msec.
```

$$\text{(D34)} \qquad \frac{X}{X^3 - 1}$$

```
(C35) /* Compute the indefinite integral of the previous expression. */
INTEGRATE(%,X);

SIN FASL DSK MACSYM being loaded
Loading done

SININT FASL DSK MACSYM being loaded
Loading done

SCHATC FASL DSK MACSYM being loaded
Loading done
Time= 419 msec.
```

$$\text{(D35)} \qquad -\frac{\log(X^2 + X + 1)}{6} + \frac{\text{ATAN}\left(\frac{2X+1}{\text{SQRT}(3)}\right)}{\text{SQRT}(3)} + \frac{\log(X-1)}{3}$$

```
(C36) /* Differentiate the result. */
DIFF(%,X);
Time= 67 msec.
```

$$\text{(D36)} \qquad \frac{2}{3\left(\frac{(2X+1)^2}{3} + 1\right)} - \frac{2X+1}{6\,(X^2 + X + 1)} + \frac{1}{3\,(X-1)}$$

```
(C37) /*  Macsyma does not automatically simplify its results so we
          must tell it to do so. */
```

```
RATSIMP(%);
Time= 79 msec.
                                      X
(D37)                                ------
                                      3
                                     X  - 1

(C38) X*SIN(X)+%E**X**2+1/LOG(X);
Time= 28 msec.
                                                        2
                                        1              X
(D38)                  X SIN(X) + ------- + %E
                                    LOG(X)

(C39) INTEGRATE(%,X);

RISCH FASL DSK MACSYM being loaded
Loading done

PFRAC FASL DSK MAXOUT being loaded
Loading done

ERF FASL DSK MAXOUT being loaded
Loading done

RPART FASL DSK MACSYM being loaded
Loading done
Time= 2488 msec.
             /
             [   1          SQRT(%PI) %I ERF(%I X)
(D39)        I ------ dX - ---------------------- + SIN(X) - X COS(X)
             ] LOG(X)                2
             /
/* In the above example Macsyma was unable to integrate the first term
   in which case it simply inserts an integral sign in front of the
   the integrand.  ERF denotes the error function. */

/* To gain more space it was necessary to restart Macsyma for the next demo.
   Thus the command numbers recycle. */

(C9) /* Assign an expression to the variable F1. */
F1:SIN(X)/X;
Time= 14 msec.
                                      SIN(X)
(D9)                                  ------
                                        X

(C10) /* Now we demonstrate a simple graphics capability of Macsyma on a
         character display.  It would look nicer on a graphics display,
         but the important point here is not the actual display on a
         character terminal but the fact that such capabilities are
         integrated into the system in a natural way. */
PLOTNUM:PLOTNUM1:50$
```

```
Time= 3 msec.

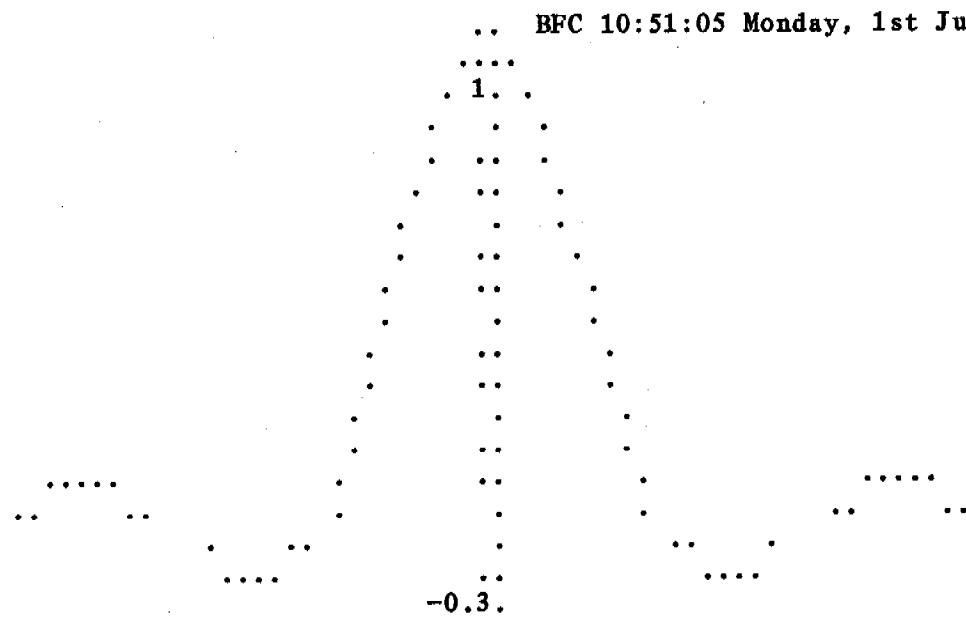
(C11) PLOT2(F1,X,-12,12);

APLOT2 FASL DSK SHARE being loaded
Loading done

TEKPLT FASL DSK SHARE being loaded
Loading done

FFORMA FASL DSK LIBLSP being loaded
Loading done

PRINT FASL DSK SHARE being loaded
Loading done
```



Xmin = -12 Xmax = 12 Ymin = -0.3 Ymax = 1

```
Time= 1909 msec.
(D11)                                DONE

(C12) INTEGRATE(F1,X,-INF,INF);
Time= 7507 msec.
(D12)                                %PI
```

The above examples display only a few of the facilities available in systems such as Macsyma and Reduce [HEAA71]. There have been dozens of computer algebra systems developed in the past fifteen years. For a directory of some of the best-known and most widely available in the U. S. see the appendix.

Computer algebra systems have been used in hundreds of applications including pure mathematics, celestial mechanics, general relativity, high energy physics, NMR imaging, economics, acoustics, computer-aided design, design of VLSI circuits, fluid mechanics, fracture mechanics, helicopter blade motion, ship hull design, underwater shock waves plus many others. For information on various applications see [PRF81, LEWV79, JENR76, NASA77, NGEW79, and WANP81].

The success of today's systems has been made possible by important improvements during the last decade in many fundamental computational algorithms plus the discovery of algorithms for some problems such as indefinite integration where no algorithm was previously known. Important progress has been made on gcd computations, factoring, resultant computations, simplification, integration, and solution of ordinary differential equations in closed form. Most of the important papers on algorithmic advances can be found in the proceedings of various ACM SIGSAM symposia [WANP81, NGEW79, JENR76, PETS71]. Some notable exceptions are papers on integration by Risch [RISR69, RISR70], the book by Davenport [DAVJ81], Singer's paper [SINM81] on solving nth order homogeneous ordinary differential equations in closed form, Gosper's paper [GOSR78] on summation of series, and the works of Musser [MUSD75], Wang [WARO75, WANP78], Yun [MOYU73, YUND74], Zassenhaus [ZASH69], and Zippel [ZIPR79] on polynomial factorization. The paper by Yun and Stoutemyer [YUST80] gives a good survey of many aspects of algebraic computation.

In the future we will see continued progress on new algorithms, continued progress on system development, and the appearance of powerful scientific workstations using the personal computers currently appearing on the market with integrated numeric, algebraic and graphics software.

3. CONCLUSION Dramatic advances are being made in scientific computation today. By the year 2000, or perhaps sooner, the scientific computation world of the average scientist or engineer will be significantly changed. Essentially all the known mathematical computational methods used with pencil and paper today will be programmed into personal workstations. putting the best and latest techniques at the fingertips of each technical worker thereby giving him or her the ability to routinely solve problems that they were previously unable to do because of a lack of personal knowledge, computing power, or both. Previously solvable problems will be doable in a fraction of the scientist's time required today thereby tremendously increasing the productivity of all technical researchers.

Indeed many of these promises are here today. Some laboratories have already made algebra systems available to their employees. The Navy has set up a network for the use of Macsyma. Other organizations are planning to make algebra systems available or are expanding current facilities while others are just beginning to realize their great potential. This author believes that nothing short of a revolution in scientific computation is underway!

REFERENCES [DAVJ81] J. H. Davenport, On the Integration of Algebraic Functions, Lecture Notes in Computer Science, Vol. 102, Springer-Verlag, New York (1981).

[GOSR78] R. W. Gosper, Jr., Decision Procedure for Indefinite Hypergeometric Summation, Proc. Nat. Acad. Sci USA 75 (1978), 40-42.

[HEAA71] A. C. Hearn, REDUCE 2 - A System and Language for Algebraic Manipulation, in [PETS71], 128-133.

[JENR76] R. D. Jenks (ed.), Proceedings of the 1976 ACM Symposium on Symbolic and Algebraic Computation, Assoc. for Computing Machinery, New York (1976).

[LEWV79] V. Ellen Lewis (ed.), Proceedings of the 1979 MACSYMA Users' Conf., MIT Laboratory for Computer Science, 545 Technology Sq., Cambridge, MA 02139.

[MATH77] MACSYMA Reference Manual, Version 9, The Mathlab Group, Laboratory for Computer Science, MIT, 545 Technology Square, Cambridge, MA 02139 (December 1977).

[MOYU73] J. Moses and D. Y. Y. Yun, The EZGCD Algorithm, Proc. of ACM National Conf. (1973), 159-166.

[MUSD75] D. R. Musser, Multivariate Polynomial Factorization, J. ACM 22, 2 (1975), 291-308.

[NASA77] Proceedings of the 1977 MACSYMA Users' Conf., available from National Technical Information Service, Springfield, VA 22161.

[NGEW79] E. W. Ng (ed.), Symbolic and Algebraic Computation, Lecture Notes in Computer Science, Vol. 72, Springer-Verlag, New York (1979).

[PRF81] R. Pavelle, M. Rothstein and J. Fitch, Computer Algebra, Scientific American, (to appear).

[PETS71] S. R. Petrick (ed.), Proceedings of Second Symposium on Symbolic and Algebraic Manipulation, Assoc. for Computing Machinery, New York (1971).

[RISR69] R. H. Risch, The Problem of Integration in Finite Terms, Trans. Amer. Math. Soc. 139 (1969), 167-189.

[RISR70] _____, The Solution of the Problem of Integration in Finite Terms, Bull. Amer. Math. Soc. 76 (1970), 605-608.

[SINM81] M. F. Singer, Liouvillian Solutions of nth Order Homogeneous Linear Differential Equations, Amer. J. Math. (to appear).

[WANP78] P. S. Wang, An Improved Multivariate Polynomial Factoring Algorithm, Math. Comp. 32 (1978), 1215-1231.

[WANP81] _____ (ed.), Proceedings of the 1981 ACM Symposium on Symbolic and Algebraic Computation, Assoc. for Computing Machinery, New York (1981).

[WARO75] _____ and L. P. Rothschild, Factoring Multivariate Polynomials over the Integers, Math. Comp. 29 (1975), 935-950.

[YUND74] D. Y. Y. Yun, The Hensel Lemma in Algebraic Manipulation, Ph.D.

Thesis, Massachusetts Institute of Technology, Cambridge, MA (1974).

[YUST80] _____ and D. Stoutemyer, Symbolic Mathematical Computation, Encyclopedia of Computer Science and Technology, 15, Marcel Dekker, New York (1980).

[ZASH69] H. Zassenhaus, On Hensel Factorization I, J. Number Theory 1 (1969), 291-311.

[ZIPR79] R. E. Zippel, Probabilistic Algorithms for Sparse Polynomials, Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA (1979).

APPENDIX

A Directory of Some Computer Algebra Systems

For a more complete directory see the article "Symbolic Mathematical Computation" by David Yun and David Stoutemyer [YUST80].

ALDES/SAC-2. A highly portable, batch system with a growing list of facilities. Detailed and accurate documentation. Available from Prof. G. E. Collins, 1210 W. Dayton St., Department of Computer Sciences, Univ. of Wisconsin, Madison, WI 53706.

ALTRAN. A highly portable, batch system restricted primarily to rational function and truncated power series computations. Excellent documentation and error diagnostics. Available from the Computing Information Library, Bell Laboratories, 600 Mountain Ave., Murray Hill, NJ 07974.

MACSYMA. The most extensive of all the computer algebra systems. Runs interactively under ITS on a PDP-10, under MULTICS on a Honeywell, and under Berkeley UNIX on a VAX. Also available via the ARPA net. For information contact Prof. Joel Moses or V. Ellen Golden, MIT Laboratory for Computer Science, 545 Technology Sq., Cambridge, MA 02139.

muMATH. A microcomputer algebra system intended for educational and personal use. Commercially available from The Soft Warehouse, P.O. Box 11174, Honolulu, Hawaii 96828.

REDUCE. A portable, interactive system with many facilities. Has been used for many applications, mostly in physics. Documentation weak. Available from Dr. A. C. Hearn, Rand Corp., 1700 Main Street, Santa Monica, CA 90406.

SCRATCHPAD. An interactive system under development at IBM Research. Has many facilities. For information contact Dr. R. D. Jenks or Dr. D. Y. Y. Yun, IBM T. J. Watson Research Center, P. O. Box 218, Yorktown Heights, NY 10598.

NUMERICAL SOFTWARE FOR FIXED POINT MICROPROCESSOR APPLICATIONS AND FOR FAST IMPLEMENTATION OF MULTIGRID TECHNIQUES

Steve McCormick
Department of Mathematics
Colorado State University
Fort Collins, Colorado 80523

ABSTRACT. The first part of this paper will describe the considerations that must be made for the development of numerical software routines for limited environment microcomputer evaluation of elementary functions. Though the presentation has broader implications, it is assumed that the target microcomputer is a single chip, binary, fixed point, truncated arithmetic, and short wordlength device. The application is assumed to demand a real-time, dedicated, special purpose device. The main feature of this part of the paper is guidelines recommended for software development in such an environment.

The second part of the paper will describe a very simplified viewpoint of multigrid methods as single grid directional minimization algorithms for variationally posed problems. This viewpoint leads to very simple, broad convergence theory, but the purpose of this talk is to describe how it can be exploited to develop test code for multigrid application. More specifically, this viewpoint suggests a means for modifying existing relaxation routines in order to produce a multigrid simulator. Such modifications involve only the relaxation process, can be implemented in a surprisingly small amount of code, do not increase storage requirements nor impact the data structure, and eliminate the need to determine the fine-to-coarse grid transfer operator and coarse grid equations. Though somewhat less efficient than the usual multigrid code, it offers a very quick way of applying multigrid to perhaps a very large and complex existing software package. Included in the talk is a description of a routine for solving general two-dimensional elliptic boundary value problems on a rectangle. It was implemented in BASIC on a Hewlett Packard 9845 in about sixty lines of code.

1. NUMERICAL SOFTWARE FOR FIXED POINT MICROPROCESSOR APPLICATIONS. The first part of this paper concerns the task of implementing numerical software in a very limited microprocessor environment. The focus is on guidelines for development of software for elementary function evaluation. These guidelines have evolved during a research project initially supported by the National Science Foundation and later by the U. S. Army Research Office. It is the culmination of the effort headed by G. Taylor, M. Andrews, and the author. Since a detailed report [1] and several research papers [2]-[6] were published containing the details of this subject, we merely paraphrase the introduction of [1] for our purposes here.

The report focus on numerical methods for limited environment microprocessor implementation: the target *microprocessor* is assumed to be a *single chip, binary, fixed point, truncated arithmetic, short wordlength* (16-bit or less) processor; and the *application* is assumed to demand a *real-time, dedicated, special-purpose* device (as opposed to an application-detached general purpose computer system) and requires *near full machine accuracy.*

The *main objective* of the report is to present guidelines for the development of software routines for evaluation of elementary functions. There is essentially no reference to sources for acquiring existing software because such sources are apparently nonexistent, although some sources seem to be on the horizon.

Narrowing the subject to elementary functions was essential. Although brief comments are made that relate to the implementation of other numerical tasks such as Fast Fourier Transforms, *explicit* computational problems (e.g., transforms, matrix multiplies, and polynomial evaluations) represent too broad of an area to treat in such a report. Moreover, except for function evaluation, there is generally too little known about solving *implicit* problems (e.g., inverse transforms, matrix equations, and differential equations) in short wordlength fixed point arithmetic.

Even with the focus limited to elementary functions, there are certain difficulties. First, existing and future microprocessors and applications are markedly diverse in nature. Tradeoffs for accuracy, speed, system resource usage, and reliability are complex and must be considered carefully for each development under design. Second, there is much controversy surrounding predictions of the future of microelectronics which complicates the task of presenting guidelines for design. Third, there is a great variety of algorithms and forms of algorithm implementations available. General recommendations are therefore difficult to make. Fourth, there are many alternatives to *machine language* or *microprogrammed* implementation of numerical algorithms including table look-up, existing numerical processor chips, and special chip design with hardwired computation.

These difficulties dictate two philosophical features evident in the report. First, the sample algorithms and implementations are not the best choice for every environment, but should prove suitable for most in the limited setting defined above. General comments and suggestions are made where appropriate so that one can view the sample algorithms as illustrations of the general concepts. The second feature is that the suggestions represent what can be done by implementing numerical function evaluations in *software*. The report makes only brief reference to the trade-offs with respect to the other alternatives. Thus, the comments should be viewed as tools for system design that can be considered along with other alternatives in light of the specific application.

Though the choice of *software* implemented function evaluation is left to the decision of the reader, there are some apparent recommendations made in the report. Perhaps the future will involve numerical algorithms implemented in customized hardwired chip designs, in chips programmed during the last stages of fabrication, for example. For such implementations, the guidelines in the report would, in fact, be relevant for designs based upon these alternatives. But for the present, the main competitors to software implementation are table look-up and floating point chips. The sample algorithms presented in the report can be implemented in very compact mode (50 or 60 words are typical), which may in some cases be implemented on the processor chip itself, and executed at a speed equivalent to at most a few fixed point multiplies/divides. (Note that multiply/divide may be hardwired or softwired, depending on the host processor.) This is significantly faster than existing floating point chips, although the speed gap will narrow dramatically with the introduction of faster floating point processors. Of course, the longer wordlength floating point chips provide greater potential accuracy. Nevertheless, software implementation may be somewhat cheaper, although program development costs must be accounted for, and should require less hardware complexity. On the other hand, table look-up is certainly faster than either alternative and attractive when memory demands permit. In a loose sense, then, software implementation dominates 16-bit applications while the table look-up and floating point chip alternatives are more competitive in shorter and longer wordlength applications, respectively. The trade-offs between table look-up and software implementation for 8- and 16-bit microprocessors is exemplified by the square root function treated in [2].

2. NUMERICAL SOFTWARE FOR FAST IMPLEMENTATION OF MULTIGRID TECHNIQUES.

There are several algebraic interpretations of multigrid methods for general matrix problems. (See [7] for example and the references cited therein.) For symmetric variationally posed problems, a very useful algebraic point of view is developed by considering the coarse grid computations as they effect the fine grid approximation. In fact, this viewpoint can be exploited [8] to achieve a theory including rigorous and sharp rates of convergence under very general conditions. However, the purpose of this second part of the paper is to describe how this viewpoint provides for a very fast and simple method of implementing multigrid in software. More precisely, this point of view defines a method that is theoretically equivalent to multigrid. Though computationally less appealing, it can be implemented with minimal design effort and in very short code, and does not involve much impact on an existing software package that is being modified for multigrid application. We begin by defining this method, which we suggestively call unigrid. (See [9] for more details.)

For focus, suppose A is an n x n real, symmetric, positive definite matrix. With f in R^n, then the problem is to find u in R^n satisfying

$$Au = f. \tag{1}$$

Many iterative methods for solving (1) can be described as directional iterations in the following sense. With an approximation, U, in R^n to u (such approximations will always be represented in capitals), then a direction d in R^n is computed (the choice of direction d defines the method) and used to update U in such a way that the new residual error is orthogonal to d. More precisely, let r = AU - f. Then an iteration with direction d is given by

$$U \leftarrow U - sd \tag{2}$$

$$s = \frac{\langle r, d\rangle}{\langle Ad, d\rangle}.$$

Here we use the arrow to denote replacement, thereby avoiding the use of subscripts for iteration indices. We write (2) in the compact form

$$U \leftarrow G_d(U). \tag{3}$$

One sweep of Gauss-Seidel applied to (1) can be written as several iterations in (3) with directions $d = e_1, e_2, \ldots, e_n$, where e_i is the i-th column of the n x n identity. Suppose for definiteness that A represents a uniform grid discretization of a *one-dimensional* elliptic boundary value problem on a finite interval with Dirichlet boundary conditions. Then vectors in R^n are thought of as vector functions of the interior grid points so that e_i is the vector function that is zero everywhere except at the i-th grid point (where the grid points are numbered, say, lexicographically). It is not difficult to see that these *spiked* directions do not reduce the error

$$E = U - u \tag{4}$$

very well. More precisely, the "oscillatory" (cf. [7]) error components are quickly reduced, but the "smoother" ones are not. The natural suggestion then is to also use smoother directions. To this end, suppose for simplicity that $n = 2^m - 1$ and define d_i^k recursively by

$$\begin{aligned} d_i^m &= e_i, && 1 \le i \le 2^m - 1 \\ d_i^k &= \tfrac{1}{4}d_{2i-1}^{k+1} + \tfrac{1}{2}d_{2i}^{k+1} + \tfrac{1}{4}d_{2i+1}^{k+1}, && 1 \le i \le 2^k - 1,\ 1 \le k < m. \end{aligned} \tag{5}$$

(These directions are actually intended for use with one-dimensional problems for which (1) is a discretization. Higher dimensional versions can be defined by combinations analogous to (5).)

These directions are progressively smoother with decreasing k. Note that d_i^1 is the tent function centered at the midpoint of the interval.

With these directions, one cycle of unigrid on "level" m is now defined recursively in terms of parameters ν, μ by

Level 1 cycle: Perform one iteration via $U \leftarrow G_{d_i^1}(U)$.

Level k cycle: Perform ν sweeps on directions d_i^k where one sweep is to do $U \leftarrow G_{d_i^k}(U)$ for $i = 1, 2, \ldots, 2^k - 1$. Now perform μ cycles on level $k - 1$.

The conventional multigrid method is a delayed displacement process of updating the fine grid approximation U after many computations are performed on coarser levels $m - 1, m - 2, \ldots, 1$. Though it would be computationally more expensive, an immediate displacement multigrid process would correct U (and compute the coarse grid equations) whenever any coarse grid approximation update is made. A somewhat subtle analysis shows that both of these methods are in fact equivalent; it is very easy to see that immediate replacement, and hence conventional, multigrid is fully equivalent to unigrid if we define the fine-to-coarse grid transfer operator I_f^c in terms of the coarse-to-fine operator I_c^f as

$$I_f^c = I_c^{f\,T} \tag{6}$$

and if the coarse grid operator, A_c, is defined in terms of the fine grid operator A_f as

$$A_c = I_f^c A_f I_c^f. \tag{7}$$

(For the finest level m, $A_f = A$.) For the way in which we have defined unigrid, I_c^f is linear interpolation although any reasonable interpolation process can be employed. We call (6) and (7) the *variational conditions* because they are naturally satisfied by finite element-type discretizations of (1).

We have described a version of multigrid designed for one-dimensional discretizations. To extend unigrid to higher dimensions, it is simple to define the corresponding smooth directions. (Actually they are the interpolants if the coordinate functions, e_i, from coarse grids.)

If unigrid is equivalent to multigrid but computationally less efficient, then what is its purpose? In addition to analytical simplicity which leads to a very simple theory [8], unigrid is very easy to program. In fact, to modify an existing possibly very complex software package (say one that solves a complex system of time dependent equations) that presently implements Gauss-Seidel (or SOR or some other relaxation scheme), it is enough to modify the relaxation routine. Thus, design involves only computing the direction (which is equivalent but somewhat simpler than defining I_c^f). Implementing unigrid does not require defining any other grid transfer operators, scale factors, or coarse grid equations. Implementation of the design principles (6) and (7) is automatic. Moreover, unigrid does not impact the software data structure. If the directions are generated each time they are used, then no coarse grid information is stored. Finally, many algorithm variations can be implemented and tested much more quickly and safely than with conventional multigrid. Once the design is completed, this multigrid "simulator" may be replaced by a careful implementation of conventional multigrid with the

confidence that a good design was achieved and with the ability to use unigrid as a benchmark to ensure the correctness of the final product.

To illustrate the simplicity of unigrid, we include the listings of a code for solving

$$-\Delta u + e^{x+y}u = \sin 3(x + y) \quad \text{in } \Omega = (0, 3) \times (0, 2)$$

(8)

$$u = \cos 3(x + y) \quad \text{on } \partial\Omega.$$

It was programmed in BASIC on an HP9845 and uses the usual five-point discretization on the fine grid (although, because of (6) and (7), it simulates nine-point stencils on coarser grids). To apply unigrid to a more general operator in (8), simple changes should be made to statements 210 and 350-380.

The cycling scheme is very simple (not as defined above). This can be seen in the sample runs which are also included in this paper. Note that level 1 denotes the finest level, that is, where Gauss-Seidel sweeps are performed. Note that the performance is the same for $h = .25$ as it is for $h = .125$.

Modifications to unigrid can be made very quickly. We have many versions now in use for research purposes and are continuing to develop others for further study (e.g., for different cycling schemes, relaxation processes, and orderings, nonlinearities, eigenproblems, irregular boundaries, nonsymmetric and/or nonpositive definite operators, and more general problems). No version has taken more than an hour (and usually just a few minutes) to produce.

```
10 DIM U(97,65)
20 DISP "# GRIDS";
30 INPUT N
40 DISP "# X PIONTS, INCL. BOUNDARY POINTS";
50 INPUT I1
60 DISP "Y POINTS";
70 INPUT J1
80 DISP "H";
90 INPUT H
100 DISP "# RELAXATIONS";
110 INPUT N0
120 DISP "CONVERGENCE TOLERANCE";
130 INPUT T
140 DISP "MAX # CYCLES";
150 INPUT C1
160 PRINT "# GRIDS=";N;" #X POINTS=";I1;" #Y POINTS=";J1;" H=";H
170 PRINT "# RELAXATIONS=";N0;" CONVERGENCE TOL=";T;" MAX # CYCLES";C1
180 C=0
190 FOR I=1 TO I1
200 FOR J=1 TO J1
210 U(I,J)=COS(3*(I+J-2)*H)
220 NEXT J
230 NEXT I
240 C=C+1
250 FOR K=1 TO N
260 M1=2^(N-K)
270 FOR N8=1 TO N0
280 E=0
290 FOR I=1+M1 TO I1-M1 STEP M1
300 FOR J=1+M1 TO J1-M1 STEP M1
310 A1=0
320 R1=0
330 FOR I3=I-M1+1 TO I+M1-1
340 FOR J3=J-M1+1 TO J+M1-1
350 D=4+EXP((I3+J3-2)*H)*H*H
360 R=D*U(I3,J3)-U(I3,J3-1)-U(I3,J3+1)-U(I3-1,J3)-U(I3+1,J3)
370 R=R-SIN(3*(I3+J3-2)*H)*H*H
380 A3=D*FND(I3,J3)-FND(I3,J3+1)-FND(I3,J3-1)-FND(I3+1,J3)-FND(I3-1,J3)
390 R1=R1+FND(I3,J3)*R
400 A1=A1+FND(I3,J3)*A3
410 NEXT J3
420 NEXT I3
430 S=R1/A1
440 E=E+R1*R1
450 FOR I3=I-M1+1 TO I+M1-1
460 FOR J3=J-M1+1 TO J+M1-1
470 U(I3,J3)=U(I3,J3)-S*FND(I3,J3)
480 NEXT J3
490 NEXT I3
500 NEXT J
510 NEXT I
520 E=SQR(E)*M1/H
530 PRINT "LEVEL=";N-K+1;"  ERROR=";E
540 NEXT N8
550 NEXT K
560 IF E<T THEN 590
570 IF C<C1 THEN 240
580 DEF FND(I3,J3)=(M1-ABS(I-I3))*(M1-ABS(J-J3))/(M1*M1)
590  END
```

UNIGRID LISTING

```
# GRIDS= 3  #X POINTS= 13  #Y POINTS= 9  H= .25
# RELAXATIONS= 3  CONVERGENCE TOL= 0  MAX # CYCLES 2


LEVEL= 3   ERROR= 159.773562628
LEVEL= 3   ERROR= 10.8376207333
LEVEL= 3   ERROR= .130523571086
LEVEL= 2   ERROR= 155.659030674
LEVEL= 2   ERROR= 10.1746496834
LEVEL= 2   ERROR= 1.07841439539
LEVEL= 1   ERROR= 41.532174974
LEVEL= 1   ERROR= 7.50157810532
LEVEL= 1   ERROR= 2.86243112818
LEVEL= 3   ERROR= 2.89527338022
LEVEL= 3   ERROR= .145588853173
LEVEL= 3   ERROR= 1.75340862750E-03
LEVEL= 2   ERROR= 2.44258558694
LEVEL= 2   ERROR= .202845696484
LEVEL= 2   ERROR= 2.21171894269E-02
LEVEL= 1   ERROR= .861920134544
LEVEL= 1   ERROR= .21532575187
LEVEL= 1   ERROR= 7.70327718248E-02
```

UNIGRID SAMPLE RUN ON 13 x 9 GRID

```
# GRIDS= 4  #X POINTS= 25  #Y POINTS= 17  H= .125
# RELAXATIONS= 3  CONVERGENCE TOL= 0  MAX # CYCLES 2


LEVEL= 4   ERROR= 601.84198548
LEVEL= 4   ERROR= 45.7094091806
LEVEL= 4   ERROR= .684507163162
LEVEL= 3   ERROR= 594.691917146
LEVEL= 3   ERROR= 56.586100575
LEVEL= 3   ERROR= 5.86425063696
LEVEL= 2   ERROR= 139.531317262
LEVEL= 2   ERROR= 17.0539436643
LEVEL= 2   ERROR= 5.53009378558
LEVEL= 1   ERROR= 35.7542783386
LEVEL= 1   ERROR= 7.20252546707
LEVEL= 1   ERROR= 3.3511930668
LEVEL= 4   ERROR= 9.83004014544
LEVEL= 4   ERROR= .489235064975
LEVEL= 4   ERROR= 7.32638861133E-03
LEVEL= 3   ERROR= 7.43719700515
LEVEL= 3   ERROR= .44716401473
LEVEL= 3   ERROR= 5.45572374272E-02
LEVEL= 2   ERROR= 3.7693134515
LEVEL= 2   ERROR= .515138079613
LEVEL= 2   ERROR= .147593559542
LEVEL= 1   ERROR= .787713994932
LEVEL= 1   ERROR= .173224759878
LEVEL= 1   ERROR= 8.31153127376E-02
```

UNIGRID SAMPLE RUN ON 25 x 17 GRID

ACKNOWLEDGEMENTS. Both parts of this paper describe projects that were joint efforts with other researchers. For the microprocessor software, recognition is due M. Andrews, G. Taylor, D. Pryor, B. Eisenberg, and D. Jaeger. The unigrid developments were in collaboration with J. Ruge.

REFERENCES

1. S.McCormick, D.Pryor and G.Taylor. "Numerical software for fixed point microcomputer applications", U.S.A.R.O. report, November, 1980.

2. M.Andrews, S.F.McCormick and G.D.Taylor. "Evaluation of functions on microcomputers: square root", Comput. Math. Appl. 4 (1979), pp. 359-367.

3. M.Andrews, B.Eisenberg, S.F.McCormick and G.D.Taylor. "Evaluation of functions on microcomputers: rational approximation of k^{th} roots", Comput. Math. Appl. 5 (1979), pp. 163-167.

4. M.Andrews, D.Jaeger, S.F.McCormick and G.D.Taylor. "Evaluation of functions on microcomputers: exp(x)", Comput. Math. Appl., to appear.

5. M.Andrews, D.Jaeger, S.F.McCormick and G.D.Taylor. "Evaluation of functions on microcomputers: ln(x)", Comput. Math. Appl., to appear.

6. M.Andrews, D.Jaeger, S.F.McCormick and G.D.Taylor. "The microcomputer numerical software project at CSU", Proc. Second Rocky Mtn. Symp. on Microcomputers: Systems, Software, Architecture, August, 1978, pp. 254-263.

7. S.McCormick. "An algebraic interpretation of multigrid methods", submitted.

8. S.McCormick and J.Ruge. "A general theory for variational multigrid", in preparation.

9. S.McCormick and J.Ruge. "Unigrid: A multigrid software simulator", in preparation.

SAFE LIFE DESIGN OF GUN TUBES - SOME NUMERICAL METHODS AND RESULTS

Anthony P. Parker[1]

Kim A. Sleeper[2]

Christopher P. Andrasic[1]

ABSTRACT. After firing a limited number of rounds, a gun tube may develop multiple radial cracks emanating from its boundaries. These cracks grow under the cyclic pressurization due to firing until they reach a critical length, at which stage catastrophic brittle failure may occur. The fundamental safety requirement is that a tube be withdrawn from service before such failure.

In order to reduce the rate of crack growth, it is common practice to induce compressive, residual stresses at the bore of a gun tube by an autofrettage process which involves suitable pressurization or swaging during manufacture.

In this paper, we describe the numerical solution of a range of problems encountered in the safe-life design of a gun tube, namely:

a. The prediction of residual stress fields arising from full or partial autofrettage.

b. The correction of these stress fields to account for the non-ideal, Bauschinger effect on unloading of the tube during manufacture.

c. Prediction of crack tip stress intensity factors for multiple cracks in pressurized, autofrettaged barrels using the modified mapping collocation method.

d. Calculation of gun tube lifetime using stress intensity factor data and a fatigue crack growth law.

Finally, some outstanding problem areas are noted, and possible numerical techniques are proposed for their solution.

1. INTRODUCTION. Fatigue crack growth arising from the cyclic pressurization of thick cylinders can produce a regular array of up to 50 equal-length radial cracks emanating from the bore [1]. A knowledge of the crack tip stress intensity factor, K is necessary in order to predict the fatigue growth rate and critical length of such cracks. Several solutions are available for the case of a cracked, pressurized thick cylinder [1-6]. It is likely that the most accurate of these solutions are those derived by use of the Modified Mapping Collocation (MMC) method. These include the solution in reference [5] for up to four internal or external radial cracks, and that in reference [6] for up to 40 internal radial cracks. The errors associated with the MMC technique are generally estimated as being less than 1%.

1. Materials Branch, Royal Military College of Science, Shrivenham, SN6 8LA, UK

2. Army Materials & Mechanics Research Center, Watertown, MA, 02172

To inhibit fatigue growth of internal cracks it is common practice to produce a more advantageous stress distribution involving residual compressive hoop stresses near the bore, by autofrettage treatment of the cylinder prior to use [7]. K solutions exist for a multiply-cracked fully autofrettaged (100% overstrain*) tube [6], [8]. Reference [6] is an MMC solution. However, the optimum autofrettage condition may not be 100% overstrain [7] since fatigue cracks may develop at the outside radius as a result of the relatively high tensile residual stress. Clearly, the choice of the optimum overstrain condition will involve a consideration of the rates at which external cracks will grow radially inwards, and the rates at which internal cracks will grow outwards. In each case, prediction of crack growth rate, critical crack length and residual strength will depend on a knowledge of the crack-tip stress intensity factor. The designer requires accurate stress intensity factors for both internally cracked and externally cracked tubes with internal pressure, and any amount of autofrettage from zero to 100% overstrain (full autofrettage).

Life prediction is normally based on the stress intensity factor calibration and an associated empirical crack growth law [9]. However, there is evidence to suggest that life predictions based on the K values obtained from 'ideal' autofrettage distributions may significantly overestimate the life of a given tube [10]. One possible explanation for this is the Bauschinger effect [11], which is evident when certain materials are loaded in compression after initial tensile loading, this causes a reduction in the 'ideal' residual stress field following autofrettage.

Each of the above aspects is considered, with particular emphasis on the numerical solution of a number of problems encountered in gun tube life prediction.

2. THE BAUSCHINGER EFFECT. In determining the residual stress field in a thick cylinder which has undergone plastic deformation it is normal to assume an elastic/perfectly plastic stress-strain curve of the form illustrated in Fig. 1(a). This behaviour implies the same magnitude of yield stress, Y in tension and compression. However, the stress-strain curve for certain gun steels may be of the form illustrated schematically in Fig. 1(b). The significant features of this 'real' behaviour are:

a. A small amount of plastic strain-hardening (slope β) typically a strain hardening exponent of 0.05. This may alter the residual stress field by 4% [12].

b. A very small modification to the residual stress field of approximately 1% as a result of compressibility of the material [13].

c. There is a significant Bauschinger effect [11], i.e. the yield stress in compression is less than that in tension.

* Overstrain is the proportion of the cylinder wall thickness that is subjected to plastic strain during the autofrettage process.

d. The shape of the unloading portion of the curves (CD and C'D') is unchanged with differing amounts of plastic flow within the plastic strain limits employed in the production of gun tubes.

For the purposes of this paper, the 'typical' behaviour illustrated in Fig. 1(b) is modelled as a series of straight lines, with zero strain hardening, and a yield strength in compression of $(-\alpha Y)$, Fig. 1(c).

3. THICK CYLINDER THEORY. Consider a tube, internal radius a external radius b, which is subjected to an internal pressure p, Fig. 2. The distribution of hoop (σ_θ) and radial (σ_r) stress in this case is given by Lame's equations as:

$$\sigma_\theta = \frac{a^2 p}{b^2-a^2}\left[1 + \frac{b^2}{r^2}\right] \qquad (1)$$

$$\sigma_r = \frac{a^2 p}{b^2-a^2}\left[1 - \frac{b^2}{r^2}\right]$$

where r is the radius at which the stress is defined.

Assuming elastic-perfectly plastic material properties, and plane strain conditions, employing Tresca's yield criterion, but omitting the analysis, the pressure p^* to cause yielding of the tube out to a radius r=c (Fig. 3) is given by [11]:

$$p^* = Y \ln (c/a) + \frac{Y}{2b^2}(b^2-c^2) \qquad (2)$$

where Y is the uniaxial yield stress for the material. This will give directly the pressure for initial yielding at the bore, p_i^*:

$$p_i^* = \frac{Y}{2b^2}(b^2-a^2) \qquad (3)$$

and the pressure for complete yielding of the tube, p_y^*:

$$p_y^* = Y \ln (b/a) \qquad (4)$$

If the cylinder is subjected to a pressure p^*, $[p_i^* < p^* < p_y^*]$, there will be partial yielding of the tube out to a radius c, Fig. 3. The hoop stresses produced by this pressurization are:

$$\left.\begin{aligned} \sigma_\theta^* &= -p^* + Y\,(1 + \ln (r/a)) \\ \sigma_r^* &= -p^* + Y \ln (r/a) \end{aligned}\right\} \quad a \leq r \leq c \qquad (5)$$

$$\left.\begin{aligned} \sigma_\theta^* &= \frac{Yc^2}{2b^2}\left[1 + \frac{b^2}{r^2}\right] \\ \sigma_r^* &= \frac{Yc^2}{2b^2}\left[1 - \frac{b^2}{r^2}\right] \end{aligned}\right\} \quad c \leq r \leq b \qquad (6)$$

If the pressure p* is subsequently removed completely, assuming that the unloading is entirely linearly elastic, with no reversed yielding (valid provided b/a < 2.22), the residual hoop stress distribution, σ_θ^R, is given by [11]:

$$\left.\begin{aligned} \sigma_\theta^R &= -p^* + Y\,(1 + \ell n\,(r/a)) - \frac{p^*a^2}{(b^2-a^2)}\left[1 + \frac{b^2}{r^2}\right] \\ \sigma_r^R &= -p^* + Y\ n\,(r/a) - \frac{p^*a^2}{(b^2-a^2)}\left[1 - \frac{b^2}{r^2}\right] \end{aligned}\right\} \quad a \leq r \leq c \qquad (7)$$

$$\left.\begin{aligned} \sigma_\theta^R &= \left[\frac{Yc^2}{2b^2} - \frac{p^*a^2}{b^2-a^2}\right]\left[1 + \frac{b^2}{r^2}\right] \\ \sigma_r^R &= \left[\frac{Yc^2}{2b^2} - \frac{p^*a^2}{b^2-a^2}\right]\left[1 - \frac{b^2}{r^2}\right] \end{aligned}\right\} \quad c \leq r \leq b \qquad (8)$$

Clearly, a re-pressurization of the tube to a pressure $p<p^*$ will produce a stress distribution which may be calculated by the addition of (7) and (1) for $r \leq c$, and (8) and (1) for $r > c$.

Assuming a reduced compressive yield strength of $(-\alpha Y)$ as a result of the Bauschinger effect, there is now the possibility of reversed yielding outwards from the bore to a radius d, Fig. 4. In the region of reversed plasticity the stresses are:

$$\left.\begin{aligned} \sigma_\theta &= -\alpha Y\,(1 + \ell n\,(r/a)) \\ \sigma_r &= -\alpha Y\,\ell n\,(r/a) \end{aligned}\right\} \quad a \leq r \leq d \qquad (9)$$

which satifies the two requirements that $\sigma_r = 0$ @ r=a and Tresca's criterion, namely $\sigma_\theta - \sigma_r = -\alpha Y$, $a \leq r \leq d$.

Consider now the elastic region $r \geq d$. As a result of unloading and yielding the elastic-plastic interface at r=d experiences an additional radial stress σ_r (=-p), given by equation (9) minus equation (5), thus:

$$-p = \sigma_r\Big|_{r=d} = p^* - (1 + \alpha)\ Y\ \ell n\,(d/a). \qquad (10)$$

Thus the stresses in the elastic region are composed of (5) and (6) plus some additional pressure p applied at r=d as a result of unloading and reversed plasticity.

If there is reversed yielding on unloading out to a radius d (d<c), material at any point r>d will see the combination loading and yielding as the application of an additional pressure p at radius d, such that

$$\left.\begin{aligned} \sigma_\theta &= p\,\frac{d^2}{b^2-d^2}\left[1+\frac{b^2}{r^2}\right] \\ \sigma_r &= p\,\frac{d^2}{b^2-d^2}\left[1-\frac{b^2}{r^2}\right] \end{aligned}\right\} \quad d \le r \le b \qquad (11)$$

The requirement for the outer region $d \le r \le c$ is that at r=d it is just yielding. The total stresses σ_θ^T and σ_r^T given by the superposition of (5) and (11) are:

$$\left.\begin{aligned} \sigma_\theta^T &= p\,\frac{d^2}{b^2-d^2}\left[1+\frac{b^2}{r^2}\right] - p^* + Y\,(1+\ell n\,(r/a)) \\ \sigma_r^T &= p\,\frac{d^2}{b^2-d^2}\left[1-\frac{b^2}{r^2}\right] - p^* + Y\,\ell n\,(r/a) \end{aligned}\right\} \quad d \le r \le c \qquad (12)$$

But we know that Tresca's criterion applies, thus:

$$(\sigma_\theta^T - \sigma_r^T) = -\alpha Y \quad @\ r=d \qquad (13)$$

and from (12) and (13)

$$-\alpha Y = Y + p\,\frac{d^2}{b^2-d^2}\,\frac{2b^2}{d^2} \qquad (14)$$

But the interface pressure is given by equation (10). Thus, combining (10) and (14):

$$\frac{p^*}{Y} = (1+\alpha)\left[\left(\frac{b^2-d^2}{2b^2}\right) + \ell n\,(d/a)\right] \qquad (15)$$

Substituting from (10) into (12), recognizing that $p = -\sigma_r\big|_{r=d}$:

$$\left.\begin{aligned}\sigma_\theta^T &= -\left\{p^* - (1+\alpha)\,Y\,\ell n\,(d/a)\right\}\frac{d^2}{b^2-d^2}\left[1+\frac{b^2}{r^2}\right] - p^* + Y\,(1+\ell n\,(r/a))\\ \sigma_r^T &= -\left\{p^* - (1+\alpha)\,Y\,\ell n\,(d/a)\right\}\frac{d^2}{b^2-d^2}\left[1-\frac{b^2}{r^2}\right] - p^* + Y\,\ell n\,(r/a)\end{aligned}\right\} \quad d \le r \le c \qquad (16)$$

Superimposing (6) and (11) and substituting from (10):

$$\left.\begin{aligned}\sigma_\theta^T &= \left[1+\frac{b^2}{r^2}\right]\left[\frac{Yc^2}{2b^2} - \left\{p^* - (1+\alpha)\,Y\,\ell n\,(d/a)\right\}\frac{d^2}{b^2-d^2}\right]\\ \sigma_r^T &= \left[1-\frac{b^2}{r^2}\right]\left[\frac{Yc^2}{2b^2} - \left\{p^* - (1+\alpha)\,Y\,\ell n\,(d/a)\right\}\frac{d^2}{b^2-d^2}\right]\end{aligned}\right\} \quad c \le r \le b \qquad (17)$$

Equations (7) and (8) together with (2) define the residual stress field after removal of autofrettage pressure when there is no reversed yielding, whilst equations (9), (16), (17) together with (2) and (15) define the residual stress field in instances where reversed yielding occurs.

For yielding not to occur on unloading:

$$(\sigma_\theta - \sigma_r)\Big|_{r=a} < -\alpha Y \qquad (18)$$

i.e. from equation (7):

$$p^* < \frac{(1+\alpha)Y}{2}\left(\frac{b^2-a^2}{b^2}\right) \qquad (19)$$

or in terms of the pressure for initial yielding p_i^*, equation (3), for no reversed yielding:

$$p^* < (1+\alpha)\,p_i^* \qquad (20)$$

For example, consider a cylinder with b/a=2, α=0.5, then from (20), for no reversed yielding:

$$p^* < 1.5\,p_i^*$$

Now since:

$$p_i^* = \frac{Y}{2b^2}\,(b^2-a^2) = .375\,Y$$

we obtain from (2):

$$1.5\ p_i^* = .5625Y = Y\ \ell n\ (\frac{c}{a}) + \frac{Y}{2b^2}\ (b^2-c^2)$$

A straightforward iterative process gives c/a=1.33, thus any overstrain in excess of 33% will cause reversed yielding at the bore.

Clearly, it will also be necessary to iterate on equation (15) in order to calculate d. Residual stress distributions for cylinder ratios (b/a) of 2.0 and 3.0 are shown in Fig. 5, for $0.25 \leq \alpha \leq 1.0$.

4. PREDICTION OF CRACK TIP STRESS INTENSITY FACTORS BY MODIFIED MAPPING COLLOCATION (MMC). Complex variable methods, due to Muskhelishvili [14] are utilized. Stresses and displacements within a body are represented in terms of complex stress functions. By employing an MMC technique as described in [5, 15] the cracked ring segment in the physical (z) plane, Fig. 6, is mapped from a retangular region in the γ (parameter) plane. Traction-free conditions along A'B' and D'E' in the parameter plane are ensured. The singularity is removed from the parameter plane by mapping a unit semi-circle onto the appropriate crack surfaces, Fig. 6. A series representation of the stress function is selected, which ensures appropriate symmetry conditions. The stress and displacement boundary conditions applicable to the problem in the physical (z) plane are:

$\sigma_r = o,\ \tau_{r\theta} = o$ over DE and BA

$u_\theta = o,\ \tau_{r\theta} = o$ over EF and AH

$\sigma_r = o,\ \tau_{r\theta} = o$ over FG and HG

$\sigma_\theta = p(r),\ \tau_{r\theta} = o$ over DC and BC

where p(r) is equal and opposite to the loading along the crack line in the unflawed structure for the case of internal pressure, autofrettage or thermal loading, the latter two stress states being essentially equivalent [16].

In the MMC method the infinite series representations of the complex stress functions are truncated to a finite number of terms. Force conditions are imposed at selected boundary points along CD, EF, and FG, which gives conditions on the unknown coefficients in the stress functions. Thus each boundary point produces two rows in the main matrix A, and two corresponding elements in the boundary conditions vector $\underset{\sim}{b}$, where:

$$A\ \underset{\sim}{x} = \underset{\sim}{b}$$

and $\underset{\sim}{x}$ is the vector of unknown coefficients. In general A is a matrix of ℓ rows and m columns, where ℓ and m depend upon the number of boundary points and unknown coefficients respectively. It was found that convergence is generally better when $2m < \ell < 2.5m$, this conforms with other workers [17]. A least-squares error minimization procedure was used to solve the overdetermined set of linear equations.

Knowing the coefficients for the stress function in the cracked region, the crack shape and stress intensity factor, K, may be determined in a straightforward manner [14]. In all cases considered there is symmetry of loading and geometry about the crack line, the only non-zero stress intensity factor being K_I.

Results for internal pressure in the bore and cracks (each of length ℓ) are presented in Fig. 7 for b/a ratio of 2.0. Equivalent results for the case of full (100%) ideal autofrettage and steady-state thermal stressing appear in Fig. 8. The form of the results at short crack lengths is shown in Fig. 9 indicating good convergence to the limiting value. Again, the short crack length convergence is good, as is that at longer crack lengths. For the particular case of 50 cracks, and b/a varying from 1.2 to 2.0, results for full autofrettage are shown in Fig. 10. By superposition of these results it is possible to determine K for any combination of internal pressure, full autofrettage or steady-state thermal loading. Furthermore, provided the crack tips do not extend beyond the minimum radius to which plastic flow was induced during the autofrettage process, it is also possible to obtain K values for partial autofrettage by a straightforward superposition [18].

A set of results for internal cracks with internal pressure and 50% overstrain based on the results of reference [6], and the superposition described in [18], is presented graphically in Fig. 11. A further set of results for external cracks with internal pressure and 50% overstrain, based on the results of reference [5] is presented graphically in Fig. 12.

5. CALCULATION OF TUBE LIFETIME. The prediction of life using Linear Elastic Fracture Mechanics and a crack growth law is well known [9]. It consists of defining the stress intensity range ΔK as:

$$\Delta K = K_{max} - K_{min} \qquad K_{min} > 0 \qquad (21)$$

$$\Delta K = K_{max} \qquad K_{min} \leq 0 \qquad (22)$$

where K_{max} and K_{min} are the effective maximum and minimum stress intensity values respectively during a given loading cycle. Equation (22) implies that the part of the fatigue cycle during which the crack is closed at its tip (i.e. $K \leq 0$) makes no contribution to crack growth. For much of a component's lifetime, the fatigue crack growth rate is related to the stress intensity factor range by [9]:

$$\frac{d\ell}{dN} = C(\Delta K)^M \qquad (23)$$

where N represents the number of cycles, and C and M are experimentally determined constants. In general C and M are also functions of the R ratio, where:

$$R = K_{min}/K_{max} \qquad K_{min} > 0 \tag{24}$$

$$R = 0 \qquad K_{min} \leq 0 \tag{25}$$

However, in this paper we ignore the (relatively) limited effects of changing R ratio, and emphasize the effects of the residual stress field contributions to K. The question of changing R ratio during fatigue crack growth through a residual stress field is considered in detail in [19]. Note that there does not appear to be any reason to assume that the superposition principle is violated by 'stress fading' during fatigue crack growth through residual stress fields at stress levels which only produce localized (crack tip) yielding [19].

Consider a tube containing a residual stress field. When a crack is introduced it has a residual stress intensity K_I^R. The tube is then subjected to a cyclic pressure loading. The stress intensity contributions produced by this loading are $K_{I_{max}}^L$ and $K_{I_{min}}^L$, the maximum and minimum values of stress intensity produced by the pressure loading.

In general we note that equations (21) and (22) give:

$$\left.\begin{aligned} \Delta K &= K_{I_{max}}^L - K_{I_{min}}^L \\ R &= \frac{K_{I_{min}}^L + K_I^R}{K_{I_{max}}^L + K_I^R} \end{aligned}\right\} \qquad K_{I_{min}}^L + K_I^R > 0 \tag{26}$$

$$\left.\begin{aligned} \Delta K &= K_{I_{max}}^L + K_I^R \\ R &= 0 \end{aligned}\right\} \qquad K_{I_{min}}^L + K_I^R \leq 0 \tag{27}$$

In order to predict lifetime to failure for a gun tube it is necessary to rearrange equation (23) to give

$$\text{Number of cycles, } N = \int_{\ell_o}^{\ell_c} \frac{d\ell}{C(\Delta K)^M} \tag{28}$$

where ℓ_o is some appropriate initial crack length, and ℓ_c is the critical crack length at which catastrophic brittle failure will occur.

In general, the integral cannot be evaluated exactly and it is necessary to integrate in a step-wise fashion in order to determine total lifetime [20]. For instance, assuming a tube with internal radius 50mm and external radius 100mm containing an array of 40 radial cracks, each of length 5mm, we may calculate the lifetime of the tube at working pressures of 400, 450 and 500 MNm^{-2}, for varying amounts of autofrettage from 0 to 100%. [The material properties assumed were: Yield strength 1200 MNm^{-2} Fracture Toughness 90 $MNm^{-3/2}$, Empirical crack growth constants, M=3.1, $C=1.455 \times 10^{-11}$ for crack growth in metres/cycle]

The results for this particular case are illustrated as continuous lines in Fig. 13, and would lead to the initial conclusion that the largest possible amount of autofrettage is required. However, this may not be the case. The dotted lines on Fig. 13 represent the lifetime for an external crack of initial length 0.05mm which grows radially inward under the same internal cyclic pressure. Thus, for pressures of 500MNm^{-2} there would be no advantage in exceeding 27% overstrain, since tube lifetime is then limited by growth of the external crack. Indeed, any increase in overstrain would tend to increase the growth rate of the external crack by causing an increase in R value. Whilst the relative positions of the lifetime curves in Fig. 13 will vary with material, initial crack lengths, working pressures and the nature of the residual stress fields, the general approach to the selection of an optimum autofrettage overstrain will be the same.

6. CONCLUSIONS AND FUTURE WORK

a) The residual stress field in an autofrettaged gun tube may be calculated exactly by assuming a simple Bauschinger effect model which accounts for a lower magnitude of the yield strength in compression than that in tension. Future work should address the problem of modelling the non-linear unloading effects which accompany this reduced yield strength.

b) The modified mapping collocation (MMC) method produces accurate, two-dimensional stress intensity factor solutions for the case of a cracked, internally pressurized tube with autofrettage. Of particular note is the accuracy of the selected MMC technique at short crack lengths, wherein lifetime estimates are most critical. A straightforward superposition allows these results to be extended to the case of partial autofrettage. Future work should include the proper representation of three-dimensional cracked configurations (e.g. thumbnail cracks).

c) Gun tubes may develop an array of multiple cracks. Future work should be aimed at understanding the factors influencing the stability of such patterns, and the effects of residual stresses on such stability.

d) Crack growth rates due to the cyclic pressurization of gun tubes may be predicted from a knowledge of the crack tip stress intensity factor range. The optimum autofrettage condition may not be 100% overstrain, since external cracks may grow inwards and produce failure. Life prediction

design curves may be generated which permit a selection of the optimum autofrettage condition. Whilst crack initiation time for internal cracks is effectively zero, there is a definite initiation period for external cracks. This initiation time should be quantified to allow accurate lifetime design predictions.

7. ACKNOWLEDGMENTS. The first author performed work on this paper during a TTCP attachment to the Engineering Mechanics Laboratory, Army Materials and Mechanics Research Center, Watertown, Mass., and wishes to acknowledge stimulating discussions with Mr. J. H. Underwood of Benet Weapons Laboratory (who pointed out the possible importance of the Bauschinger effect on the autofrettage distribution in gun tubes), and Dr. D. M. Tracey of the Army Materials and Mechanics Research Center. Finally, the third author acknowledges support from a UK Ministry of Defence research contract as a Research Scientist at the Royal Military College of Science.

8. REFERENCES

1. Goldthorpe, B. D. "Fatigue and Fracture of Thick Walled Cylinders and Gun Barrels", "Case Studies in Fracture Mechanics" AMMRC MS 77-5 (1977).

2. Pu, S. L. and Hussain, M. A. "Stress Intensity Factors for a Circular Ring with Uniform Array of Radial Cracks Using Cubic Isoparametric Singular Elements", "Trans. 24th Conference of Army Mathematicians", ARO Report 79-1 (1979).

3. Tweed, J. and Rooke, D. P. "The Stress Intensity Factor for a Crack in a Symmetric Array Originating from a Circular Hole in an Infinite Solid", J. Engng. Sci., 13, 653-662 (1975).

4. Baratta, F. I., "Stress Intensity Factors for Internal Multiple Cracks in Thick-Walled Cylinders Stressed by Internal Pressure Using Load Relief Factors", Engng. Frac. Mech., 10, 691-697 (1978).

5. Tracy, P. G., "Elastic Analysis of Radial Cracks Emanating from the Outer and Inner Surfaces of a Circular Ring", Engng. Frac. Mech., 11, 291-300 (1979).

6. Parker, A. P. and Andrasic, C. P., "Stress Intensity Prediction for a Multiply-Cracked, Pressurised Gun Tube with Residual and Thermal Stresses", U.S. Army Symposium on Solid Mechanics, AMMRC MS 80-5, 35-39, (1980).

7. Kapp, J. A. and Eisenstadt, R., "Crack Growth in Externally Flawed, Autofrettaged Thick-Walled Cylinders and Rings", "Fracture Mechanics", ASTM STP 677, C. W. Smith (ed), 746-756 (1979).

8. Parker, A. P. and Farrow, J. R., "Stress Intensity Factors for Multiple Radial Cracks Emanating from the Bore of an Autofrettaged or Thermally Stressed Thick Cylinder", Engng. Frac. Mech., 14, 237-241 (1981).

9. Paris, P. C. and Erdogan, F., "A Critical Analysis of Crack Propogation Laws", Trans. ASME, J. Bas. Engng., 85, 528-534 (1963).

10. Underwood, J. H. and Throop, J. F., "Residual Stress Effects on Fatigue Cracking of Pressurized Cylinders and Notched Bending Specimens", Presented at SESA Spring Meeting, Boston, Mass., May (1980).

11. Hill, R. "The Mathematical Theory of Plasticity", Oxford University Press (1967).

12. Chen, P. C. T., "Generalized Plane-Strain Problems in an Elastic-Plastic Thick-Walled Cylinder", Technical Report ARLCB-TR-80028, Benet Weapons Lab, Watervliet Arsenal, NY (1980).

13. Chen, P. C. T., "Numerical Prediction of Residual Stresses in an Autofrettaged Tube of Compressible Material", US Army Numerical Analysis & Computers Conference, Huntsville, (1981), in press.

14. Muskhelishvili, N. I., "Some Basic Problems of the Mathematical Theory of Elasticity", Noordhoff Ltd. (1953).

15. Andrasic, C. P. and Parker, A. P., "Weight Functions for Cracked, Curved Beams", 2nd Int. Conf. on Numerical Methods in Fracture Mechanics, Swansea, 67-82 (1980).

16. Parker, A. P., Farrow, J. R., "On the Equivalence of Axisymmetric Bending Thermal and Autofrettage Stress Fields", J. Strain Analysis, 15, 1, 51-52 (1980).

17. Eason, E. D., "A Review of Least Squares Methods for Solving Partial Differential Equations", Int. J. Num. Meth. Engng., 10, 1021-1046 (1976).

18. Parker, A. P., "Stress Intensity and Fatigue Crack Growth in Multiply-Cracked, Pressurized, Partially Autofrettaged Thick Cylinders", Fatigue of Engng. Materials and Structures (1981), in press.

19. Parker, A. P., "Stress Intensity Factors, Crack Profiles and Fatigue Crack Growth Rates in Residual Stress Fields", To be presented at ASTM Meeting on the Effect of Residual Stress on Fatigue Performance, Phoenix, AZ, 11-14 May (1981).

20. "Examples of the Use of Data Items on Fatigue Crack Propogation Rates", Engineering Sciences Data Unit, Item No. 74017 (1977).

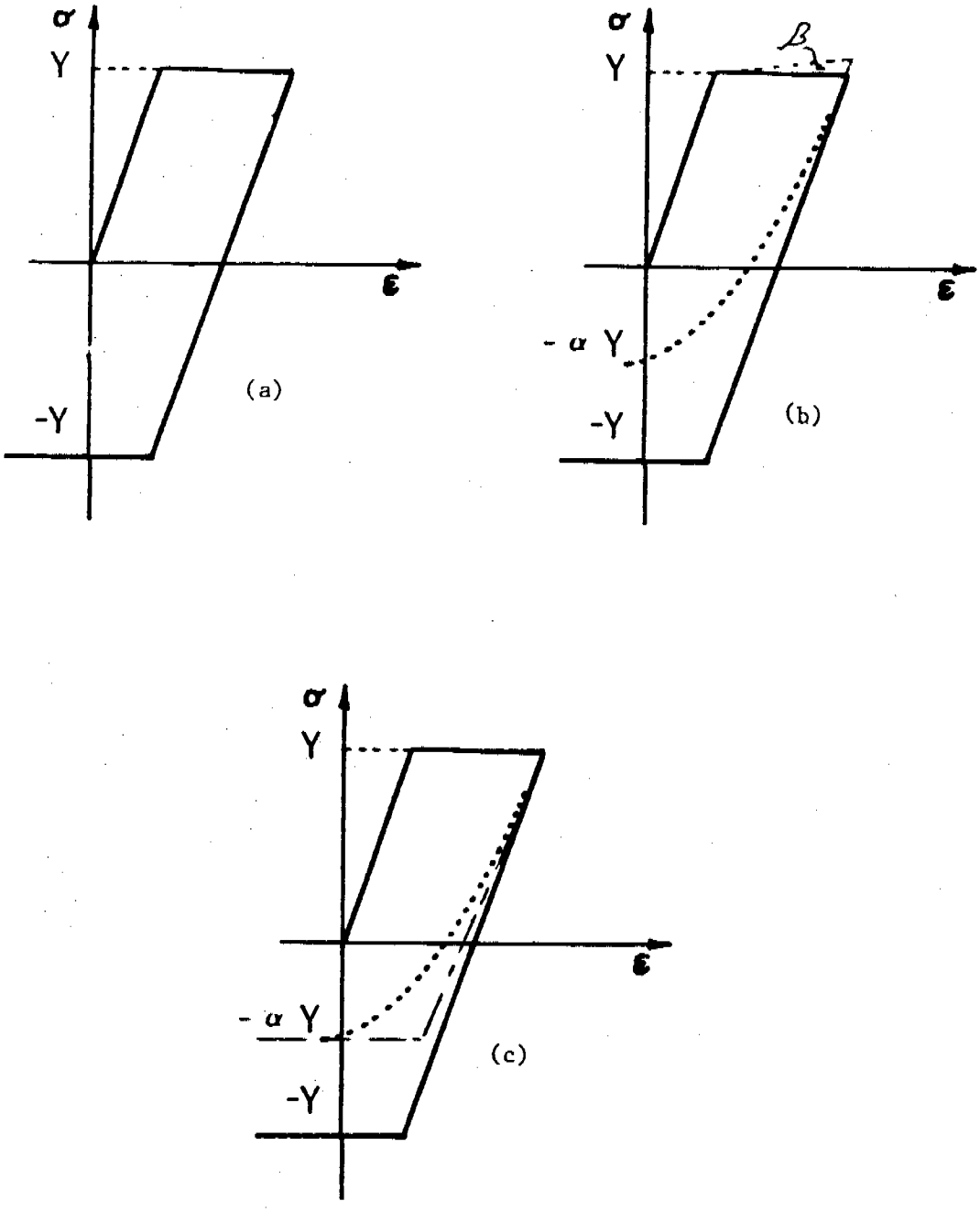


Figure 1 : Elastic-Plastic Stress-Strain Curves.

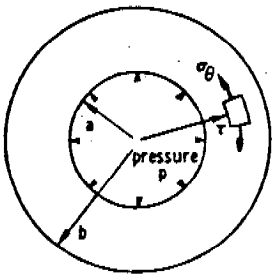


Figure 2 : Internally pressurized Thick Cylinder.

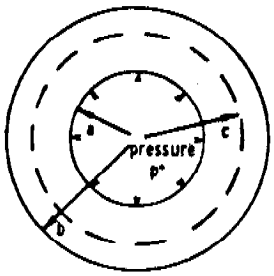


Figure 3 : Internally Pressurized, Partially Plastic Thick Cylinder.

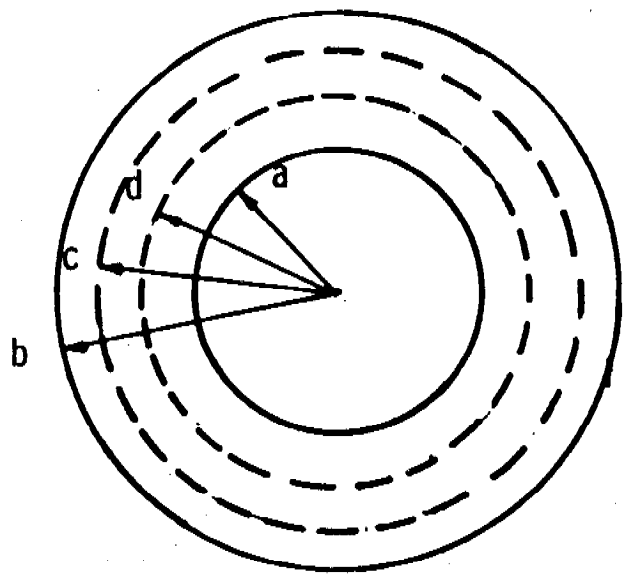
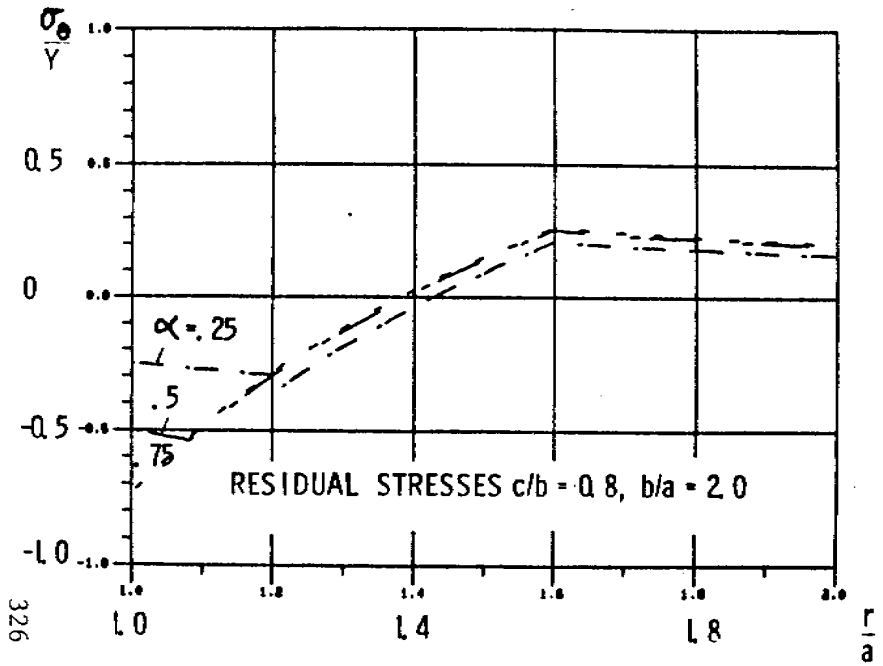
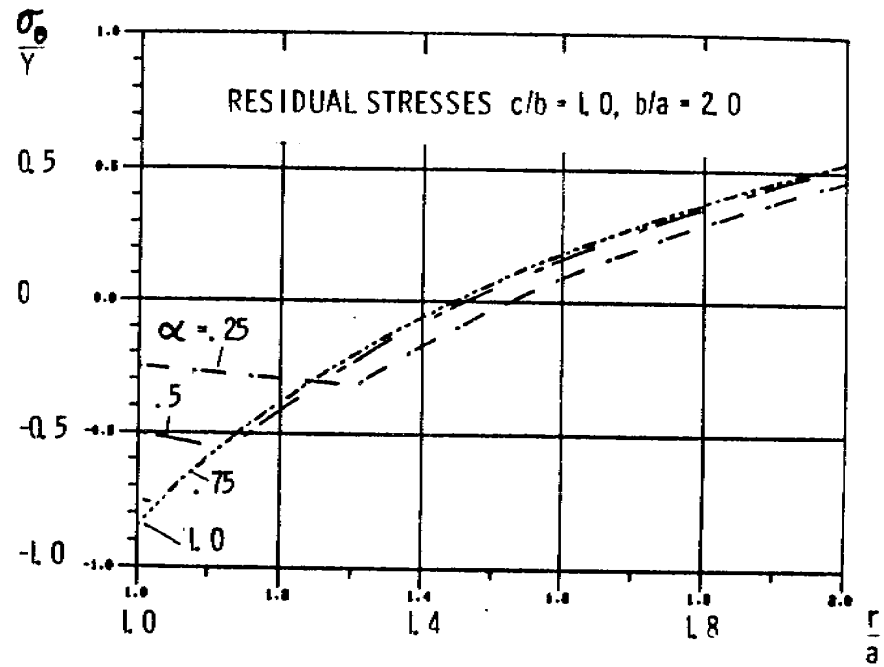
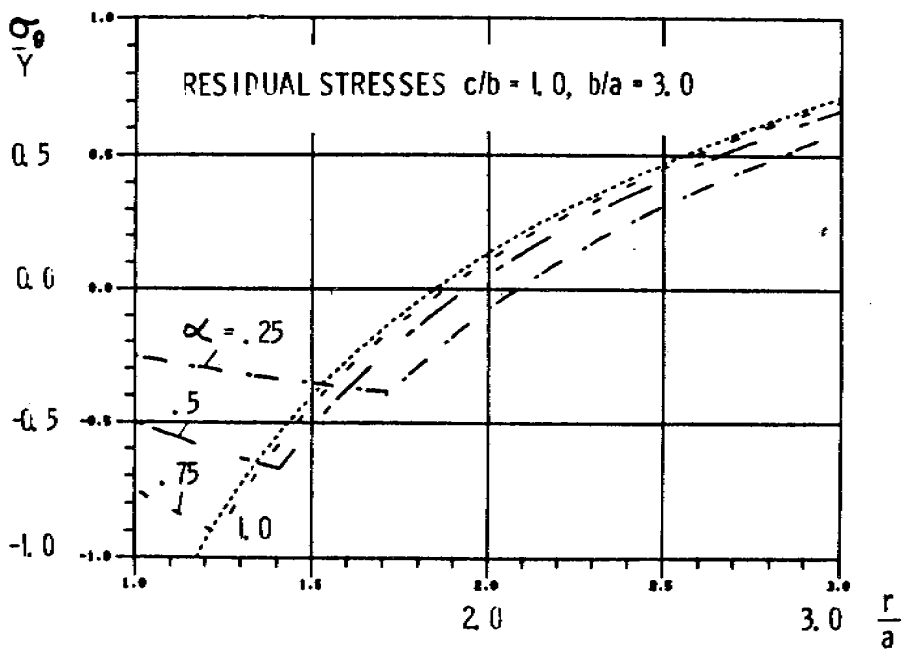


Figure 4 : Unpressurized, Autofrettaged Thick Cylinder With Reversed Yielding.







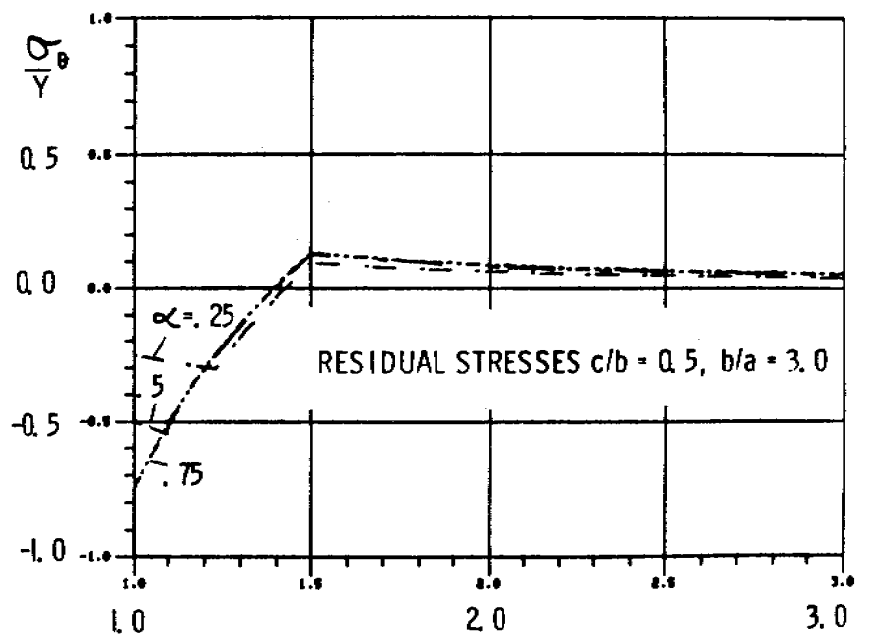


Figure 5 : Residual Stress Fields in Various Autofrettaged Tubes With Reversed Yielding.

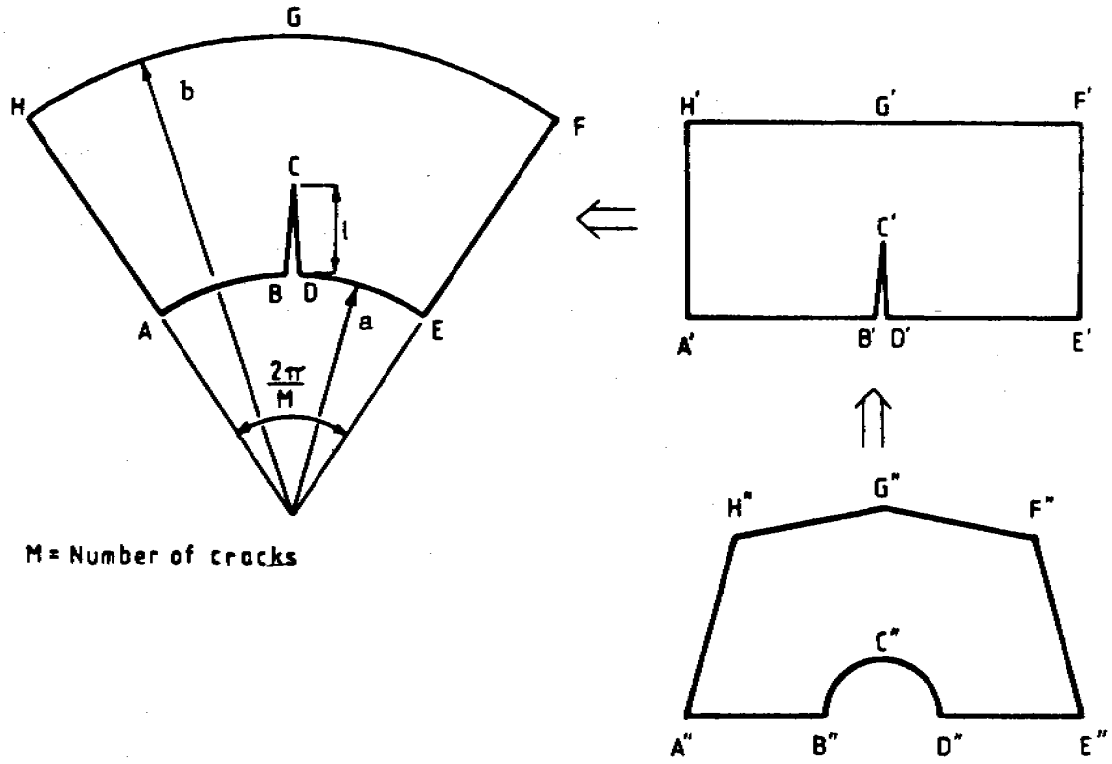


Figure 6 : Physical and Mapped Planes.

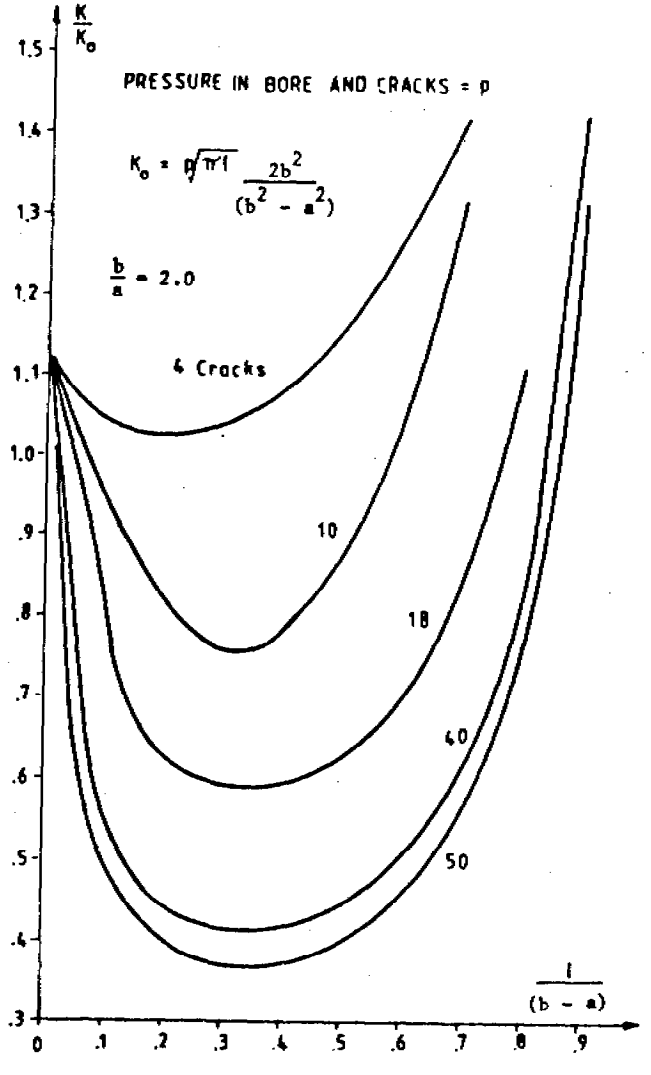


Figure 7: Stress Intensity Factors for Pressurised Thick Cylinder.

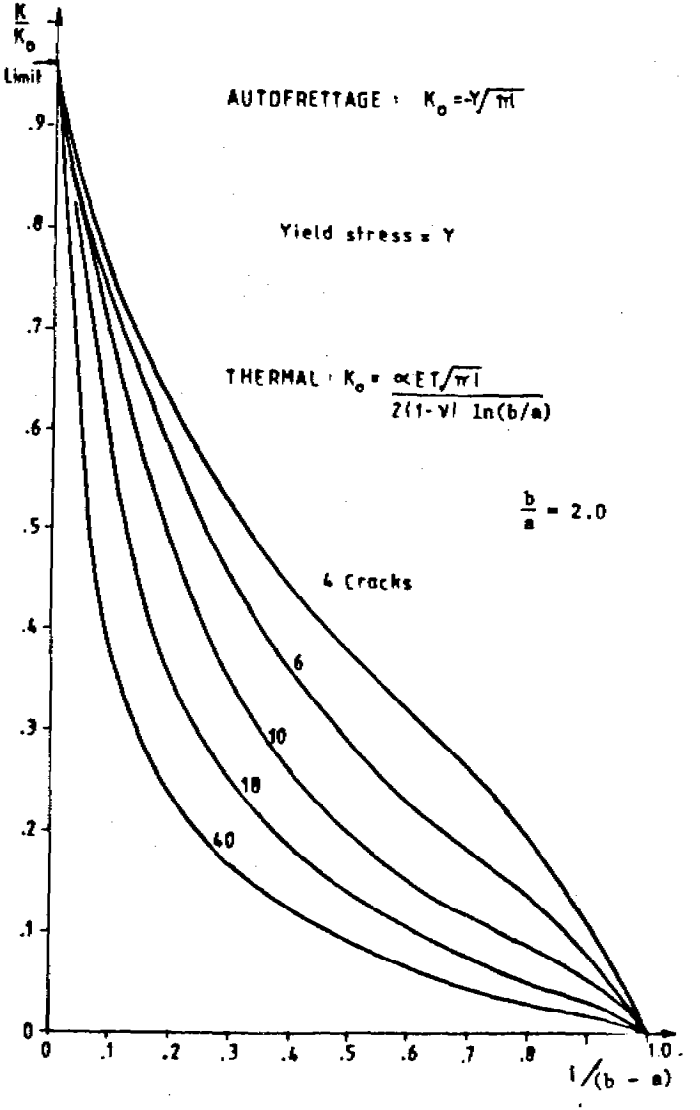
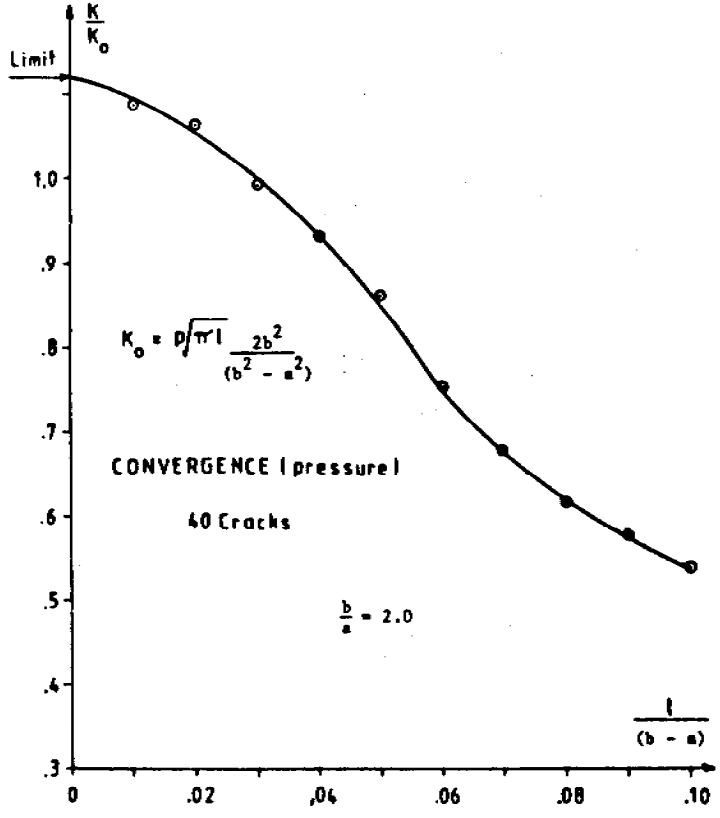
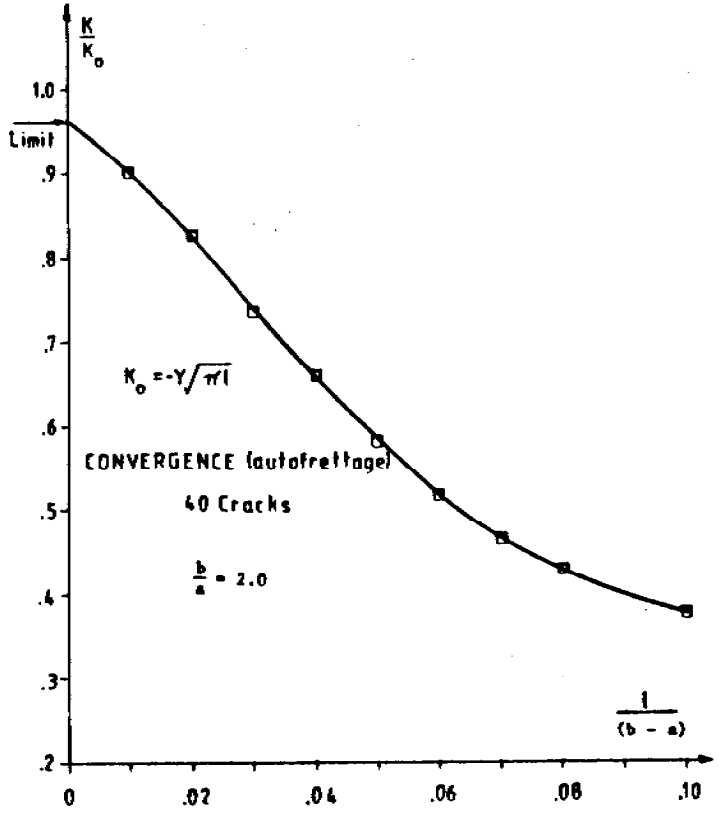


Figure 8: Stress Intensity Factors for Autofrettaged or Thermally Stressed Thick Cylinder.



Short Crack Length Convergence Characteristics - Pressurised Thick Cylinder.



Short Crack Length Convergence Characteristics - Autofrettaged Thick Cylinder.

Figure 9 : Short Crack Length Convergence Characteristics.

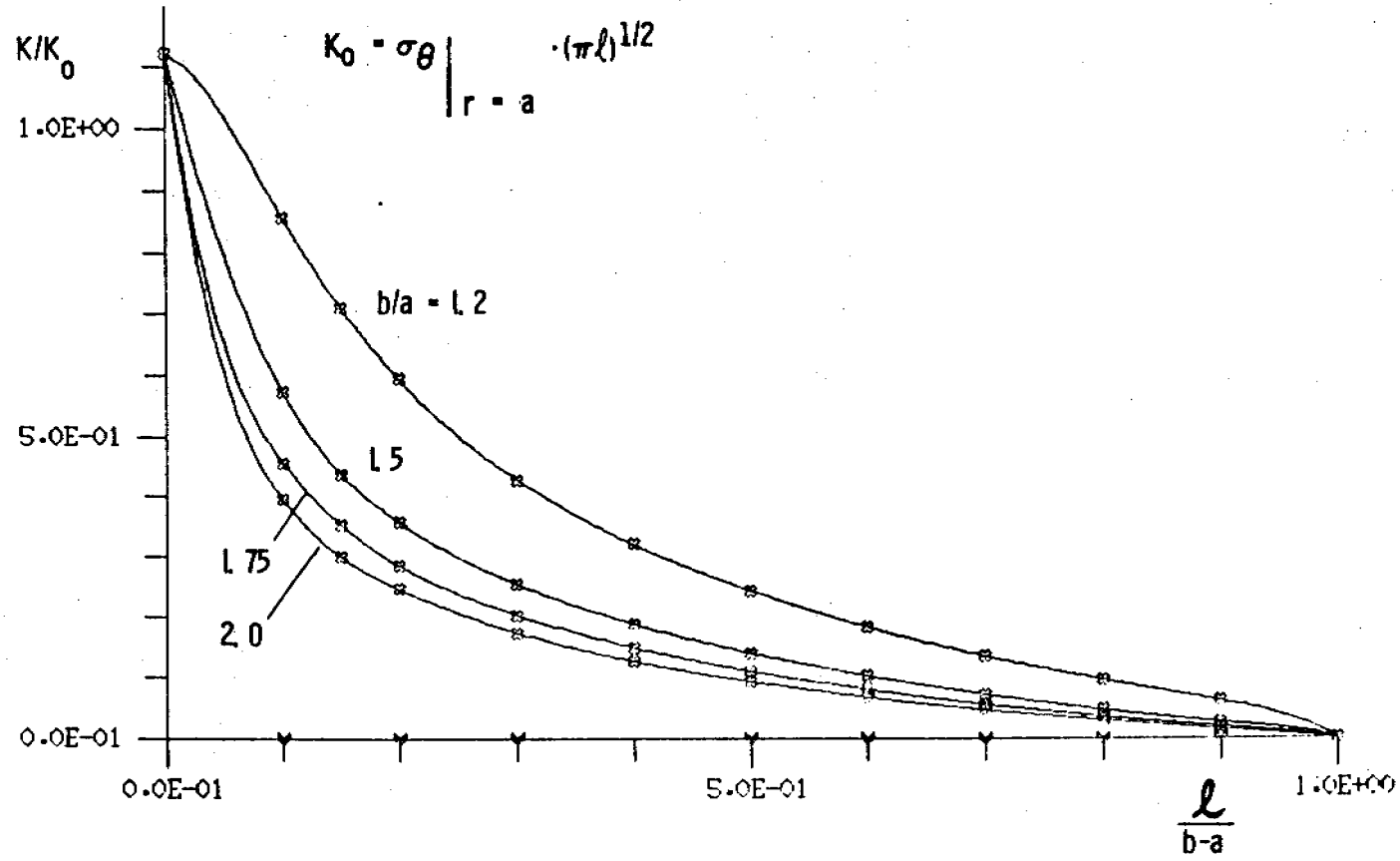


Figure 10 : Stress Intensity Factors for Autofrettaged Tube With 50 Radial Cracks.

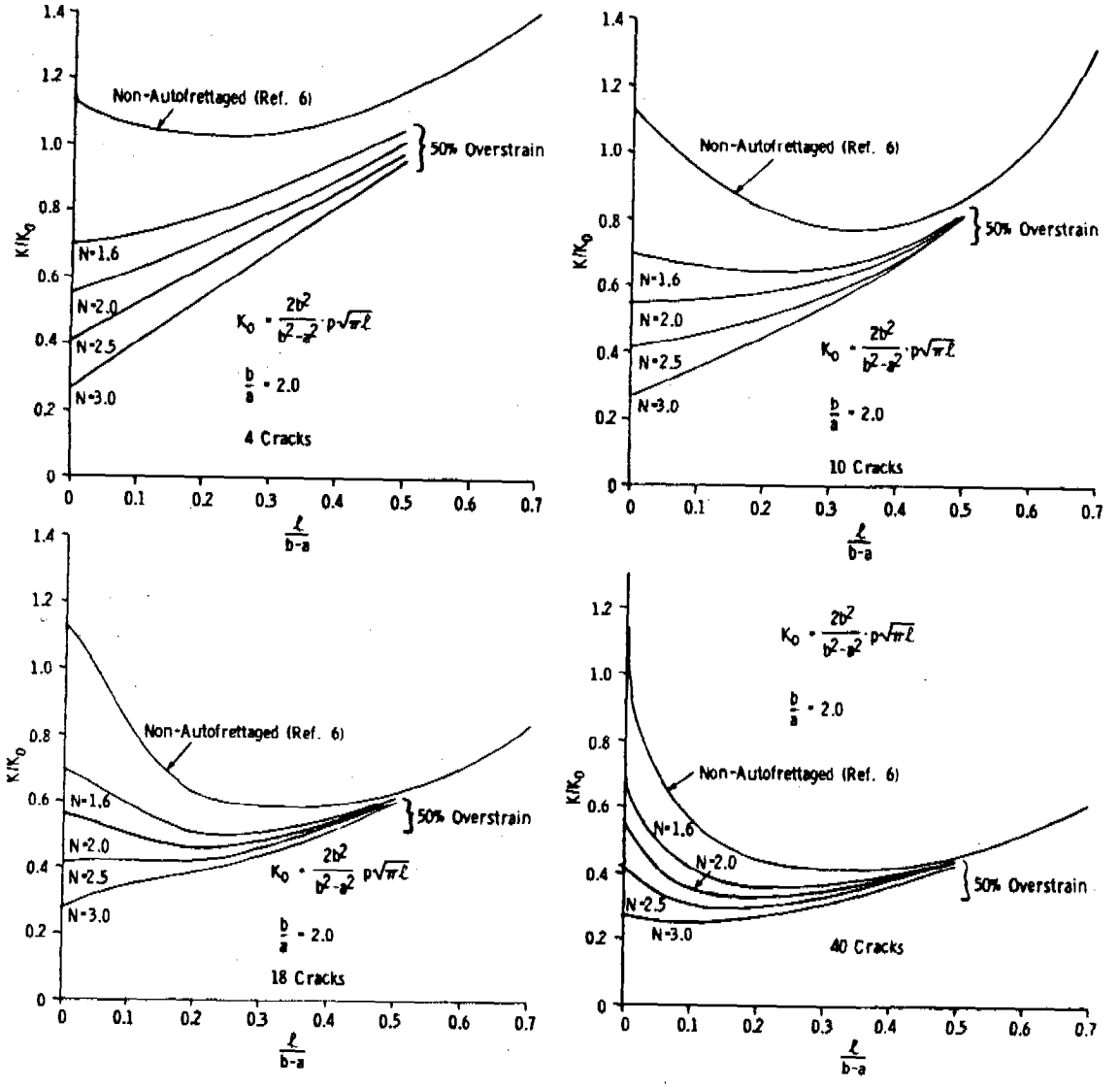


Figure 11 : Stress Intensity Factors for Internally Cracked, Pressurized Tube with 50% Autofrettage.

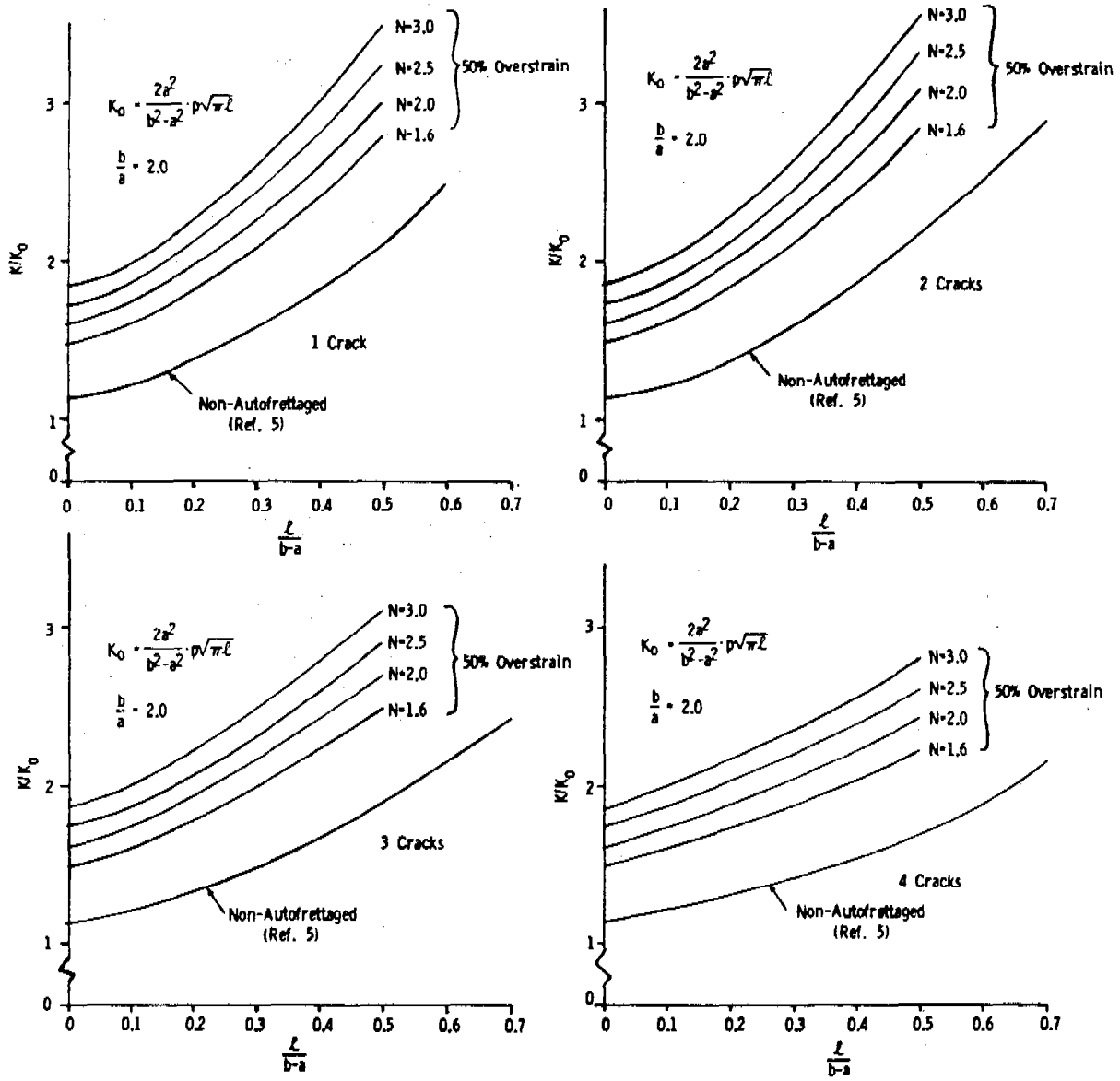


Figure 12 : Stress Intensity Factors for Externally Cracked, Pressurized Tube with 50% Autofrettage.

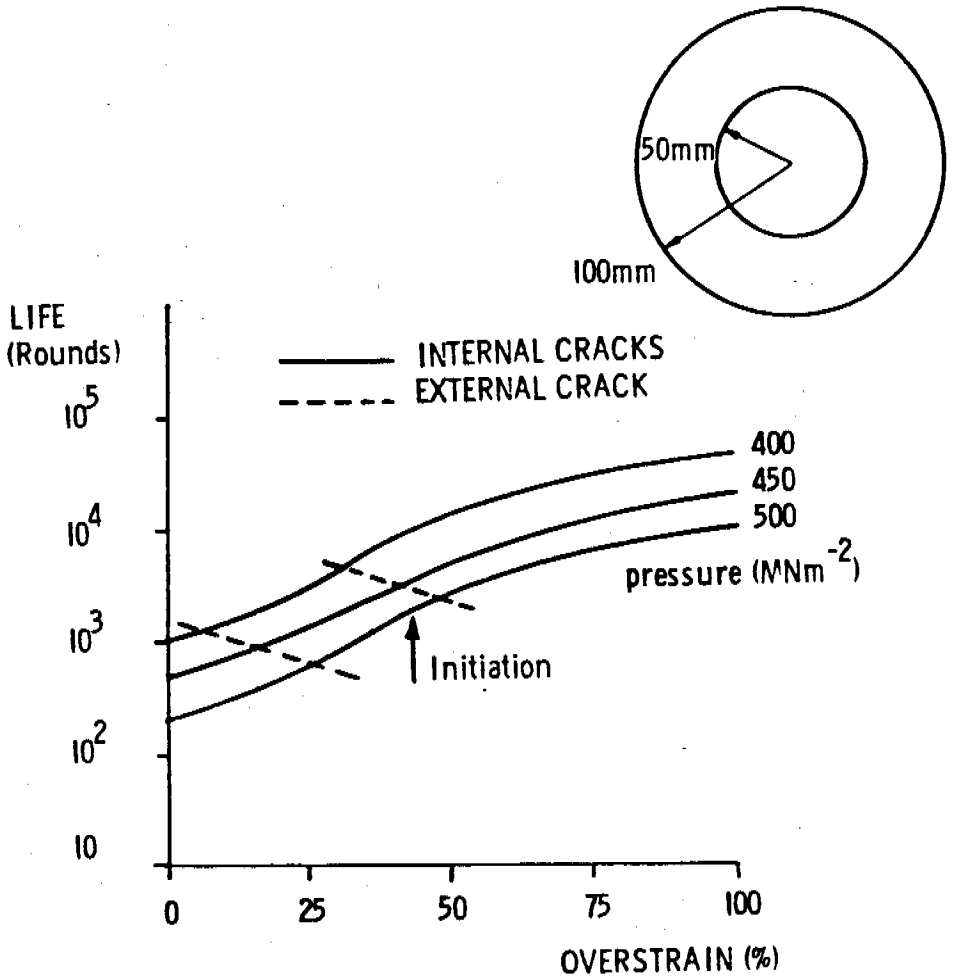


Figure 13 : Calculated Lifetimes for Tube with 40 Internal Cracks or One External Crack Subjected to Cyclic Pressure Loadings of 400, 450 and 500 MNm^{-2}.

GUN TUBE FATIGUE LIFE ESTIMATES- INFLUENCE OF RESIDUAL STRESS, CRACK GROWTH LAW AND LOAD SPECTRA

Donald M. Neal
Anthony P. Parker
Edward M. Lenoe
US Army Materials and Mechanics Research Center
Watertown, Massachusetts, 02172

ABSTRACT. A gun tube should be withdrawn from service before crack-like defects within the tube can achieve a critical length and cause catastrophic brittle failure. The objective of this study is to conduct a sensitivity analysis of the relative importance of fracture toughness, yield strength, proportion of autofrettage, initial crack length firing pressure and crack growth law parameters in the determination of safe life estimates for gun tubes. By recognizing the importance of the individual parameter in the life prediction procedure, requirements for accurate determination of the parameter can be established.

The Monte Carlo method was applied in order to simulate parameter variability in the life time estimating process. A normal distribution function was assumed where a specific coefficient of variation (C.V.) described the relative amount of variability. The largest dispersion in the life time estimates resulted from 5% variation in the power term of the crack growth law. The other parameter contributed by a considerable lesser amount in the life variability. The results also indicated a considerable advantage when the autofrettage process was applied to the gun tube. The lognormal probability density function "best" represented probability ranked life estimates when compared to the Weibull and normal functions.

NOMENCLATURE

a	Inner tube radius
b	Outer tube radius
c	Autofrettage radius
C	Coefficient in Paris' crack growth law
F	Proportion of autofrettage
K	Stress intensity factor (range)
K_{max}	Maximum value of stress intensity during loading cycle
K_{min}	Minimum value of stress intensity during loading cycle
K_c	Fracture toughness
l	Crack length
l_i	Initial crack length
l_c	Critical crack length
m	Exponent in Paris' law
N	Number of loading cycles
p	Pressure
Q	Configuration correction factor
Y	Yield strength
	Factor employed in determination of K for partial autofrettage

1. INTRODUCTION. The fundamental safety requirement for a gun tube is that it should be withdrawn from service before crack-like defects which develop in the tube during initial firing can grow to a critical length and cause catastrophic, brittle failure of the tube. Ideally the fatigue life should exceed the wear life of the tube, and tube inspection should not be necessary during service life. However, there have been in-service failures of gun tubes [1], and there is evidence to suggest that a relatively small increase in firing pressures (e.g. for the firing of long-rod projectiles) or an improvement in wear characteristics of gun tubes may make fatigue life the dominant limiting factor in life assessment [2].

A linear-elastic fracture mechanics approach to crack growth rate prediction implies the need to calculate accurate stress intensity factor data, and to fully understand the effect of autofrettage residual stresses [3] and multiple cracking on stress intensity calibrations. Deterministic studies relating to each of these problem areas are reported elsewhere in this publication [3]. The objective of this study is to conduct a sensitivity analysis, utilizing standard Monte Carlo simulation techniques, in order to gain some understanding of the relative importance of Fracture Toughness, yield strength, proportion of autofrettage, initial crack length, firing pressure and crack growth law on the fatigue life of gun tube.

2. METHOD OF LIFE PREDICTION. For much of the lifetime of a cracked component, the fatigue crack growth rate is given by Paris' law:

$$\frac{d\ell}{dN} = C(K)^m \qquad (1)$$

where ℓ is the crack length, N is the number of cycles and K is the stress intensity factor range, $K_{max}-K_{min}$ [3], where K_{max} and K_{min} are the maximum and minimum values respectively of the stress intensity during the loading cycle. C and m are empirical constants, which are determined for the particular material and thickness in a standard test.

In order to predict lifetime to failure for a gun tube, we write equation (1) such that:

$$N = \int_{\ell_i}^{\ell_c} \frac{d\ell}{C(K)^m} \qquad (2)$$

where ℓ_i is some initial crack length (in the case of a gun tube this is normally taken as the depth of the heat-check craze cracking which appears at the bore after the first few rounds are fired) and ℓ_c is the critical crack length associated with some critical value of K_{max}, termed the fracture toughness or designated K_c.

A typical cracked gun-tube geometry is illustrated in Fig. 1. The tube has internal radius a, external radius b and has been autofrettaged to a radius c. [Autofrettage is a process in which plastic flow is induced in the tube during manufacture. This plastic flow commences at the bore, and spreads radially outwards. The process induces an advantageous distribution of compressive residual stresses in the inner portion of the tube which tend to reduce the stress intensity of cracks emanating from the inner radius.]

In the case of a pressurized tube, it is standard practice to express the stress intensity factor range, K, as:

$$K = Q(\ell)\ p(\pi\ \ell)^{1/2} \tag{3}$$

where p is the maximum pressure during the firing cycle, and $Q(\ell)$ is some configuration correction factor which includes the effects of loading and geometry. In the case of an autofrettaged tube:

$$K = \alpha K_p + K_A \tag{4}$$

where K_p is the stress intensity contribution due to internal pressure in the bore and the cracks, K_A is the (negative) stress intensity due to full (100%) autofrettage (i.e. c = b). Numerical solutions for K_p and K_A appear in [3]. α is a function of the ratio of material yield strength Y to working pressure, p, given by:

$$\alpha = \left[1 + \frac{Y}{p}\ \ell n\ \left(\frac{b}{c}\right) - \frac{Y}{p}\ \frac{(b^2-c^2)}{2b^2}\right] \tag{5}$$

whilst the autofrettage radius, c is given by:

$$c = F(b-a) + a \tag{6}$$

F being the proportion of autofrettage (i.e. percentage overstrain = 100 x F.)

3. SELECTED PARAMETERS. The mean parameter values considered are listed in Fig. 2. The working pressure, p ($400 MNm^{-2}$), material yield strength, Y ($1200\ MNm^{-2}$) and fracture toughness, K_c ($90\ MNm^{-3/2}$) were selected to be typical of gun tube operation and material. The proportion of overstrain, F spans the whole range from zero to 100% autofrettage. ℓ_i is typical of measured heat-check crack depths. The Paris' law constants, $C=1.45 \times 10^{-11}$ and m=3.1 (giving crack growth rates in meters /cycle from stress intensity in $MNm^{-3/2}$) are also characteristic of gun steel. The tube was assumed to have an inner radius of 50mm and outer radius 100mm.

The coefficient of variation (C.V.) in each of the parameters is taken as 5% throughout, with the exception of working pressure (2%) and initial crack length (10%), in order to model some of the "real-life" variations. The parameters marked with an 'O' in Fig. 2 were not varied during the tests, since Y cannot influence life with zero autofrettage, and variations of autofrettage below zero and above full autofrettage are physically unacceptable. All of the parameters marked 'XX' were insensitive at the C.V. levels employed.

4. MONTE CARLO SIMULATION. In the simulation scheme a probability distribution function for N described in (2) is determined. The necessary parameters C, M, ℓ_i and those related to K (4, 5, and 6) are represented by a normal distribution function with appropriate means and C.V. (See Fig. 2).

A random selection from each of the parameter distributions is inserted in (2) and solution for K is obtained. This process is repeated until all functional values have been selected. Note, an equal number of random values for each individual parameters should be generated. The resultant number for the life time distribution will be the same as those determined for the parameters. Although an initial assumption of normality existed for each of the parameters, the resultant life estimate distribution was not normal. This situation often occurs in the Monte Carlo process.

The random numbers for the individual functions are obtained from generation of uniform random numbers and solving for X in the relation.

$$\int_{\alpha}^{X} f_i \, dX = R \qquad (7)$$

where R = Uniform random number and
f_i = normal frequency distribution.

If the probability distributions of the controlling parameter are known from some experimental results or from an analytic basis, then the appropriate distribution function f_i may be used.

An examination of the relative change in the third and fourth moments (Skewness and Kurtosis) as related to the increasing number of simulations provided the necessary mechanism for determining an acceptable number of simulations. Observing the Tabulation of Moments vs. Number Trials in Fig. 3 indicated approximately 2000 simulations would be sufficient. Acceptable convergence of Skewness and Kurtosis indicates functional distribution form does not vary due to increasing number trials.

5. SENSITIVITY ANALYSIS. Fig. 4 shows probability density histograms for the case of a non-autofrettaged tube with 40 internal radial cracks. This number was selected as being typical of crack patterns observed in non-autofrettaged gun tubes. The results indicate a relative insensitivity to variations in p of 2%, of 2 and 5 percent for p and K_c respectively, but a most significant dependence on m with 5% variation. Fig. 4(d) shows the effect of varying all parameters simultaneously. 99.9% life with all parameters varying is 111 rounds. (Probability that lifetime of tube will exceed 111 rounds is .999.)

Fig. 5 illustrates the results for a tube with 75% autofrettage (i.e. F = 0.75) with 4 internal radial cracks. This number appears typical of crack patterns in autofrettaged tubes. The parameters, in increasing order of sensitivity are ℓ_i, F and m. The 99.9% life in this case with all parameters varying is 1869 rounds.

Finally, Fig. 6 illustrates the results for 100% autofrettage (F = 1.0), with 4 internal cracks. The sensitive parameters, in increasing order, are p, ℓ_i and m. The 99.9% life is 4683 rounds. With increasing amounts of autofrettage, it becomes more important to include variations in all parameters, not just m (see Figs. 4, 5, 6).

As the amount of autofrettage is increased, K_c effect on variability decreases while ℓ_i exhibits the opposite characteristics. This appears reasonable on physical grounds, since more of the tube lifetime is expended at very short crack lengths as the amount of autofrettage is increased. Conversely, the proportion of life spent at longer crack lengths becomes less significant.

Inspection of the variation in mean, standard deviation and 99.9% life indicates the very considerable advantages associated with the autofrettage process. In particular, the 99.9% life is increased from 111 rounds with zero autofrettage, to 1869 with 75% autofrettage and 4683 with 100% autofrettage. Since gun tubes would normally be required to guarantee something like 1500 rounds, and a factor of safety is required, it is clear that the non-autofrettaged tube would not be acceptable, whilst the 75% and 100% autofrettage tube would represent viable options, the only additional cost being the autofrettage process itself, no modifications to the material being required.

6. CUMULATIVE PROBABILITY FITTING. Considering a random selection of 300 data values with parameter variations of the specified amount listed in Fig. 2, we obtain the Normal, Weibull and Lognormal density function representation of the ranked data illustrated in Figs. 7, 8 and 9, for zero, 75% and 100% overstrain respectively. In all cases the RMS errors and graphical results indicate that the Lognormal best represent the data. This observation is readily understood when we recall that the distribution is dominated by the effect of m, the exponent in Paris' law. The mean, standard deviation and Weibull parameter are listed in Figs. 7, 8 and 9 with their corresponding 90% tolerance limits. The two parameter Weibull was considered to be a more accurate representation of the data than the three parameter results. The appearance of outliers in Figs. 7 and 9 do not effect either inference results or selection of the proper functional representation.

7. CONCLUSIONS & DISCUSSION. The combination of linear elastic fracture mechanics and an empirical crack growth law has become the standard method for the calculation of fatigue life in gun tubes. The fundamental requirement of such an approach is that the gun tube should be withdrawn from service before catastrophic brittle failure can occur.

The sensitivity analysis conducted herein indicates that in non-autofrettaged gun tubes the most sensitive parameter is m, the exponent in Paris' crack growth law. With increasing amounts of autofrettage the initial crack length and proportion of autofrettage are also significant factors. For cumulative failure probability, the lognormal distribution is superior to both normal and Weibull distributions for zero, 75% and 100% autofrettage.

The improvement in 99.9% life resulting from large amounts of autofrettage, based on typical materials and loadings, indicates the great advantages which autofrettage may provide. One particularly important feature of the results reported here is that the current practice of applying 75% autofrettage and limiting life to approximately 2000 rounds for typical pressures and gun tube steels, is consistent with the results presented in Fig. 5 which were obtained using the Monte Carlo scheme.

Whilst this study relates to a particular, axisymmetric geometry containing residual stresses, the benefits of introducing advantageous residual stresses in more complex geometrical configurations, such as pin-loaded lugs and welded joints, are already becoming apparent. This method of sensitivity analysis would also be applicable to such configurations.

8. ACKNOWLEDGEMENT. One of us (APP) performed work on this paper during a TTCP attachment to the Mechanics and Engineering Laboratory, AMMRC, Watertown, a secondment from the Royal Military College of Science, Shrivenham, Wiltshire, SN6 8LA, England. The authors are indebted to Sheila Evans of AMMRC for the excellent work in preparing the manuscript.

9. REFERENCES

[1] Shinozuka, M., Vaicaitis, R., and Lenoe, E. M., "Sensitivity Analysis of Fatigue Life Estimates in Cannon Bores", Proc. US Army Symp. on Solid Mechanics, 'Case Studies on Structural Integrity and Reliability', AMMRC MS 78-3 (1978).

[2] Tabone, M. V., Burns, I. W. and Gibson, A. F., "A Review of Fatigue Life Prediction for In-service Ordnance", Army Staff Course Project, Royal Military College of Science, England (1980).

[3] Parker, A. P., Sleeper, K. A. and Andrasic, C. P., "Safe Life Design of Gun Tubes - Some Numerical Methods & Results", US Army Numerical Analysis & Computers Conference, Huntsville, Alabama (1981), in press.

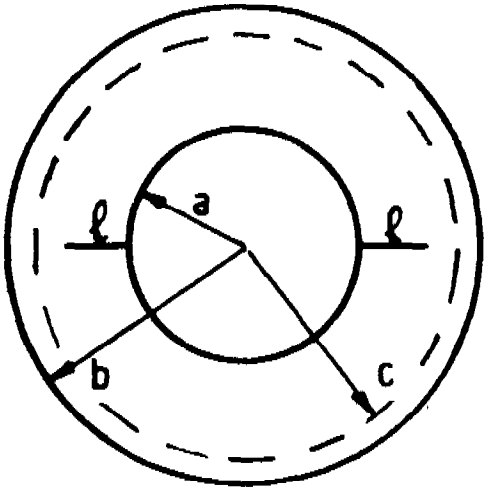


Fig. 1: CRACKED THICK CYLINDER GEOMETRY

		C. V.	F=0	F=.75	F=1.0
p	(400 MN m^{-2})	.02	X	X	X
Y	(1200 MN m^{-2})	.05	0	XX	XX
F	(0, .75, 1.0)	.05	0	X	0
K_c	(90 MN $m^{-3/2}$)	.05	X	XX	XX
ℓ_i	(.002 m)	.1	XX	X	X
m	(3.1)	.05	X	X	X
C	(1.45 x 10^{-11})	.05	XX	XX	XX

X TESTED
XX TESTED BUT INSENSITIVE
0 NOT TESTED

Fig. 2: PARAMETERS TESTED IN MONTE CARLO SIMULATION

NUMBER OF TRIALS	MEAN	S. D.	SKEW.	KURT.
100	1042	124	. 071	2. 96
200	1046	108	. 111	2. 79
400	1066	107	. 144	3. 39
1000	1058	108	. 240	2. 94
1500	1058	108	. 227	3. 40
2000	1057	106	. 181	2. 97
3000	1056	108	. 220	2. 95

BASED ON PRESSURE VARIATION OF 2% IN NON-AUTOFRETTAGED BARREL.

Fig. 3 : CONVERGENCE CHARACTERISTICS FOR MONTE CARLO SIMULATION.

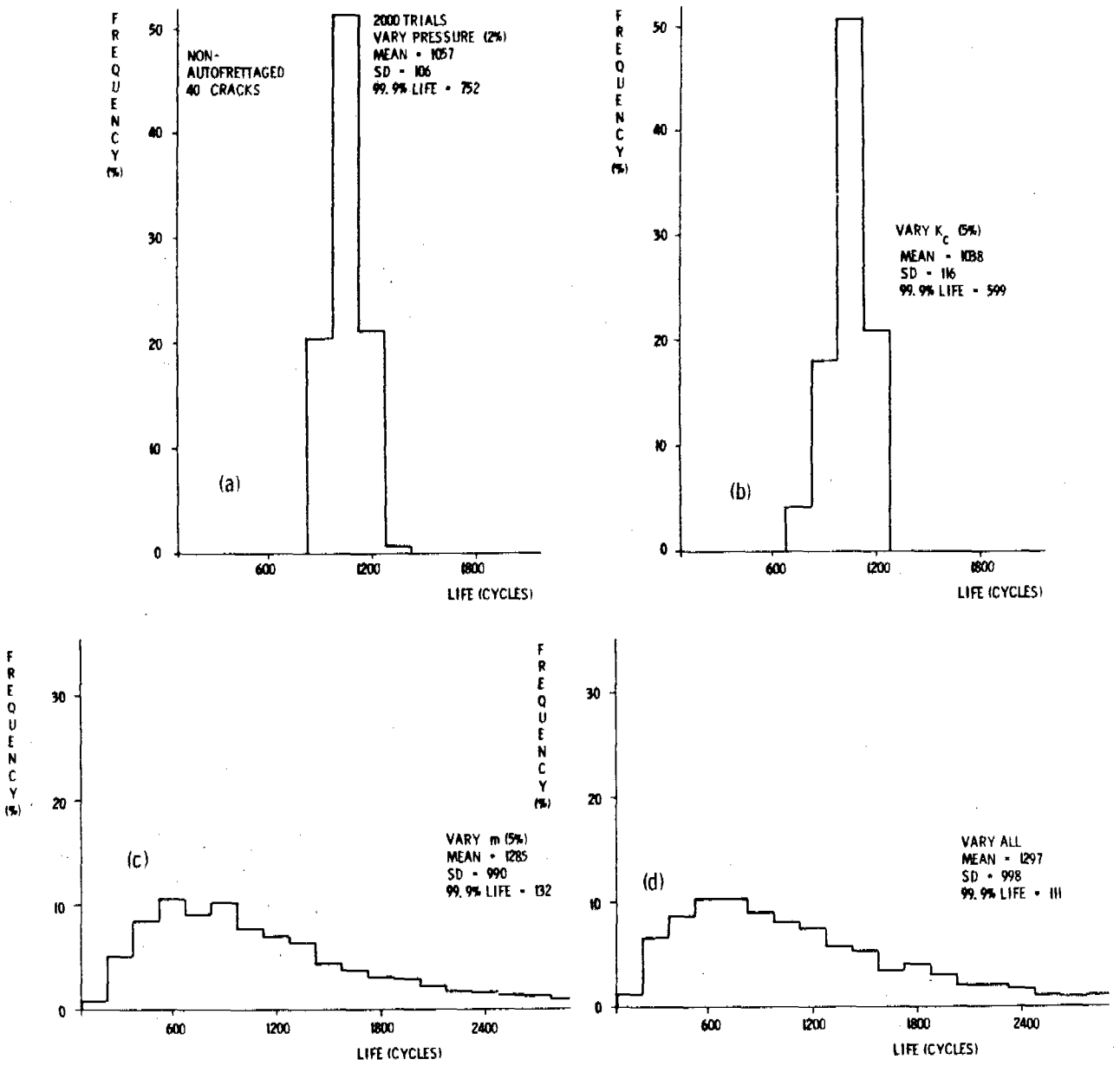


Fig. 4 : PROBABILITY DENSITY FOR NON-AUTOFRETTAGED TUBE WITH 40 CRACKS.

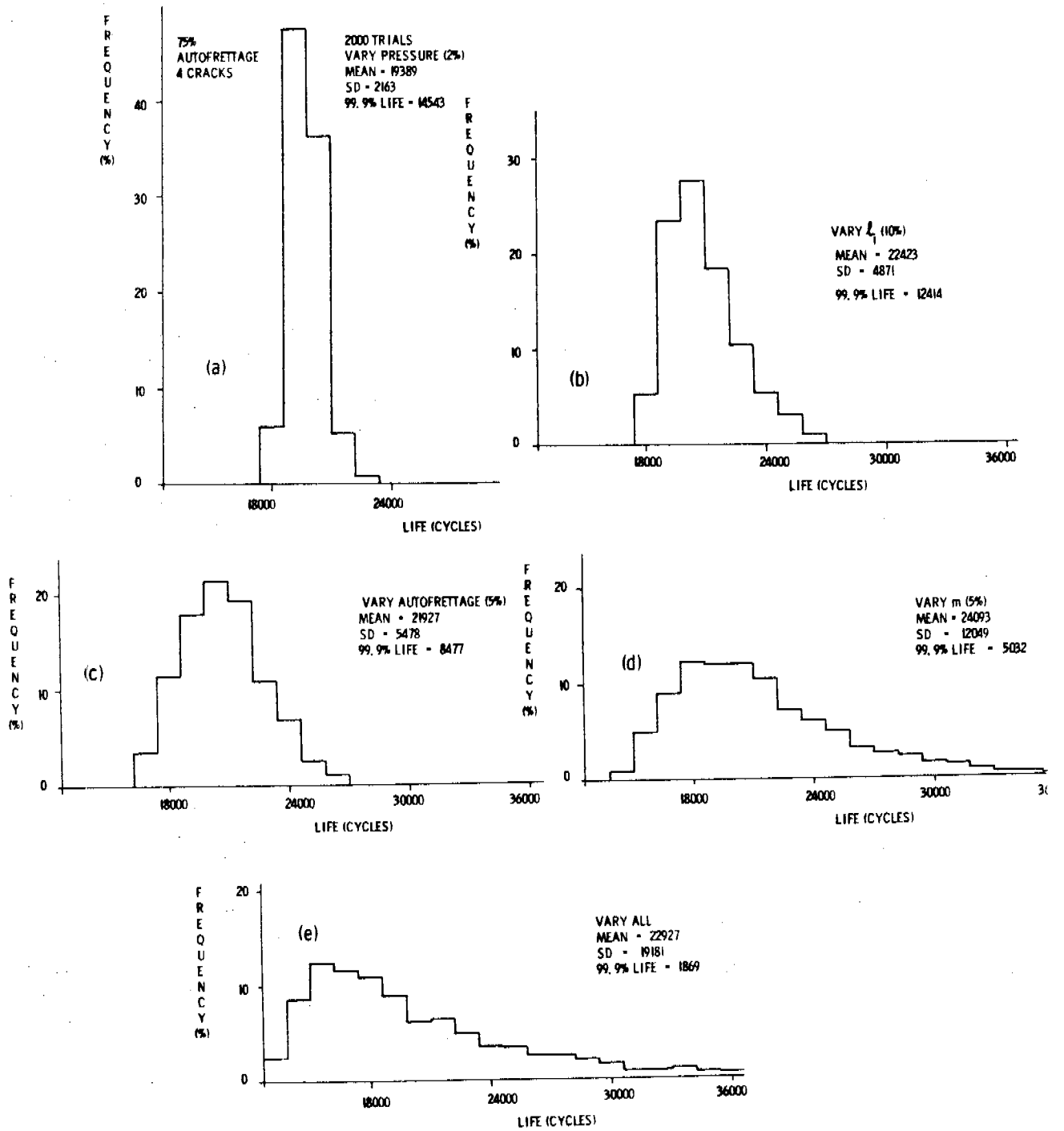


Fig. 5 : PROBABILITY DENSITY FOR 75% AUTOFRETTAGED TUBE WITH 4 CRACKS.

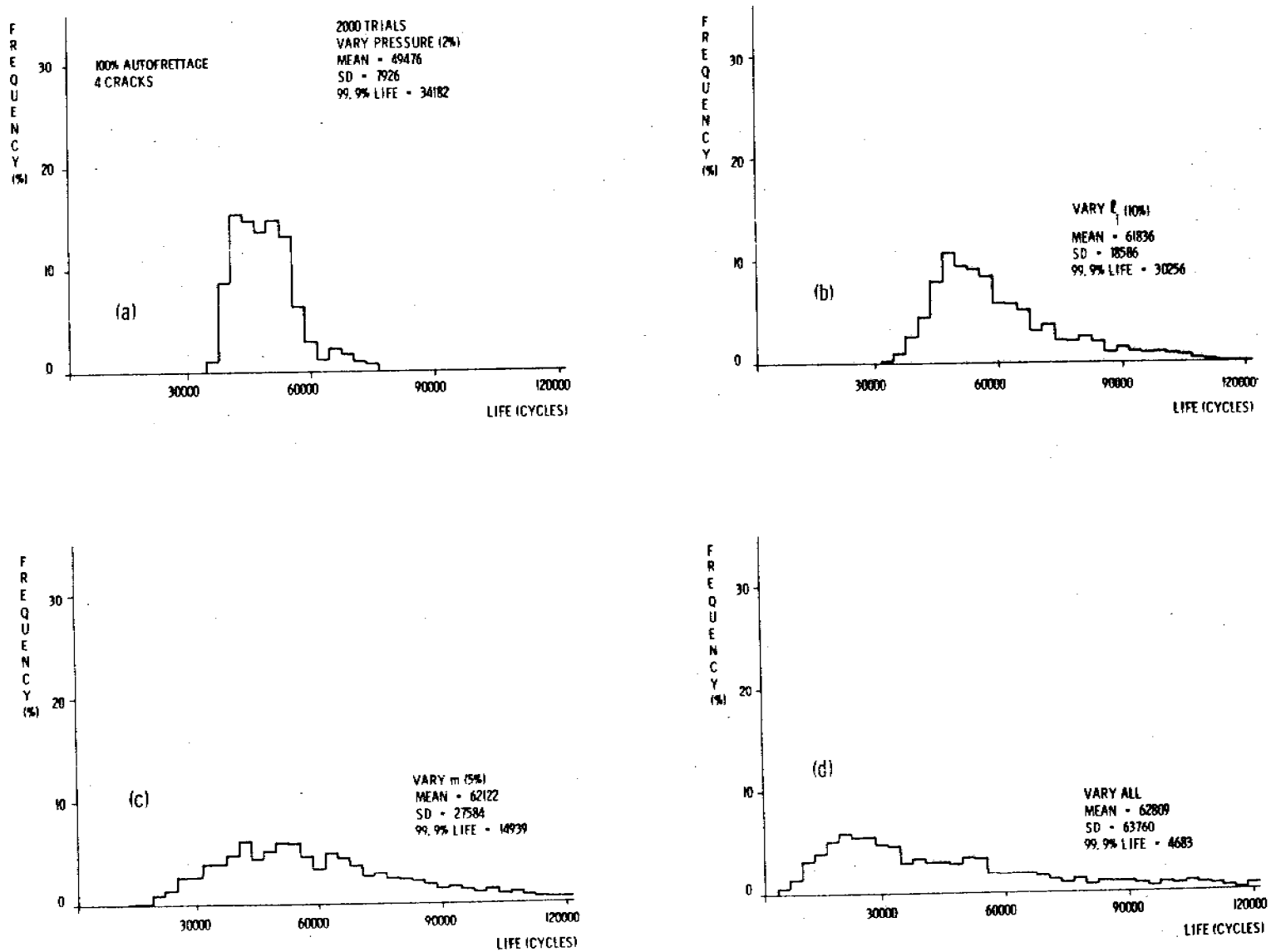


Fig. 6 : PROBABILITY DENSITY FOR 100% AUTOFRETTAGED TUBE WITH 4 CRACKS.

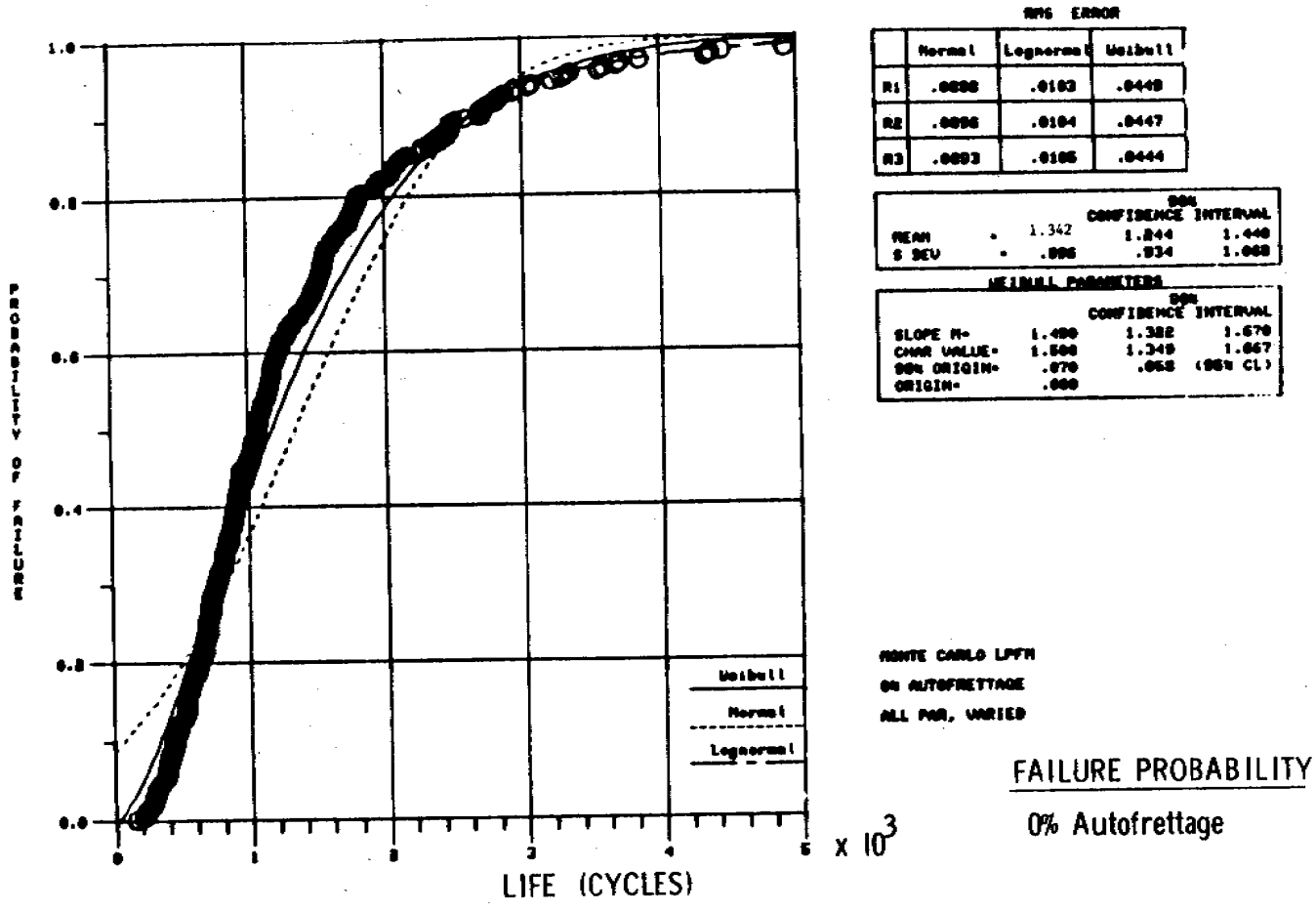


Fig. 7 : CUMULATIVE DISTRIBUTION FUNCTIONS FOR NON-AUTOFRETTAGED TUBE WITH 40 CRACKS.

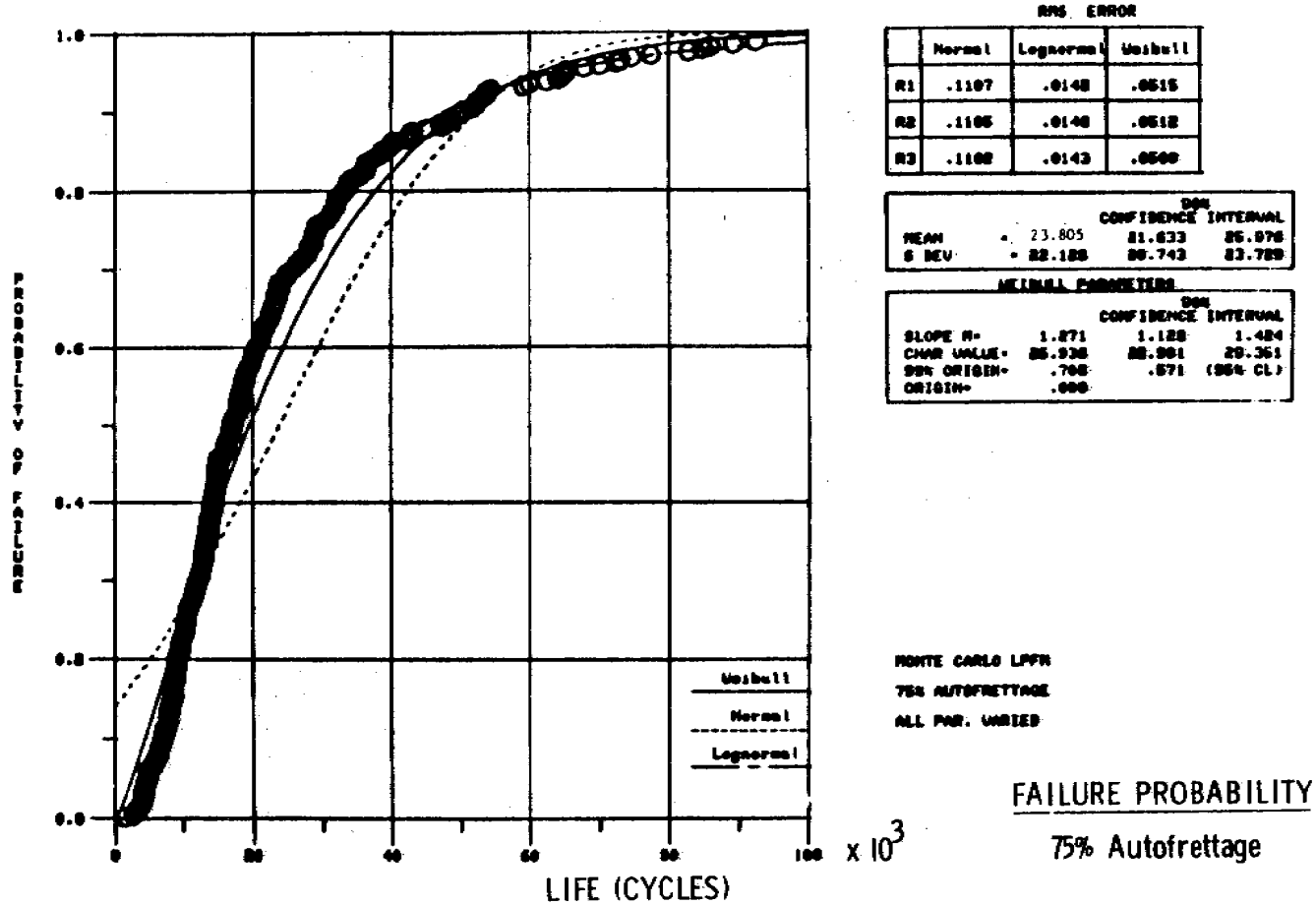


Fig. 8 : CUMULATIVE DISTRIBUTION FUNCTIONS FOR 75% AUTOFRETTAGED TUBE WITH 4 CRACKS

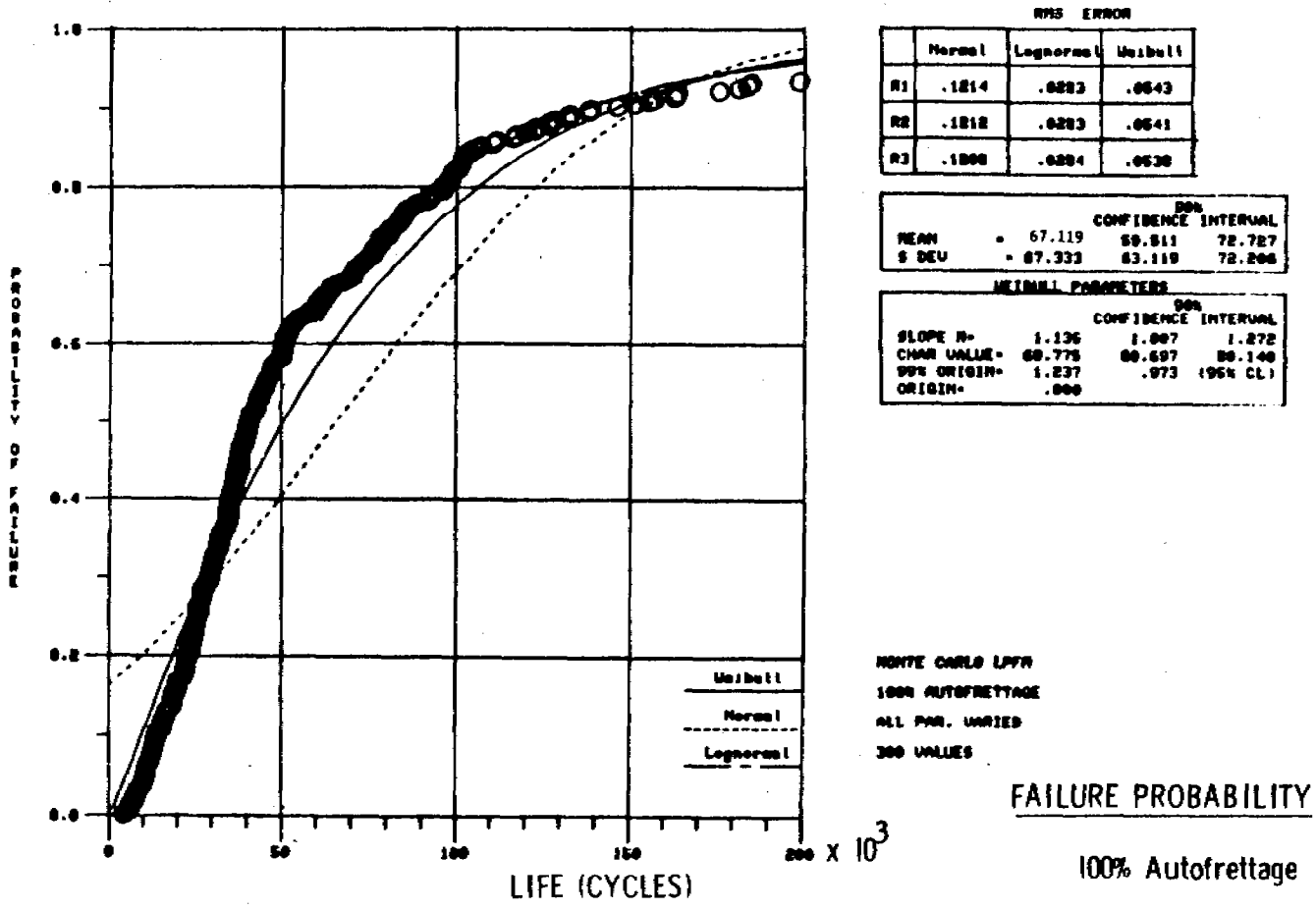


Fig. 9 : CUMULATIVE DISTRIBUTION FUNCTIONS FOR 100% AUTOFRETTAGED TUBE WITH 4 CRACKS.

NUMERICAL PREDICTION OF RESIDUAL STRESSES IN AN AUTOFRETTAGED TUBE OF COMPRESSIBLE MATERIAL

P. C. T. Chen
U. S. Army Armament Research and Development Command
Large Caliber Weapon Systems Laboratory
Benet Weapons Laboratory
Watervliet, NY 12189

ABSTRACT. The residual stresses in an autofrettaged tube of compressible material are obtained by a new finite difference approach. The tube is assumed to obey the Mises' yield criterion, the Prandtl-Reuss flow theory and the isotropic-hardening rule. In order to test the accuracy of the computer program, a convergence study for a nearly incompressible tube has been made and compared with the exact solution as well as the simulated results for residual stresses in an incompressible tube.

1. INTRODUCTION. The importance of favorable residual stresses in an autofrettaged tube is well known [1]. Many methods for predicting residual stresses have been reported [2-4]. For an elastic-plastic material which obeys the Mises' yield criterion and the associated flow rules, a closed form solution exists only in the plane strain case neglecting strain hardening and compressibility [5]. Recently a method to simulate this problem by thermal loads has been devised by Hussain et al [6]. For a compressible material with or without strain hardening, a new finite difference approach has been developed by this author [7]. Two types of incremental loadings have been discussed. In the present paper, the numerical prediction of residual stresses in an autofrettaged tube of compressible material will be reported. The effect of Poisson's ratio will be discussed. In order to test the accuracy of the computer program, a convergence study for a nearly incompressible tube has been made and compared with the exact solution as well as the simulated results for residual stresses in an incompressible tube.

2. INCOMPRESSIBLE TUBE. For an ideally-plastic incompressible tube which obeys the Mises' yield criterion and the associated flow rules, a closed form solution exists in the plane strain case. The residual stresses and displacement after complete elastic unloading in a partially autofrettaged tube are given by [5],

$$\left.\begin{matrix}\sigma_r \\ \sigma_\theta\end{matrix}\right\} = \frac{\sigma_o}{\sqrt{3}}\left[\left(\frac{\rho^2}{b^2} - 2\log\frac{\rho}{r} \mp 1\right) - p_1\left(1 \mp \frac{b^2}{\rho^2}\right)\right] \qquad a \leq r \leq \rho \tag{1}$$

$$\left.\begin{matrix}\sigma_r \\ \sigma_\theta\end{matrix}\right\} = \frac{\sigma_o}{\sqrt{3}}\left(\frac{\rho^2}{b^2} - p_1\right)\left(1 \mp \frac{b^2}{\rho^2}\right) \qquad \rho \leq r \leq b \tag{2}$$

$$\sigma_z = \begin{cases} \dfrac{\sigma_o}{\sqrt{3}}\,(\rho^2/b^2 - 2\log \rho/r - p_1) & a \leq r \leq \rho \\ \dfrac{\sigma_o}{\sqrt{3}}\,(\rho^2/b^2 - p_1) & \rho \leq r \leq b \end{cases} \tag{3}$$

$$u/r = (\sqrt{3}/2)(\sigma_o/E)(\rho/r)^2 \tag{4}$$

where

$$p_1 = (1 - \rho^2/b^2 + 2\log \rho/a)/(b^2/a^2 - 1) \tag{5}$$

and ρ is the radius of the autofrettaged interface.

According to Hussain et al [6], the distribution of radial and hoop stresses can be simulated by a steady state thermal loading. The equivalence between the temperature gradient and the yield stress is

$$\frac{E\alpha(T_a - T_\rho)}{2(1-\nu)\log(\rho/a)} = \frac{2\sigma_o}{\sqrt{3}} \tag{6}$$

and the temperature distribution is given by

$$\begin{aligned} T &= T_a - \frac{(T_a - T_\rho)}{\log(\rho/a)} \log (r/a) && a \leq r \leq \rho \\ T &= T_\rho && \rho \leq r \leq b \end{aligned} \tag{7}$$

3. FINITE DIFFERENCE APPROACH. For a compressible material with or without strain hardening, a new finite difference approach has been developed by this author [7]. An incremental procedure is used for pressure beyond the elastic limit and the elastic solution is used as the initial condition. The cross section of the tube is divided into n rings and we want to determine all incremental quantities at all grid points in each incremental step. In the plastic region, the incremental stresses are related to the incremental strains by the incremental form

$$\Delta\sigma_i = d_{ij}\,\Delta\varepsilon_j \quad \text{for} \quad i,j = r,\theta,z \tag{8}$$

and

$$d_{ij}/2G = \nu/(1-2\nu) + \delta_{ij} - \sigma_i'\sigma_j'/S \tag{9}$$

where E is Young's modulus, ν is Poisson's ratio, δ_{ij} is the Kronecker delta,

$$S = \frac{2}{3}\,(1 + \frac{1}{3}\,H'/G)\sigma^2 \quad , \quad 2G = E/(1+\nu)$$

$$\sigma_m = (\sigma_r+\sigma_\theta+\sigma_z)/3 \quad , \quad \sigma_i' = \sigma_i - \sigma_m$$

$$\sigma = (1/\sqrt{2})[(\sigma_r-\sigma_\theta)^2 + (\sigma_\theta-\sigma_z)^2 + (\sigma_z-\sigma_r)^2]^{1/2} > \sigma_o \qquad (10)$$

and σ_o is the yield stress in simple tension or compression. For a strain hardening material, H' is the slope of the effective stress/plastic strain curve. For an ideally-plastic material, H' = 0. When $\sigma < \sigma_o$ or $d\sigma < 0$, the state of stress is elastic and the third term in equation (9) disappears. Using equation (8) and $\Delta u = r\Delta\varepsilon_\theta$, there exists only two unknowns at each station that have to be determined for each increment of loading. The unknown variables in the present formulation are $(\Delta\varepsilon_\theta)_i$, $(\Delta\varepsilon_r)_i$, for i = 1,2,...n,n+1.

The equation of equilibrium and the equation of compatibility are valid for both the elastic and the plastic regions of a thick-walled tube. The finite-difference forms of these two equations at i = 1,...,n are given by

$$(r_{i+1}-2r_i)(\Delta\sigma_r)_i - (r_{i+1}-r_i)(\Delta\sigma_\theta)_i + r_i(\Delta\sigma_r)_{i+1}$$
$$= (r_{i+1}-r_i)(\sigma_\theta-\sigma_r)_i - r_i[(\sigma_r)_{i+1} - (\sigma_r)_i] \qquad (11)$$

for the equation of equilibrium, and

$$(r_{i+1}-2r_i)(\Delta\varepsilon_\theta)_i - (r_{i+1}-r)(\Delta\varepsilon_r)_i + r_i(\Delta\varepsilon_\theta)_{i+1}$$
$$= (r_{i+1}-r_i)(\varepsilon_r-\varepsilon_\theta)_i - r_i[(\varepsilon_\theta)_{i+1} - (\varepsilon_\theta)_i] \qquad (12)$$

for the equation of compatibility.

With the aid of the incremental stress-strain relations (equation (8)), equation (11) can be rewritten as

$$[(r_{i+1}-2r_i)(d_{12})_i + (-r_{i+1}+r_i)(d_{22})_i](\Delta\varepsilon_\theta)_i$$
$$+ [(r_{i+1}-2r_i)(d_{11})_i + (-r_{i+1}+r_i)(d_{21})_i](\Delta\varepsilon_r)_i$$
$$+ r_i(d_{12})_{i+1}(\Delta\varepsilon_\theta)_{i+1} + r_i(d_{11})_{i+1}(\Delta\varepsilon_r)_{i+1}$$
$$= (r_{i+1}-r_i)(\sigma_\theta-\sigma_r) - r_i[(\sigma_r)_{i+1} - (\sigma_r)_i] \qquad (13)$$

The boundary conditions for the problem are

$$\Delta\sigma_r(a,t) = -\Delta p \quad , \quad \Delta\sigma_r(b,t) = 0 \qquad (14)$$

Using the incremental relations (equation (8)), we rewrite equation (11) as

$$(d_{12})_1(\Delta\varepsilon_\theta)_1 + (d_{11})_1(\Delta\varepsilon_r)_1 = -\Delta p \qquad (15)$$

and

$$(d_{12})_{n+1}(\Delta\varepsilon_\theta)_{n+1} + (d_{11})_{n+1}(\Delta\varepsilon_r)_{n+1} = 0 \qquad (16)$$

Now we can form a system of 2(n+1) equations for solving 2(n+1) unknowns, $(\Delta\varepsilon_\theta)_i$, $(\Delta\varepsilon_r)_i$, for i = 1,2,...,n,n+1. Equations (15) and (16) are taken as the first and last equations, respectively, and the other 2n equations are set up at i = 1,2,...,n using equations (12) and (13). The final system is an unsymmetric band matrix with the nonzero terms clustered about the main diagonal, two below and one above.

When the total applied pressure p is given, it is natural to divide the loading path into m equal fixed increments with $\Delta p = (p-p^*)/m$ where p^* is the pressure corresponding to initial yielding. These fixed increments need not be equal for all steps and any sequence of m increments can be supplied by the user. In [7], an adaptive algorithm to generate a sequence of load increments was described.

4. NUMERICAL RESULTS AND DISCUSSIONS. In order to test the accuracy of the computer program, a convergence study for a nearly incompressible tube (ν = .4999999) has been made and compared with the exact solution for an incompressible tube (ν = 1/2). The numerical results for a tube with b/a = 2 and H' = 0 are very accurate as shown in Table 1 for 30, 60, and 100 percent overstrain. A comparison of the calculated residual hoop stresses with the exact solution as well as the simulated results is shown in Table 2. The finite difference approach can generate more accurate results than the method of simulation by thermal load for incompressible material. In order to discuss the effect of compressibility, we calculate the residual stresses for a tube with b/a = 2, H' = 0, n = 400, ν = 0, 0.3, 0.4999. The results are shown in Tables 3, 4, and 5 for residual hoop, radial, and axial components, respectively. The effect of hardening on the residual stresses can be discussed in a similar way. The results for a tube with b/a = 2, ν = 0.3, n = 400, H'/E = 0, 1/9, 1/19 (w = Et/E = 0, 0.05, 0.0) are shown in Tables 6, 7, and 8 for residual hoop, radial, and axial components, respectively. It can be seen that the effect of hardening on residual hoop stress is larger than that of compressibility.

REFERENCES

1. Hill, R., Mathematical Theory of Plasticity, Oxford University Press, 1950.

2. Hodge, P. G. and White, G. N., "A Quantitative Comparison of Flow and Deformation Theories of Plasticity," J. Appl. Mech., Vol. 17, 1956, pp. 180-184.

3. Chu, S. C., "A More Rational Approach to the Problem of an Elastoplastic Thick-Walled Cylinder," J. of the Franklin Institute, Vol. 294, pp. 57-65.

4. Chen, P. C. T., "The Finite Element Analysis of Elastic-Plastic Thick-Walled Tubes," Proceedings of Army Symposium on Solid Mechanics, 1972, The Role of Mechanics in Design-Ballistic Problems, pp. 243-253.

5. Prager, W. and Hodge, P. G., Theory of Perfectly Plastic Solids, Dover Publications, Inc., 1950.

6. Hussain, M. A., Pu, S. L., Vasilakis, J. D., and O'Hara, P., "Simulation of Partial Autofrettage by Thermal Loads," J. Pressure Vessel Technology, 1980, pp. 314-318.

7. Chen, P. C. T., "An Adaptive Algorithm For Exact Solution of an Over-Strained Tube," Proceedings 1980 Army Numerical Analysis and Computer Conference, pp. 347-355.

TABLE 1. CONVERGENCE STUDY FOR A NEARLY INCOMPRESSIBLE TUBE UNDER INTERNAL PRESSURE ($b/a = 2$, $H' = 0$, $\nu = .4999999$)

O.S.	n	P/σ_o	MAX σ_θ/σ_o	Inside σ_z/σ_o	$\frac{E}{\sigma_o}\frac{U_a}{a}$
30%	10	.64630	.80697	-.06895	1.54781
	20	.64099	.81444	-.06364	1.50104
	50	.63815	.81861	-.05080	1.47764
	100	.63725	.81996	-.05990	1.47047
	200	.63681	.82062	-.05946	1.46699
	400	.63659	.82095	-.05924	1.46528
	*	.63637	.82128	-.05902	1.46358
60%	10	.77375	.93345	-.19640	2.49329
	20	.76123	.94049	-.18388	2.33897
	50	.75464	.94438	-.17729	2.26259
	100	.75257	.94563	-.17522	2.23922
	200	.75156	.94625	-.17421	2.22805
	400	.75105	.94655	-.17371	2.22251
	*	.75056	.94685	-.17321	2.21703
100%	10	.82096	1.15470	-.24361	4.14111
	20	.80999	1.15470	-.23264	3.76669
	50	.80408	1.15470	-.22673	3.57791
	100	.80221	1.15470	-.22486	3.51990
	200	.80129	1.15470	-.22394	3.49173
	400	.80083	1.15470	-.22348	3.47785
	*	.80038	1.15470	-.22303	3.46410

* Exact solution.

TABLE 2. A COMPARISON OF RESIDUAL HOOP STRESS (σ_θ/σ_o) FOR b/a = 2, H' = 0

O.S.	r/a	ν = .5 Exact	n = 400 ν = .4999	ν = .3000 Simulation
30%	1.0	-0.54224	-0.54317	-0.54645
	1.1	- .28497	- .28582	- .29157
	1.2	- .07250	- .07329	- .08021
	1.3	+ .10709	+ .10636	+ .09897
	1.4	+ .09672	+ .09587	+ .08962
	1.5	+ .08835	+ .08774	+ .08205
	1.6	+ .8150	+ .08056	+ .07582
	1.7	+ .07583	+ .07487	+ .07065
	1.8	+ .07107	+ .07102	+ .06630
	1.9	+ .06705	+ .06610	+ .06261
	2.0	+ .06361	+ .06267	+ .05945
60%	1.0	-0.84679	-0.84865	-0.85480
	1.1	- .56305	- .56468	- .57384
	1.2	- .33048	- .33191	- .34250
	1.3	- .13525	- .13652	- .14766
	1.4	+ .03190	+ .03076	+ .01955
	1.5	+ .17737	+ .17635	+ .16534
	1.6	+ .30575	+ .30483	+ .29416
	1.7	+ .28446	+ .28345	+ .21408
	1.8	+ .26662	+ .26555	+ .25270
	1.9	+ .25152	+ .25042	+ .24288
	2.0	+ .23863	+ .23752	+ .23062
100%	1.0	-0.97964	-0.98130	-0.99326
	1.1	- .68437	- .68579	- .70058
	1.2	- .44303	- .44425	- .46027
	1.3	- .24098	- .24203	- .25841
	1.4	- .06842	- .06933	- .08559
	1.5	+ .08142	+ .08063	+ .06474
	1.6	+ .21338	+ .21268	+ .19729
	1.7	+ .33099	+ .33037	+ .31553
	1.8	+ .43681	+ .43634	+ .42205
	1.9	+ .53306	+ .53259	+ .51886
	2.0	+ .62111	+ .62069	+ .60749

TABLE 3. THE EFFECT OF COMPRESSIBILITY ON THE RESIDUAL STRESS σ_θ/σ_o ($b/a = 2$, $H' = 0$, $n = 400$)

O.S.	r/a	$\nu = .4999$	$\nu = .3000$	$\nu = .0000$
30%	1.0	-0.54317	-0.53992	-0.51455
	1.1	- .28528	- .28233	- .25808
	1.2	- .07329	- .07127	- .05712
	1.3	+ .10636	+ .16389	+ .09593
	1.4	+ .09587	+ .09358	+ .08647
	1.5	+ .08774	+ .08530	+ .07887
	1.6	+ .08056	+ .07854	+ .07266
	1.7	+ .07487	+ .07297	+ .06753
	1.8	+ .07012	+ .06831	+ .06324
	1.9	+ .06610	+ .06437	+ .05962
	2.0	+ .06267	+ .06102	+ .05653
60%	1.0	-0.84865	-0.84138	-0.80090
	1.1	- .56468	- .55776	- .51977
	1.2	- .33191	- .32513	- .28850
	1.3	- .13652	- .13036	- .09892
	1.4	+ .03076	+ .03487	+ .05298
	1.5	+ .17635	+ .17160	+ .17167
	1.6	+ .30483	+ .29721	+ .26278
	1.7	+ .28345	+ .27635	+ .24434
	1.8	+ .26555	+ .25889	+ .22892
	1.9	+ .25042	+ .24414	+ .21587
	2.0	+ .23752	+ .23155	+ .20474
100%	1.0	-0.98130	-0.97388	-0.92931
	1.1	- .68579	- .67902	- .63864
	1.2	- .44425	- .43792	- .40015
	1.3	- .24203	- .23600	- .20018
	1.4	- .06933	- .06370	- .03171
	1.5	+ .08063	+ .08531	+ .10837
	1.6	+ .21268	+ .21530	+ .22222
	1.7	+ .33037	+ .32918	+ .31266
	1.8	+ .43634	+ .42900	+ .38327
	1.9	+ .53259	+ .51654	+ .43768
	2.0	+ .62069	+ .59296	+ .47918

TABLE 4. THE EFFECT OF COMPRESSIBILITY ON THE RESIDUAL STRESS σ_r/σ_o
(b/a = 2, H' = 0, n = 400)

O.S.	r/a	ν = .4999	ν = .3000	ν = .0000
30%	1.0	0.00000	0.00000	0.00000
	1.1	- .03732	- .03684	- .25808
	1.2	- .04891	- .04812	- .04462
	1.3	- .04368	- .04287	- .03940
	1.4	- .03320	- .03255	- .02994
	1.5	- .02477	- .02427	- .02234
	1.6	- .01789	- .01752	- .01613
	1.7	- .01220	- .01194	- .01100
	1.8	- .00744	- .00728	- .00671
	1.9	- .00342	- .00335	- .00309
	2.0	0.00000	0.00000	0.00000
60%	1.0	0.00000	0.00000	0.00000
	1.1	- .06371	- .66302	- .05957
	1.2	- .09539	- .09415	- .08795
	1.3	- .10581	- .10413	- .09578
	1.4	- .10182	- .09986	- .09032
	1.5	- .08797	- .08598	- .07658
	1.6	- .06731	- .06566	- .05803
	1.7	- .04593	- .04480	- .03960
	1.8	- .02804	- .02735	- .02417
	1.9	- .01291	- .01259	- .01113
	2.0	0.00000	0.00000	0.00000
100%	1.0	0.00000	0.00000	0.00000
	1.1	- .07520	- .07453	- .07076
	1.2	- .11561	- .11444	- .10780
	1.3	- .13282	- .13125	- .12233
	1.4	- .13424	- .13235	- .12165
	1.5	- .12474	- .12262	- .11079
	1.6	- .10764	- .10541	- .09335
	1.7	- .08523	- .08307	- .07197
	1.8	- .05911	- .05728	- .04850
	1.9	- .03042	- .02929	- .02423
	2.0	0.00000	0.00000	0.00000

TABLE 5. THE EFFECT OF COMPRESSIBILITY ON THE RESIDUAL STRESS σ_z/σ_o
(b/a = 2, H' = 0, n = 400)

O.S.	r/a	ν = .4999	ν = .3000	ν = .0000
30%	1.0	-0.27153	-0.15819	+0.01264
	1.1	- .16153	- .08015	+ .03964
	1.2	- .06108	- .02272	+ .02990
	1.3	+ .03133	+ .01831	+ .00000
	1.4	+ .03133	+ .01831	+ .00000
	1.5	+ .03133	+ .01831	+ .00000
	1.6	+ .03133	+ .01831	+ .00000
	1.7	+ .03133	+ .01831	+ .00000
	1.8	+ .03133	+ .01831	+ .00000
	1.9	+ .03133	+ .01831	+ .00000
	2.0	+ .03133	+ .01831	+ .00000
60%	1.0	-0.42426	-0.28532	-0.07295
	1.1	- .31413	- .18422	+ .01377
	1.2	- .21360	- .10266	+ .06134
	1.3	- .12112	- .03886	+ .07385
	1.4	- .03551	+ .00946	+ .06108
	1.5	+ .04419	+ .04477	+ .03373
	1.6	+ .11874	+ .06946	+0.00000
	1.7	+ .11874	+ .06946	+0.00000
	1.8	+ .11874	+ .06946	+0.00000
	1.9	+ .11874	+ .06946	+0.00000
	2.0	+ .11874	+ .06946	+0.00000
100%	1.0	-0.49052	-0.36683	-0.16290
	1.1	- .38037	- .25538	- .05105
	1.2	- .27984	- .15795	+ .03647
	1.3	- .18737	- .07470	+ .09527
	1.4	- .10177	- .00538	+ .12490
	1.5	- .02210	+ .05073	+ .12951
	1.6	+ .05243	+ .09474	+ .11609
	1.7	+ .12243	+ .12800	+ .09177
	1.8	+ .18842	+ .15179	+ .06210
	1.9	+ .25084	+ .16184	+ .03078
	2.0	+ .31028	+ .17789	+ .00000

TABLE 6. THE EFFECT OF HARDENING ON THE RESIDUAL STRESS σ_θ/σ_o
(b/a = 2, ν =.3, n = 400)

O.S.	r/a	w = 0.00	w = 0.05	w = 0.10
30%	1.0	-0.53992	-0.50612	-0.47446
	1.1	- .28233	- .26457	- .24824
	1.2	- .07127	- .06644	- .06254
	1.3	+ .16389	+ .09831	+ .09216
	1.4	+ .09358	+ .08861	+ .08306
	1.5	+ .08530	+ .08082	+ .07575
	1.6	+ .07854	+ .07445	+ .06978
	1.7	+ .07297	+ .06919	+ .06485
	1.8	+ .06831	+ .06480	+ .06073
	1.9	+ .06437	+ .06108	+ .05725
	2.0	+ .06102	+ .05792	+ .05428
60%	1.0	-0.84138	-0.78984	-0.74017
	1.1	- .55776	- .52382	- .49114
	1.2	- .32513	- .30556	- .28679
	1.3	- .13036	- .12276	- .11559
	1.4	+ .03487	+ .03245	+ .02989
	1.5	+ .17610	+ .16532	+ .15461
	1.6	+ .29721	+ .27952	+ .26204
	1.7	+ .27635	+ .25991	+ .24364
	1.8	+ .25889	+ .24349	+ .22825
	1.9	+ .24414	+ .22962	+ .21524
	2.0	+ .23155	+ .21778	+ .20414
100%	1.0	-0.97388	-0.91430	-0.85591
	1.1	- .67902	- .63781	- .59733
	1.2	- .43792	- .41165	- .38574
	1.3	- .23600	- .22219	- .20843
	1.4	- .06370	- .06050	- .05708
	1.5	+ .08531	+ .07938	+ .07388
	1.6	+ .21530	+ .20147	+ .18825
	1.7	+ .32918	+ .30854	+ .28866
	1.8	+ .42906	+ .40260	+ .37700
	1.9	+ .51634	+ .48518	+ .45473
	2.0	+ .59296	+ .55752	+ .52301

TABLE 7. THE EFFECT OF HARDENING ON THE RESIDUAL STRESS σ_r/σ_o
($b/a = 2$, $\nu = .3$, $n = 400$)

O.S.	r/a	w = 0.00	w = 0.05	w = 0.10
30%	1.0	0.00000	0.00000	-0.00000
	1.1	-0.03684	- .03467	- .03250
	1.2	-0.04812	- .04531	- .04249
	1.3	-0.04287	- .04039	- .03788
	1.4	-0.03255	- .03070	- .02879
	1.5	- .02427	- .02290	- .02149
	1.6	- .01752	- .01654	- .01551
	1.7	- .01194	- .01128	- .01057
	1.8	- .00728	- .00688	- .00645
	1.9	- .00335	- .00317	- .00297
	2.0	0.00000	0.00000	-0.00000
60%	1.0	0.00000	0.00000	-0.00000
	1.1	-0.06302	- .05919	- .05544
	1.2	- .09415	- .08844	- .08286
	1.3	- .10413	- .09784	- .09169
	1.4	- .09986	- .09386	- .08799
	1.5	- .08598	- .08084	- .07580
	1.6	- .06566	- .06174	- .05790
	1.7	- .04480	- .04213	- .03951
	1.8	- .02735	- .02572	- .02411
	1.9	- .01259	- .01184	- .01110
	2.0	- .00000	0.00000	-0.00000
100%	1.0	.00000	0.00000	-0.00000
	1.1	-0.07453	-0.06991	- .06543
	1.2	- .11444	-0.10736	- .10050
	1.3	- .13125	-0.12316	- .11530
	1.4	- .13235	-0.12421	- .11631
	1.5	- .12262	-0.11511	- .10781
	1.6	- .10541	0.09898	- .09272
	1.7	- .08307	- .07803	- .07312
	1.8	- .05728	- .05382	- .05045
	1.9	.02929	- .02753	- .02582
	2.0	0.00000	0.00000	0.00000

TABLE 8. THE EFFECT OF HARDENING ON THE RESIDUAL STRESS σ_z/σ_o
($b/a = 2$, $\nu = .3$, $n = 400$)

O.S.	r/a	w = 0.00	w = 0.05	w = 0.10
30%	1.0	-0.15819	-0.14544	-0.13389
	1.1	- .08015	- .07403	- .06852
	1.2	- .02272	- .02097	- .01953
	1.3	+ .01831	+ .01737	+ .01628
	1.4	+ .01831	+ .01737	+ .01628
	1.5	+ .01831	+ .01737	+ .01628
	1.6	+ .01831	+ .01737	+ .01628
	1.7	+ .01831	+ .01737	+ .01628
	1.8	+ .01831	+ .01737	+ .01628
	1.9	+ .01831	+ .01737	+ .01628
	2.0	+ .01831	+ .01737	+ .01628
60%	1.0	-0.28532	- .25926	-0.23525
	1.1	- .18422	- .16735	- .15185
	1.2	- .10266	- .09318	- .08450
	1.3	- .03886	- .03494	- .03142
	1.4	+ .00946	+ .00946	+ .00931
	1.5	+ .00447	+ .04219	+ .03959
	1.6	+ .06946	+ .06533	+ .06124
	1.7	+ .06946	+ .06533	+ .06124
	1.8	+ .06946	+ .06533	+ .06124
	1.9	+ .06946	+ .06533	+ .06124
	2.0	+ .06946	+ .06533	+ .06124
100%	1.0	-0.36683	-0.33011	-0.29555
	1.1	- .25538	- .22763	- .20188
	1.2	- .15795	-0.13872	- .12109
	1.3	- .07470	-0.06310	- .05265
	1.4	- .00538	-0.00025	+0.00416
	1.5	+ .05073	+0.05065	- .05022
	1.6	+ .09474	+0.09068	- .08655
	1.7	+ .12800	+0.12107	- .11426
	1.8	+ .15197	+0.14312	- .13449
	1.9	+ .16814	+0.15811	- .14835
	2.0	+ .17789	+0.16726	+ .15690

DYNAMIC GUN TUBE BENDING ANALYSIS

Richard A. Lee, Jonathan F. Kring, and Dana S. Charles
US Army Tank-Automotive Command, Warren, Michigan

ABSTRACT. A simulation is presented of a gun barrel and its support at the trunnion. The simulation was programmed on an EAI 781 hybrid computer. from a magnetic field test tape. Errors due to dynamic gun tube bending are presented.

OBJECTIVE. Our objective is to evaluate the error due to gun tube flexure introduced from vehicle motions while firing-on-the-move. The analysis done will be applicable to the dynamic bending of any beam-like structure.

INTRODUCTION. In recent years there has been an increased emphasis on firing a combat vehicle's main weapon while the vehicle was moving. This has been called "firing-on-the-move" (FOM). Stabilization systems were added to vehicles that were designed to perform accurate stationary firing with the idea that stabilizing the gun in elevation and azimuth would allow the vehicle to perform accurate firing while moving. However, this was not the case. Errors occurred while firing-on-the-move that are not significant when firing from a stationary vehicle. Some of these errors are the horizontal and vertical vehicle velocities, stabilization errors, combined pitching and rolling motions, and gun tube flexure. This report is concerned with evaluating the error due to gun tube flexures that are introduced from vehicle motions.

A gun tube can bend or take non-uniform shape due to disturbances or phenomena that are not vehicle introduced. These can be caused from firing the gun or from sunlight heating one side of the gun tube. These errors are not included in this simulation. The static or quasi-static error caused from thermal gradients in the tube is corrected for in current vehicles with a muzzle reference system. This system has a small mirror mounted on the muzzle end of the tube. A light beam is reflected off the mirror to align the sight with the tube muzzle. This system performs very well for these quasi-static corrections but cannot be used for dynamic tube leveling on the moving vehicle.

It is extremely difficult to measure the dynamic bending of a gun tube in a vehicle traversing cross-country terrain. A one-mil angular bending error in a tube will produce approximately a five-foot error firing at a target 1600 meters away. This is a significant error and one must measure the tube bending to considerably less than one mil. To give some indication of the angular size this corresponds, i.e., the angle a golf ball subtends a football field away is about 0.3 mils.

The derivation of the equations that were programmed on the computer is given in Appendix A. The equations and computer programs are in a general form and are applicable to any symmetrical gun tube. Realistic dimensions and material data were chosen for the analysis. The gun tube was separated into eighteen uniform elements, with each finite element having uniform characteristics over its length. One thing to note in the equations is that the gun

tube rigidity increases as the fourth power of the diameter. Thus, larger caliber gun tubes are considerably more rigid that small ones.

The model was implemented and solved on a hybrid computer. The gun was modeled on an analog computer and forcing functions were supplied by the digital computer via D/A. The vehicle ride was obtained from magnetic tape recordings of field data. These rides were digitized and stored in the digital computer for use as the gun forcing functions. The input into the gun was only in the vertical direction; consequently, the error data presented are for the gun tube flexure in a vertical plane. In reality, there is some flexing in the horizontal direction but that is not considered here.

DISCUSSION. The purpose of this study was to measure by computer techniques the muzzle error at a mile range of a gun barrel subjected to dynamic inputs at the trunnion. To simulate the gun tube, it was divided into sections to analyze its response using Euler's equation for the flexure of a beam.

The equations of motion as applied to the sectioned tube are as follows:

1. Basic equation for gun barrel without support:

$$M_L \ddot{Y}_L = \frac{2(EI)_L}{X_L^3} * \left(Y_{L+1} - 2Y_L + Y_{L-1}\right) - \frac{(EI)_{L+1}}{X_L X_{L+1}^2} * \left(Y_{L+2} - 2Y_{L+1} + Y_L\right)$$

$$- \frac{(EI)_{L-1}}{X_L X_{L-1}^2} * \left(Y_L - 2Y_{L-1} + Y_{L-2}\right)$$

WHERE: L = Subscript to designate the section
M = Mass
E = Modulus of elasticity
I = Moment of inertia
X = Length
Y = Vertical Displacement
$\ddot{Y}$ = Vertical acceleration

2. Basic equations for gun barrel with support acting on 1st, 2nd, and 11th sections:

a. 1st Section

$$M_1 \ddot{Y}_1 = -\frac{(EI)_2}{X_1 X_2^2} * \left(Y_3 - 2Y_2 + Y_1\right) - K_s * Y_1$$

WHERE: K_s = Spring constant of support (12,200 lbs/in)

b. 2nd Section

$$M_2\ddot{Y}_2 = \frac{2(EI)_2}{X_2^3} * (Y_3 - 2Y_2 + Y_1) - \frac{(EI)_3}{X_2X_3^2} * (Y_4 - 2Y_3 + Y_2) - K_s * Y_2$$

c. 11th Section

$$M_{11}\ddot{Y}_{11} = \frac{2(EI)_{11}}{X_{11}^3} * (Y_{12} - 2Y_{11} + Y_{10}) - \frac{(EI)_{12}}{X_{11}X_{12}^2} * (Y_{13} - 2Y_{12} + Y_{11})$$

$$- \frac{(EI)_{10}}{X_{11}X_{10}^2} * (Y_{11} - 2Y_{10} + Y_9) - K_s * Y_{11}$$

NOTE: A detailed description in the development of the equations of motion is noted in Appendix A.

The equations of motion were simulated on the analog portion of the hybrid computer. A typical analog circuit that generates sections 1, 2, and 3 is shown in Figure 2.

The muzzle error due to the flexure of the gun tube has two components, one based on the bending displacement and one based on the rate of change of that bending. We refer to these as angular error and velocity error and their sum as total error. If the tube were completely rigid, this error would be zero. Bending from gravity occurs, but since the error from this is well-known and compensated for, it is removed prior to a simulation run.

At the start of a simulation run, the static error due to analog noise was measured and removed. The model was run 100 times slower than real time and 20 sample measurements of the error each second were taken to avoid interference from the natural frequency of the tube, which was approximately 500 Hz. Seven and one-half seconds of each ride was studied to obtain a representative sampling of the error. The vertical displacements of the trunnion were inputted dynamically, and the resulting error measurements saved in computer storage for processing after the run. Refer to Appendix B for details on the trunnion inputs.

Six different vehicle rides were studied, each with and without the additional support. For each of the types of error collected, distributions were determined with regard to the gun aiming at a target 1600 meters distant. The range of error was divided into classes and histograms of the frequency that the error fell into each class were made. Time histories of the total error were also plotted.

Hit probability curves were generated based on each type of error. For ten selected target sizes the percentage of hits given, the measured errors were calculated. A smooth curve was fit through the ten target size points. Since an enemy tank would be approximately 2.5 meters high, hit probabilities for this particular target size are displayed in Figure 3.

A major concern was the relative contribution of the velocity error, as a compensating system for this does not yet exist. For all the rides studied, the velocity error averaged 3.2 percent of the total error without the support and 15.6 percent with the support. In the latter case, the increase is probably due to the higher total accuracy of the system with the extra support. However, in both cases, the contribution is minor. These results are displayed in Figure 4.

By referring to Figure 3, the effect of the additional support can be easily seen. For the 2.5 meter target, hit probability increased from an average of 12.9 percent to an average of 79.3 percent. This large improvement in performance shows that if firing on the move is desired, additional rigidity of the gun barrel will greatly reduce the error caused by the dynamic motion of the vehicle.

CONCLUSIONS. To perform accurate firing on the move, the gun tube flexure due to vehicle motion must be considered.

For the rides and gun used in this simulation, traversing Course 4 at 7 mph resulted in the gun being on a 2.5 meter target 1600 meters away less than 10 percent of the time. This error was due only to gun tube bending--the sight and breech end of the gun were pointing at the center of the target.

Providing a rigid support for the gun tube resulted in an increase in the hit probability for the "bending" condition of a factor greater than 7.

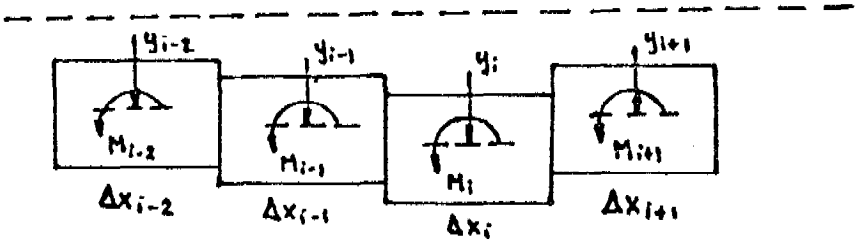
Providing a rigid support for a gun tube will significantly decrease the bending error.

The tube bending error is due almost entirely to the tube's angular position. The error due to muzzle velocity was insignificant.

For some conditions the gun tube bending error can be the most significant error occurring while firing on the move.

APPENDIX A

Equations of Motion Derivation



Euler's equation for the flexure of a beam:

$$\frac{d^2}{dX^2}\left(EI\,\frac{d^2Y}{dX^2}\right) = \frac{w}{g}\,\frac{d^2Y}{dt^2} \tag{A-1}$$

The slope across an element i is given by:

$$\frac{dY_i}{dX_i} = \frac{(Y_{i-1} - Y_i)}{\Delta X_i} \tag{A-2}$$

Y_i is the vertical distance moved for element i from an arbitrary reference line.

The second derivative or rate of change of slope is the difference between the left and right faces of the element.

i.e.

$$\frac{d^2Y_i}{dX_i^2} = \frac{Y'_{i-1} - Y'_{i+1}}{\Delta X_i^2} \tag{A-3}$$

Where the prime denotes derivative

Then:

$$\frac{d^2Y_i}{dX_i^2} = \frac{Y_{i-1} - 2Y_i + Y_{i+1}}{\Delta X_i^2} \tag{A-4}$$

The bending moment at each element is given by:

$$M_i = (EI)_i \frac{d^2Y_i}{dX_i^2} \tag{A-5}$$

Then for EI constant over element i the bending moment of element i is given by:

$$M_i = \frac{(EI)_i (Y_{i-1} - 2Y_i + Y_{i+1})}{\Delta X_i^2} \tag{A-6}$$

Euler's equation states:

$$\frac{d^2M}{dX^2} = \frac{w}{g} \frac{d^2Y}{dt^2} \tag{A-7}$$

To take the derivative of the bending moment EI must be constant over the element.

The rate of change of bending movement over the element is given by:

$$\frac{dM_i}{dX_i} = \frac{M_{i-1} - M_i}{\Delta X_i} \tag{A-8}$$

The second derivative is then given by:

$$\frac{d^2M_i}{dX_i^2} = \frac{M'_{i-1} - M'_{i+1}}{\Delta X_i} \tag{A-9}$$

Then:

$$\frac{d^2M_i}{dX_i^2} = \frac{M_{i-1} - 2M_i + M_{i+1}}{\Delta X_i^2} \tag{A-10}$$

Writing each moment equation

$$M_{i-1} = \frac{(EI)_{i-1}\,(Y_{i-2} - 2Y_{i-1} + Y_i)}{\Delta X_{i-1}^2} \tag{A-11}$$

$$M_i = \frac{(EI)_i\,(Y_{i-1} - 2Y_i = Y_{i+1})}{\Delta X_i^2} \tag{A-12}$$

$$M_{i+1} = \frac{(EI)_{i+1}\,(Y_i - 2Y_{i+1} + Y_{i+2})}{X_{i+1}^2} \tag{A-13}$$

The mass of each element is the mass per unit length times the length of the element.

$$M_i = \frac{w}{g}\ _i\ \Delta X_i \tag{A-14}$$

Euler's equation is then written as:

$$M_i \frac{d^2Y_i}{dt^2} = \frac{(EI)_{i-1}\,(Y_{i-2} - 2Y_{i-1} + Y_i)}{\Delta X_i\ \Delta X_{i-1}^2} - \frac{2(EI)_i\,(Y_{i-1} - 2Y_i + Y_{i+1})}{\Delta X_i^3} + \frac{(EI)_{i+1}\,(Y_i - 2Y_{i+1} + Y_{i+2})}{\Delta X_i\ \Delta X_{i+1}^2} \tag{A-15}$$

Evaluating the end conditions:

There is no bending moment on the end element

$$\frac{d^2M_{end}}{dX_{end}^2} = \frac{M_{i+1}}{\Delta X_i^2} \tag{A-16}$$

Second from end

$$\frac{d^2M_{end+1}}{dX_{end+1}^2} = \frac{-2M_i + M_{i+1}}{\Delta X_i^2} \tag{A-17}$$

The opposite end

$$\frac{d^2M_{end}}{dX^2_{end}} = \frac{M_{i-1}}{\Delta X_i^2} \tag{A-18}$$

and

$$\frac{d^2M_{end-1}}{dX^2_{end-1}} = \frac{M_{i-1} - 2M_i}{\Delta X_i^2} \tag{A-19}$$

APPENDIX B

Trunnion Movement Generation

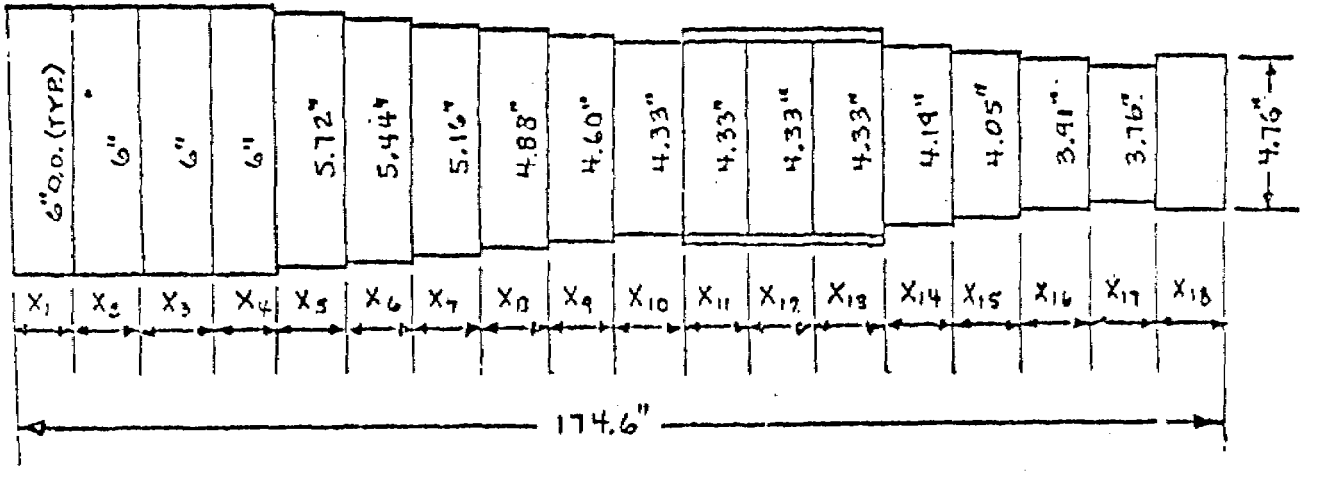
A 14-channel magnetic tape of analog field data from tests conducted at Fort Knox, Kentucky was used to provide center of gravity, vertical, pitch, and roll acceleration signals. These were combined to produce a vertical trunnion acceleration signal which was digitally sampled at 100 times per second, using a high-speed digital-to-analog converter, and stored for later use.

The analog simulation was run 100 times slower than real time to obtain a more accurate simulation. This also allowed us to observe high-frequency gun tube movements which would have been difficult to follow with the naked eye.

The acceleration signal was digitally integrated twice to provide a displacement signal which was applied to the trunnion during simulation. The displacement signal was inputted into the feedback inverter of the supports (see Figure 2), which caused the displacement of the gun to match the driving signal.

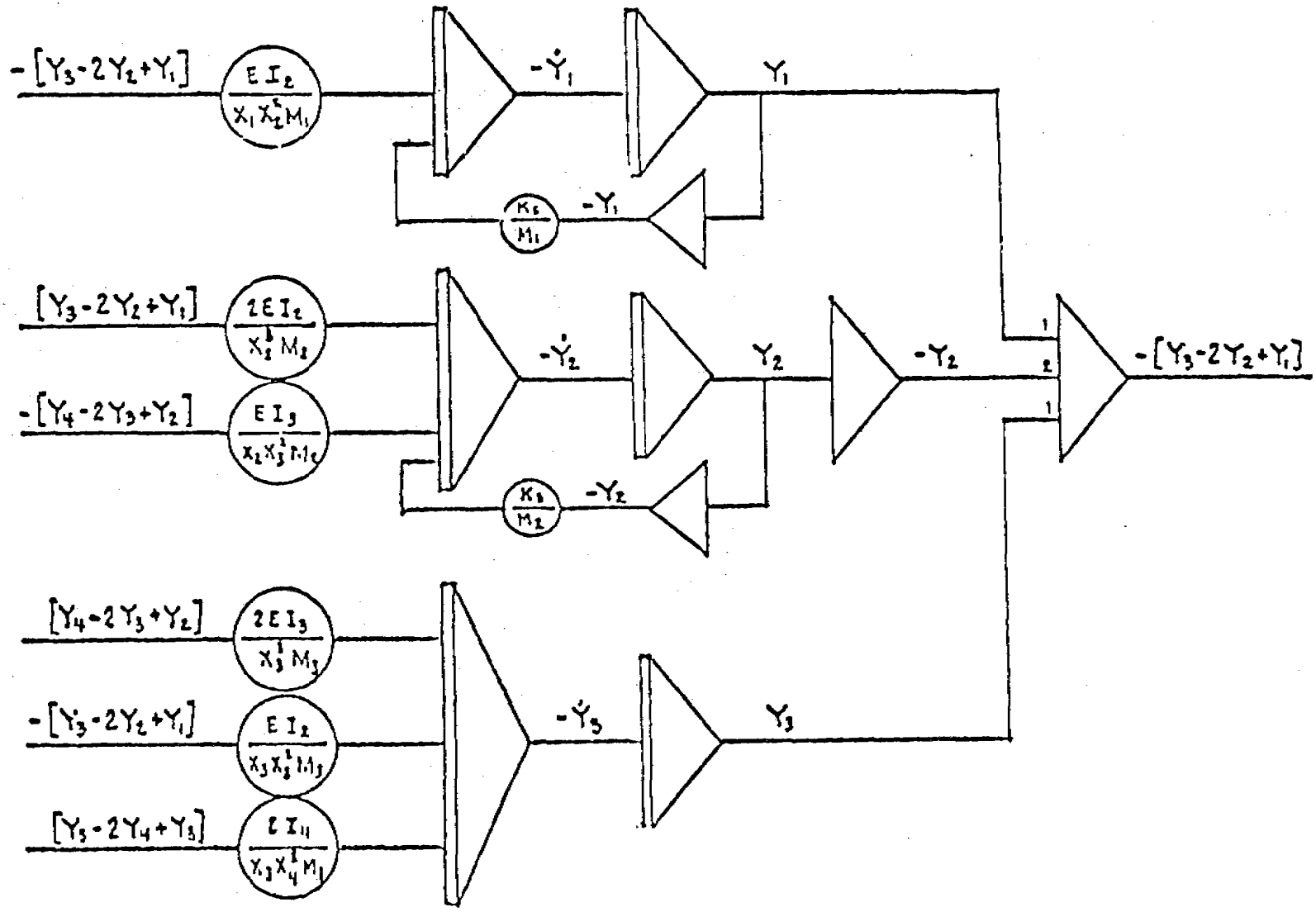
Figure 5 shows a typical displacement signal used as input to the trunnion.

Figure 1



SIMULATED GUN BARREL

Figure 2



TYPICAL ANALOG CIRCUIT

Figure 3

HIT PROBABILITY FOR 2.5 METER TARGET

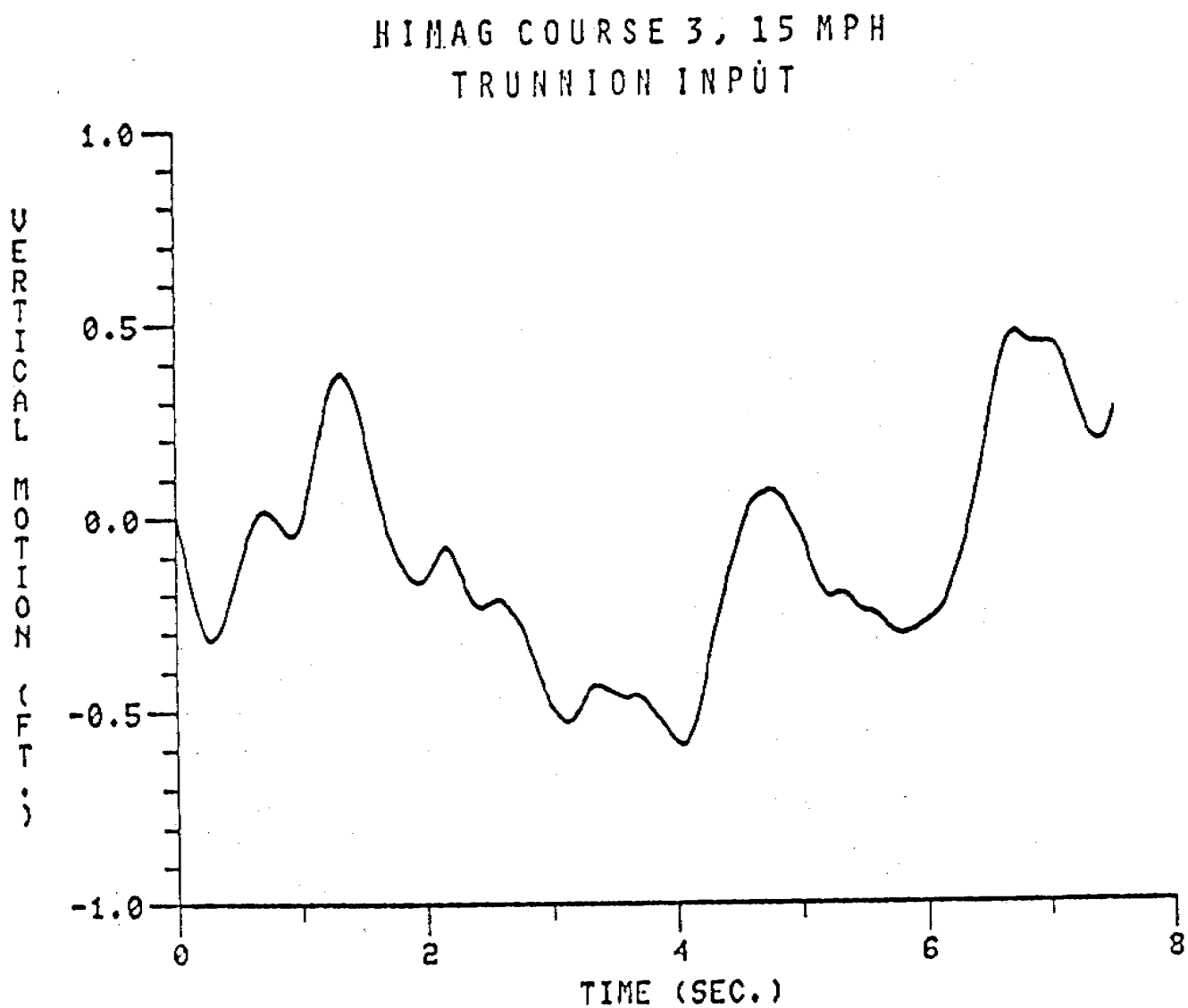
COURSE 5, 03 MPH WITHOUT SUPPORT	11.5 PERCENT
COURSE 5, 03 MPH WITH SUPPORT	81.6 PERCENT
COURSE 4, 07 MPH WITHOUT SUPPORT	7.9 PERCENT
COURSE 4, 07 MPH WITH SUPPORT	57.5 PERCENT
COURSE 3, 15 MPH WITHOUT SUPPORT	8.9 PERCENT
COURSE 3, 15 MPH WITH SUPPORT	66.9 PERCENT
COURSE 2, 30 MPH WITHOUT SUPPORT	21.4 PERCENT
COURSE 2, 30 MPH WITH SUPPORT	85.4 PERCENT
COURSE 2, 25 MPH WITHOUT SUPPORT	14.2 PERCENT
COURSE 2, 25 MPH WITH SUPPORT	92.1 PERCENT
COURSE 2, 10 MPH WITHOUT SUPPORT	13.5 PERCENT
COURSE 2, 10 MPH WITH SUPPORT	92.0 PERCENT

Figure 4

CONTRIBUTION OF VELOCITY ERROR

COURSE 5, 03 MPH WITHOUT SUPPORT	2.7 PERCENT
COURSE 5, 03 MPH WITH SUPPORT	15.5 PERCENT
COURSE 4, 07 MPH WITHOUT SUPPORT	2.7 PERCENT
COURSE 4, 07 MPH WITH SUPPORT	13.3 PERCENT
COURSE 3, 15 MPH WITHOUT SUPPORT	3.2 PERCENT
COURSE 3, 15 MPH WITH SUPPORT	16.5 PERCENT
COURSE 2, 30 MPH WITHOUT SUPPORT	3.9 PERCENT
COURSE 2, 30 MPH WITH SUPPORT	15.1 PERCENT
COURSE 2, 25 MPH WITHOUT SUPPORT	3.4 PERCENT
COURSE 2, 25 MPH WITH SUPPORT	15.5 PERCENT
COURSE 2, 10 MPH WITHOUT SUPPORT	3.1 PERCENT
COURSE 2, 10 MPH WITH SUPPORT	18.0 PERCENT

Figure 5


HIMAG COURSE 3, 15 MPH
TRUNNION INPUT
VERTICAL MOTION (FT.)
1.0
0.5
0.0
-0.5
-1.0
0
2
4
6
8
TIME (SEC.)

FINITE ELEMENT MODELING OF THE VULNERABILITY OF AN M-15 LAND MINE USING AN EXPLICIT INTEGRATION SCHEME

Frederick H. Gregory
U.S. Army Ballistic Research Laboratory
U.S. Army Armament Research and Development Command
Aberdeen Proving Ground, Maryland 21005

ABSTRACT. A finite element model of the body of an M-15 land mine has been formulated using an axisymmetric two-dimensional mesh with both rigid and nonlinear spring base support boundary conditions to simulate the soil. This model has been analyzed with the ADINA finite element structural response code. An analysis of various implicit/explicit time integration schemes showed that the explicit central difference time marching method gave the best solution in terms of displacements and stresses. A numerical study was conducted to determine the optimum time for which convergence of the solution was obtained.

The two basic materials of which the mine is composed, steel and high explosive, were assumed to have nonlinear constitutive material models. The steel case was found to be markedly inhomogeneous via 1-D tensile tests of specimens cut from various areas of the same. This material was modeled with a bilinear stress-strain curve, von Mises yield condition, and kinematic hardening rule. A tension cut-off elastic-plastic model of the explosive which employed a bulk modulus versus volume strain relation, was derived from a Mie-Grüneisen shock wave equation of state. This model allowed a tension cut-off plane to form in a direction normal to the principal tensile stress whenever the strain initially exceeded 0.1% in tension.

Solution of this problem out to 2 msec of real time required about 4 hours of cpu time on the CDC 7600 computer for a transient shock load imposed on the top and sides of the mine. Failure of the mine case was predicted, based on a comparison of the value of the three-dimensional second invariant of plastic strain with that of the one-dimensional value measured in the tensile tests.

1. INTRODUCTION. This paper describes the response of an antitank mine to a transient blast load. The rationale for this analysis is the need to develop a remote, expeditious means of clearing a path through an enemy mine field. A technique has been suggested (Ref. 1) by which a relatively large transient pressure is delivered to the surface of the earth by means of explosives. The object of this study was to determine the extent of structural damage to an M-15 mine body from a given level of blast wave amplitude and shape. The principal kill mechanism is to be a serious distortion or rupture of the mine body. It was not intended that advantage be taken of some nuance of component design such as fuze initiation or pressure plate removal, etc. Some considerations such as the latter have been examined in Ref. 2.

The paper is divided into four major areas as follows: (a) problem definition, (b) determination of material properties and selection of failure criteria, (c) finite element model description and calculations, and (d) analysis of predicted response.

2. PROBLEM DEFINITION.

A. M-15 Antitank Mine Description. The M-15 mine has a cylindrical steel body with a primary fuze well in the center of the top and two secondary fuze wells, one on the side and one on the bottom. The center of the top of the mine has a depressed area which houses the pressure plate assembly. Drawings of the mine are shown in Figures 1 and 2. The mine has a nominal diameter of 32.13 cm, height of 9.88 cm, and weighs 14.3 kg.

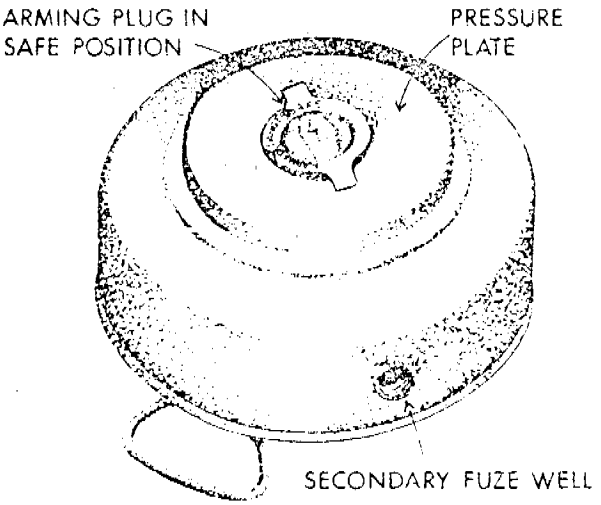
The mine body is made essentially of two pieces of WD-1010 steel which are joined at the lower periphery by a 360° crimp. The upper part of the mine body is formed by a deep drawing operation which results in very inhomogeneous materials properties. The central cavity shown in the lower halves of Figures 1 and 2 is filled with 10 kilograms of composition B explosive. This filling operation is done with the explosive in a molten state.

The normal method of activation of the fuze is by means of force applied to the pressure plate (1250 to 2000 newtons) which in turn is transferred to the belleville springs. At a certain deflection, the belleville springs snap through, driving the firing pin into the detonator. The explosion of the detonator activates the tetryl booster which in turn detonates the primary composition B charge. There are two auxiliary fuze wells on the M-15 mine which give it an anti-disturbance capability (See Figure 2).

B. Guidelines for the Numerical Model. In keeping with the philosophy of identifying a failure mechanism which is as general as possible and is not dependent upon some specific design feature, the pressure plate, fuze, and belleville springs were omitted from the finite element model. This was done in consonance with the previously stated guideline of not identifying failures of the fuze components. The part of the mine which constitutes our model is shown in the lower part of Figure 2, not including the secondary fuzes and filling hole.

There are a large number of antitank mines, both foreign and of U.S. manufacture, which consist basically of a round thin metal body filled with explosive. This type of antitank mine constitutes a large part of the inventory of U.S. and Soviet mines. The component which is most distinctive is the fuze mechanism. There are a variety of radically different fuzes for these mines, different both in mechanical design and different in the selection of some particular signature of combat tanks which is required to activate the fuze train. Therefore, the numerical model adapted for the M-15 mine is representative of a basic, necessary component of a large class of both foreign and U.S. mines.

The auxiliary fuze wells were eliminated from the finite element model for two reasons. First, these fuze wells make the mine body more susceptible to damage due to stress concentrations which occur in the neighborhood of the joint between the secondary fuze and the mine body. Thus, the simplified model is more conservative in terms of the blast load required for mine defeat. Secondly, the inclusion of these wells in the finite element mesh would have necessitated the use of a three-dimensional (3-D) finite element model. The 3-D model would have required a very large increase in the amount of computing time used in obtaining the dynamic response of the structure. The four lobed



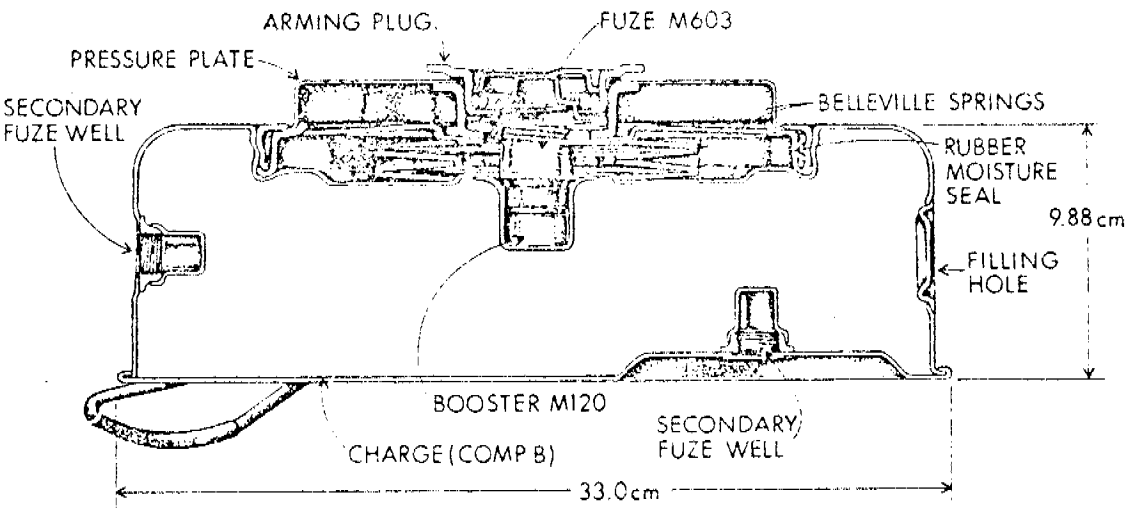


Figure 1. U.S. M-15 Antitank Mine

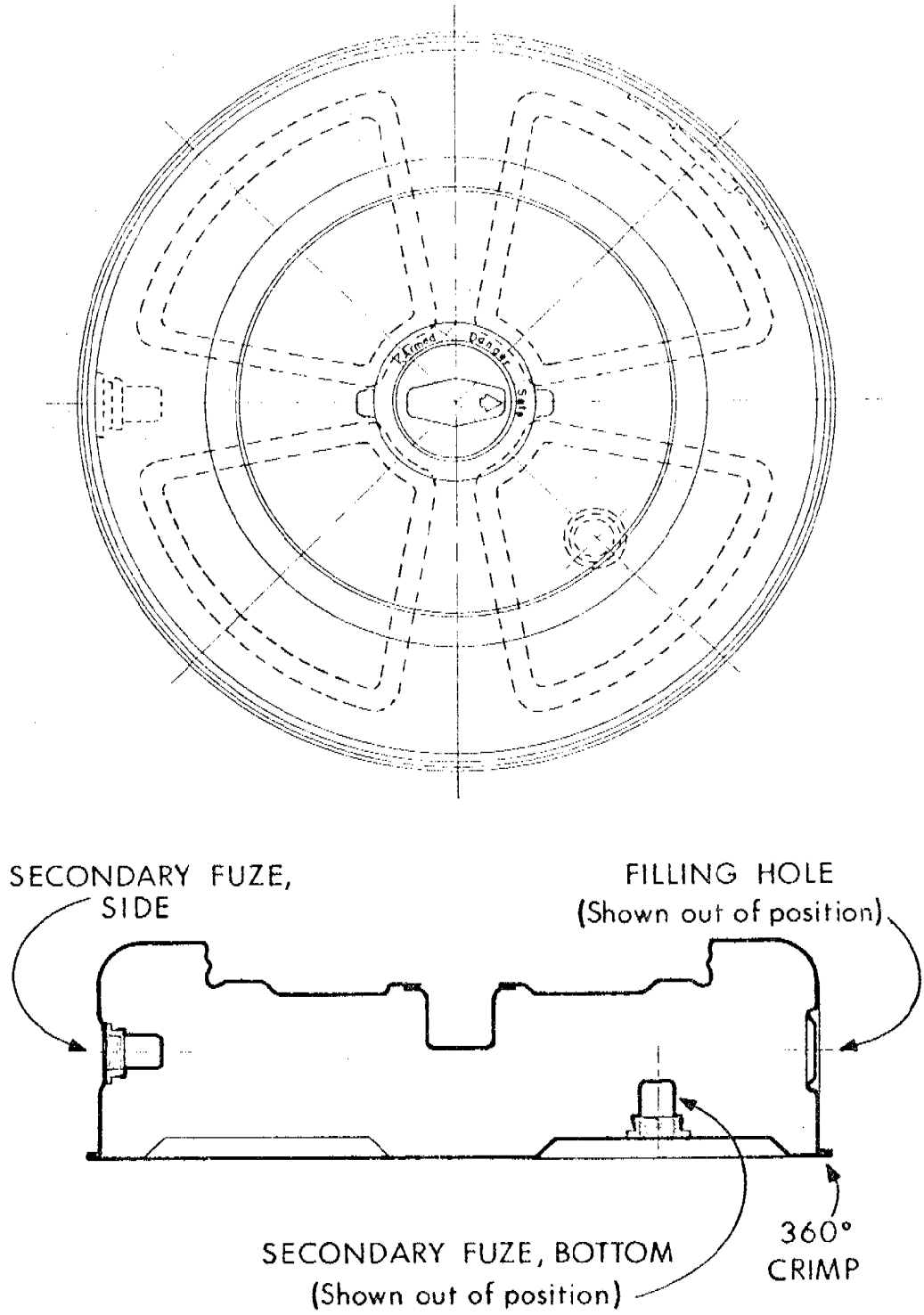


Figure 2. M-15 Mine Base Assembly

shaped dimples in the base of the mine body shown in Figure 2 were omitted for the same reasons. The result of these simplifications was a 2-D model in which we needed consider only a pie shaped section of the axisymmetric body.

C. Base Support and Surface Loading. When employed in the field, mines of the M-15 type are usually buried and covered with a shallow layer of soil for concealment. In a few cases, the mine may be placed on the surface and covered with grass, leaves, etc., for concealment. In either case, the mine will experience a transient pressure load on its top surface when the countermine explosive is detonated in the vicinity. For buried mines, the sides will experience a lesser pressure pulse, the magnitude of which will depend on several factors such as how well the soil is tamped, type of soil, and depth of burial. The base of the mine will pick up a load at a later time from that of the top surface loading. The magnitude and shape of this base support will depend upon the downward acceleration/movement of the mine and the dynamic properties of the soil medium.

The method used to simulate the soil support was by means of nonlinear, upward acting springs. The detailed description will be given in the next Section. This calculation will be referred to as Case A.

In order to compare the predicted structural response with experimental data presented in Reference 1, a second base support condition was used in which the motion of the base in the vertical direction was restrained. This support condition will be referred to as Case B.

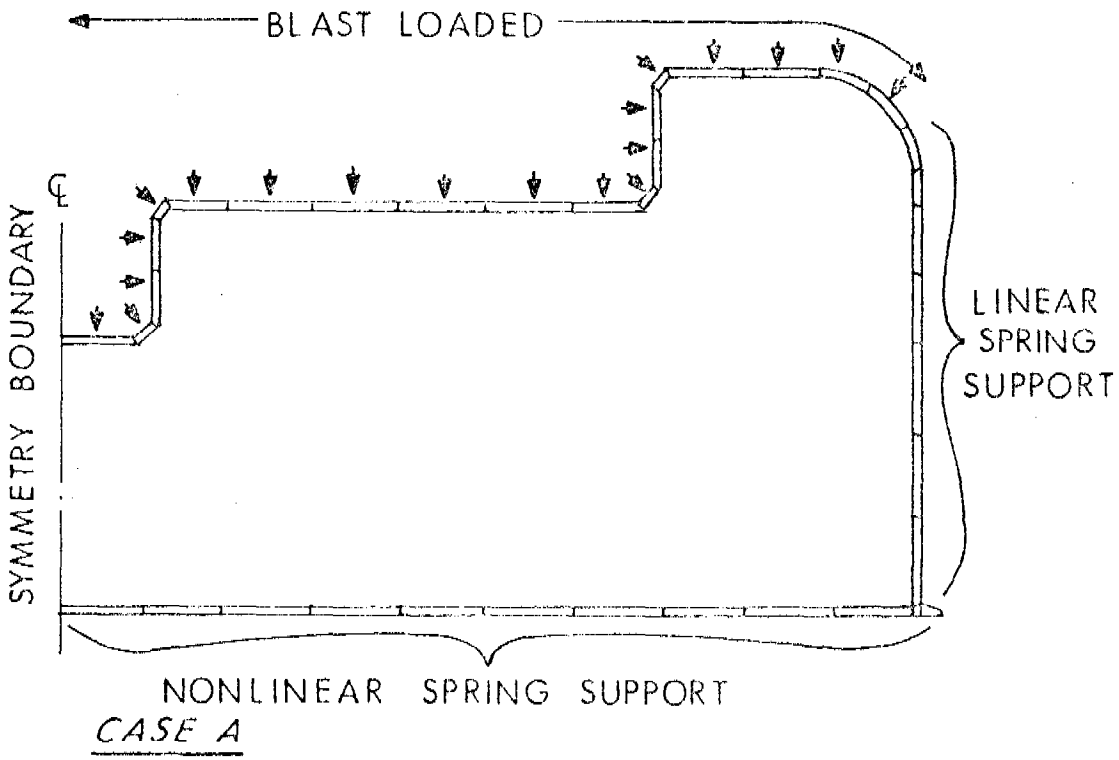
The boundary support configurations for Cases A and B are shown in Figure 3. Also shown there are the portions of the surface that were loaded. In Case A, the mine is simulated as being buried in soil up to its top surface; whereas, in Case B it is assumed sitting on a rigid surface.

Reference 1 describes some experiments conducted with mine clearance types of explosives. In these experiments, two types of U.S. mines were exposed to the resulting blast loading. The pressure pulse used in this paper was designed to simulate the peak pressure and impulse measured in these experiments. The peak pressure was 13.8 MPa and the impulse delivered was 6.5 kPa-sec. A decaying exponential function was fitted to these parameters resulting in the following equation

$$P(t) = 13.76\ e^{-2117t} \qquad (1)$$

A curve of this function varying in time is shown in Figure 4. All points on the surface of the mine indicated in Figure 3 were loaded with this transient load beginning at zero seconds.

3. MATERIAL PROPERTIES AND FAILURE CRITERIA. Material properties were required for the steel jacket, the composition B explosive filler, and the soil in which the mine is emplaced. Of these three, mechanical properties were measured only for the steel jacket. The data for the explosive and soil were taken from available publications. Failure criteria are developed for the steel jacket and the explosive.



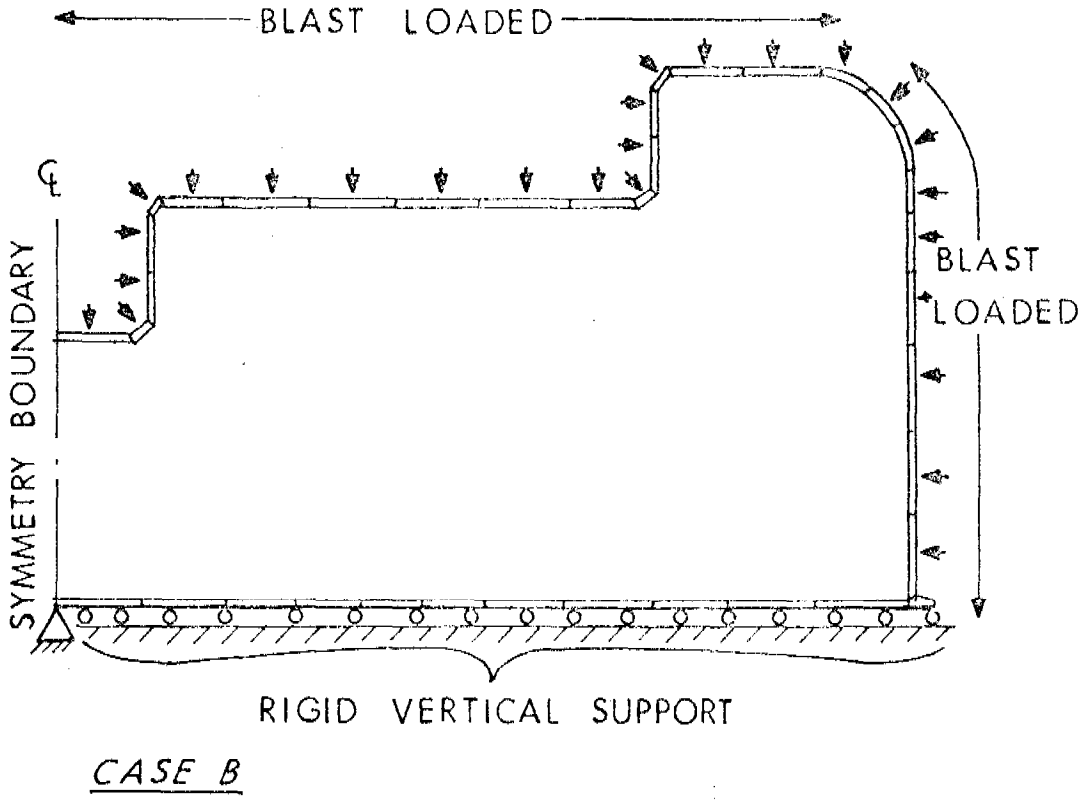


Figure 3. Base Support and Loaded Surfaces

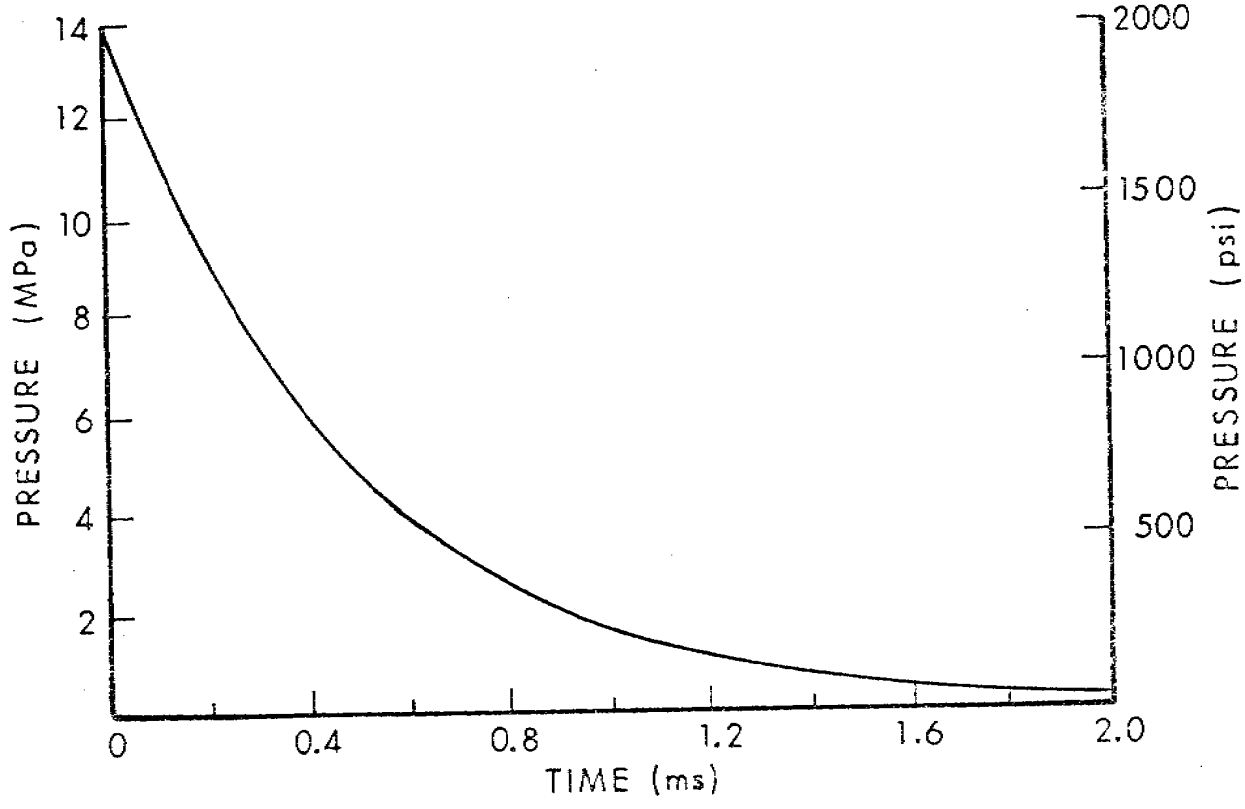


Figure 4. Shock Loading Function for M-15 Antitank Mine

A. WD-1010 Steel. The M-15 jacket is made of a medium strength steel alloy with a density of 7.80 g/cm^3 and a thickness of 0.94 mm. Six tensile specimens were cut from an inert training mine as shown in Figure 5. Two specimens were cut from each of the significant surfaces of the mine body. These specimens were machined with a large radius on the test section as shown in Figure 5(b). A biaxial strain gage was attached at the location of the minimum width. The specimens were then tested in an Instron Testing Machine. The stress-strain curves resulting are shown in Figures 6-8, plotted in pairs according to the location of the specimens on the mine surface. The stress-strain curves for the pairs of specimens are similar when comparing a curve with that of a mating specimen, but were surprisingly dissimilar when compared with specimens taken from different areas on the mine body. Of the three pairs of stress-strain results, the data for the top annular surface showed the most disparity. The other two sets of data indicate very closely matched properties. It is evident that the metal is work hardened in various areas by the stamping operation by which the upper part of the mine jacket is shaped.

As can be seen in the vertical section in Figure 2, the jacket is formed from two sheet steel blanks, which after being shaped are joined by a 360° crimp around the lower periphery of the mine. The bottom of the mine was assumed to have the same properties as the pressure plate well area because neither are deformed appreciably in the forming operation. The metal in the pressure plate well area exhibits properties typical of mild steel (Figure 6). There is a slight overshoot of the yield stress, a relatively flat, low modulus section followed by a large strain to failure.

Bilinear approximations to the stress-strain curves are also shown in Figures 6-8. These bilinear approximations were obtained by averaging the data for the individual specimens. The version of the ADINA (References 3 and 4) finite element code used in this analysis has a bilinear, elastic-plastic, von Mises yield condition, kinematic hardening, axisymmetric 2-D element which was used to model the steel jacket. A summary of the significant parameters is given in Table 1.

Because the steel in the mine jacket has an appreciable amount of ductility, it was desired to apply a failure criterion which included a measure of the deviatoric strain. The deviatoric strain is defined by the usual formula

$$\&_{ij} = \epsilon_{ij} - \epsilon_{kk}\delta_{ij}/3 \qquad (2)$$

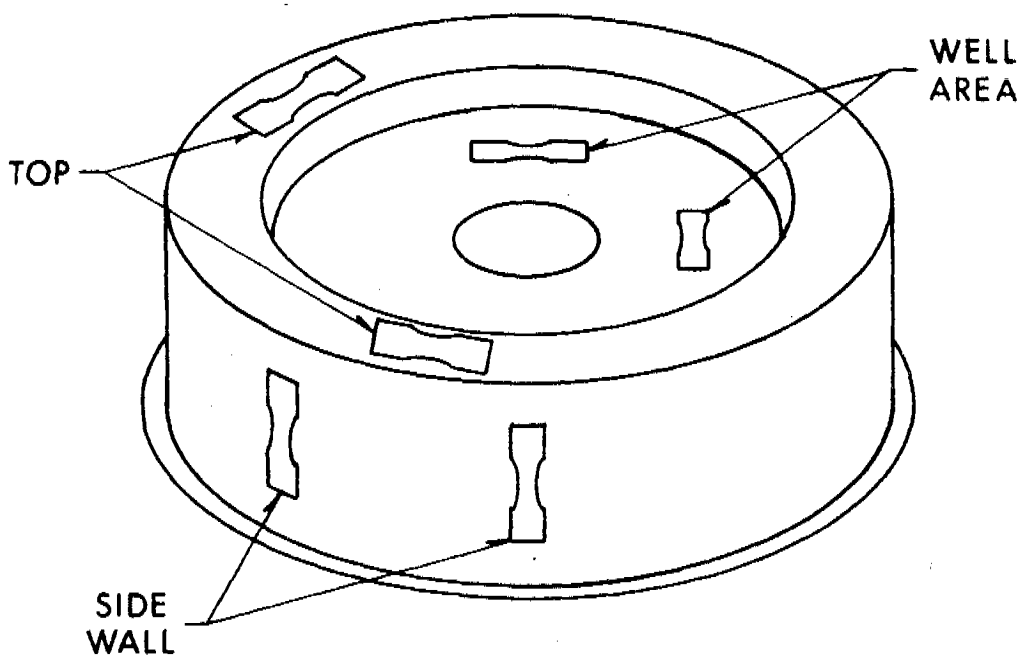
where

$\&_{ij}$ is the deviatoric strain component

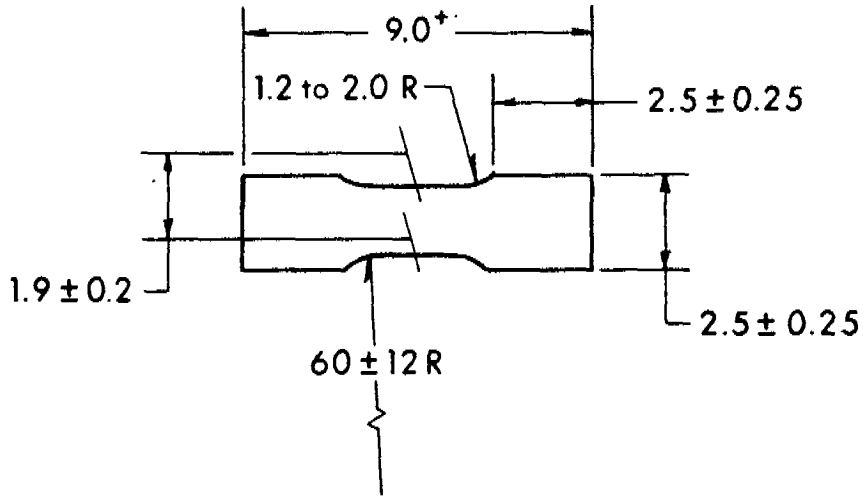
ϵ_{ij} is the total strain component

δ_{ij} is the Kronecker delta

and repeated indices imply a summation. The strain is assumed composed of an



(a) LOCATION OF SPECIMENS



(b) PREPARATION OF SPECIMEN DIMENSIONS (cm)

Figure 5. Tensile Stress-Strain Specimens for the M-15 Mine Base Assembly

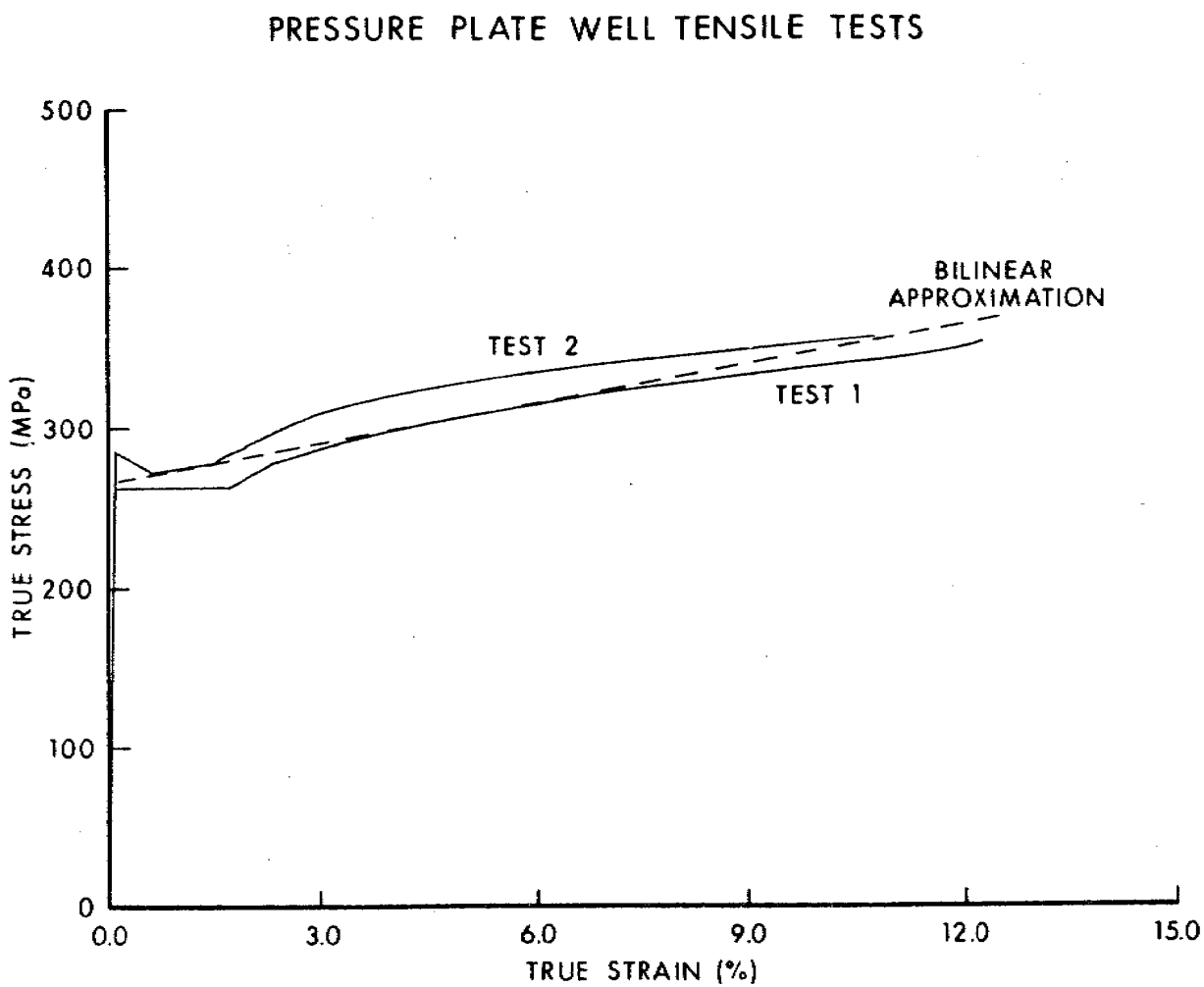


Figure 6. Stress-Strain Curves for Pressure Plate Well Tensile Specimens

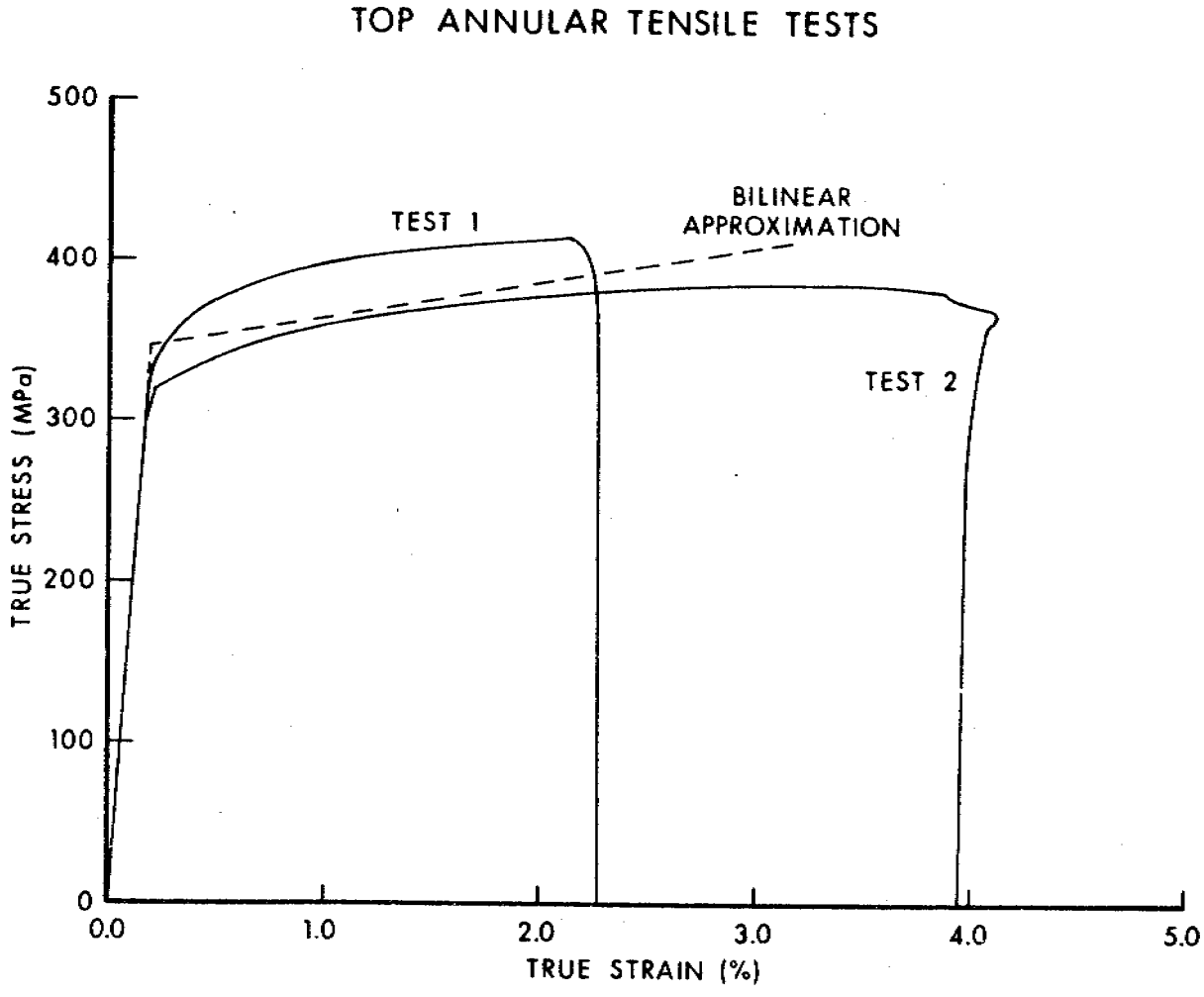


Figure 7. Stress-Strain Curves for Top Annular Tensile Specimens

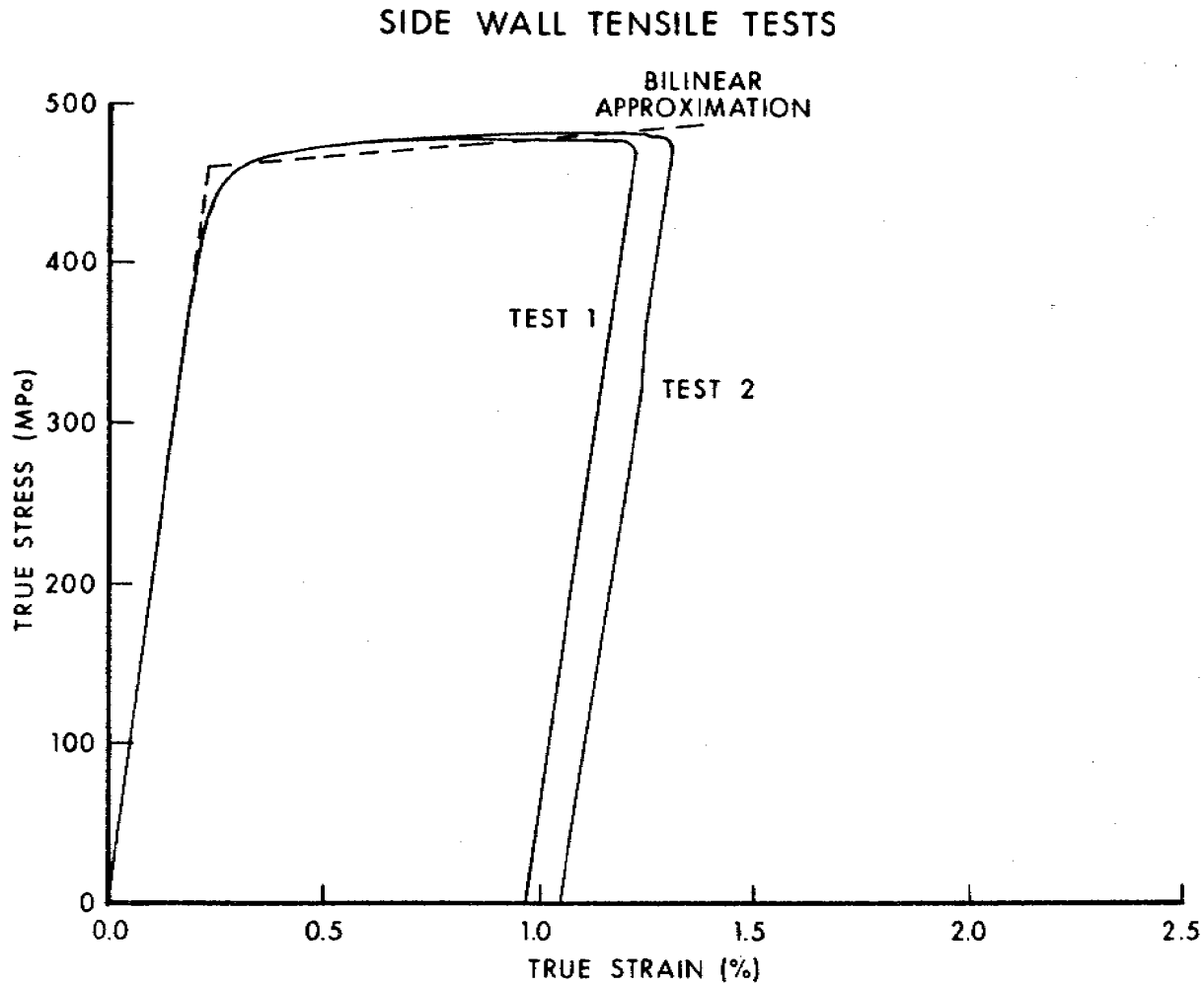


Figure 8. Stress-Strain Curves for Sidewall Tensile Specimens

TABLE 1. WD-1010-T4 SHEET STEEL STRESS-STRAIN DATA

Specimen Location	Specimen Number	ν^*	σ_y (MPa)	ϵ_y (%)	E_y (GPa)	σ_f^{**} (MPa)	ϵ_f (%)	E_t (GPa)	$I_2\left(\frac{\epsilon^P}{1-D}\right)_f$
Pressure plate well	1	‡	257	.132	195	351	12.3	.77	
Pressure plate well	2	.278	277	.126	219	370	10.8	.87	
Average	1+2	—	267	.129	207	360	11.55	.82	9.7×10^{-3}
Top annular	1	‡	365	.171	213	424	2.31	2.76	
Top annular	2	.257	331	.172	193	403	4.00	1.88	
Average	1+2	—	348	.171	203	413	3.16	2.32	6.55×10^{-4}
Side wall	1	.273	460	.220	209	482	1.22	2.20	
Side wall	2	.286	462	.230	201	485	1.31	2.13	
Average	1+2	.280	461	.225	205	484	1.265	2.165	7.94×10^{-5}

* For definition of symbols, see the text.

** Determined by extrapolation of stress to failure strain on bilinear fit to data.

‡ Only an axial strain gage was used in these two tests.

elastic and a plastic component such that

$$\epsilon_{ij} = \epsilon^{E}_{ij} + \epsilon^{P}_{ij}. \tag{3}$$

The plastic component of the spherical strain is assumed zero, i.e.,

$$\epsilon^{P}_{kk} = 0. \tag{4}$$

Upon substitution of Equation (3) into Equation (2) and using an equation similar to Equation (3) for the total deviatoric strain, $\&_{ij}$, and further using

$$\&^{E}_{ij} = \epsilon^{E}_{ij} - \epsilon^{E}_{kk}\delta_{ij}/3,$$

and Equation (4), one finds the result

$$\&^{P}_{ij} = \epsilon^{P}_{ij}. \tag{5}$$

The criterion which was selected to predict failure of the steel casing material was the second invariant of plastic deviatoric strain, $I_2(\&^P)$. This quantity is defined by

$$I_2(\&^P) = \tfrac{1}{2}\,\&^{P}_{ij}\&^{P}_{ij} \tag{6}$$

where summation is implied.

In the uniaxial tension test where the load is applied in the z-direction, we have

$$\&^{P}_{xx} = \&^{P}_{yy}$$

such that

$$\&^{P}_{kk} = 0 = \&^{P}_{zz} + 2\&^{P}_{xx}$$

or

$$\&^{P}_{xx} = -\tfrac{1}{2}\,\&^{P}_{zz}. \tag{7}$$

Thus, for 1-D tension test we have

$$I_2\left(\&^{P}_{1\text{-}D}\right) = 3/4\,\left(\&^{P}_{zz}\right)^2. \tag{8}$$

The value of this quantity at the failure strain can be obtained from the 1-D tensile test data given in Table 1 by the formula

$$\&^{P}_{zz_f} = \in_{zz_f} - \sigma_{zz_f}/E_y \tag{9}$$

where E_y is Young's modulus. The value of the stress, σ_{zz}, at the failure strain is given by (writing the quantities without the zz subscript for convenience)

$$\sigma_f = \sigma_y + E_t\ (\in_f - \sigma_y/E_y) \tag{10}$$

where σ_y is the yield stress, and

E_t is the plastic tangent modulus.

Upon substitution of Equation (10) into Equation (9), one finds for the deviatoric plastic strain at failure

$$\&^{P}_{f} = (\in_f - \sigma_y/E_y)(1 - E_t/E_y) \tag{11}$$

where all the quantities in this equation are measured in the 1-D tensile test.

The values of the invariant $I_2\left(\&^{P}_{1\text{-}D}\right)_f$ at the failure strain are listed in Table 1. It is of interest to note that the values of this quantity range over nearly two orders of magnitude for the three sets of tensile specimens.

B. Composition B-3 Explosive. Composition B-3 explosive is a viscoelastic material which is available in three forms, pressed, cast, and powder. In most munitions, the explosive is normally inserted into its container by pouring in the molten state, so that it is called cast. Composition B explosive and composition B-3 explosive are similar, but the B-3 form has no wax content (Ref. 5).

	RDX	TNT	Wax
Composition B	63%	36%	1%
Composition B-3	60%	40%	——

The materials properties used herein are those for composition B-3 and are taken from Reference 6, primarily.

After surveying the available material properties of composition B-3 explosive and the various 2-D axisymmetric materials models in the ADINA code, it was decided that the curve description material model was the appropriate model to use (See pp. XII. 16-21 of Ref. 3). This model requires tables of

bulk moduli and shear moduli versus volume strain. Specifically, the loading bulk modulus, the unloading bulk modulus, and the loading shear modulus as functions of volume strain are required.

A relationship between the volume strain and the bulk modulus was obtained from the Mie-Grüneisen equation of state (EOS) and certain other assumptions which are detailed in Reference 7. The equation relating the bulk modulus to the volume strain is

$$\kappa = \frac{\Gamma(\Gamma+1)(A\mu^2+B\mu^3+C\mu^4)}{2 - \mu\Gamma} + A + A'\mu + B'\mu^2 + C'\mu^3 \qquad (12)$$

where

κ = the loading bulk modulus

Γ = the Grüneisen coefficient

A,B,C = the coefficients appearing in the Grüneisen EOS in terms of μ

$A' = A(\Gamma+1) + 2B$

$B' = B(\Gamma+2) + 3C$

$C' = C(\Gamma+3)$

$\mu = \epsilon_v/(1-\epsilon_v)$

$\epsilon_v = (V_o-V)/V_o$, volume strain taken positive in compression

$V_o = 1/\rho_o$ = specific volume at normal conditions.

The values taken from Reference 6 for the materials constants are as follows:

$\rho_o = 1.68$ g/cm^3

$\Gamma = 0.947$ (Assumed invariant as a function of V)

A = 13.5 GPa

B = 9.5 GPa

C = 100.6 GPa

$\nu = 0.29$ = Poisson's ratio

Note that when $\epsilon_v = 0$, $\mu = 0$, $\kappa_o = A$ and $V = V_o$.

Also, in the Grüneisen EOS, at $\epsilon_v = 0$, we take the pressure and internal energy to be zero, $P_o = E_o = 0$.

Because no data were available to relate the unloading bulk modulus to the volume strain for composition B-3 explosive, the same values of the bulk modulus for unloading as for loading were used. The loading shear modulus was obtained from the loading bulk modulus by use of the relationship

$$G_\ell = \frac{3\kappa_\ell(1-2\nu)}{2(1+\nu)} . \tag{13}$$

Figure 9 shows the graphical relationship represented by Equations (12) and (13). Table 2 gives the values of the two moduli as they were used in the ADINA program. ADINA uses linear interpolation between discrete points.

The tensile volumetric strain at failure for composition B-3 explosive is given in Reference 6 as -0.1 per cent. This criterion was used in all calculations presented in this report. The technique used in the ADINA code to apply this failure criterion is by the artifice of superimposing on the applied load-produced strains, an in-situ gravity pressure sufficient to cause a hydrostatic compression equal in magnitude to the tensile failure

TABLE 2. ADINA INPUT VALUES FOR BULK AND SHEAR MODULI FOR COMPOSITION B-3 EXPLOSIVE

Point No.	ϵ_v (%)	κ_ℓ (GPa)	κ_u (GPa)	G_ℓ (GPa)
1	0	13.52	13.52	6.60
2	1.0	14.00	14.00	6.84
3	2.5	14.91	14.91	7.28
4	3.75	15.83	15.83	7.73
5	5.0	16.92	16.92	8.26
6	10.0	23.36	23.36	11.41

strain. Then, when the total strain becomes negative, a tension cut-off plane is assumed to form normal to the principal strain. The normal and shear stiffnesses across this plane are reduced by a factor determined by an input value. One or two additional planes orthogonal to existing tension cut-off plane(s) are allowed to form if the strain criterion is met. The plane(s) becomes inactive if compression again develops in the direction normal to it.

The pseudo-hydrostatic pre-strain is applied by positioning the vertical coordinate (z-coordinate) at the proper negative value. The hydrostatic pressure applied at an element integration point is given for an element, j, by

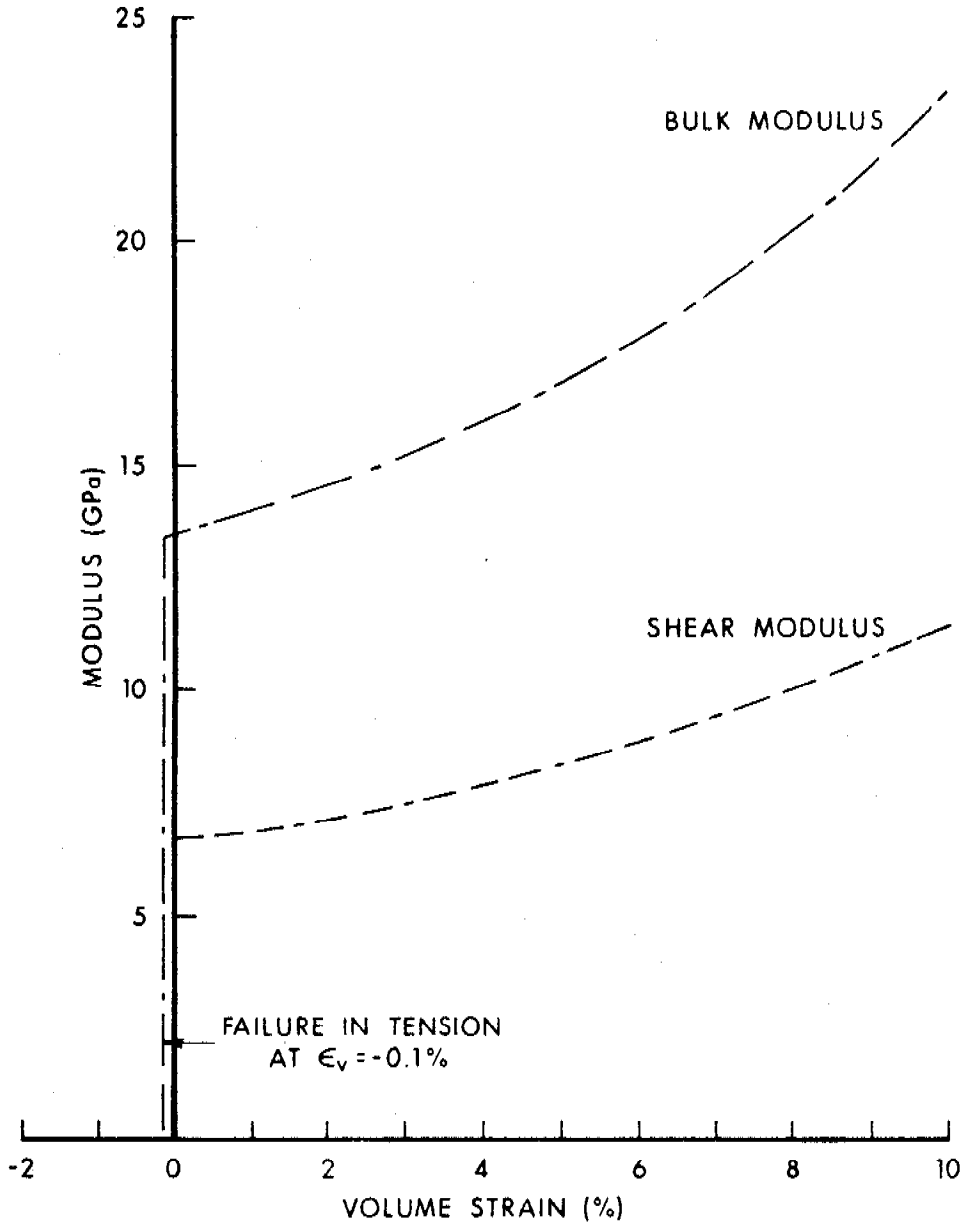


Figure 9. Relationship of Bulk Modulus and Shear Modulus to Volume Strain for Composition B-3 Explosive

$$P_j = - \rho_e \sum_{i=1}^{N} h_{ij} z_{ij} \qquad (14)$$

where

ρ_e is the density of the overburden

h_{ij} is the shape function for node i of element j

z_{ij} is the vertical coordinate for node i in element j.

The position of the system vertical coordinate can be obtained from the equation

$$z_{ave} = \frac{\kappa_o \epsilon_v^f}{g\rho_e} \qquad (15)$$

where

κ_o is the initial bulk loading modulus

ϵ_v^f is the volumetric failure strain, negative in tension

g is the acceleration of gravity.

C. Soil Simulation. For the structural response calculations denoted by Case A, nodal tie elements were used to model the base support as nonlinear springs. No simulation of the soil was necessary for the rigid support calculations of Case B.

The nodal tie element is an element which is available in the Ballistic Research Laboratories version of the ADINA code. Of the three types of nodal tie elements available, the one which was appropriate was the boundary type element. This element is defined by one node only and is capable of three translational degrees of freedom (DOF) and three rotational DOF's. In the application at hand, the elements along the base of the mine were used to transmit a vertical force (F_z), while those along the side transmitted a horizontal force (F_y).

Due to the large variety of soils in which mines would be emplaced, it is obvious that one can only select a soil simulation model which would be representative of some subclass of soils. With this in mind, a typical load deflection curve (Reference 8) was selected to define the nodal tie element properties. The average load-deflection for the elastic loading range given in Reference 8 is .0815 MPa/cm (30 psi/inch). The load-deflection response quoted is for slowly varying loads. For shock loads, the soil would be stiffer.

In an attempt to account for the dynamic response of soil, a nonlinear quadratic component was added to the force-deflection property. The magnitude of the nonlinear response term was made equal to the linear component at a deflection of 2.5 cm. The nodal spring constant and spring force as a function of vertical displacement are shown in Figure 10. These values were used for the nodal tie elements along the base of the mine. It should be noted that when the displacement is positive, the spring force and spring constant are zero.

For the support along the vertical sides of the mine, a linear spring force is used. This was done with two thoughts in mind. First, the movement of the mine in the lateral direction is small. Second, the soil on the sides of the mine is disturbed when the mine is emplaced and the soldier is not going to tamp the soil there, except lightly, while the mine is armed.

In the ADINA input data, the foregoing nonlinear stiffness values are adjusted by a factor proportional to an annular sector of π radians and a radial extent appropriate for the particular nodal tie element. Along the vertical side, the linear nodal tie element stiffness values are proportional to the height of the particular element onto which the nodal tie boundary element is attached.

4. FINITE ELEMENT MODEL DESCRIPTION AND CALCULATIONS.

A. Mesh Generation. The finite element mesh for the mine was generated with the aid of the GEN3D mesh generator code (Ref. 9). The resulting mesh for the steel and explosive element groups are shown in Figures 11 and 12. A six node QUAD element with quadratic displacement interpolation functions in the direction parallel to the surface was used for the steel casing. This element models the bending of the thin metal casing better than a four node QUAD. A four node QUAD was used to model the explosive except at the material interface.

In ADINA, each material having a distinct model for response must be modeled as a separate element group. For the Case A calculations with the nodal tie support elements, four element groups were required: (1) linear node ties on the side of the mine, (2) nonlinear node ties on the base to simulate soil support, (3) nonlinear 2-D solid elements for the steel case, and (4) nonlinear curve description 2-D solid elements for the explosive. For the steel case, three material subtypes were used to model the steel properties as shown in Table 1. The assignment of the material subtypes to the areas of the mine case is shown in Figure 11. For the Case B calculations, the nodal tie element groups were not required.

B. Time Step Solution. In ADINA, one has the choice of marching the dynamic solution forward via explicit or implicit finite difference techniques. In general, it is difficult to make absolute statements as to which is best for a given application. For shock loads such as indicated in Figure 4, it has been our experience that the explicit method gives the higher quality solution. The subject problem was run for a relatively large number of cycles using both implicit (with equilibrium iterations included) and explicit time integration solutions. After a given amount of problem

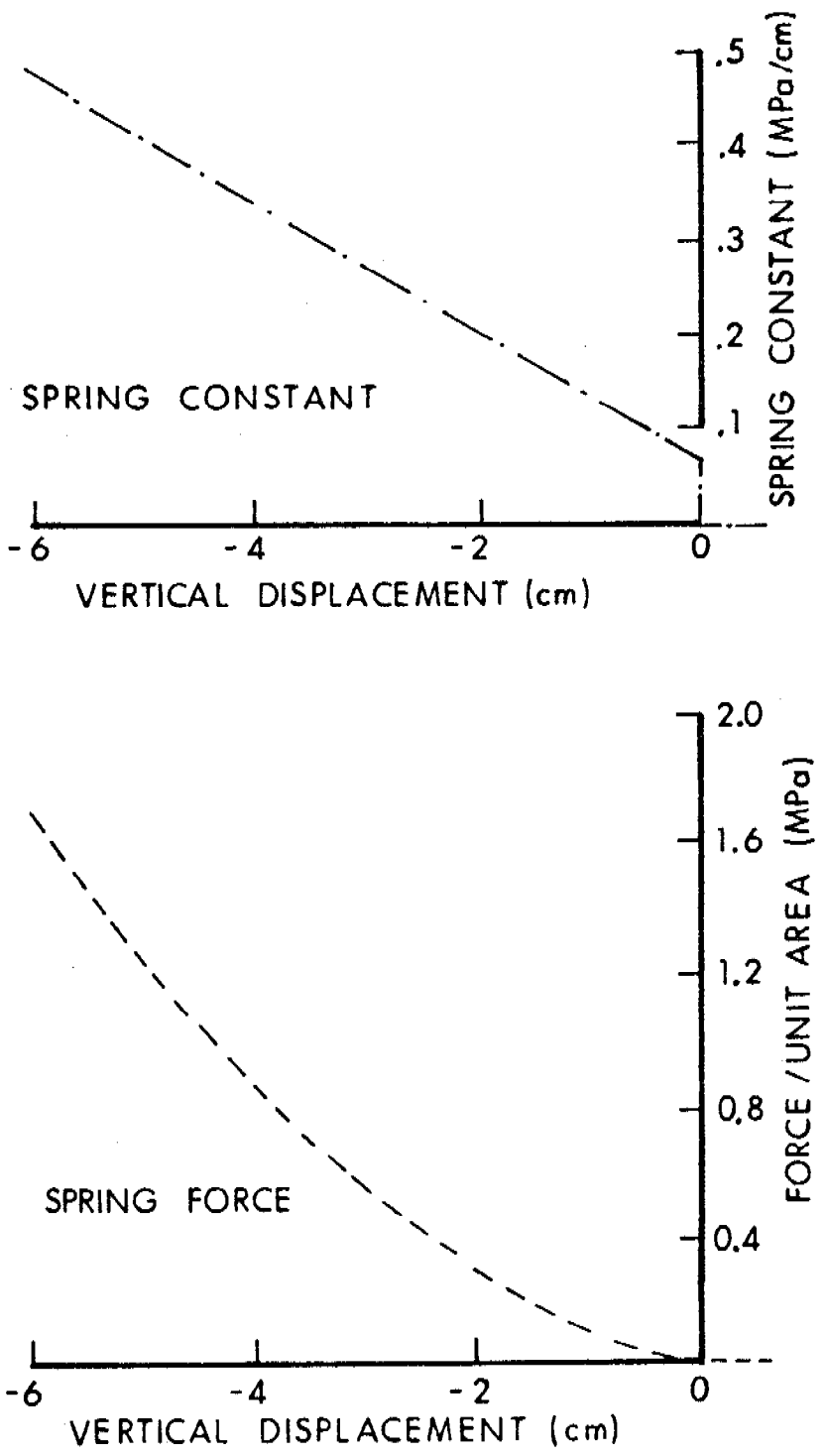


Figure 10. Nonlinear Elastic Nodal Tie Boundary Stiffness

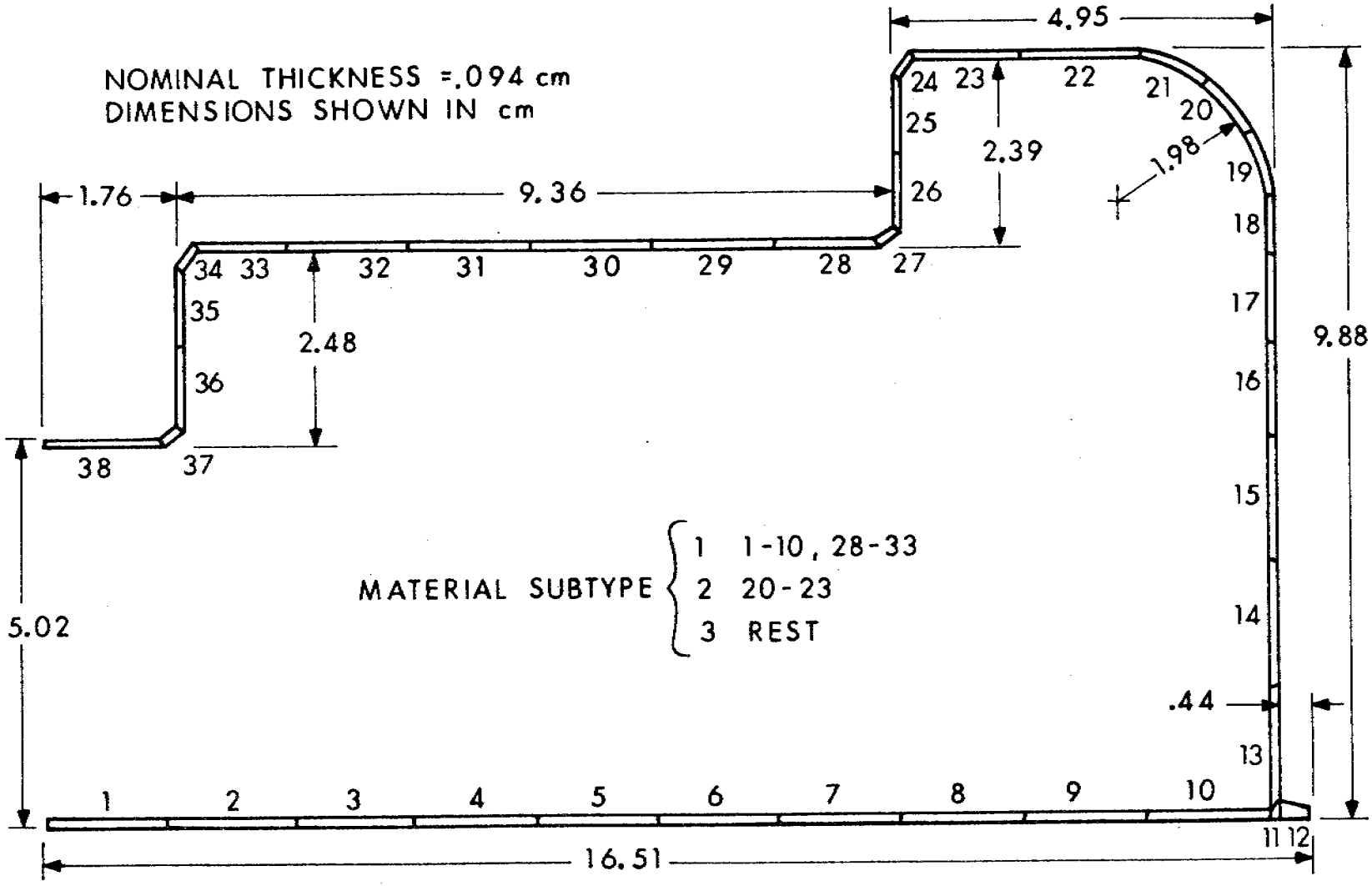


Figure 11. Element Configuration for Steel Casing, M-15 Antitank Mine

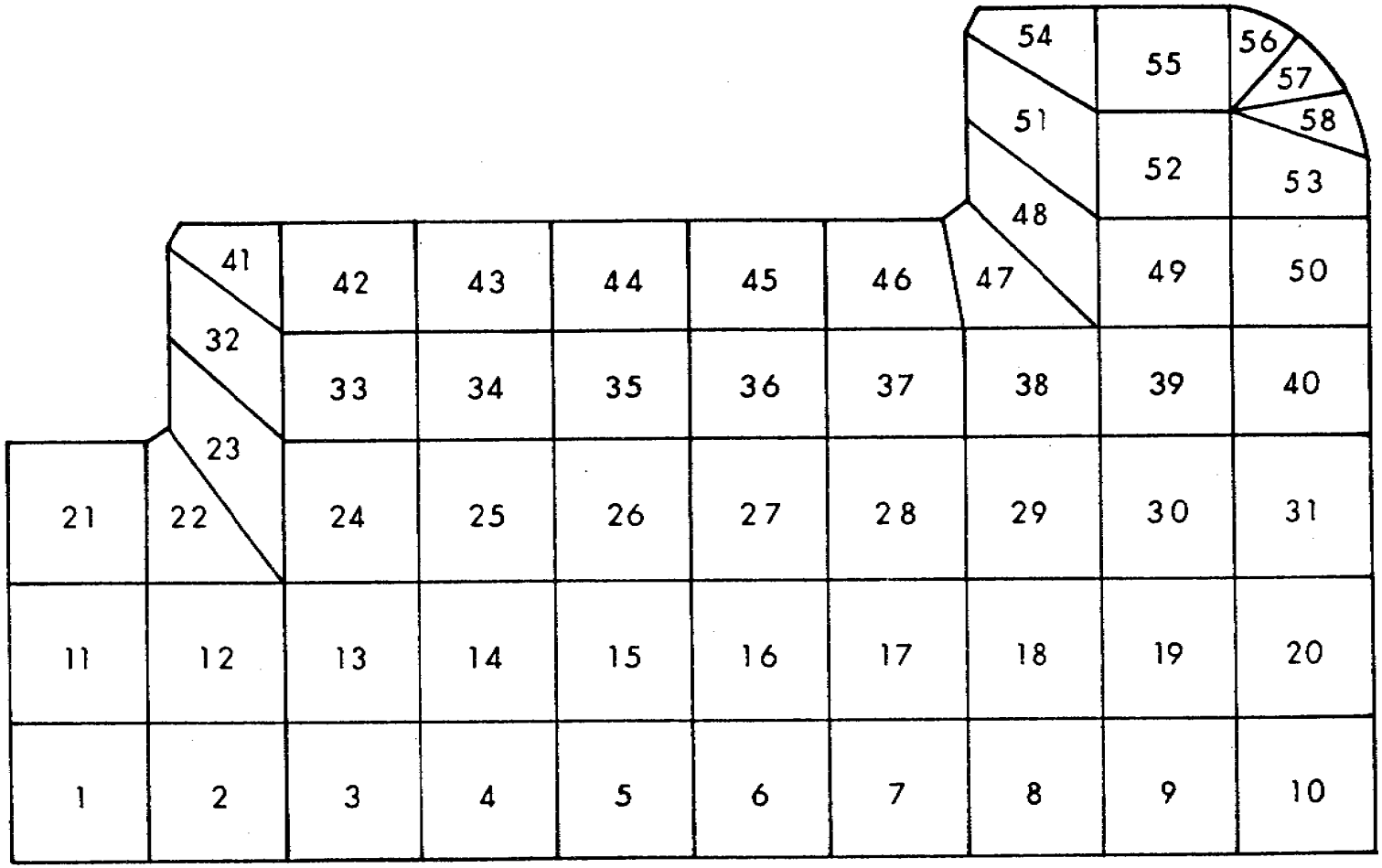


Figure 12. Element Configuration for Composition B-3 Explosive, M-15 Antitank Mine

solution time, the results were compared. The explicit solution had a smoother variation in both displacements and stresses. For this reason, we selected the explicit solution method.

The time step selected for the explicit scheme was sized to the thickness of the metal casing.

$$\Delta t = \frac{\Delta \ell}{3\sqrt{E_y/\rho}} = \frac{.00094}{3[2 \times 10^{11}/7800]^{\frac{1}{2}}} \approx 60 \text{ nsec}$$

A time step of 50 nanoseconds was used for all calculations.

Eigenfrequencies and mode shapes were calculated for the lowest four modes for both Cases A and B. The result is shown in Table 3 for the former.

TABLE 3. EIGENFREQUENCIES AND PERIODS

Case A		Case B	
Frequency (cps)	Period (sec)	Frequency (cps)	Period (sec)
36.44	2.744×10^{-2} (Rigid Body Mode)	6426	1.556×10^{-4}
3636	2.750×10^{-4}	7899	1.266×10^{-4}
6710	1.490×10^{-4}	9685	1.032×10^{-4}
8531	1.172×10^{-4}	12186	8.205×10^{-5}

The 0.5×10^{-7} second time step may be compared to the 1.5×10^{-4} second period of the fundamental mode of the rigid support configuration. It is readily seen that there is no danger of not capturing the response of the fundamental eigenfrequency as well as many of the higher frequency modes.

C. ADINA Program Modifications. Three significant modifications to the ADINA program were required in the course of performing the calculations herein. First, it was necessary to correct the stress state to lie on the yield surface to a higher order of accuracy than that fulfilled in the standard ADINA program. In the ADINA program, the von Mises yield criterion may be applied as

$$\Sigma_{ij}\,\Sigma_{ij} \leq 2/3\ \sigma_s^{\,2} \tag{16}$$

where

$$\Sigma_{ij} = S_{ij} - \alpha_{ij}$$

S_{ij} = deviatoric stress component

α_{ij} = tensor defining the center of the current yield surface

σ_s = current yield stress.

Upon taking a time step during which plasticity occurs, the new yield criterion is

$$(\Sigma_{ij} + \Delta\Sigma_{ij})(\Sigma_{ij} + \Delta\Sigma_{ij}) \leq 2/3\ \sigma_s{}^2. \tag{17}$$

The quadratic term, $\Delta\Sigma_{ij}\Delta\Sigma_{ij}$, was neglected in ADINA and caused the stress state to gradually creep outside the yield surface. At some point, this causes a negative square root to be encountered, which in turn, aborted the solution. Upon inclusion of the second order correction, no further difficulties of this type were encountered. Further details of this correction may be found in References 10 and 11.

The second major modification of ADINA involved the addition of subroutines and modification of current routines to allow the monitoring of extremal principal stresses and strains for the steel element group. ADINA normally does not print strains, and extremal values of the stresses are often difficult to discern. The routines included here check the values of the three principal stresses and three principal strains at each time step against currently stored maxima/minima for similar quantities. Any new extremal values found are placed in a save table which also includes the time and location of occurrence as well as a snap shot of the cartesian stress-strain state. This table may be printed when desired. It is also included in the restart file for consistency.

The third modification of ADINA was required to monitor the failure criterion for the steel casing. Provision was made to input for each material subtype the 1-D measured value at failure of the second invariant of plastic strain. At each integration point for each element for each time step, the 3-D value of $I_2\left(\varepsilon^P_{3\text{-}D}\right)$ is calculated and compared to the indicated failure value for that material. Information is saved, giving the maximum value of $I_2\left(\varepsilon^P_{3\text{-}D}\right)$ for each element as well as the integration point number and the time of occurrence. A compact table is printed as desired showing for each element whether plasticity has occurred; if it has, what the maximum value of $I_2\left(\varepsilon^P_{3\text{-}D}\right)$ is; whether the value has exceeded the input failure value; and if so, at what time and location.

5. ANALYSIS OF RESULTS. The output from these calculations included the usual printed output from ADINA giving cartesian displacements and stresses, forces in the node tie elements, as well as the tables of extremal stresses and strains and tables associated with the failure criteria. In addition, plot files were saved every 2.5 microseconds, giving a complete picture of the solution. The latter were used in conjunction with two post-processors, PLOT1D and PLOT3D (Ref. 9) to provide a graphical picture of such quantities

as deformed shapes, contour plots of various stress and strain components as well as time varying plots of quantities of interest.

The calculations for both Case A and Case B were carried out to a response time of 2 milliseconds. For Case A, four failures of the steel jacket were predicted as shown in Figure 13. The failures predicted were all in the area of the primary fuze well. However, other areas on the steel jacket had significant plastic flow during the first two milliseconds of response (2nd invariant of plastic strain attained at least one-half of the failure value). Figure 14 shows the deformed shape of the configuration at the time of the first predicted failure in the steel jacket. In order to make the deformation of the structure more meaningful, the rigid body motion on the vertical springs has been subtracted out. Also, the vector displacement as measured from the reference points, indicated by the crosses, has been magnified by an appropriate factor. Figure 15 shows a contour plot of the hoop stress in the critical fuze well area at the time of the first predicted failure. It is readily seen that there are strong bending stresses occurring near the corners of the fuze well. Figure 16 shows contours of constant radial strain at this same time for the nonlinear spring supported configuration.

Figures 17-20 show similar plots for Case B with the rigid support as Figures 13-16 show for Case A. For Case B, it was not necessary to subtract rigid body motion since the base of the mine is restrained in the vertical direction. Notice that the first failure for Case B occurred earlier than for Case A and the center of fuze well is deflecting downward rather than upward. In Case B, the vertical sides of the mine were loaded by the transient pressure. In general, Case B is a much more highly constrained structure than Case A. The deformation of Case B at approximately ¼ microsecond time is generally inward in compression. Figure 19 shows the center of the fuze well with intense stress gradients. From Figure 20, it is evident that this area is undergoing a rather strong shearing action.

The printed output from these calculations showed that the explosive filler was developing cracks in many locations and directions according to the -0.1% tensile failure criterion. These cracks were indicated very early in the response and continued to develop at later times. Thus, we expect the internal mass of explosive to be nearly pulverized by the time of the casing ruptures indicated. Whenever the casing ruptures, it is assumed that a large volume of the explosive would be ejected. This was, in fact, the indication in the tests conducted (Ref. 1). It is not possible to compare the deformed shapes resulting from the tests described in Reference 1 with those predicted herein. The only results given in Reference 1 are photographs of the final result of the deformation. Also, for economy reasons, the calculation was not run far enough to get a final deformed shape.

Shown in Figure 21 is the motion of the center of mass of the mine for Case A as a function of time. It is seen that the mine is deflecting into the soil under the intense pressure pulse. At approximately 1.4 milliseconds, the spring forces and the transient top surface load equilibrate. Thereafter, the base support load becomes larger than the surface loading and the mine begins to decelerate. At 2.0 milliseconds, the average pressure differential between the surface loading and the base reaction is 1.28 MPa.

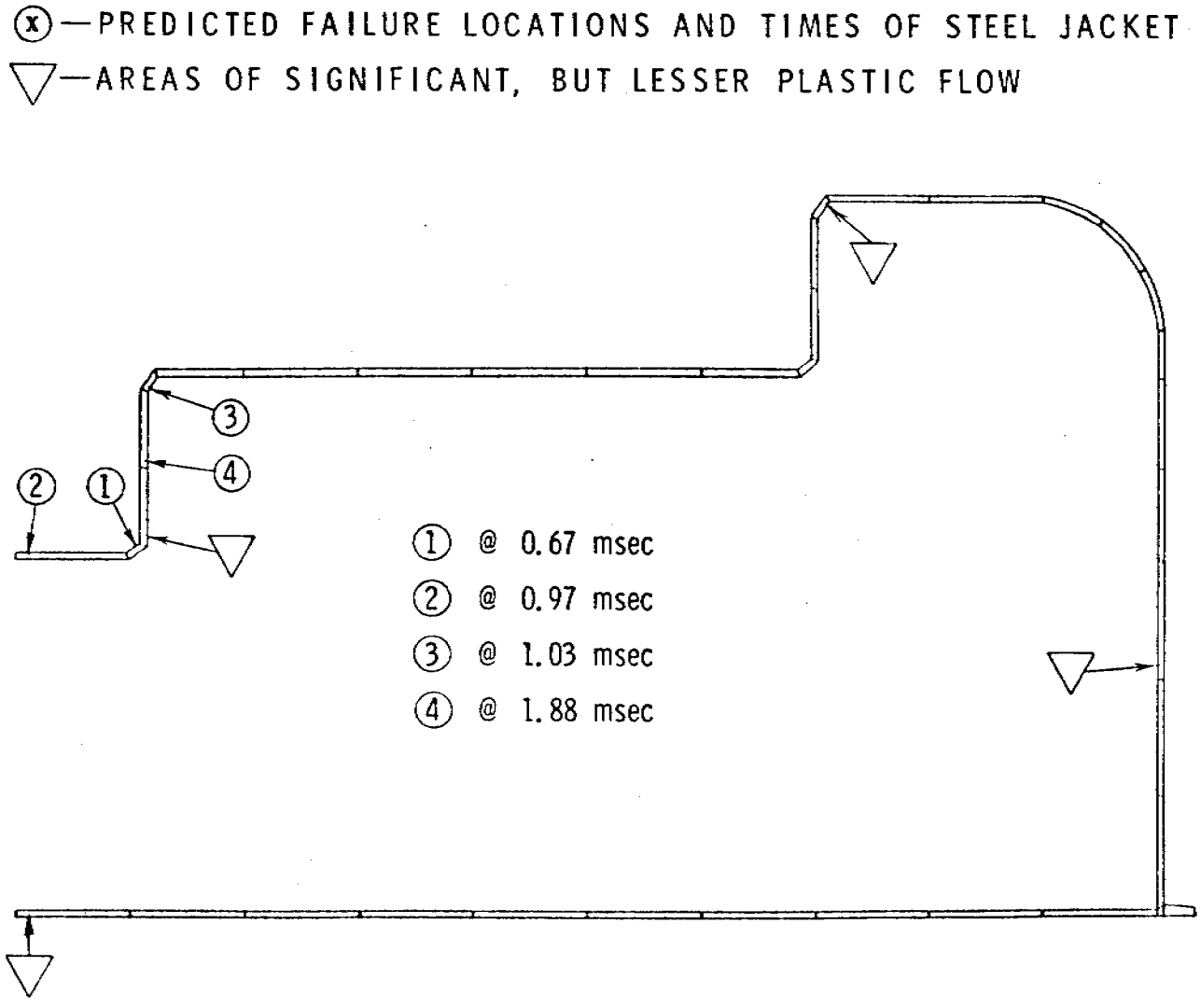


Figure 13. Predicted Failures of M-15 Mine Jacket with Nonlinear Spring Support, Case A

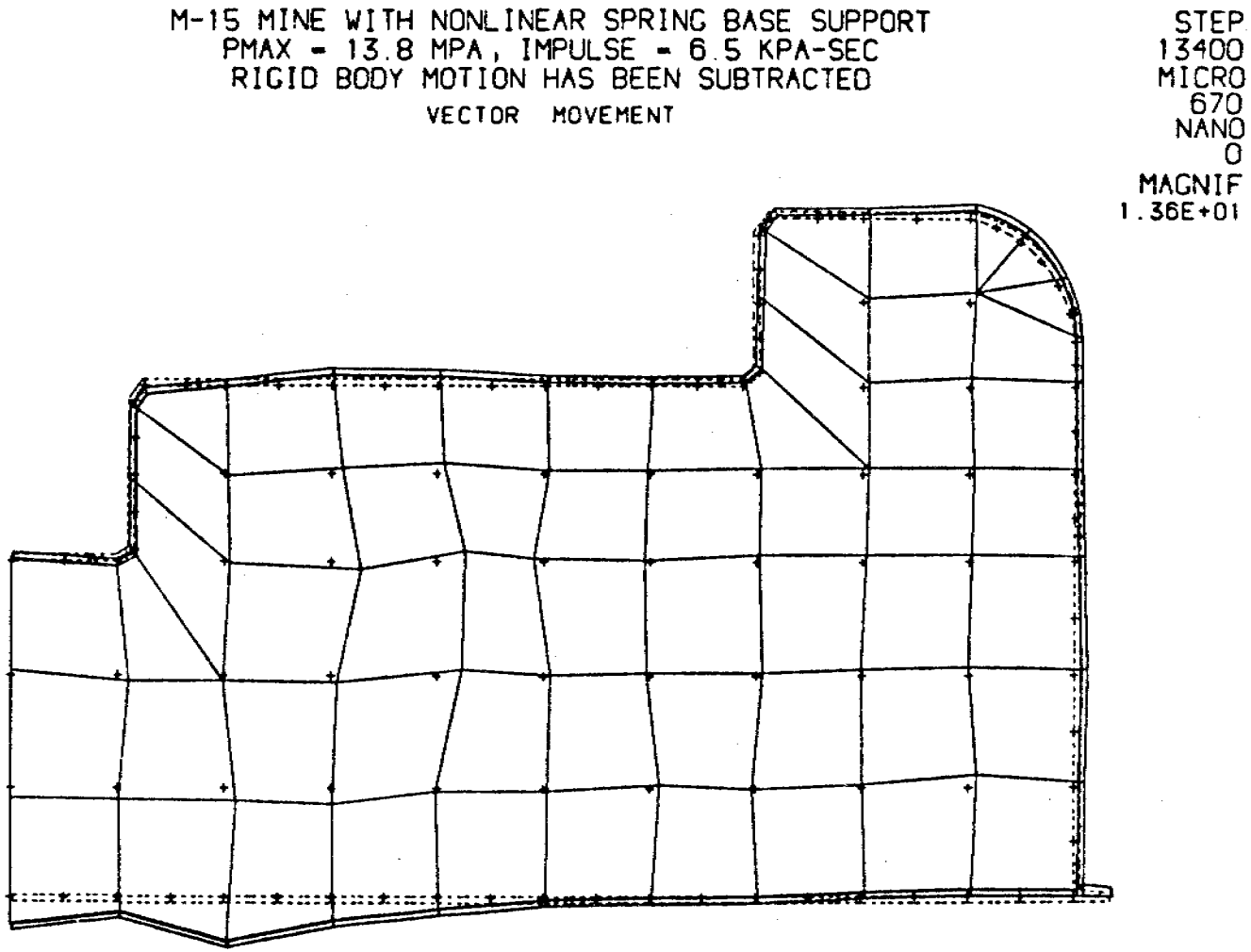


Figure 14. Deformed Shape of M-15 Mine at the Time of First Predicted Failure, Case A

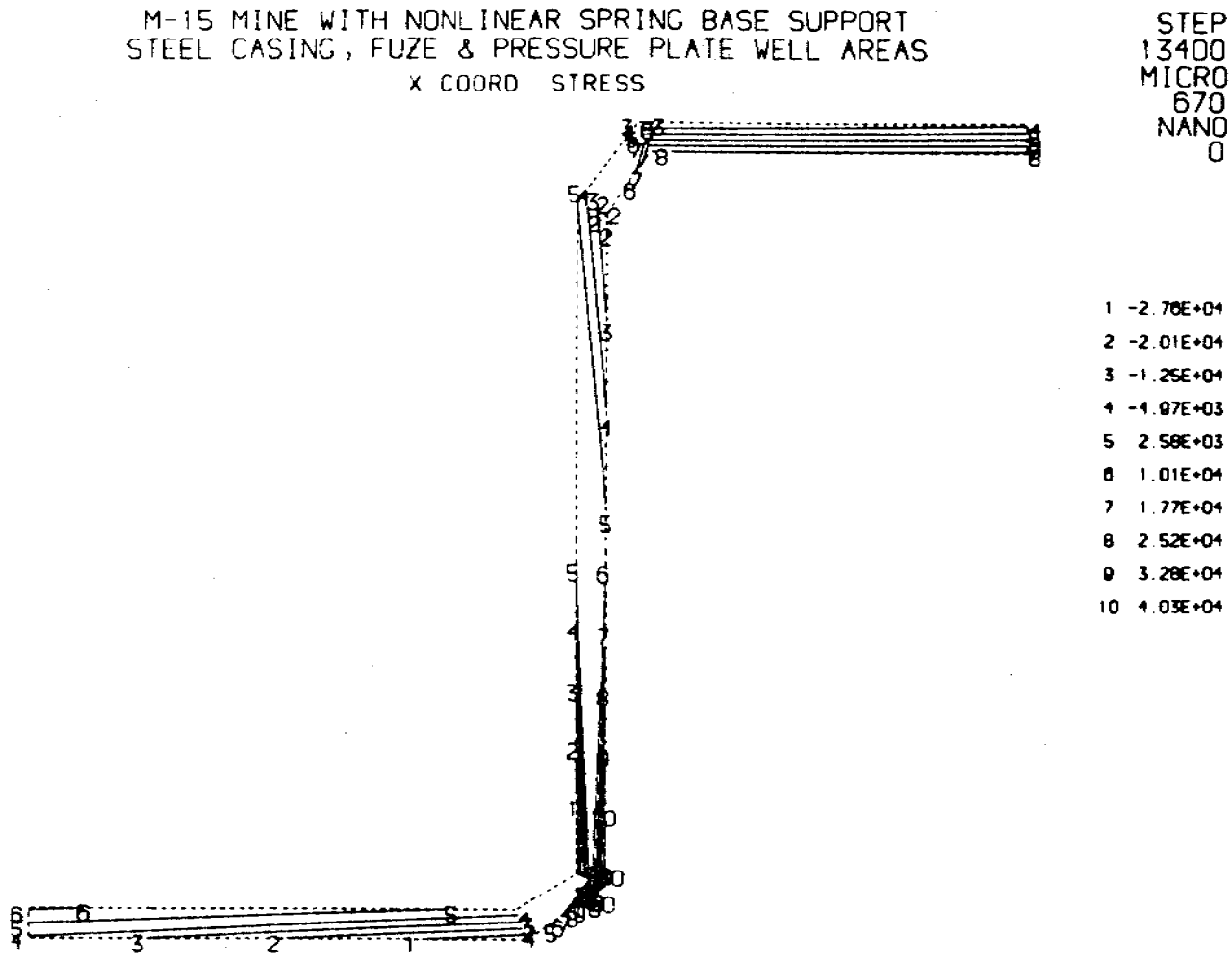


Figure 15. Contour Plot of Hoop Stress in Fuze Well Area at the Time of First Predicted Failure, Case A

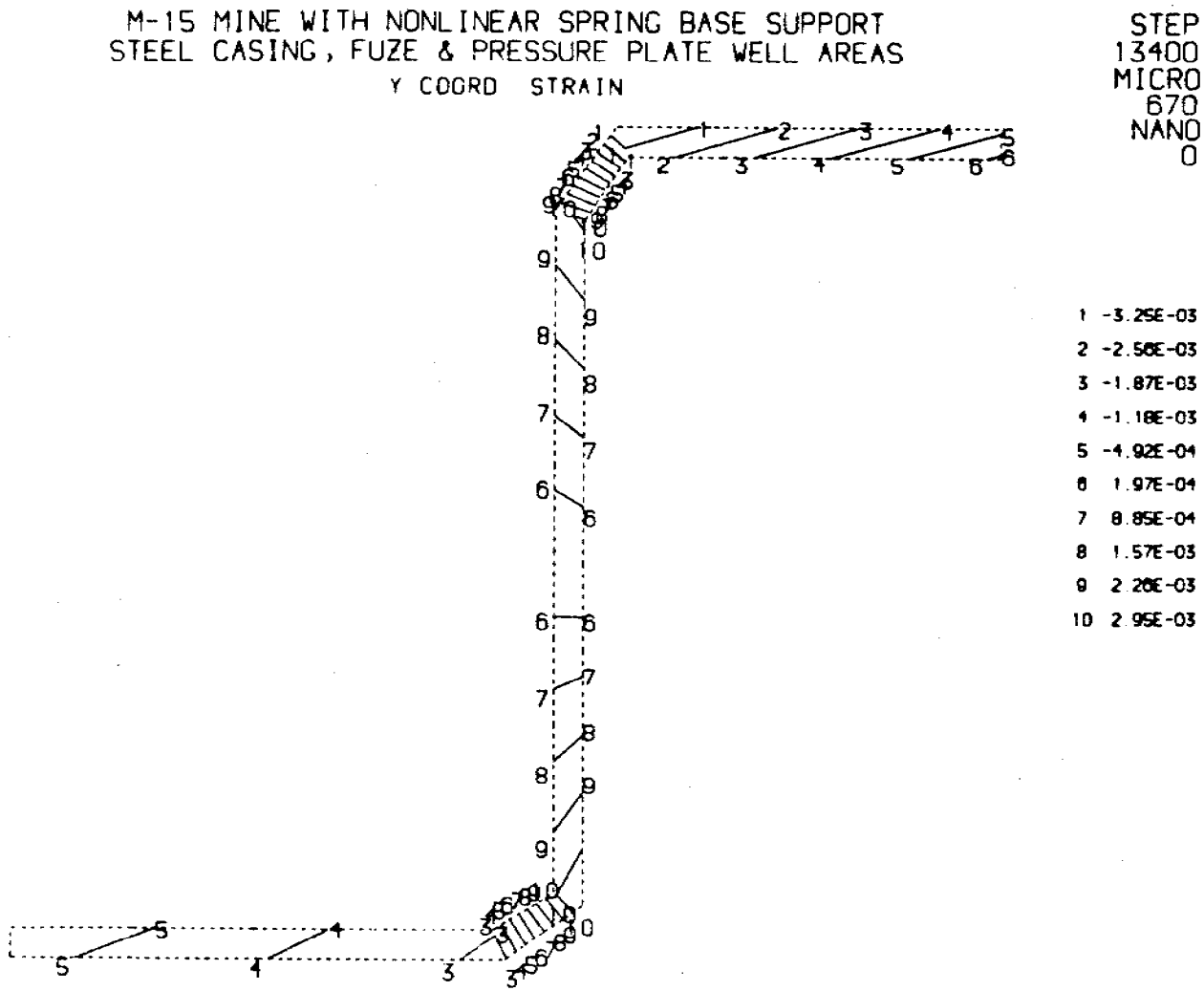


Figure 16. Contour Plot of Radial Strain in Fuze Well Area at the Time of First Predicted Failure, Case A

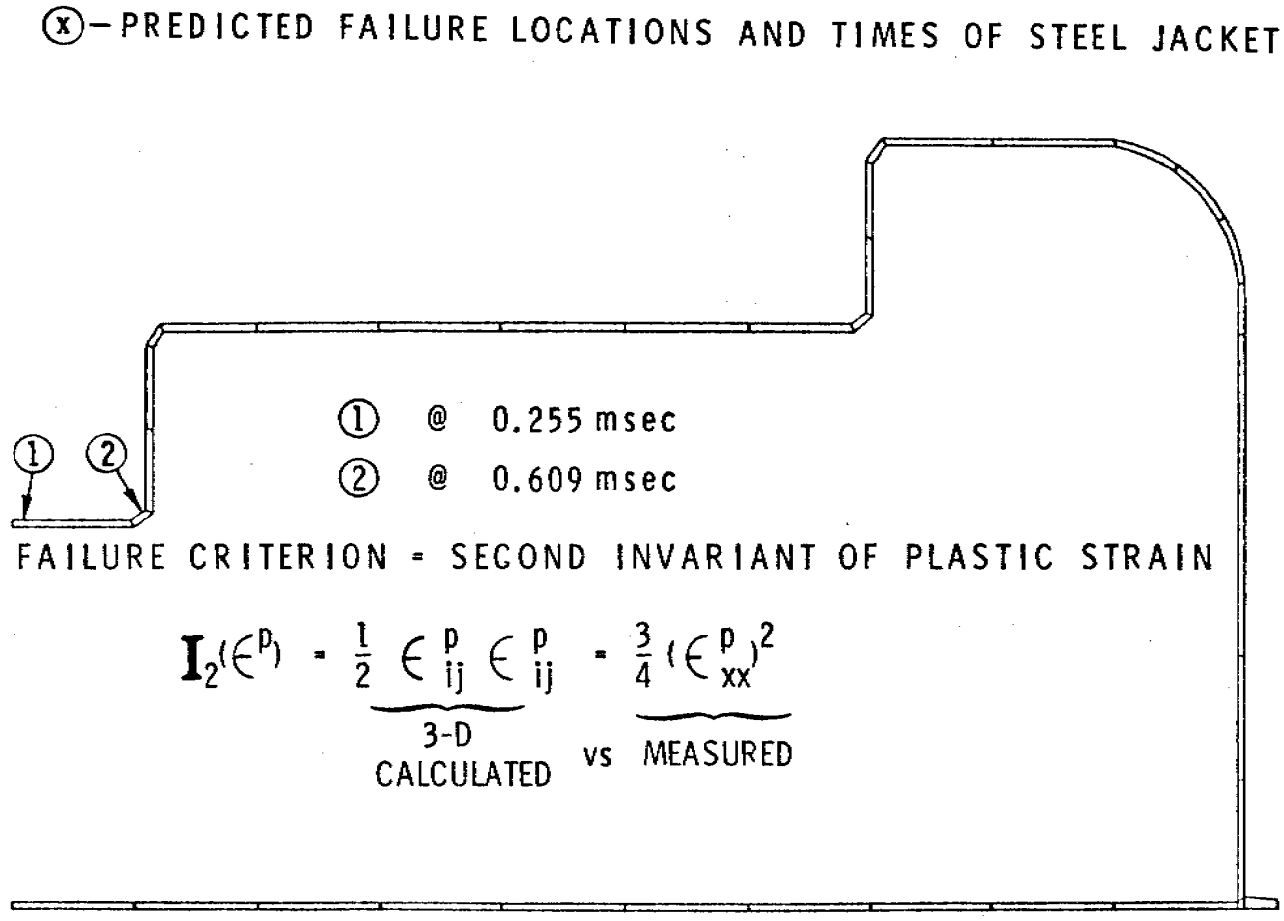


Figure 17. Predicted Failures of M-15 Mine with Rigid Base Suport, Case B

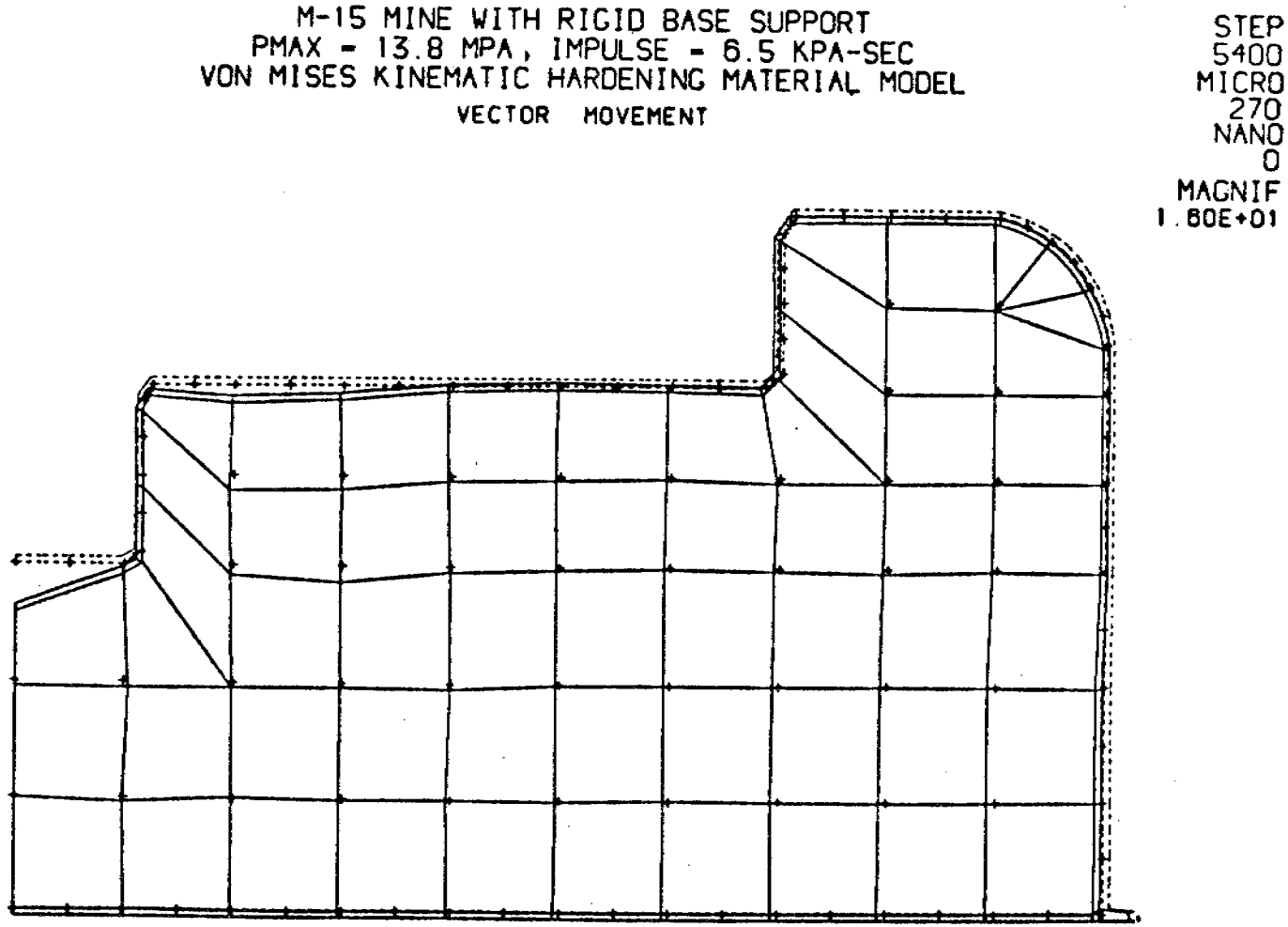


Figure 18. Deformed Shape of M-15 Mine at Time of First Predicted Failure, Case B

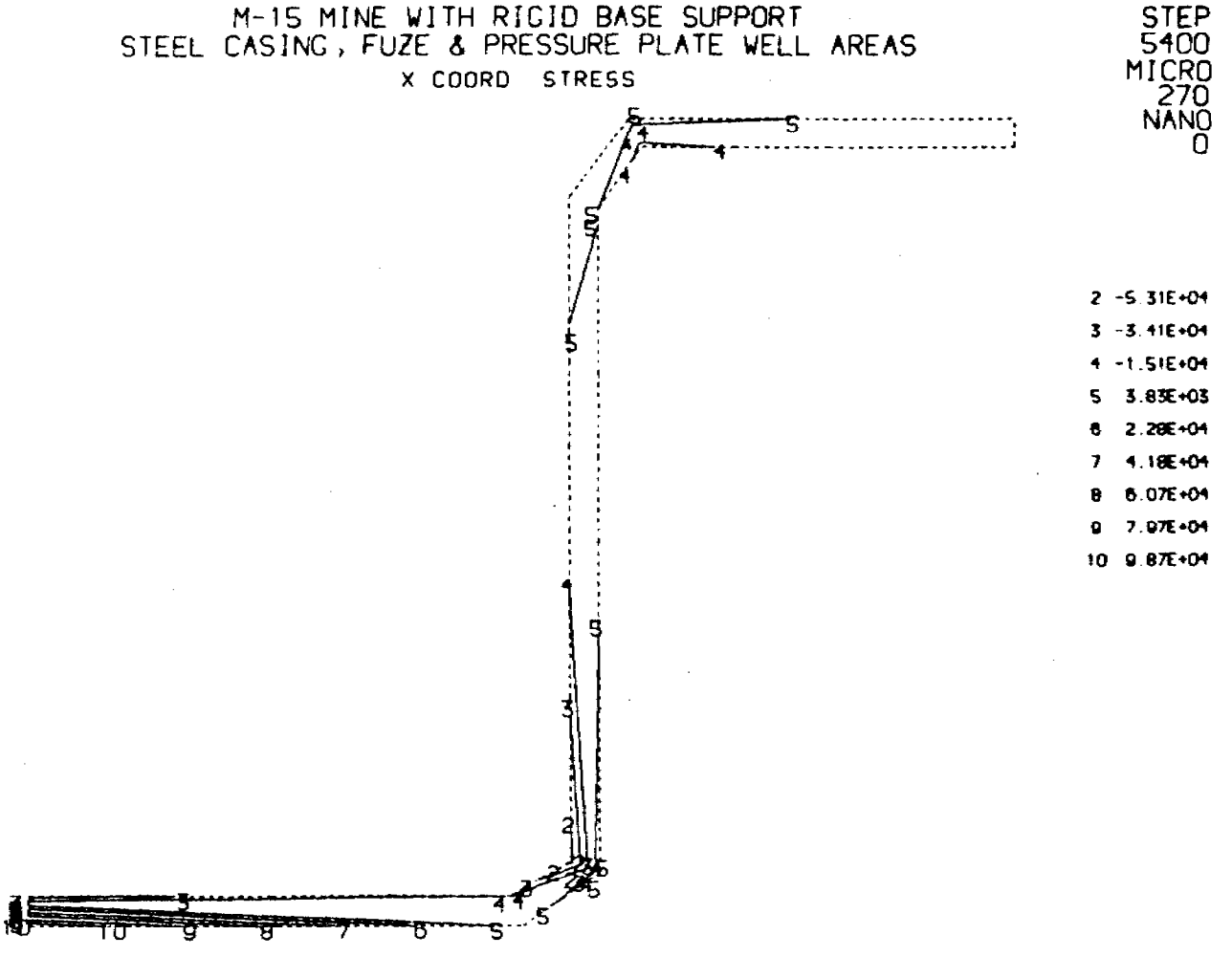


Figure 19. Contour Plot of Hoop Stress in Fuze Well Area at the Time of First Predicted Failure, Case B

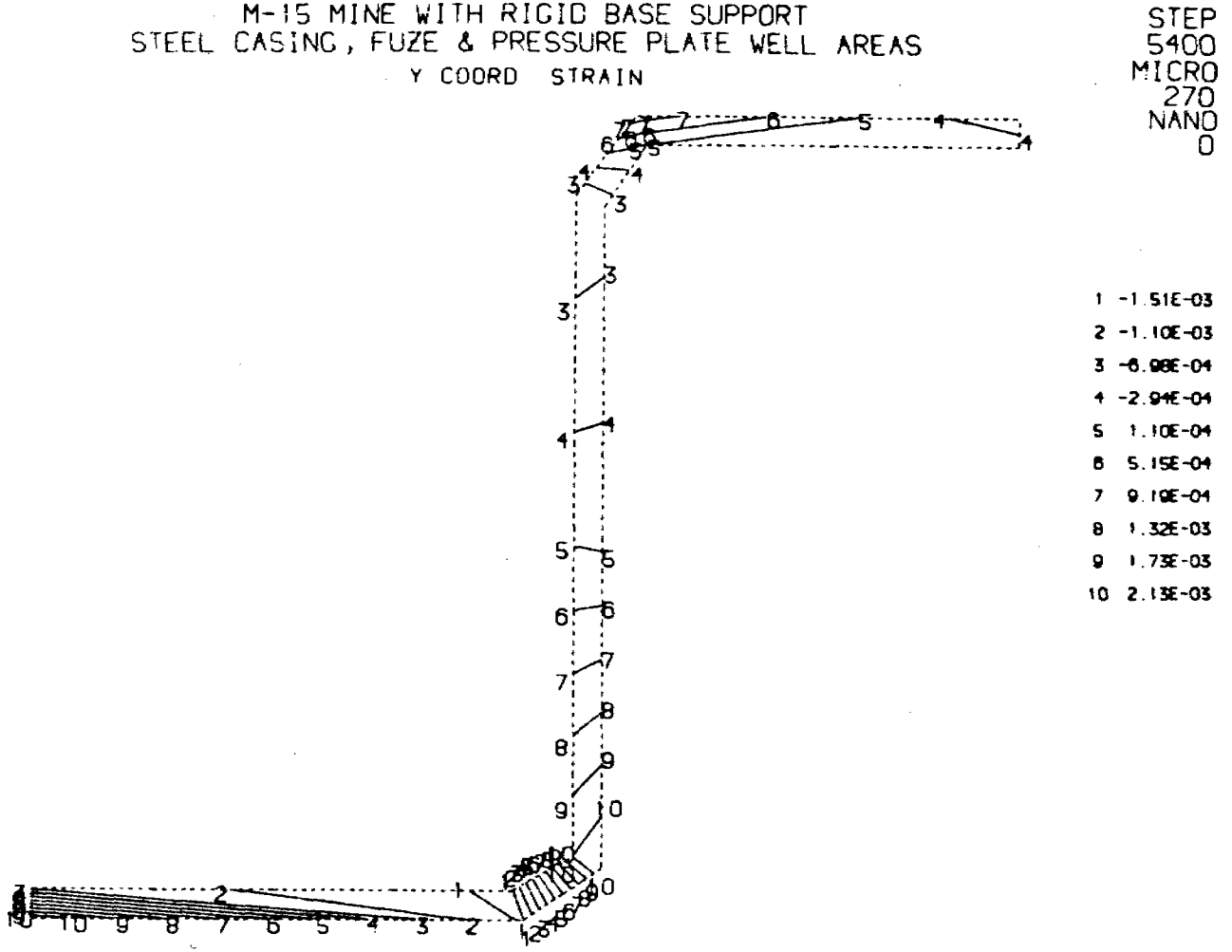


Figure 20. Contour Plot of Radial Strain in Fuze Well Area at Time of First Predicted Failure, Case B

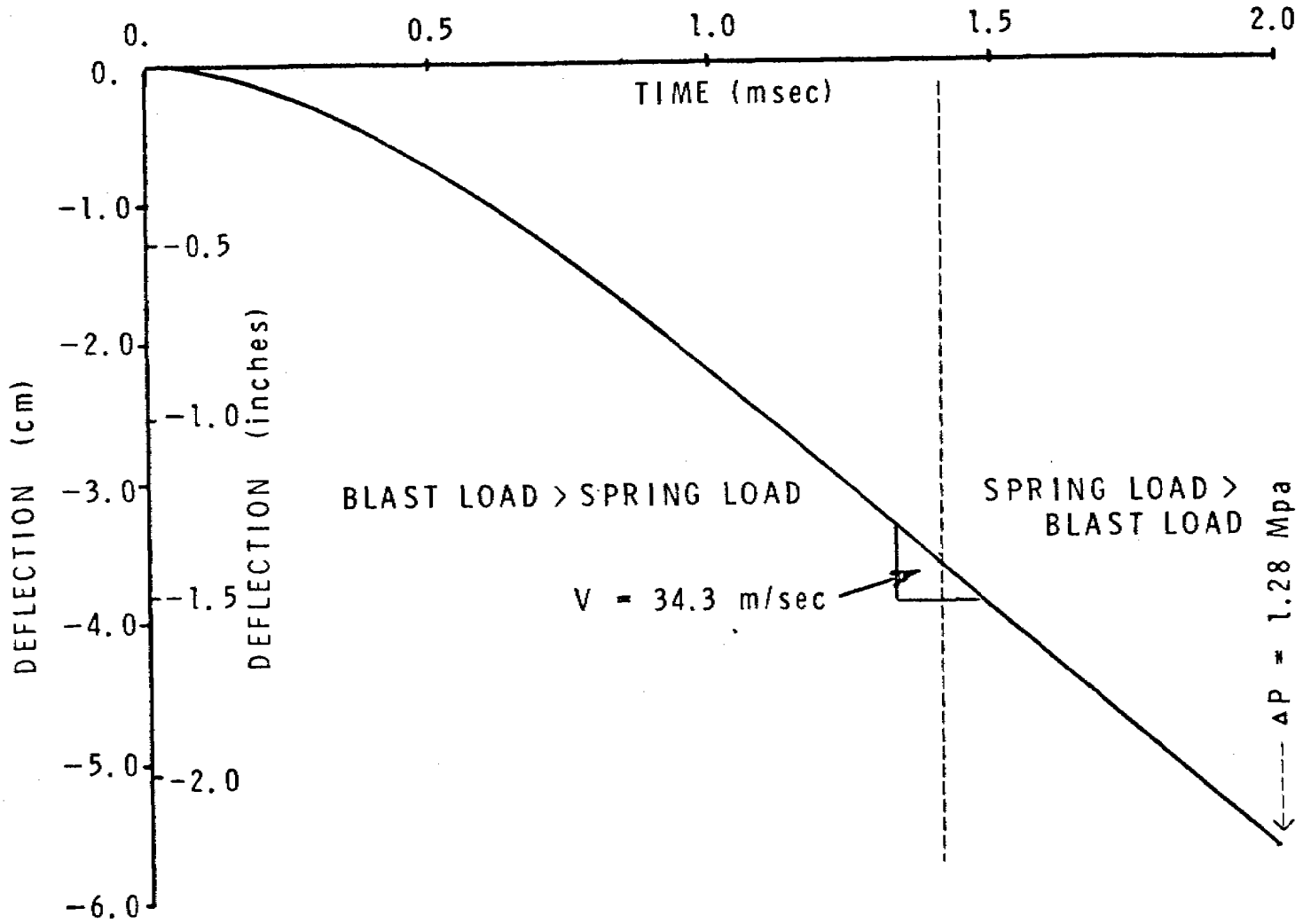


Figure 21. Motion of Center of Mass of M-15 Mine with Nonlinear Spring Support

6. CONCLUSIONS. The M-15 mine is predicted to fail in the area of the central fuze cavity at the 13.8 MPa peak pressure, 6.5 kPa-sec impulse level in both the spring support and rigid support conditions. This agrees with experimental tests, which showed catastrophic failure of the metal casing and ejection of the secondary fuze wells.

The explicit time integration method gave the most accurate results for the shock loaded mine. This statement is based on the smoothness of calculated displacements and stresses.

The M-15 mine case is highly inhomogeneous in its constitutive structural properties. Follow-on work should take account of these inhomogenities. A liberal sampling of stress-strain data measurements is indicated for such deep drawn thin metal components.

Second order corrections to assure that the stress state is on the yield surface during plastic flow are required.

The soil medium supporting the mine and the nature of the loading of the sidewall have a significant influence on the resulting response. It is recommended that the soil medium be included explicitly in any additional studies. Attenuation of the shock in the area of the sidewall should be investigated.

REFERENCES.

1. Allen J. Tulis et al (IITRI) and David C. Heberlein et al (MERADCOM), "Improved Fuel-Air Explosives", USA Mobility Equip. R. & D. Command Report 2222, Sep 1977 (S).

2. Andrew Mark, "M-15 Antitank Mine Vulnerability", BRL Report 2211, Jan 1980.

3. K. J. Bathe, "ADINA, A Finite Element Program for Automatic Dynamic Incremental Nonlinear Analysis", MIT-82448-1, Revised Dec 1978.

4. K. J. Bathe, "Static and Dynamic Geometric and Material Nonlinear Analysis Using ADINA", MIT-82448-2, Revised May 1978.

5. Brigitta M. Dobratz, "Properties of Chemical Explosives and Explosive Simulants", UCRL-51319 (Rev 1), July 1974.

6. M. S. Chawla and R. B. Frey, "A Numerical Study of Projectile Impact on Explosives", BRL Memorandum 2741, April 1977.

7. Frederick H. Gregory, "Failure of the M-15 Antitank Mine Due to Blast Loads", ARBRL Technical Report in publication.

8. C. A. Hogentogler, Engineering Properties of Soil, First Edition, McGraw-Hill Book Co., 1937, p. 223.

9. John E. Crawford, Private Communication, Civil Engineering Laboratory, Naval Construction Center, Port Hueneme, California, Jan 1980.

10. J. M. Santiago, "On the Accuracy of Flow Rule Approximations Used in Structural and Solid Response Computer Programs", 1981 Army Numerical Analysis and Computers Conference, Huntsville, Alabama, 26-27 Feb 1981.

11. A. D. Gupta, J. M. Santiago, and H. L. Wisniewski, "An Improved Strain Hardening Characterization in the ADINA Code Using the Mechanical Sub-layer Concept", First Chautauqua on Finite Element Modeling, Ed. J. H. Conaway, Schaeffer Analysis, Inc., Sep 1980.

ON THE ACCURACY OF FLOW RULE APPROXIMATIONS USED IN STRUCTURAL AND SOLID RESPONSE COMPUTER PROGRAMS

Joseph M. Santiago
U.S. Army Ballistic Research Laboratory
U.S. Army Armament Research and Development Command
Aberdeen Proving Ground, Maryland 21005

ABSTRACT. Computer programs for calculating the elastic-plastic response of structures and solids employ a variety of plasticity algorithms which basically differ in the approximations used for the flow rule and the yield condition. This paper focuses on the linear kinematic hardening model of plastic behavior, presenting a numerical comparison between a number of commonly used approximations and an exact solution to this model. The exact solution is obtained by quadrature based on assuming proportional straining during an increment. Comparisons are presented for biaxial and triaxial states of stress and recommendations are made as to the "best" approximation and the "optimum" number of subincrements needed for a given accuracy.

1. INTRODUCTION. Computer programs that treat the elastic-plastic behavior of materials usually employ an incremental approximation to the plasticity equations, with the stress being calculated at discrete steps. This usually entails using a finite difference approximation to the flow rule. In addition, for the sake of simplicity, a linear approximation to the yield condition is often imposed on the stress.

This paper concerns itself with code approximations to linear kinematic hardening based on the Prager model of a yield surface translating in stress space (ref. 1). The von Mises yield condition is employed with the associative flow rule. Hardening is taken to be proportional to the plastic strain. Elastic-perfectly plastic behavior is automatically included in the analysis by setting the hardening parameter equal to zero. Both biaxial (plane stress) and triaxial states of stress are analyzed.

2. REVIEW OF PLASTICITY EQUATIONS. We briefly summarize the Prandtl-Reuss theory of elastic-plastic behavior (ref. 2) as background. The theory assumes the additive decomposition of the strain ε_{ij} into an elastic strain ε^e_{ij} and a plastic strain ε^p_{ij}

$$\varepsilon_{ij} = \varepsilon^e_{ij} + \varepsilon^p_{ij} \quad * \tag{1}$$

*Cartesian tensor notation, including the summation convention, is employed throughout, with Latin indicies i, j, k, ... = 1, 2, 3 and Greek indicies α, β, δ, ... = 1,2.

with the plastic portion taken to be incompressible

$$\varepsilon^{p}_{kk} = 0 \tag{2}$$

The stress σ_{ij} is assumed to be related to the elastic strain through Hooke's law for an isotropic material

$$\sigma_{ij} = \frac{E}{1+\nu} \left[\varepsilon^{e}_{ij} + \frac{\nu}{1-2\nu} \varepsilon^{e}_{kk} \delta_{ij} \right] \tag{3}$$

where E is Young's modulus and ν is Poisson's ratio. As already mentioned, the Prager model of kinematic hardening (ref. 1) is assumed, so that the von Mises yield condition takes the form

$$\frac{1}{2} \Sigma_{ij} \Sigma_{ij} = \frac{1}{3} \sigma_y^2 \tag{4}$$

where

$$\Sigma_{ij} = S_{ij} - \alpha_{ij} \tag{5}$$

and σ_y is the yield stress from a uniaxial tensile test, α_{ij} measures the translation of the yield surface in stress space, and S_{ij} is the deviator of the stress:

$$S_{ij} = \sigma_{ij} - \frac{1}{3} \sigma_{kk} \delta_{ij} \tag{6}$$

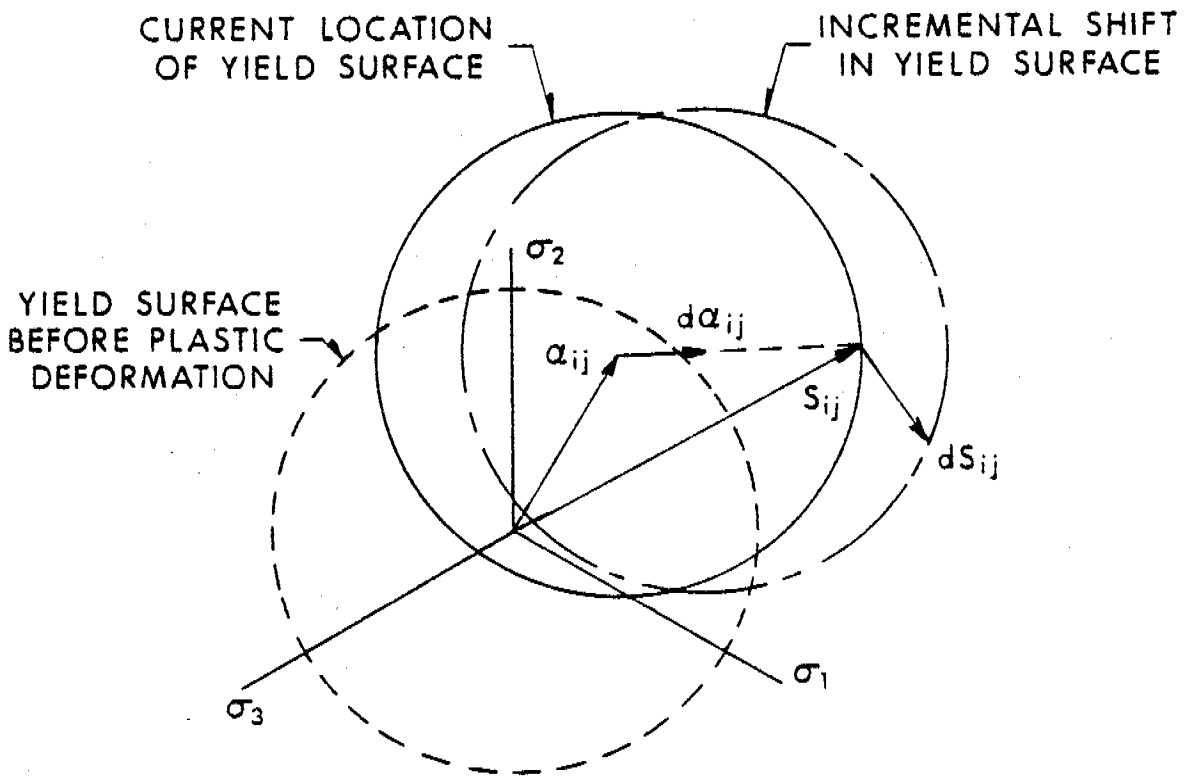


Figure 1. Representation of the von Mises yield condition in stress space as viewed along the diagonal from the positive tensile stress octant.

As shown by Figure 1, in stress space the yield condition represents a cylindrical (hyper-) surface of constant radius $(2/3)^{\frac{1}{2}}\,\sigma_y$ with its axis parallel to the diagonal from the origin into the positive tensile stress octant. Initially, the axis passes through the origin, but as plastic deformation progresses, the surface shifts in accordance with the hardening rule, which for a linear hardening model is directly proportional to the plastic strain

$$\alpha_{ij} = \frac{2}{3}\frac{E\,E_p}{E-E_p}\,\varepsilon^p_{ij} \tag{7}$$

where E_p is the plastic modulus as specified by the slope of the stress-strain curve in the plastic range. Lastly, the plastic strain is determined using the associative flow rule, which for the von Mises condition is

$$d\varepsilon^p_{ij} = \Sigma_{ij}\,d\lambda, \tag{8}$$

where $d\lambda \geq 0$ is the flow parameter which is adjusted so as to maintain the stress on the yield surface during plastic flow. Combining (7) and (8) we see that instantaneously the yield surface will translate in the direction of the normal at the point of stress application, as portrayed in Figure 1. As a special case these equations include elastic-perfectly plastic response by setting $E_p = 0$ so that the yield surface does not translate.

For purposes of analysis in this paper, the foregoing equations are reduced to a fundamental set of equations

$$d\Sigma_{ij} = dS^e_{ij} - \kappa\,\Sigma_{ij}\,d\tilde{\lambda} \tag{9}$$

$$dS^e_{ij} = \frac{E}{1+\nu}\left[d\varepsilon_{ij} - \frac{1}{3}\,d\varepsilon_{kk}\,\delta_{ij}\right] \tag{10}$$

$$\Sigma_{ij}\,\Sigma_{ij} = \frac{2}{3}\,\sigma_y^2 \tag{11}$$

on the unknown components of Σ_{ij}, where

$$\kappa = 1 + \frac{2}{3}\,\frac{1+\nu}{\frac{E}{E_p} - 1} \tag{12}$$

depends on the material constants and the flow parameter is redefined as

$$d\tilde{\lambda} = \frac{E}{1+\nu}\,d\lambda \tag{13}$$

In equation (9) dS^e_{ij} operates as a forcing function specified by (10) as a function of the prescribed strain differential. The yield condition (11) acts as a constraint on the values of Σ_{ij}. The flow parameter is controlled so that $d\tilde{\lambda} = 0$ when the stress is inside the yield surface or when $\Sigma_{ij}\, dS^e_{ij} < 0$ and the stress is on the yield surface; otherwise, when $\Sigma_{ij}\, dS^e_{ij} > 0$ the flow parameter takes on positive values that assure that the stress is maintained on the yield surface. The solution of the system determines the history of the stress at a point as a function of the strain history.

In the case of biaxial states of stress or plane stress situations, where the stress components for one coordinate, for example the 3 coordinate, vanish: $\sigma_{13} = \sigma_{23} = \sigma_{33} = 0$, the previous system reduces to

$$d\Sigma_{\alpha\beta} = d\sigma^e_{\alpha\beta} - [\kappa\, \Sigma_{\alpha\beta} - \mu\, \Sigma_{\gamma\gamma}\, \delta_{\alpha\beta}]\; d\tilde{\lambda} \quad * \tag{14}$$

$$\Sigma_{\alpha\beta}\, \Sigma_{\alpha\beta} + \Sigma_{\alpha\alpha}\, \Sigma_{\beta\beta} = \frac{2}{3}\, \sigma_y^{\,2} \tag{15}$$

where now the prescribed forcing term is

$$d\sigma^e_{\alpha\beta} = \frac{E}{1+\nu}\; [d\varepsilon_{\alpha\beta} - \mu\, d\varepsilon_{\gamma\gamma}\, \delta_{\alpha\beta}] \tag{16}$$

and a second material parameter appears

$$\mu = \frac{1}{3}\; \frac{1-2\nu}{1-\nu} \tag{17}$$

This system turns out to be more complicated than the system for triaxial states of stress principally because of coefficient μ. For incompressible materials ($\nu = \frac{1}{2}$) μ vanishes and the two systems behave similarly. It should also be noted that the yield condition (15) is now represented by an ellipsoid with a major axis along the σ_{12} coordinate axis and the remaining axes in the plane of the σ_{11} and σ_{22} coordinate axes at angles of 45° to these axes.

3. CODE APPROXIMATIONS TO PLASTICITY EQUATIONS. As already pointed out, computer programs for calculating the elastic-plastic response of bodies commonly determine the strain and the stress throughout the body at discrete time or loading steps. The usual procedure involves first determining the change in the strain from the prior step to the current step without recourse to plasticity equations. Then the values of the stress and the strain at the prior step and the strain at the current step are used in a plasticity algorithm to calculate the current value of the stress. Some programs will iterate on the current strain and stress values, but basically plasticity algorithms involve the determination of a current stress from a prescribed strain increment and a known prior stress.

* Greek indicies are over the range 1, 2.

Because the strains are prescribed at discrete intervals, plasticity algorithms are almost invariably based upon plasticity equations in which finite differences replace derivatives. Hence, for example, in the case of triaxial stress, given:

σ°_{ij} = prior value of the stress,

ε°_{ij} = prior value of the strain,

ε_{ij} = current value of the strain,

the plasticity algorithm will employ the following approximation to equation (9)

$$\Delta\Sigma_{ij} = \Delta S^{e}_{ij} - \kappa\, \Sigma^{*}_{ij}\, \Delta\tilde{\lambda} \tag{18}$$

where

$$\Delta S^{e}_{ij} = \frac{E}{1+\nu}\left[\Delta\varepsilon_{ij} - \frac{1}{3}\,\Delta\varepsilon_{\gamma\gamma}\,\delta_{\alpha\beta}\right] \tag{19}$$

is known from the prescribed strain increment $\Delta\varepsilon_{ij} = \varepsilon_{ij} - \varepsilon^{\circ}_{ij}$ and where

$$\Sigma^{*}_{ij} = \Sigma^{\circ}_{ij} + \omega\,\Delta\Sigma_{ij} \qquad (0\leq\omega\leq 1) \tag{20}$$

is some intermediate value between the prescribed prior value Σ°_{ij} and the as yet to be determined current value $\Sigma_{ij} = \Sigma^{\circ}_{ij} + \Delta\Sigma_{ij}$. Equation (18) can be regarded as a forward, central, or backward finite difference depending on the value of ω:

$$\omega = \begin{cases} 0 \ ; \text{ forward difference} \\ \tfrac{1}{2} \ ; \text{ central difference} \\ 1 \ ; \text{ backward difference} \end{cases}$$

A second approximation often employed in algorithms (ref. 3, 4) involves replacing the yield condition imposed on the current value of the stress by a linear approximation to this condition. In terms of the previous example, this means that rather than the current stress satisfying the exact yield condition

$$2\Sigma^{\circ}_{ij}\,\Delta\Sigma_{ij} + \Delta\Sigma_{ij}\,\Delta\Sigma_{ij} = 0 \quad ^{*} \tag{21}$$

it is required to satisfy the linear approximation

---

*This relation follows from (11) on assuming that the prior stress satisfies the yield condition. This is safely assumed since a simple elastic calculation can be used to eliminate any portion of the strain increment inside the yield surface, for example see Table 4.7 in ref. 5.

$$2\,\Sigma^{o}_{ij}\,\Delta\Sigma_{ij} = 0 \qquad (22)$$

This approximation assumes that the stress increment is small enough to permit the square terms to be neglected in comparison with the linear terms. However, the approximation results in an error by computing a stress outside the yield surface. To understand the reason for this, consider equation (22) as a vector relation in stress space, as illustrated in Figure 2 for the case of a forward difference

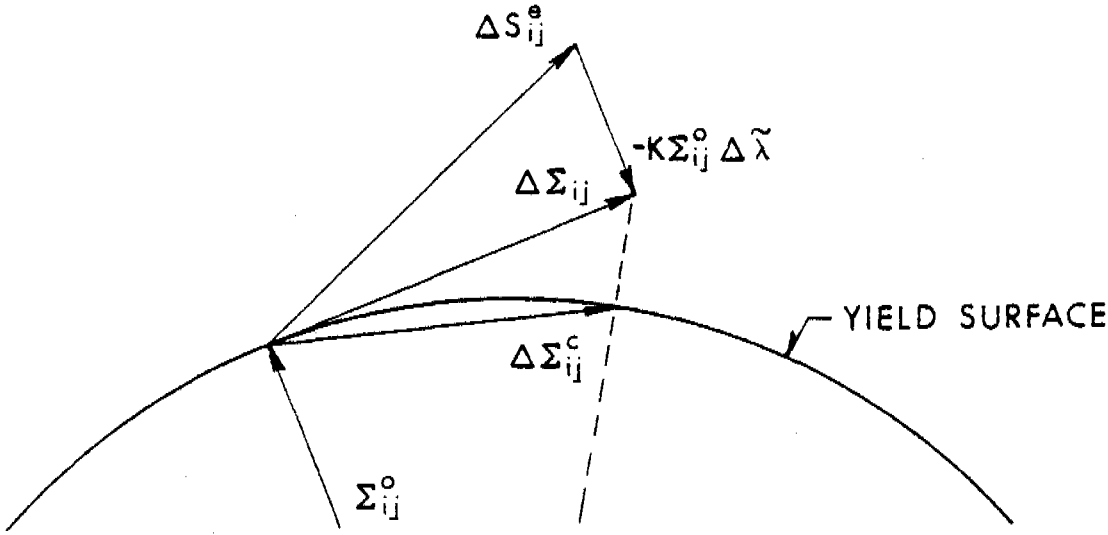


Figure 2. Illustration showing how the use of the linearized yield condition determines a stress increment outside the yield surface that requires correcting.

($\omega = 0$) flow rule approximation. Equation (22) requires the stress increment $\Delta\Sigma_{ij}$ to be perpendicular to the direction of the yield surface normal and hence tangent to the surface. Because the von Mises surface is strictly convex, the increment will determine a stress outside the yield surface unless a correction is applied, as for example shown in Figure 2 where a corrected stress increment Σ^{c}_{ij} is obtained by a radial correction. As pointed out in references 6 and 7, the lack of a correction in the kinematic hardening subroutine of the ADINA finite element program (ref. 3) had been found to cause a premature termination due to the cumulative effect of uncorrected increments. A correction to this subroutine based on the combined use of a central flow rule approximation and the exact yield condition has been implemented in the ADINA code and is detailed in reference 7. On the other hand, when the exact yield condition (20) is used, as illustrated in Figure 3, the parameter $\Delta\tilde{\lambda}$ takes on just the right value to give a stress on the yield surface.

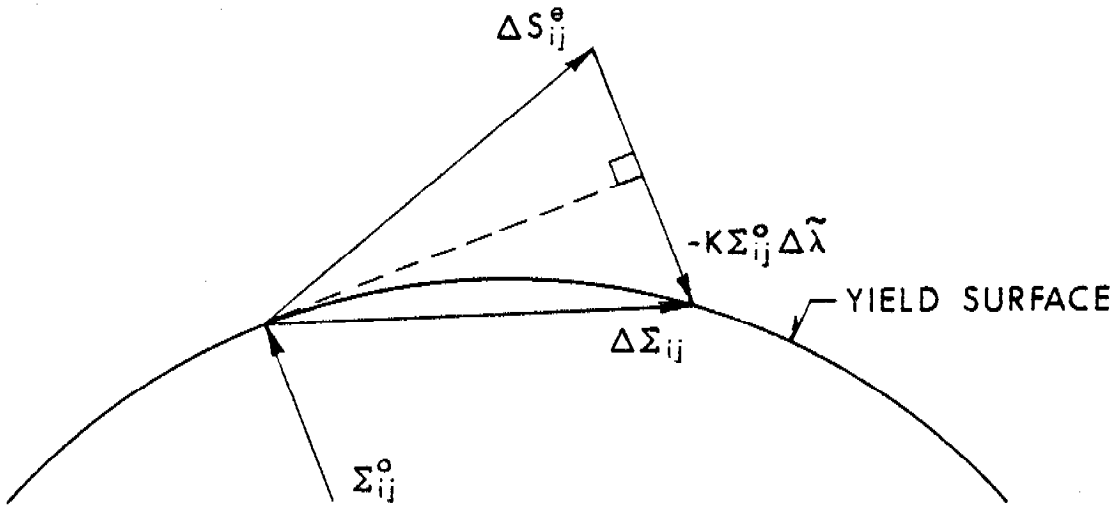


Figure 3. Illustration showing how the use of the exact yield condition maintains the stress on the yield surface.

While the use of the flow rule approximation is more or less dictated by the discrete nature of the code calculations, the use of the linearized yield condition is hard to justify. Since modern computers can solve complex algebraic relations very efficiently, there seems little reason not to use the exact quadratic expression. In fact, in the case of triaxial stress, when the exact condition (21) is used, the equation to be solved for $\Delta\tilde{\lambda}$ is at worst quadratic and if the central flow rule approximation ($\omega = \frac{1}{2}$) is used the relation on $\Delta\tilde{\lambda}$ becomes linear (ref. 7).

It is not uncommon practice in codes (ref. 3 and 8) to subincrement the plasticity algorithm in order to increase accuracy. Subincrementing involves dividing the strain increment into a number of equal subincrements, as depicted for example in Figure 4 for three subincrements. The plasticity algorithm is applied sequentially to each subincrement with the prior stress being updated at each step: $\Sigma^o_{ij} \to \Sigma^1_{ij} \to \Sigma^2_{ij}$, so that the normal direction changes at each subincrement step. In this way, subincrementing results in a closer simulation of the differential system (9) through (11). Subincrementing can be used with any of the flow rule approximations in combination with either the exact yield condition or the linearized condition (plus correction). We shall be evaluating the accuracy of a number of these approximations by comparing them with an exact solution that represents the limit of the subincrement process as the individual subincrements go to zero.

Corresponding remarks about approximating the flow rule and the yield condition and about subincrementing also apply in the case of biaxial stress.

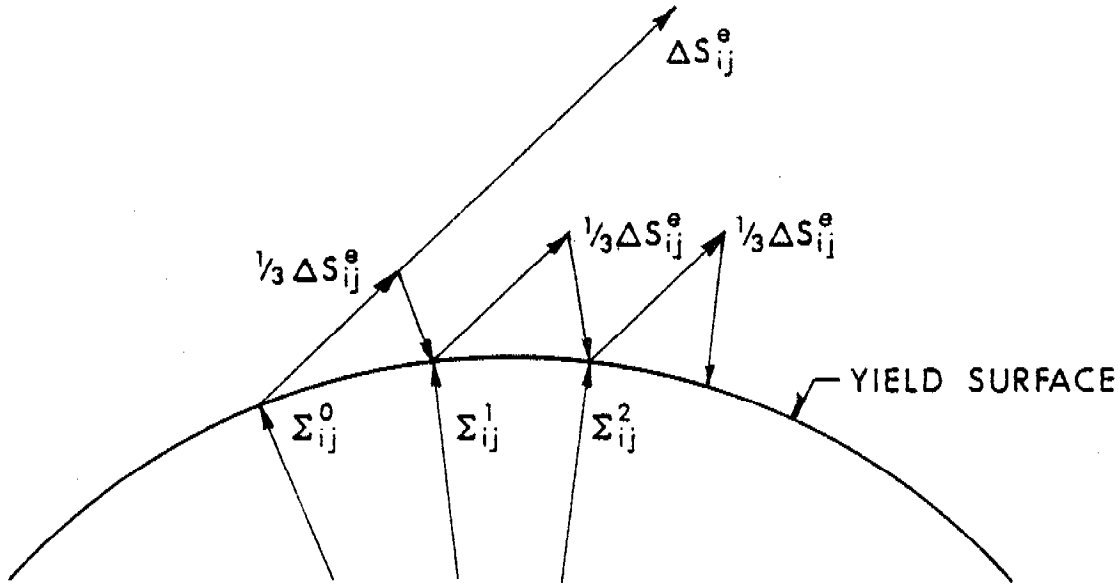


Figure 4. Illustration showing the division of the increment ΔS^e_{ij} into three equal subincrements in order to increase the accuracy of the plasticity calculation.

4. PROPORTIONAL INCREMENTAL STRAINING. In order to integrate the differential system (9) through (11) it is necessary to make an assumption about how the strain varies from the prior to the current step. We simply assume that the strain increases in some continuous way from its initial to its final value along the direction defined by the discrete strain increment. Hence, the components of the continuously increasing strain increment will be in the same proportion as the corresponding components of the discrete strain increment. In the case of triaxial stress, this assumption means that during the strain increment the forcing function S^e_{ij} in (9) is proportional to the increment ΔS^e_{ij}:

$$S^e_{ij} = f(\lambda)\ \Delta S^e_{ij} \tag{23}$$

where $f(\lambda)$ is a continuous function that increases from zero to unity as λ goes from zero to $\Delta\tilde{\lambda}$. With this assumption, (9) becomes

$$\frac{d\Sigma_{ij}}{d\lambda} = \frac{df}{d\lambda}\ \Delta S^e_{ij} - \kappa\ \Sigma_{ij} \tag{24}$$

where Σ_{ij} satisfies the yield condition (11) for all values of λ between the limits $0 \le \lambda \le \Delta\lambda$. At the limits we have:

$$\begin{aligned} \lambda = 0 \quad &\Rightarrow \quad f = 0 \quad \& \quad \Sigma_{ij} = \Sigma^\circ_{ij} \\ \lambda = \Delta\tilde{\lambda} \quad &\Rightarrow \quad f = 1 \quad \& \quad \Sigma_{ij} = \Sigma^\circ_{ij} + \Delta\Sigma_{ij} \end{aligned} \tag{25}$$

Equation (24) can be regarded as the continuous counterpart of the discrete flow equation (18). For the case of biaxial stress, the corresponding assumption yields a similar simplification of (14).

5. TRANSFORMATION OF TRIAXIAL EQUATIONS. In order to facilitate the analysis of the previous equations, a tranformation of coordinates in stress space is required. For the triaxial stress case, we transform to a set of orthogonal coordinates in the plane of the vectors Σ°_{ij} and ΔS_{ij}, while simultaneously normalizing coordinates with respect to the yield stress σ_y. Although not necessary, for convenience one coordinate axis is made to coincide with the direction of Σ°_{ij} by defining the basis vector a_{ij} as follows:

$$a_{ij} = \sqrt{\frac{3}{2}} \frac{\Sigma^{\circ}_{ij}}{\sigma_y} \tag{26}$$

It follows from the yield condition (11) that a_{ij} is a unit vector (i.e., $a_{ij}\, a_{ij} = 1$). Using the Gram-Schmitt process, a second unit vector b_{ij} orthogonal to the first (i.e., $a_{ij}\, b_{ij} = 0$) is defined by requiring that it satisfy the equation

$$\sqrt{\frac{3}{2}} \frac{\Delta S^{e}_{ij}}{\sigma_y} = \eta_1\, a_{ij} + \eta_2\, b_{ij} \tag{27}$$

The components of ΔS^{e}_{ij} relative to this new basis will be

$$\eta_1 = \sqrt{\frac{3}{2}} \frac{\Delta S^{e}_{ij}\, a_{ij}}{\sigma_y} \quad \& \quad \eta_2 = \sqrt{\frac{3}{2}} \frac{\Delta S^{e}_{ij}\, b_{ij}}{\sigma_y} \tag{28}$$

Resolving Σ_{ij} relative to this basis

$$\sqrt{\frac{3}{2}} \frac{\Sigma_{ij}}{\sigma_y} = \tau_1\, a_{ij} + \tau_2\, b_{ij} \tag{29}$$

the components of Σ_{ij} become

$$\tau_1 = \sqrt{\frac{3}{2}} \frac{\Sigma_{ij}\, a_{ij}}{\sigma_y} \quad \& \quad \tau_2 = \sqrt{\frac{3}{2}} \frac{\Sigma_{ij}\, b_{ij}}{\sigma_y} \tag{30}$$

It follows from (26) that initially these components will have the values $\tau^{\circ}_1 = 1$ and $\tau^{\circ}_2 = 0$. By substituting (27) and (29) in (18), the discrete flow equation is

transformed into the pair

$$\Delta\tau_1 = \eta_1 - \tau_1^* \, \Delta\Lambda \quad \& \quad \Delta\tau_2 = \eta_2 - \tau_2^* \, \Delta\Lambda \tag{31}$$

where, as before, τ_a^* (a = 1, 2) are intermediate values:

$$\tau_a^* = \tau_a^\circ + \omega \, \Delta\tau_a \qquad (0 \leq \omega \leq 1) \tag{32}$$

and where the flow parameter is redefined as

$$\Delta\Lambda = \kappa \, \Delta\tilde{\lambda} \tag{33}$$

With this transformation the yield condition (11) becomes the unit circle

$$\tau_1^{\,2} + \tau_2^{\,2} = 1 \tag{34}$$

which both the initial and final values of τ_1 and τ_2 are to satisfy.

Applying the same transformation to the equation of proportional incremental straining causes the continuous flow equation (24) to reduce to

$$\frac{d\tau_1}{d\Lambda} = \frac{df}{d\Lambda} \; \eta_1 - \tau_1 \quad \& \quad \frac{d\tau_2}{d\Lambda} = \frac{df}{d\Lambda} \; \eta_2 - \tau_2 \tag{35}$$

From (25) the limit values follow:

$$\begin{aligned} &\Lambda = 0 \;\Rightarrow f = 0 \quad \& \quad \tau_a = \tau_a^\circ \\ &\Lambda = \Delta\Lambda \Rightarrow f = 1 \quad \& \quad \tau_a = \tau_a^\circ + \Delta\tau_a \end{aligned} \tag{36}$$

where a = 1, 2. The components τ_a are required to satisfy the yield condition (34) continuously between the limits. The results of this transformation for both the discrete and the continuous case are summarized in Table 1.

The problem in transformed coordinates is portrayed in Figure 5, reduced to its essential ingredients. Because the yield condition corresponds to a unit circle, it will be automatically satisfied when polar coordinates are employed. Hence, let us set

$$\begin{aligned} &\tau_1 = \cos\theta \qquad \tau_2 = \sin\theta \\ &\eta_1 = \eta \cos\alpha \quad \eta_2 = \eta \sin\alpha \end{aligned} \tag{37}$$

Table 1. Summary of Transformed Equations

		Triaxial Stress	Biaxial Stress
Coordinate Transformation		$\Sigma_{ij} \rightarrow \tau_1, \tau_2$ $\Delta S^e_{ij} \rightarrow \eta_1, \eta_2$	$\Sigma_{11}, \Sigma_{12}, \Sigma_{22} \rightarrow \tau_1, \tau_2, \tau_3$ $\Delta\sigma^e_{11}, \Delta\sigma^e_{12}, \Delta\sigma^e_{22} \rightarrow \eta_1, \eta_2, \eta_3$
Flow Equations	Discrete	$\Delta\tau_1 = \eta_1 - \tau_1^* \Delta\Lambda$ $\Delta\tau_2 = \eta_2 \quad \tau_2^* \Delta\Lambda$	$\Delta\tau_1 = \eta_1 - (1-\delta)\tau_1^* \Delta\Lambda$ $\Delta\tau_2 = \eta_2 - \tau_2^* \Delta\Lambda$ $\Delta\tau_3 = \eta_3 - \tau_3^* \Delta\Lambda$
	Continuous	$\frac{d\tau_1}{d\Lambda} = \frac{df}{d\Lambda} \eta_1 - \tau_1$ $\frac{d\tau_2}{d\Lambda} = \frac{df}{d\Lambda} \eta_2 - \tau_2$	$\frac{d\tau_1}{d\Lambda} = \frac{df}{d\Lambda} \eta_1 - (1-\delta)\tau_1$ $\frac{d\tau_2}{d\Lambda} = \frac{df}{d\Lambda} \eta_2 - \tau_2$ $\frac{d\tau_3}{d\Lambda} = \frac{df}{d\Lambda} \eta_3 - \tau_3$
Yield Condition		$\tau_1^2 + \tau_2^2 = 1$	$\tau_1^2 + \tau_2^2 + \tau_3^2 = 1$

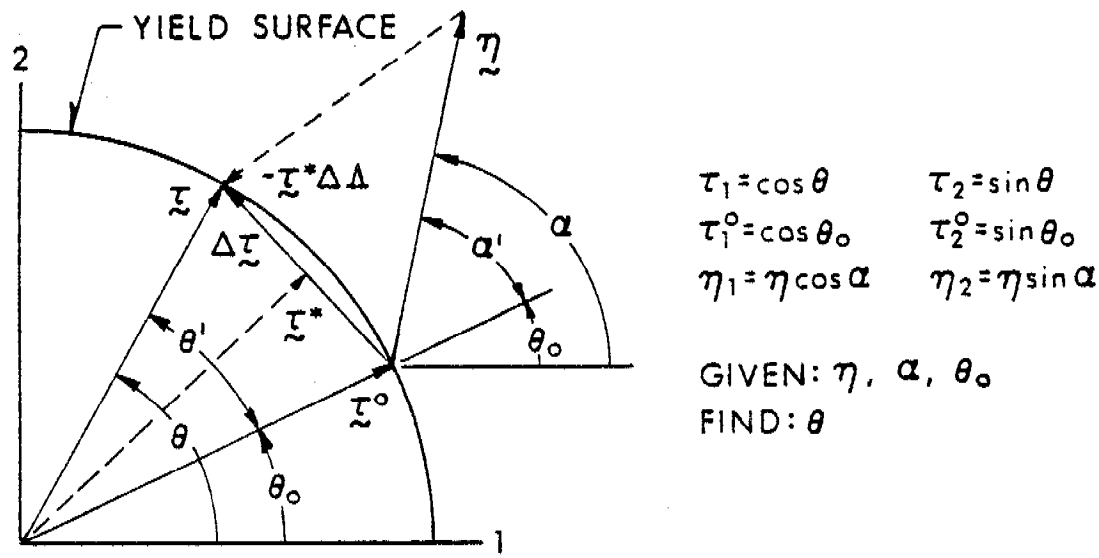


Figure 5. Representation of triaxial stress problem in transformed stress coordinates.

where

$$\eta = \sqrt{\frac{3}{2}} \frac{(\Delta S^e_{ij} \; \Delta S^e_{ij})^{\frac{1}{2}}}{\sigma_y} \tag{38}$$

Substituting (37) into the continuous flow equations (35), we obtain a differential equation

$$\frac{d\theta}{df} = \eta \sin(\alpha-\theta) \tag{39}$$

on the single unknown θ. Integrating this equation between the limits (36), we obtain the solution

$$\tan\frac{\theta-\alpha}{2} = \tan\frac{\theta_o-\alpha}{2} \; e^{-\eta} \tag{40}$$

where θ_o represents the initial value of the polar angle and θ the final value. Making a similar substitution in the discrete flow equations (31) and (32) results in

$$\sin(\theta-\theta_o) = \eta\,[(1-\omega)\sin(\alpha-\theta_o) + \omega \sin(\alpha-\theta)] \tag{41}$$

Notice that both the discrete and the continuous solutions are independent of the initial stress, as specified by θ_o; that is, both solutions are essentially

functions of differences

$$\theta' = \theta - \theta_o \quad \& \quad \alpha' = \alpha - \theta_o \tag{42}$$

We shall find that this is not true in the case of biaxial stress. It is for the sake of comparison with the biaxial case that we have chosen not to utilize the fact that initially $\theta_o = 0$, which earlier was shown to follow from (25).

6. TRANSFORMATION OF BIAXIAL EQUATIONS. The transformation of coordinates for the biaxial stress case is composed of a rotation that brings the coordinate axes into coincidence with the principal axes of the yield surface ellipsoid followed by a normalization of the ellipsoid to a unit sphere. The transformation is defined by the equations

$$\tau_1 = \frac{3}{2}\,\frac{\Sigma_{11} + \Sigma_{22}}{\sigma_y}, \quad \tau_2 = \frac{\sqrt{3}}{2}\,\frac{\Sigma_{11} - \Sigma_{22}}{\sigma_y}, \quad \tau_3 = \sqrt{3}\,\frac{\Sigma_{12}}{\sigma_y}$$

$$\eta_1 = \frac{3}{2}\,\frac{\Delta\sigma^e_{11} + \Delta\sigma^e_{22}}{\sigma_y}, \quad \eta_2 = \frac{\sqrt{3}}{2}\,\frac{\Delta\sigma^e_{11} - \Delta\sigma^e_{22}}{\sigma_y}, \quad \eta_3 = \sqrt{3}\,\frac{\Delta\sigma^e_{12}}{\sigma_y} \tag{43}$$

so that τ_a and η_a (a = 1, 2, 3) are the transformed components of $\Sigma_{\alpha\beta}$ and $\Delta\sigma^e_{\alpha\beta}$, respectively. Under this transformation the yield condition (15) becomes

$$\tau_1^2 + \tau_2^2 + \tau_3^2 = 1 \tag{44}$$

On comparing (9) and (14) it easily follows that the discrete flow equation, which is the biaxial stress counterpart of (18), is transformed into the trio

$$\begin{aligned} \Delta\tau_1 &= \eta_1 - (1-\delta)\,\tau_1^*\,\Delta\Lambda \\ \Delta\tau_2 &= \eta_2 - \tau_2^*\,\Delta\Lambda \\ \Delta\tau_3 &= \eta_3 - \tau_3^*\,\Delta\Lambda \end{aligned} \tag{45}$$

where τ_a^* (a = 1, 2, 3) are intermediate values as given by (32), where $\Delta\Lambda$ is defined by (33), and where

$$\delta = 2\,\frac{\mu}{\kappa} \tag{46}$$

Similarly, the continuous flow equation for biaxial stress is tranformed into

$$\frac{d\tau_1}{d\Lambda} = \frac{df}{d\Lambda}\eta_1 - (1-\delta)\tau_1$$

$$\frac{d\tau_2}{d\Lambda} = \frac{df}{d\Lambda}\eta_2 - \tau_2 \qquad (47)$$

$$\frac{d\tau_3}{d\Lambda} = \frac{df}{d\Lambda}\eta_3 - \tau_3$$

where the limit values (36) apply with a = 1, 2, 3.

For the sake of comparison, the transformed biaxial equations are also summarized in Table 1. It is clear that for both the discrete and the continuous equations, the triaxial stress equations are simpler than the biaxial stress equations, there being only two equations involved and these being more symmetric. The biaxial equations are more complex because of the occurrence of the δ term which as we can see from (17) and (46) is caused by the elastic compressibility of the material. In fact, when the material is incompressible, so that $\nu = \frac{1}{2}$ and hence $\delta = 0$, a simple coordinate rotation bringing a coordinate plane into the plane of the vectors τ° and η transforms the three biaxial equations into the two triaxial equations.

As in the triaxial stress case, because the yield condition for the biaxial case (44) corresponds to a unit sphere, it will be automatically satisfied when the angle coordinates θ and ϕ, as given by equations

$$\tau_1 = \cos\theta \qquad \tau_2 = \sin\theta\cos\phi \qquad \tau_3 = \sin\theta\sin\phi \qquad (48)$$

are used. Writing vector η in terms of angle coordinates:

$$\eta_1 = \eta\cos\alpha \qquad \eta_2 = \eta\sin\alpha\cos\gamma \qquad \eta_3 = \eta\sin\alpha\sin\gamma \qquad (49)$$

where

$$\eta = \sqrt{\frac{3}{2}}\,\frac{(\Delta\sigma^e_{\alpha\beta}\,\Delta\sigma^e_{\alpha\beta} + \Delta\sigma^e_{\alpha\alpha}\,\Delta\sigma^e_{\beta\beta})^{\frac{1}{2}}}{\sigma_y} \qquad (50)$$

and substituting these and the previous relations in the continuous flow equations, a pair of differential equations on θ and ϕ are obtained:

$$\frac{d\theta}{df} = \eta\,\frac{(1-\delta)\cos\theta\sin\alpha\cos(\gamma-\phi) - \sin\theta\cos\alpha}{1 - \delta\cos^2\theta}$$

$$\frac{d\theta}{df} = \eta\,\frac{\sin\alpha\sin(\gamma-\phi)}{\sin\theta} \qquad (51)$$

These equations have so far resisted efforts to find an exact integral. Consequently, a simpler case of the biaxial stress equations was selected for study by assuming that $\phi = \gamma$ (= 0 for convenience), so that only the single equation

$$\frac{d\theta}{df} = \eta \frac{(1-\delta) \cos\theta \sin\alpha - \sin\theta \cos\alpha}{1 - \delta \cos\theta} \tag{52}$$

needs to be integrated. This equation still is more complex than its triaxial counterpart (39), again because of the δ term. Physically it corresponds to biaxial stress situations in which the shear component of stress vanishes (i.e. $\sigma_{23} = \Sigma_{23} = 0$), as for example happens in cylindrical symmetric thin shell problems. In such situations the last of the discrete biaxial equations (45) is trivially satisfied, while the substitution of (48) and (49) into the first two reduces them to the single equation

$$\begin{aligned} &\sin(\theta-\theta_o) - \delta(\sin\theta - \sin\theta_o)[(1-\omega)\cos\theta_o + \omega\cos\theta] \\ &\quad = \eta\{(1-\delta)\sin\alpha[(1-\omega)\cos\theta_o + \omega\cos\theta] \\ &\qquad - \cos\alpha[(1-\omega)\sin\theta_o + \omega\sin\theta]\} \end{aligned} \tag{53}$$

which is the counterpart of (41). Before analyzing these equations, we replace the angle α by β, defined by the relation

$$\tan\beta = (1-\delta)\tan\alpha \tag{54}$$

so that (52) and (53) become:

$$(1-\delta\cos^2\theta)\frac{d\theta}{df} = \eta' \sin(\beta-\theta) \tag{55}$$

and

$$\begin{aligned} &\sin(\theta-\theta_o) - \delta(\sin\theta-\sin\theta_o)[(1-\omega)\cos\theta_o + \omega\cos\theta] \\ &\quad = \eta'[(1-\omega)\sin(\beta-\theta_o) + \omega\sin(\beta-\theta)] \end{aligned} \tag{56}$$

where

$$\eta' = \eta[\cos^2\alpha + (1-\delta)^2\sin^2\alpha]^{\frac{1}{2}} \tag{57}$$

Notice that if $\delta = 0$, then, as expected, (55) and (56) reduce to the triaxial equations (39) and (41).

The solution to (55) with the appropriate limit values is obtained in implicit form as

$$\tan \frac{\theta-\beta}{2} = \tan \frac{\theta_o-\beta}{2} e^{-H(\theta)} \tag{58}$$

where

$$H(\theta) = \frac{\eta' - \delta [\cos (\theta+\beta) - \cos (\theta_o+\beta)]}{1 - \delta \cos^2 \beta} \tag{59}$$

7. COMPARISON OF APPROXIMATIONS: TRIAXIAL STRESS. In this section we present a comparison of the differences between the exact solution to the triaxial case (40) and four particular approximations to (41):

a. Forward difference (ω=0) and linearized yield (22) (plus radial correction)

b. Forward difference (ω=0) and exact yield (21)

c. Central difference (ω=½) and exact yield

d. Backward difference (ω=1) and exact yield

We have already noted that in the triaxial stress case the solutions are independent of θ_o, so that only α' and η need be varied, see Figure 5. For each value of α' and η the difference $\Delta\theta$ between the value of θ' computed using one of the above approximations and the value using the exact solution will be determined. Due to symmetry, we need only consider values of α' between 0° and 90°; notice that when α' = 0 both the exact solution and the approximations give the trivial solution θ' = o for all values of η. The value of η, as we can see from its definition (38) or from Figure 5, measures the ratio of the magnitude of the increment ΔS^e_{ij} to the yield stress. The angle θ' (in radians) can be regarded as measuring the magnitude of the stress trajectory on the yield surface relative to the yield stress.

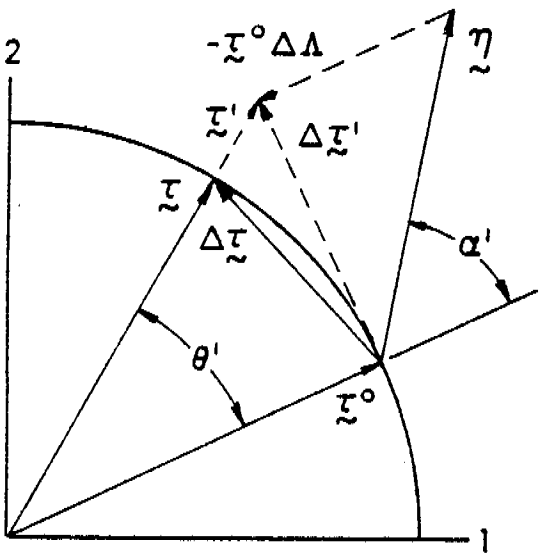
The solutions for the four approximations can be obtained by geometric construction, as shown in Figures 6 through 9. The exact solution is obtained from (40):

$$\tan \frac{\theta'}{2} = \frac{\sinh \frac{\eta}{2} \sin \alpha'}{\cosh \frac{\eta}{2} + \sinh \frac{\eta}{2} \cos \alpha'} \tag{60}$$

The difference between the value of θ' determined from an approximation and the exact value is defined as the error:

$$\Delta\theta = \theta'_{\text{Approx}} - \theta'_{\text{Exact}} \tag{61}$$

Within the range $0° \leq \alpha' \leq 90°$, the solution θ' will be positive; hence, $\Delta\theta$ will be positive or negative depending on whether an approximation over or underpredicts.



FORWARD DIFFERENCE

$$\Delta\underset{\sim}{\tau}' = \underset{\sim}{\eta} - \underset{\sim}{\tau}^{\circ}\Delta\Lambda$$

LINEARIZED YIELD

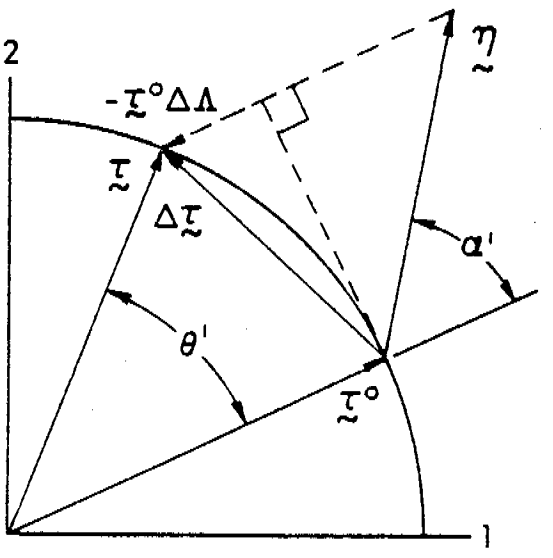
$$\underset{\sim}{\tau}^{\circ} \cdot \Delta\underset{\sim}{\tau}' = 0$$

RADIAL CORRECTION

$$\underset{\sim}{\tau} = \frac{\underset{\sim}{\tau}'}{|\underset{\sim}{\tau}'|}$$

$$\boxed{\tan\theta' = \eta \sin\alpha'}$$

Figure 6. Solution for the forward difference and linearized yield condition approximation with radial correction.



FORWARD DIFFERENCE

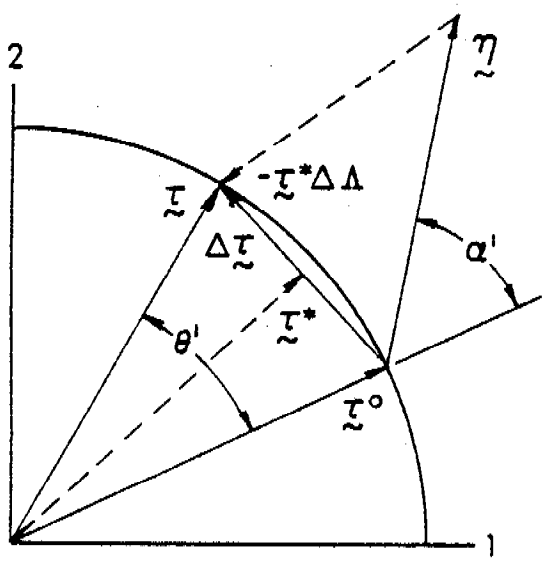
$$\Delta\underset{\sim}{\tau} = \underset{\sim}{\eta} - \underset{\sim}{\tau}^{\circ}\Delta\Lambda$$

EXACT YIELD

$$\underset{\sim}{\tau} \cdot \underset{\sim}{\tau} = 1$$

$$\boxed{\sin\theta' = \eta \sin\alpha'}$$

Figure 7. Solution for the forward difference and exact yield condition approximation.



CENTRAL DIFFERENCE

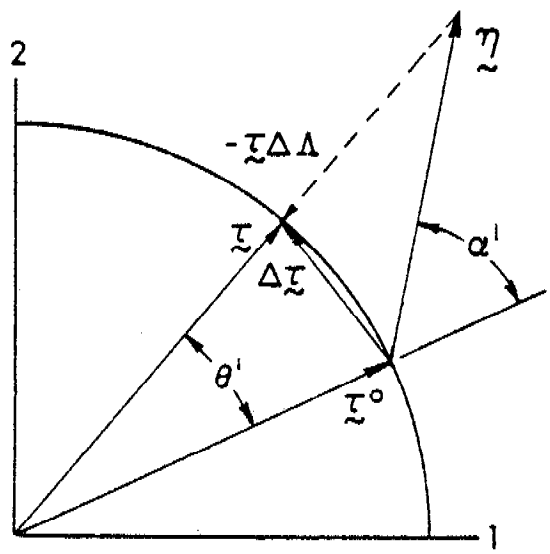
$$\Delta\underset{\sim}{\tau} = \underset{\sim}{\eta} - \underset{\sim}{\tau}^{*}\Delta\Lambda \; ; \; \underset{\sim}{\tau}^{*} = \frac{\underset{\sim}{\tau}^{\circ} + \underset{\sim}{\tau}}{2}$$

EXACT YIELD

$$\underset{\sim}{\tau} \cdot \underset{\sim}{\tau} = 1$$

$$\tan\frac{\theta'}{2} = \frac{\eta \sin\alpha'}{2 + \eta\cos\alpha'}$$

Figure 8. Solution for the central difference and exact yield condition approximation.



BACKWARD DIFFERENCE

$$\Delta\underset{\sim}{\tau} = \underset{\sim}{\eta} - \underset{\sim}{\tau}\Delta\Lambda$$

EXACT YIELD

$$\underset{\sim}{\tau} \cdot \underset{\sim}{\tau} = 1$$

$$\tan\theta' = \frac{\eta \sin\alpha'}{1 + \eta\cos\alpha'}$$

Figure 9. Solution for the backward difference and exact yield condition approximation.

In Figure 10 we plot the errors $\Delta\theta$ at 5° interval in α' for the four approximations with r = 10% and 50%. For both values of η the graphs show the same trends. When the exact yield condition is used, both the forward and central differences overpredict over the entire range, while the backward difference underpredicts. When the linearized yield condition and the forward difference is used, the approximation overpredicts over most of the range except near the end as α' approaches 90° where the error crosses over and underpredicts. Also, except for the central difference approximation which achieves its maximum error at the end of the range, all the other approximations reach their maximum errors more or less near mid-range. It is clear that over the entire range the central difference approximation is clearly superior, with only the forward difference/linearized yield approximation giving a smaller error in the neighborhood of the cross-over point.

Table 2 gives the maximum error for each approximation over a range of values

Table 2. Maximum Error as a Function of η

η	Linearized Yield	Exact Yield		
	Forward Diff	Forward Diff	Central Diff	Backward Diff
.05	.000610	.000632	.000010	-.000596
.10	.002379	.002555	.000083	-.002271
.15	.005210	.005844	.000279	-.004897
.20	.009007	.010554	.000657	-.008329
.25	.013672	.016739	.001274	-.012440
.30	.019111	.024500	.002181	-.017116
.35	.025306	.034043	.003426	-.022359
.40	.032135	.045368	.005050	-.027999
.45	.039519	.058597	.007090	-.033943
.50	.047384	.074306	.009576	-.040110

of η between 5% and 50%. At 5% the central difference approximation is about 60 times more accurate than any of the other approximations. As η increases to 50%, the advantage of the central approximation diminishes to an accuracy of four to eight times greater than its competitors. As is to be expected, the table makes clear the benefits of subincrementing for any approximation. For example, using the central difference approximation, if η = .5, the maximum possible

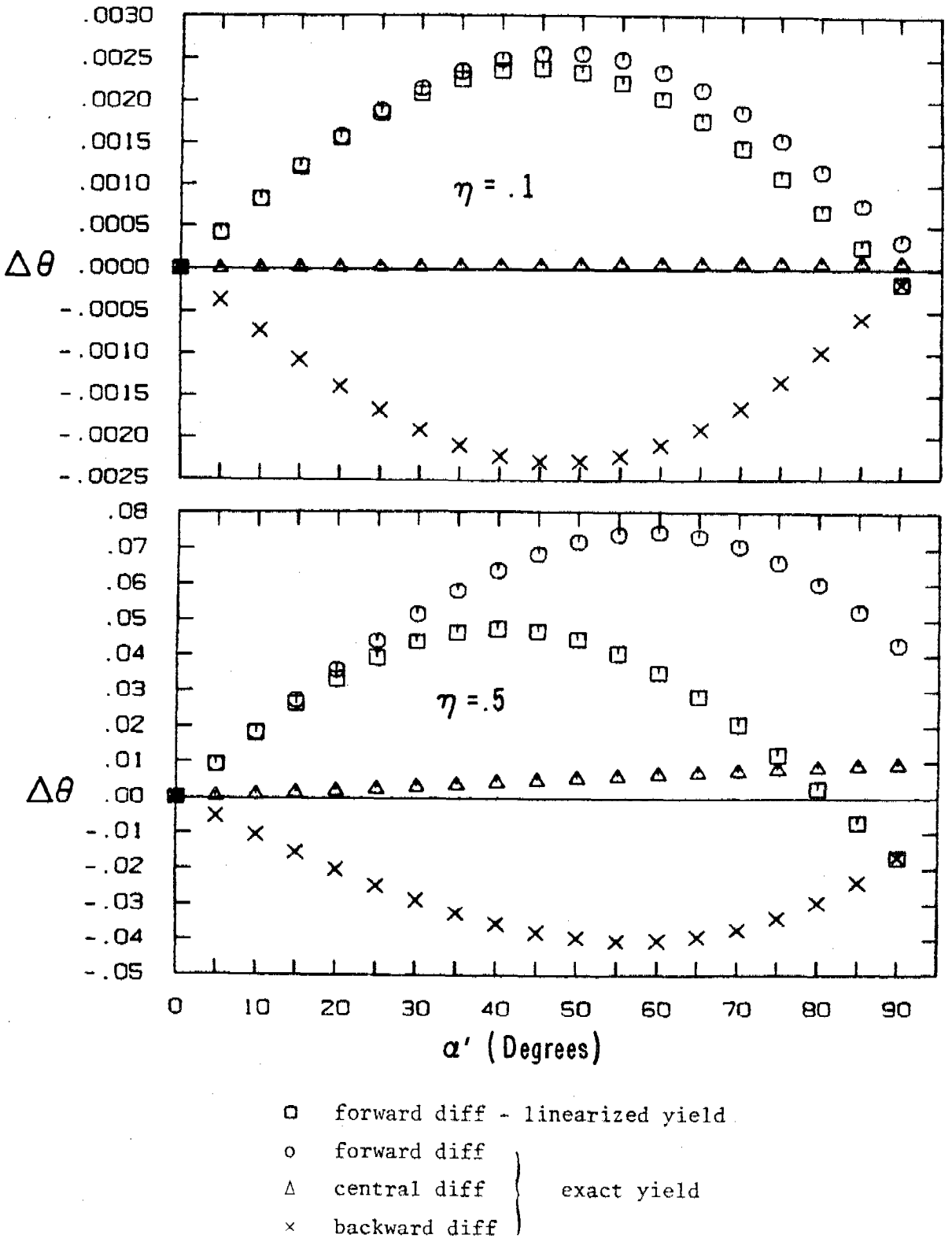


Figure 10. Graphs of the error $\Delta\theta$ for triaxial stress approximations versus the angle α' for two values of the ratio η.

error can be .009576; if five subincrements are used so that $\eta = .1$ for each subincrement, the maximum possible error is reduced to .000415; and if ten subincrements are used, the maximum error is further reduced to .00020. It is also clear that if subincrementing is automatically performed whenever $\eta > .05$, then the central approximation will retain its 60 : 1 advantage in accuracy over the other approximations.

8. COMPARISON OF APPROXIMATIONS: BIAXIAL STRESS. In this section we compare the deviations from the exact solution that result from three particular approximations to the biaxial stress equations. As remarked in Section 6, the comparison will not be for the general biaxial equations, but for the special case where the shear component vanishes. However, this comparison should provide some estimates on the magnitude of the errors connected with these approximations for the general case.

The three approximations to be compared are:

a. Forward difference ($\omega=0$) and linearized yield (plus radial correction)

b. Forward difference ($\omega=0$) and exact yield

c. Central difference ($\omega=\frac{1}{2}$) and exact yield

The equation for the first approximation follows from equations (21), (43), (45), (48), (49), and (54) after considerable manipulation and can be written in terms of polar coordinates as follows:

$$\tan \theta' = \frac{\eta' \sin\beta'}{1-\delta \cos^2\theta_o} \tag{62}$$

where θ' and η' are defined by (42) and (57) and where

$$\beta' = \beta - \theta_o \tag{63}$$

with β given by (54). The equations for the second and third approximations follow from equation (56) after setting $\omega = 0$ and $\frac{1}{2}$, respectively:

$$[Y(\theta')]^2 - 2(1-\delta \cos^2\theta_o)\, Y(\theta') + \eta' \sin\beta'(\eta' \sin\beta' - 2\,\delta \sin\theta_o \cos\theta_o) = 0 \tag{64}$$

$$[Y(\theta')]^3 - [2(1-\delta \cos^2\theta_o) + \eta' \cos\beta']\,[Y(\theta')]^2 + \eta' \sin\beta'\,(\eta' \sin\beta' - 4\,\delta \sin\theta_o \cos\theta_o)\, Y(\theta') - (\eta' \sin\beta')^2\,[2(1-\delta \sin^2\theta_o) + \eta' \cos\beta'] = 0 \tag{65}$$

where the function containing the unknown θ' is defined as:

$$Y(\theta') = \frac{\eta' \sin\beta'}{\tan\frac{\theta'}{2}} \tag{66}$$

Notice that the second approximation involves solving a quadratic equation for the unknown and the third involves a cubic. Interestingly, when these approximations are implemented in a computer code for the general biaxial stress situation, the equations used to determine the flow parameter increments $\Delta\lambda$ turn out to be, respectively, quadratic and cubic.

The exact value of θ' to which the above obtained approximate values will be compared is found by solving the transcendental equation (derived from (58) and (59))

$$2\delta \left[\frac{(\cos\beta + X \sin\beta)^2}{1 + X^2} - \frac{(\cos\beta + X_o \sin\beta)^2}{1 + X_o^2}\right] = \eta' + (1-\delta \cos^2\beta) \ln\frac{X}{X_o} \tag{67}$$

for the variable

$$X = \tan\frac{\beta'-\theta'}{2} \tag{68}$$

where

$$X_o = \tan\frac{\beta'}{2} \tag{69}$$

This transcendental equation is easily solved on the computer using a Newton-Raphson algorithm.

As in the previous section, the error is defined as the difference between the values of θ' obtained from each of the three approximations and the exact solution. Unlike the triaxial case, both the exact and the approximate solutions will depend on the initial value θ_o, so that now, in additon to α' and η, θ_o will have to be varied over a suitable range. Moreover, due to the θ_o dependence, the equations no longer have a trivial solution when $\alpha' = 0$ and the solutions are no longer symmetrically distributed about $\alpha' = 0$. Hence, α' must be varied over the range values from -90° to +90°, while due to symmetry θ_o need only vary from 0 to 90°. Also, the trival solution is obtained when $\beta' = 0$, corresponding to the value

$$\alpha' = \alpha_o' \equiv \tan^{-1}\left[\frac{\delta \sin\theta_o \cos\theta_o}{1 - \delta \cos^2\theta_o}\right] \tag{70}$$

At this angle, exact and approximate solutions change sign. Therefore, to insure that overpredictions will be positive and underpredictions negative, the error needs to be redefined as

$$\Delta\theta = [\theta'_{Approx} - \theta'_{Exact}] \text{ sign}(\theta'_{Exact}) \tag{71}$$

With this definition in mind, the errors resulting from solving the previous equations at 10° intervals in α' for the values $\theta_o = 0°$, 45°, 90° and $\eta = 10\%$ (assuming that $\delta = \frac{1}{3}$) are plotted in Figure 11. Comparison with the $\eta = 10\%$ graph in Figure 10 shows that the magnitudes of the errors are approximately the same and that the same general trends persist: the central approximation is clearly superior and achieves its maximum error at the ends of the interval ($\alpha' = \pm 90°$), while the linearized forward and exact forward approximations reach their maxima near the middle range ($\alpha' = \pm 45°$). We also see the effects of the dependence on the initial angle θ_o in the diminishing errors as $\theta_o : 0° \to 90°$ and in the shifting of the $\Delta\theta = 0$ solutions from the origin; e.g., when $\theta_o = 45°$ the trivial solutions occur at $\alpha_o' = 11.3°$, but at the values $\theta_o = 0°$ and 90° there is no shifting due to symmetry.

Figure 12 shows the graphs of the errors for the same three values of θ_o when $\eta = 50\%$. The general trends noted before are still present with the central approximation still superior and comparison with the $\eta = 50\%$ graph in Figure 10 confirms that magnitudes of the errors are still close.

The comparison of the magnitudes of the errors between the triaxial and the biaxial cases suggests that for a given value of α' the triaxial error for each approximation is the average of the biaxial errors over the range of θ_o. This more or less is confirmed numerically in Figures 13 and 14, where the errors using the central differences approximation with $\eta = 10\%$ are plotted against the angle θ_o for values of $\alpha' = 0°$, 30°, 60°, 90°. Each graph shows how the error varies over the range $-90° \quad \theta_o \quad 90°$ for $\delta = \frac{1}{3}$ (representing a genuine biaxial situation) and for $\delta = 0$ (corresponding to the triaxial situation, as remarked at the end of Section 7). We see that except for the case where $\alpha' = 0$, the errors in the biaxial case do indeed tend to cluster about the error in the triaxial case. Hence, the errors computed using the simpler triaxial stress equations should provide good estimates on the magnitudes of the errors to be expected when using biaxial stress approximations. As further confirmation of this supposition, we present in Table 3 for a range of values of θ_o, the maximum errors for the three approximations considered here when $\eta = 10\%$. Comparison of these errors with the maximum errors in Table 2 when $\eta = 10\%$ shows that the errors are of the same order.

Table 3. Maximum Error as a Function of θ_o using the Biaxial Stress Approximations with $\eta = .10$.

θ_o	Linearized Yield	Exact Yield	
	Forward Diff	Forward Diff	Central Diff
0°	.003557	.003782	.000123
15°	.003257	.004057	.000128
30°	.002920	.003776	.000117
45°	.002279	.003110	.000095
60°	.001906	.002448	.000074
75°	.001670	.001969	.000061
90°	.001540	.001718	.000056

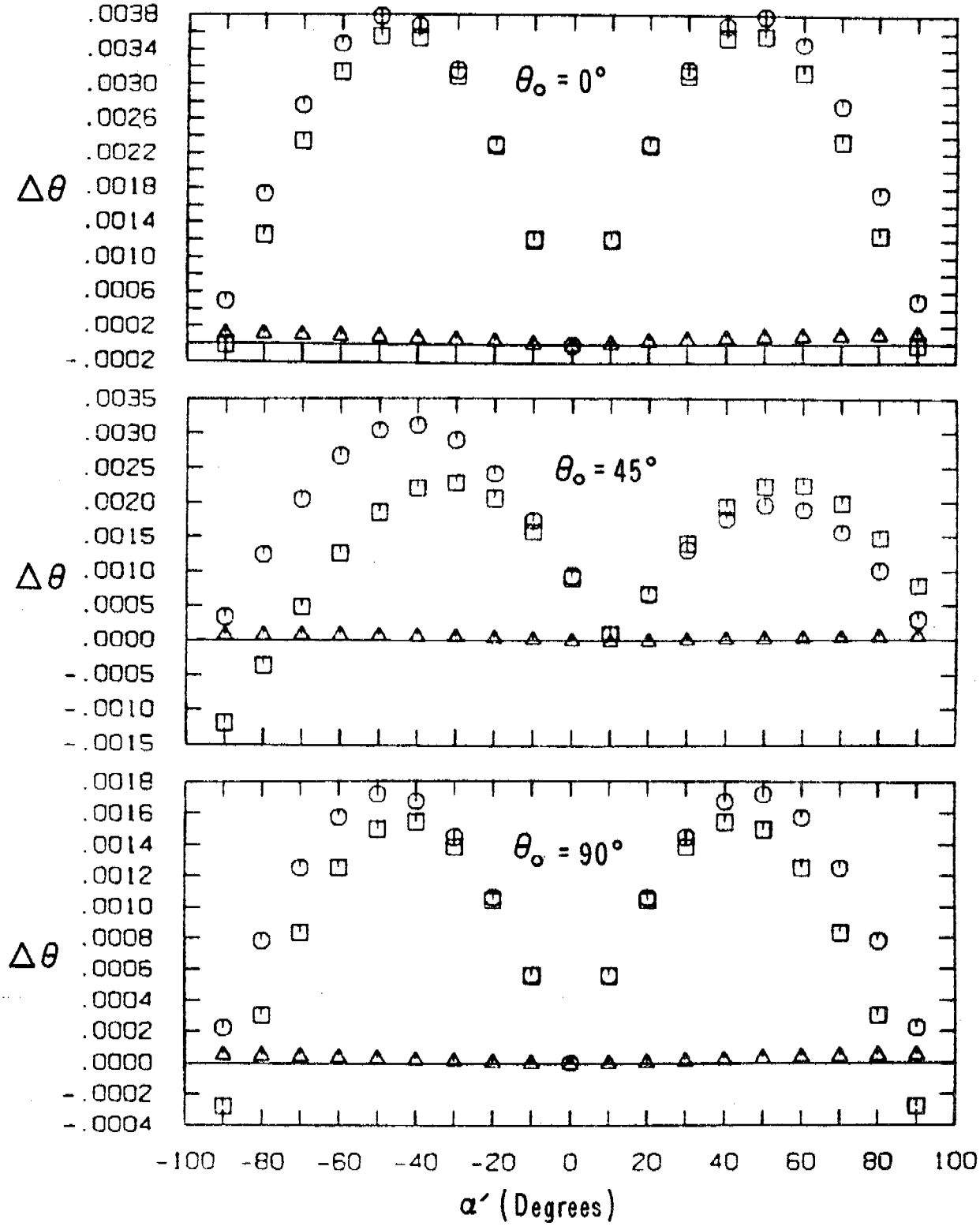


Figure 11. Graphs of the error $\Delta\theta$ for the biaxial approximations versus the angle α' for three values of θ_o when $\eta = .10$.

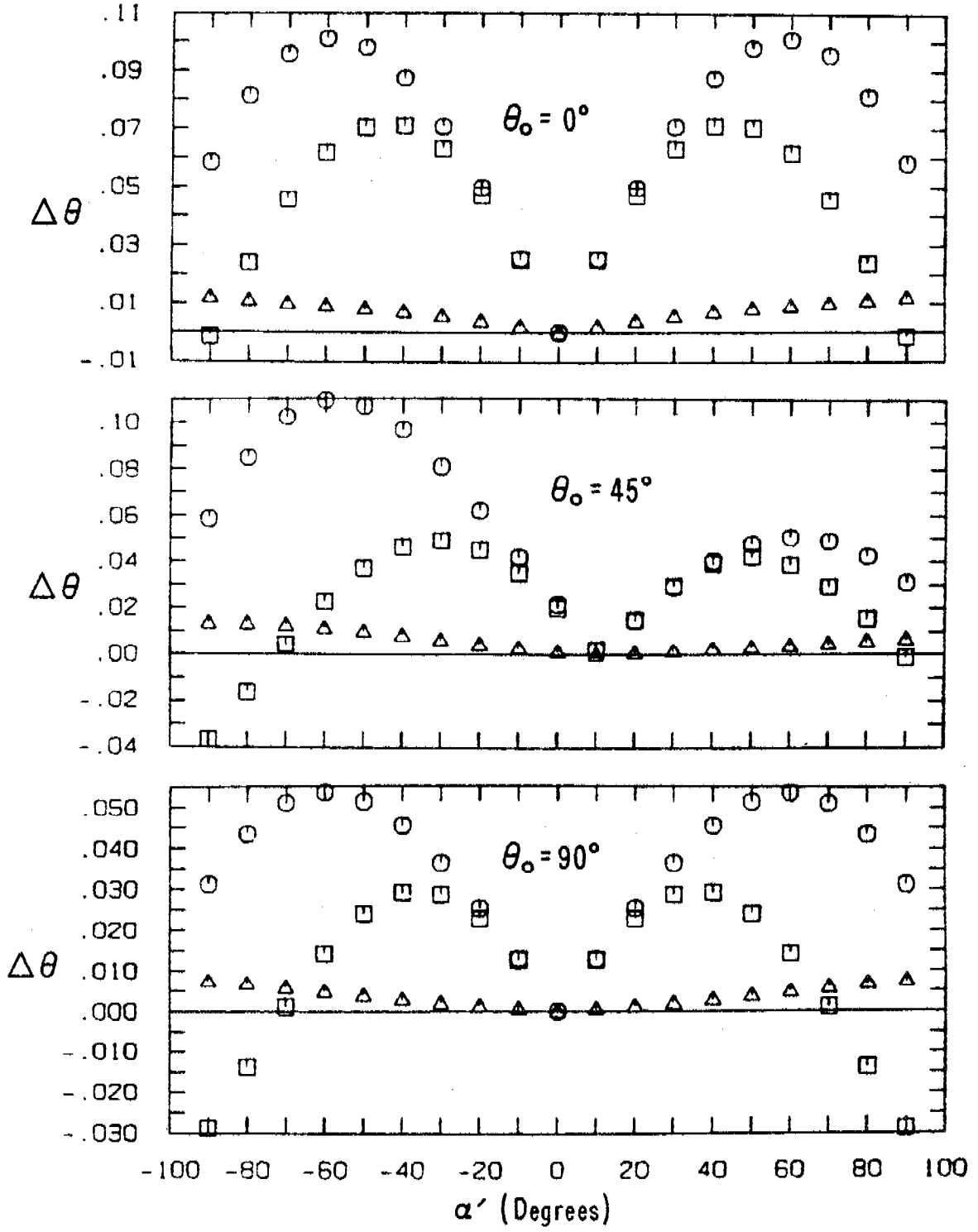


Figure 12. Graphs of the error $\Delta\theta$ for the biaxial approximations versus the angle α' for three values of θ_o when $\eta = .50$.

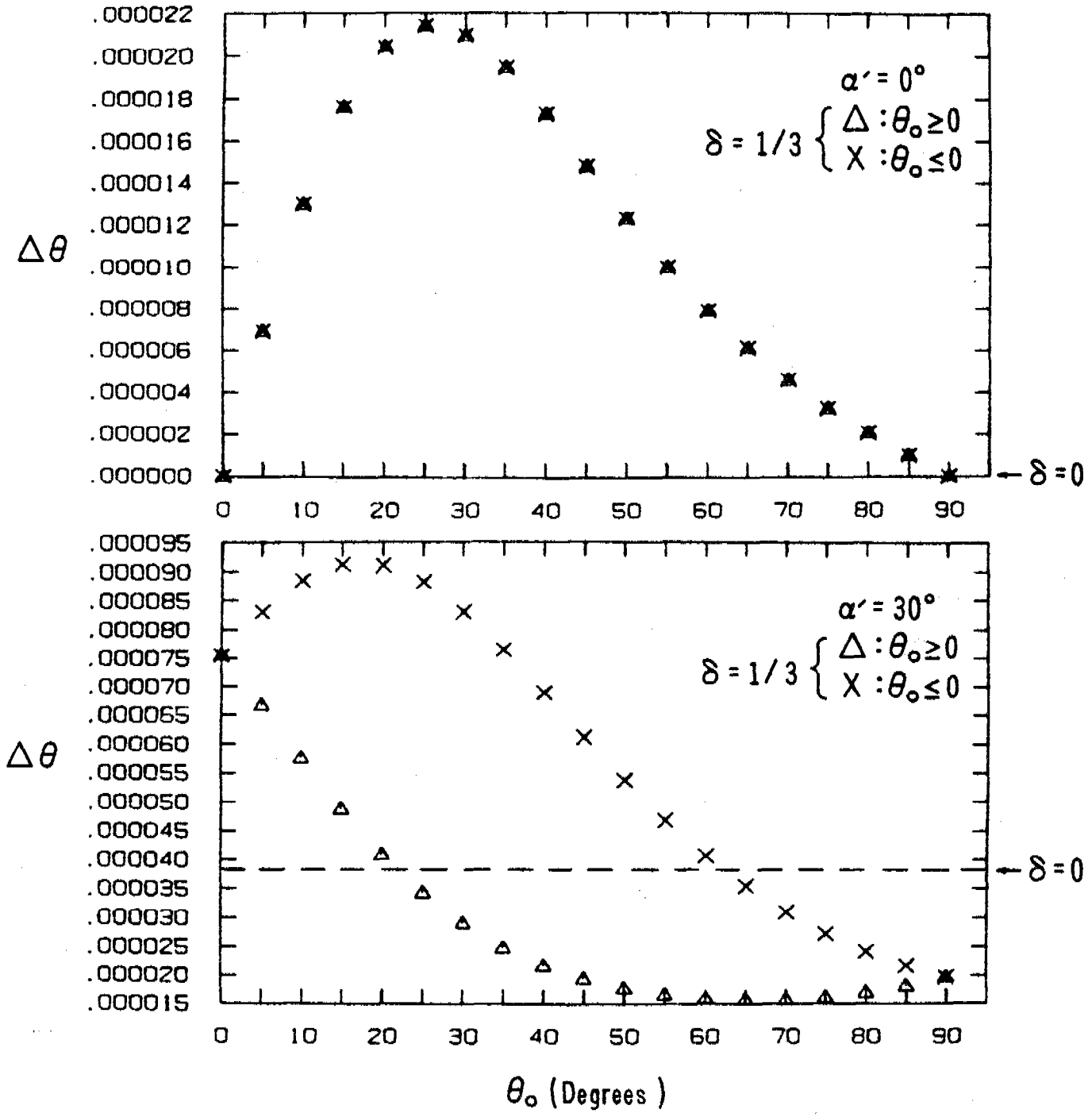


Figure 13. Graphs of the variation in the errors between the triaxial ($\delta = 0$) and biaxial ($\delta = \frac{1}{3}$) central difference approximation for $\alpha' = 0°$ & $30°$.

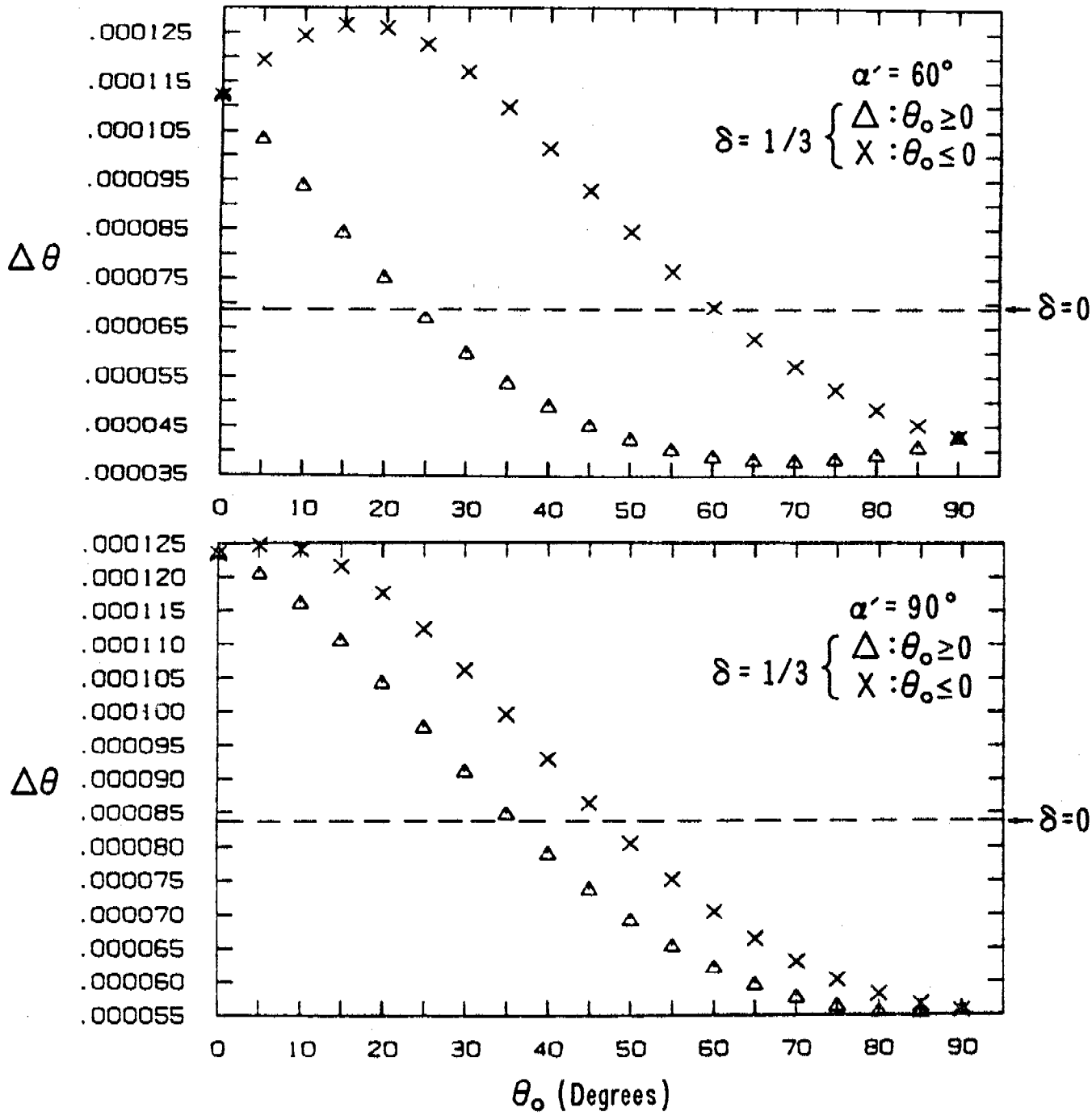


Figure 14. Graphs of the variation in the errors between the triaxial ($\delta = 0$) and biaxial ($\delta = \frac{1}{3}$) central difference approximation for $\alpha' = 60°$ & $90°$.

9. SUMMARY AND CONCLUSION. The Prandtl-Reuss equations for a linear strain hardening material have been integrated exactly for a prescribed discrete strain increment for the cases of biaxial stress and triaxial stress. A comparison of these solutions with a number of approximations to these equations commonly used in response programs has been performed. The comparison shows that the central finite difference approximation to the flow rule in combination with the exact yield condition is the most accurate. For example, the maximum error using this approximation will be in the order of .01% of the yield stress for an elastic stress increment equal to 10% of yield. The comparison also enables us to quantify the beneficial effects of subincrementing in plasticity approximations. Hence, if subincrementing is automatically employed whenever the elastic stress increment exceeds 5% of yield, then the maximum error for an elastic increment as great as 50% of yield will be limited to approximately .01% of yield. Also, the comparison shows that the simpler triaxial stress analysis provides good estimates on the errors to be expected with the equivalent biaxial approximations.

The central finite difference approximation, which strangely enough is little used, has been recently implemented in the ADINA response program (ref. 3 and 7) for the case of triaxial stress and will soon be programed for the biaxial stress case. It is also planned to implement the central difference approximation in the REPSIL (ref. 8) and in the PETROS (ref. 9 and 10) series of shell response programs at the earliest opportunity.

An intriguing question that might have occurred to the reader is why not implement the exact solution. In the triaxial stress case where the solution (60) is explicit there is little doubt that it would be more effective, not so much in improving accuracy, for the central approximation is very accurate, but in obviating the need for subincrementing. Hence, the implementation of the exact triaxial stress solution is one item in future plans.

As for implementing the exact solution in the biaxial stress case, the main objection is that the solution is for a special plane stress situation in which there is no shear stress component. Hence, the solution cannot be utilized in most programs that analyze plane stress or Kirchhoff shell problems. Moreover, the solution is implicit and requires some numerical scheme, such as Newton-Raphson, to obtain an answer; hence, there may be little to choose in terms of efficiency between solving for the implicit exact solution and using the central difference approximation with subincrementing.

ACKNOWLEDGEMENTS. This research was performed with funds from the U.S. Army Research in Ballistics Program. The assistance of Mr. Henry L. Wisniewski in selecting numerical techniques and generating counterpart computer programs is gratefully acknowledged.

REFERENCES

1. W. Prager, "The Theory of Plasticity - A Survey of Recent Achievements", Proc. Instn. Mech. Engr., London, Vol. 169, 1955, pp. 41-57.

2. R. Hill, The Mathematical Theory of Plasticity, Oxford University Press, London, 1950, pp. 38-45.

3. K. J. Bathe, "ADINA, A Finite Element Program for Automatic Dynamic Incremental Nonlinear Analysis", Report No. 82448-1, Sep. 1975 (revised Nov. 1979), Acoustics and Vibration Lab., Mech. Engrg. Dept., M.I.T., Cambridge, MA.

4. J. W. Leech, "Finite-Difference Calculation Method for Large Elastic-Plastic Dynamically-Induced Deformations of General Thin Shells", AFFDL-TR-66-171, Dec. 1966, Air Force Flight Dynamics Lab., Wright-Patterson Air Force Base, OH.

5. K. J. Bathe, "Static and Dynamic Geometric and Material Nonlinear Analysis Using ADINA", Report No. 82448-2, May 1976 (revised May 1977), Acoustics and Vibration Lab., Mech. Engrg. Dept., M.I.T., Cambridge, MA.

6. A. D. Gupta, J. M. Santiago, and H. L. Wisniewski, "An Improved Strain Hardening Characterization in the ADINA Code Using the Mechanical Sublayer Concept", First Chautauqua on Finite Element Modeling, Harwichport, MA, Sep. 15-17, 1980.

7. J. M. Santiago and H. L. Wisniewski, "An Improved Formulation of the Kinematic Hardening Model in the ADINA Code", to be submitted for publication as a BRL Report, U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, MD.

8. J. M. Santiago, H. L. Wisniewski, and N. J. Huffington, Jr., "A User's Manual for the REPSIL Code", BRL Report No. 1744, Oct. 1974, U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, MD.

9. S. D. Pirotin, B. A. Berg, and E. A. Witmer, "PETROS 3.5: New Developments and Program Manual for the Finite-Difference Calculation of Large Elastic-Plastic Transient Deformations of Multilayer Variable-Thickness Shells", BRL Contract Report No. 211, February 1975, U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, MD.

10. S. D. Pirotin, B. A. Berg, and E. A. Witmer, "PETROS 4: New Developments and Program Manual for the Finite-Difference Calculations of Large Elastic-Plastic, and/or Viscoelastic Transient Deformations of Multilayer Variable-Thickness (1) Thin Hard-Bonded, (2) Moderately-Thick Hard-Bonded, or (3) Thin Soft-Bonded Shells", BRL Contract Report No. 316, September 1976, U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, MD.

MICROCOMPUTER IMPLEMENTATION OF CONTROL ALGORITHMS FOR WEAPON POINTING AND STABILIZATION

Jonathan Korn
ALPHATECH, Inc.
Burlington, Massachusetts

and

Edward Carroll
Ronald Johnson
Barry Kullack
Headquarters Army Armament
Research & Development Command
Dover, New Jersey

ABSTRACT

This paper describes two discrete-time LQ regulator algorithms which have been implemented and tested on a microcomputer-based servo control system located in the Stabilization Research Laboratory facility at ARRADCOM. These algorithms include reduced-order observers for estimating system states and disturbances, and LQ-based digital control laws for precision stabilization in the presence of external disturbances.

1. INTRODUCTION

Modern control theory and digital microprocessors represent two emerging technologies which have significantly altered the perception of the gun pointing and stabilization problem from that of an isolated subsystem design process to one in which all system state variables and error sources (including target-induced errors [1]) are considered in the gun control law development. Reference [2], in particular, illustrates the performance advantages associated with modern gun control law design. In this example, an LQ controller was designed, implemented and evaluated in live firing tests on a helicopter armament system. The dispersion associated with the modern control design was 1.26 mr as compared to 4.2 mr for the original design. The control law implementation in this effort was performed using fixed configuration analog electronics which imposed practical constraints in terms of exploiting the full benefits of modern observer theory and disturbance accomodation methodology. These constraints are, for the most part, eliminated by performing the control law implementation in software on a microcomputer-based controller. The advantages in this approach include reduced hardware complexity, cost, increased design flexibility, commonality and grown potential, as well as improved system performance.

The discrete-time LQ regulator designs presented in this paper represent the first of a series of microcomputer-based control concepts which will be subjected to extensive laboratory testing, followed by implementation and evaluation of the XM-97 Helicopter Turret System, discussed in [2]. One of the designs discussed includes a four-state observer for estimating and suppressing recoil torque disturbances. This design is similar to

the one proposed in [2], but not implemented, due to the limitations of the analog electronics used for the control law implementation.

2. DESCRIPTION OF LABORATORY TEST FIXTURE

The inertia wheel test fixture used to evaluate the discrete-time regulator algorithms developed in this paper consist of two DC torque motors which drive the inertial wheel, an 800 Hz preamp, a demodulator, and a torque drive amplifier. The testing facility and a representative block diagram of the test fixture are shown in Figs. 1a and 1b. Note that one torque motor, denoted by subscript 1, is used for regulation and tracking while the second troque motor (denoted by subscript 2) is used to generate torque input disturbances. A listing of the motor/inertia wheel parameters is given in Table 1.

For practical reasons we neglect the armature inductance in the motors' model. The open-loop state-space representation of the inertia wheel test fixture is then given by:

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{1}{J}\left(\frac{K_{T_1}}{R_1}K_{b_1} + \frac{K_{T_2}}{R_2}K_{b_2}\right) \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{K_{in}K_{T_1}}{J\,R_1} \end{bmatrix} e_{v_1} + \begin{bmatrix} 0 \\ \frac{1}{J} \end{bmatrix} (T_2 - f) \qquad (2\text{-}1)$$

where θ and $\dot{\theta}$ represent motor shaft angular motion and velocity, f is coulomb friction (modeled as a constant bias disturbance), and T_2 is an external torque disturbance. In our work, T_2 simulates recoil disturbance, and we replace this torque notation with r(t). In general, recoil disturbance is modeled as a damped sinusoid. Because of practical constraints, we simplify this model to a pure sinusoid with frequency ω_r. The equation representing the disturbance states is therefore,

$$\begin{bmatrix} \dot{r} \\ \ddot{r} \\ \dot{f} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -\omega_r^2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} r \\ \dot{r} \\ f \end{bmatrix} \qquad (2\text{-}2)$$

Note that only the position $y=K_{out}$ is available for online measurement.

3. CONTROL LAW SYNTHESIS

3.1 Problem Formulation

The problem is restated here in a form amenable for the control law synthesis. We are given the dynamic system represented in the state-space as

$$\dot{\underline{x}}_1 = A_1\underline{x}_1 + F\underline{x}_2 + \underline{b}_1 u \qquad (3\text{-}1)$$

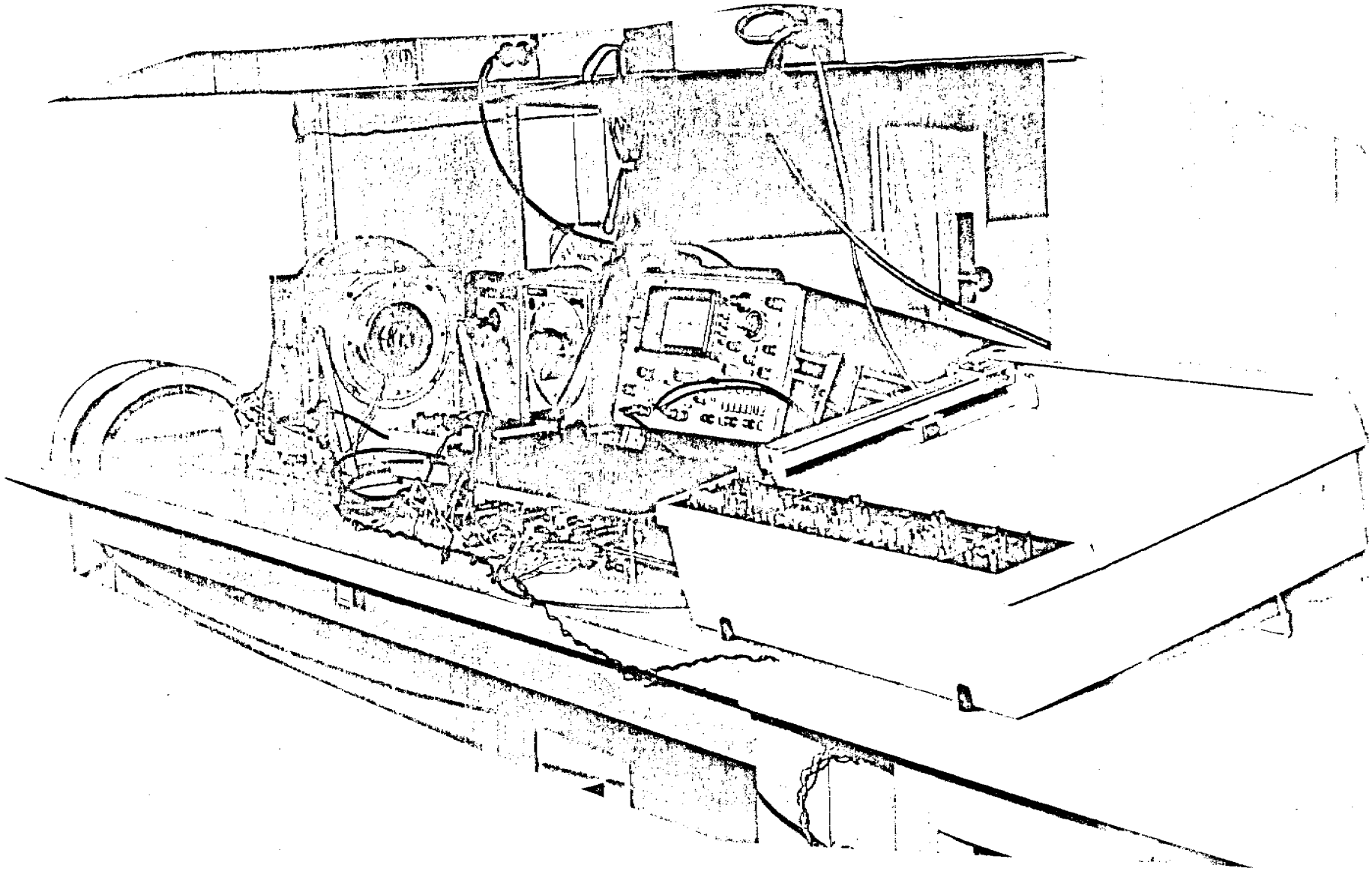


Figure 1a. Testing Facility at ARRADCOM.

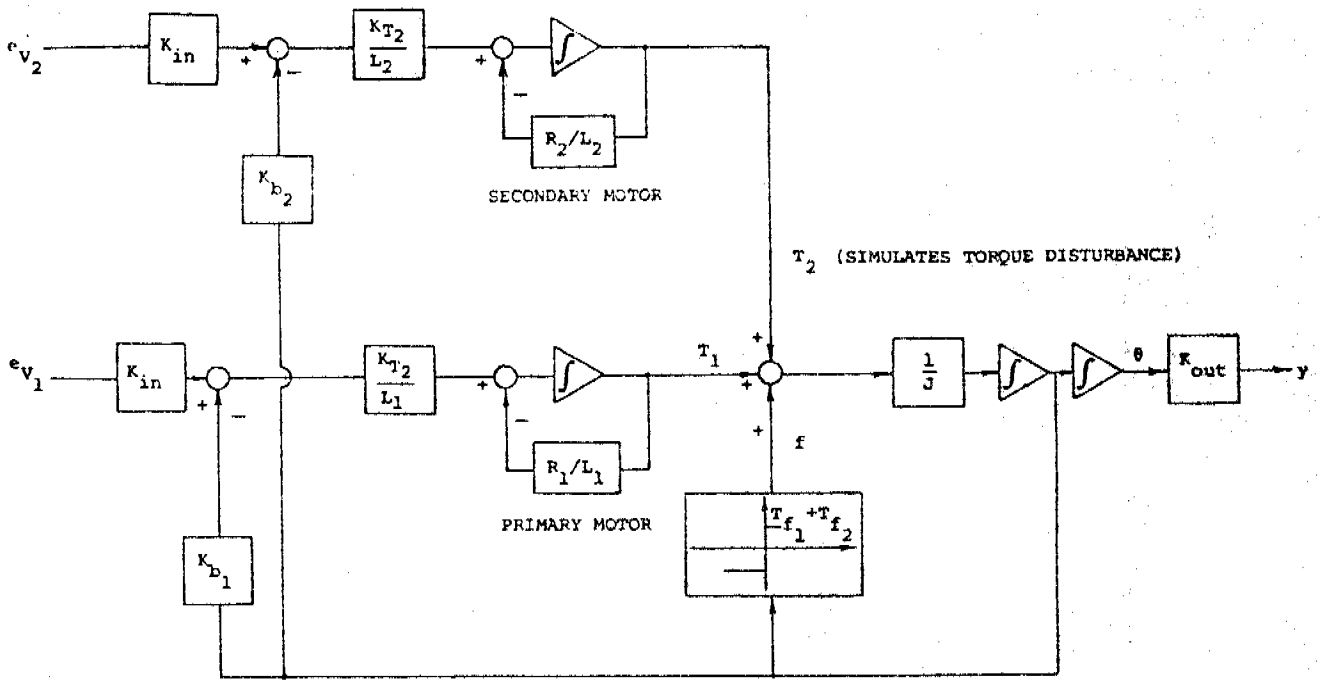


Figure 1b. Two Torque Motor Diagram of Inertia Wheel.

TABLE 1. MOTOR/INERTIA WHEEL PARAMETERS

	Motor 1	Motor 2	Units
Back emf, K_b	0.347	0.54	vdc/rad/sec
Torque, K_T	0.257	0.4	ft-#/amp
Armature Resistance, R	28	55	ohms
Armature Inductance, L	0.0125	0.025	henry
Armature Time Constant (Measured), $T = \frac{L}{R}$	$\frac{1}{1500}$	$\frac{1}{1500}$	sec
Torque at 30V	0.218	0.258	ft-#
Maximum Friction, T_f	0.1	0.05	ft-#
Maximum unbalance	0.1	0.1	ft-#
Wheel Inertia, J	0.416		in-#-sec^2
K_{in} = 12 K_{out} = 10			

$$\dot{\underline{x}}_2 = A_2\underline{x}_2$$
$$y = C\underline{x}_1 \qquad (3\text{-}1)$$

where $\underline{x}_1$ represents the plant state, $\underline{x}_2$ the torque disturbance state, $u \triangleq e_{v_2}$ control signal, and y the plant output. The state variables are as follows:

$$\underline{x}_1' = [\theta, \dot{\theta}]$$
$$\underline{x}_2' = [r \quad \dot{r} \quad f] \qquad (3\text{-}2)$$

where θ = motor shaft angular displacement (gun pointing angle)
$\dot{\theta}$ = motor shaft angular velocity
r = recoil torque disturbance
$\dot{r}$ = recoil torque rate
f = coulomb friction and any other torque bias disturbance.

The matrices A_1, A_2, and $\underline{b}_1$ are as per Eqs. 2-1 and 2-2, and $F = \begin{bmatrix} 0 & 0 & 0 \\ \frac{1}{J} & 0 & -\frac{1}{J} \end{bmatrix}$, $c = [K_{out} \quad 0]$.

The simplified plant/disturbance configuration is shown in Fig. 2.

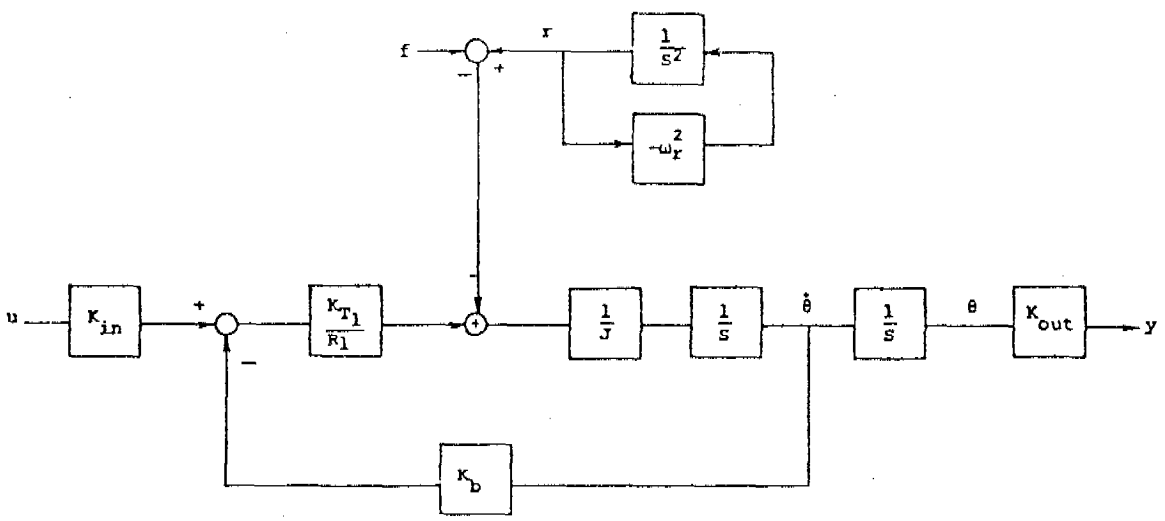


Figure 2. Plant/Torque Disturbance Configuration.

Several remarks are pertinent:

1. The back emf gains K_{b_1} and K_{b_2} are combined into a single gain, viz.,

$$K_b = K_{b_1} + \frac{R_1}{K_{T_1}} \frac{K_{T_2}}{R_2} K_{b_2} \qquad (3\text{-}3)$$

2. The linearized model (Eq. 3-1) requires that the coulomb friction term, f, be represented as a simple torque bias term, independent of $\dot{\theta}$.

3. For our design, we consider the power amplifier gain K_{in}=12, and the resolver gain K_{out}=10 as a part of the physical plant.

The design problem now is twofold:

1. Given the plant output measurement, y(t), obtain an estimate of the state $\underline{x}_1$ and $\underline{x}_2$.

2. Design a control law that would regulate the pointing angle, $\theta(t)$, in face of any persisting torque disturbances. In other words, the control law requirement is to null-out $\underline{x}_2$ and to regulate the plant state $\underline{x}_1$.

We consider first the control law design.

3.2 Control

It is convenient to synthesize the control law first, and then to proceed with the estimation scheme. The usual practice is to rewrite the system (Eq. 3-1) with the augmented plant/distrubance dynamics state

$$\underline{x} = [\underline{x}_1' \quad \underline{x}_2']' \tag{3-4}$$

such that

$$\begin{aligned} \dot{\underline{x}} &= A\underline{x} + \underline{b}u \\ y &= C\underline{x} \end{aligned} \tag{3-5}$$

where

$$A = \begin{bmatrix} A_1 & F \\ 0 & A_2 \end{bmatrix} \quad \underline{b} = \begin{bmatrix} \underline{b}_1 \\ 0 \end{bmatrix} \quad C = [C_1 \quad 0] \quad . \tag{3-6}$$

Upon substitution of the parameter values of Table 1, and with the "recoil" frequency of 10 Hz (ω_r = 62.83 rad/sec), one obtains

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & -0.2 & 28.85 & 0 & -28.85 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -3948 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \underline{b} = \begin{bmatrix} 0 \\ 3.18 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad C = [10 \quad 0 \quad 0 \quad 0 \quad 0] \tag{3-7}$$

We obtain now the feedback control law, $u = -K_C\hat{\underline{x}}$, by a straightforward application of optimal control theory. The optimal gains, K_C, are derived by minimizing the cost functional

$$J(u) = \int_0^\infty (\underline{x}'Q\underline{x}+u'Ru)dt \quad , \quad Q \geq 0 \quad , \quad R > 0 \quad . \tag{3-8}$$

where R=1 and Q=diag[$5\cdot10^4$ 0 0 0 0]. This choice yields the continuous-time gains

$$K_c = [223.6 \quad 11.8 \quad 1.851 \quad 0.084 \quad -9.07] \tag{3-9}$$

which yield the closed-loop pole locations for $\theta,\dot{\theta}$ at $-18 \pm j18$.

However, since our design requires a discrete-time control law, we reformulate our optimal control problem. This process is carried out in two steps, as indicated in the appendix. It yields the equivalent discrete-time system

$$\underline{x}_{k+1} = \Phi\underline{x}_k + \underline{\Gamma}u_k \tag{3-10}$$

and with a sampling interval $\Delta T = 0.01$ sec,

$$\Phi = \begin{bmatrix} 1 & 9.990\ 10^{-3} & 1.395\ 10^{-3} & 4.712\ 10^{-6} & -1.442\ 10^{-3} \\ 0 & 9.980\ 10^{-1} & 2.696\ 10^{-1} & 1.395\ 10^{-3} & -2.882\ 10^{-1} \\ 0 & 0 & 8.090\ 10^{-1} & 9.355\ 10^{-3} & 0 \\ 0 & 0 & -36.93 & 8.090\ 10^{-1} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \underline{\Gamma} = \begin{bmatrix} 1.589\ 10^{-4} \\ 3.177\ 10^{-2} \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3-11}$$

The equivalent discrete-time control law is then

$$u_k = -K_d\underline{x}_k = -[182.8 \quad 10.72 \quad 1.563 \quad 0.0775 \quad -9.07]\underline{x}_k \tag{3-12}$$

3.3 Estimation

Implementation of the control law (Eq. 3-12) requires that an estimate be made of the state $\underline{x}_k$. This estimate is made using a discrete-time, reduced-order observer, with the equations summarized in the appendix, this design procedure yields an observer state, $\underline{z}_k$, which evolves according to

$$\underline{z}_{k+1} = A_1\underline{z}_k + \underline{A}_2y_k + \underline{A}_3u_k \quad , \tag{3-13}$$

and the state estimate is given by

$$\hat{\underline{x}}_k = A_z\underline{z}_k + \underline{A}_yy_k \quad . \tag{3-14}$$

A necessary constraint in this design is the observer bandwidth, as it must not exceed the Nyquist sampling rate.

In this paper, we treat two basic cases:

1. The "recoil" disturbance is not operative, and the only torque that disturbes the gun is coulomb friction. The observer needs to estimate $\dot{\theta}$ and f only.

2. All disturbances are operative; the observer estimates $\dot{\theta}$, r, $\dot{r}$, and f.

3.3.1 Two-State Observer — Since the recoil dynamics can be eliminated from the state Eq. 3-10, we obtain the second-order observer which estimates the angle-rate and friction terms:

$$\underline{z}_{k+1} = \begin{bmatrix} 0.3711 & -0.1978 \\ 0.6855 & 0.9011 \end{bmatrix} \underline{z}_k + \begin{bmatrix} -2.590 \\ 4.981 \end{bmatrix} y_k + \begin{bmatrix} 0.0218 \\ 0.0109 \end{bmatrix} u_k$$

$$\hat{\underline{x}}_k = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \underline{z}_k + \begin{bmatrix} 0.1 \\ 6.275 \\ -6.862 \end{bmatrix} y_k \qquad (3\text{-}15)$$

The design variables $\tilde{Q}$ and $\tilde{R}$ (see appendix) were chosen as

$$\tilde{R} = 1 \quad , \quad \tilde{Q} = \text{diag}(0,100) \quad . \qquad (3\text{-}16)$$

and the observer poles are at 0.64 ± j25. The selection of $\tilde{R}$ and $\tilde{Q}$ is not arbitrary. If we assume that the observer (Eq. 3-13) has a continuous-time equivalent, it would be desired to locate the poles of such an observer to the left of the (continuous-time) optimal controller closed-loop poles. The equivalent observer poles are taken as the eigenvalues of the matrix $(\ln A_1)/\Delta T$,* and for the chosen $\tilde{R}$ and $\tilde{Q}$ they are -38 ± j38.

3.3.2 Four-State Observer — We design now a reduced-order observer which estimates all four unmeasured states: $\dot{\theta}$, r, $\dot{r}$, and f. Three design cases are considered; in subsection 3.4 we compare these designs by evaluating the closed-loop disturbance rejection properties. In all three designs we use $\tilde{R}$=1 and vary $\tilde{Q}$. The equivalent continuous-time observer poles and the respective $\tilde{Q}$ values are given in Table 2.†

3.4 Closed-Loop Disturbance Rejection Properties

Figure 3 shows the closed-loop configuration, where the torque T represents any unmodeled disturbance which may excite the gun inertia. It is of

*This matrix may be computed by its Taylor series expansion.

†The observer numerical values are obtained in the same straightforward manner as those of the two-state observer, and are not given here.

major interest to evaluate the capability of the control algorithm to suppress such torque disturbance at the gun output, θ. Simultaneously, one must ensure appropriate loop stability margins, It is required, therefore, to examine the closed-loop frequency response.

TABLE 2. EQUIVALENT OBSERVER POLES

		Observer Poles (Continuous-Time Equivalent)
I	$\tilde{Q}=\text{diag}(0,10^2,0,10^2)$	$-20\pm j62$, $-32\pm j39$
II	$\tilde{Q}=\text{diag}(0,1,0,10^3)$	$-1\pm j63$, $-66\pm j70$
III	$\tilde{Q}=\text{diag}(0,7,0,10^4)$	$-0.9\pm j63$, $-137\pm j141$

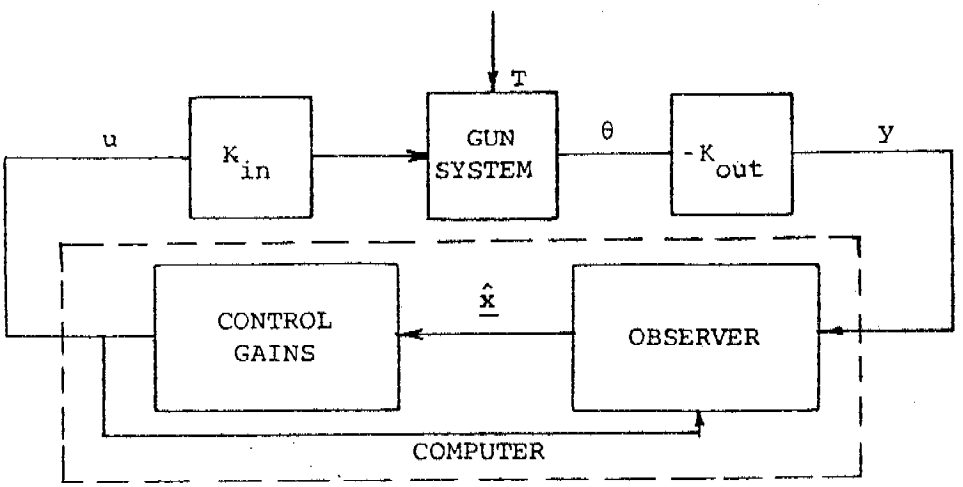


Figure 3. Closed-Loop Configuration.

One can represent the closed-loop by an n+(n-m)-th order state equation [3], viz.,

$$\begin{bmatrix} \underline{x}_{k+1} \\ \underline{z}_{k+1} \end{bmatrix} = \begin{bmatrix} \Phi-\underline{\Gamma}K_d\underline{A}_y C & -\underline{\Gamma}K_d A_z \\ \underline{A}_2 C-\underline{A}_3 K_d \underline{A}_y C & A_1-\underline{A}_3 K_d A_z \end{bmatrix} \begin{bmatrix} \underline{x}_k \\ \underline{z}_k \end{bmatrix} = \Phi_{CL} \begin{bmatrix} \underline{x}_k \\ \underline{z}_k \end{bmatrix} \qquad (3\text{-}17)$$

In order to facilitate the frequency analysis we transform Eq. 3-17 into a continuous-time equivalent via a logarithmic transformation, as indicated in subsection 3.3.1. The unmodeled torque disturbance, T, is then appended, resulting in the state equation

$$\begin{bmatrix} \dot{\underline{x}} \\ \dot{\underline{z}} \end{bmatrix} = A_{CL} \begin{bmatrix} \underline{x} \\ \underline{z} \end{bmatrix} + \underline{b}_T T \quad , \quad A_{CL} = \frac{1}{\Delta T} \ln \Phi_{CL} \quad , \qquad (3\text{-}18)$$

where $\underline{b}_T' = [0 \quad 1/J \quad 0 \quad 0 \quad 0 \;\vdots\; \underline{0}'_{n-m}]$. The transfer function of interest is

$$\frac{\theta}{T}(s) = C_o(sI-A_{CL})^{-1}\underline{b}_T \tag{3-19}$$

where $C_o = (1 \quad 0 \quad 0 \quad 0 \quad 0 \;\vdots\; \underline{0}'_{n-m})$.

Bode plots of $|\theta(s)/T(s)|$ are shown in Figs. 4 through 7. Figure 4 shows the simple design case for which the recoil disturbance is inoperative (subsection 3.3.1). Two cases are presented:

1. The friction estimate is fed-back, and the discrete control gains are

$$K_d = [182.8 \quad 10.72 \quad -9.07] \quad . \tag{3-20}$$

2. The friction estimate is not used in the feedback loop, i.e., the $\hat{f}$ gain (-9.07) is set to zero.

Case 1 is characterized by a low frequency gain droop which demonstrates the closed-loop capability to cancel out low frequency (bias) disturbances. This is not the case, however, when the friction term is excised from the feedback path as shown in Fig. 4.

Gain and phase margins are obtained by breaking the loop at u, and by computing the u-to-u open-loop transfer function. For case 1 the margins are $(\infty, 50^o)$ and for case 2 (12 dB, 45^o).

The full recoil disturbance case is presented in Figs. 5 through 7. Shown are the three design cases (I through III) as indicated in Table 2. Several comments are in order:

1. The three designs exhibit a notch at (or just about) the recoil frequency (10 Hz).

2. The notches' depth and width vary considerably. Table 3 lists the uncancelled poles and zeros of $|\theta(s)/T(s)|$ for all three cases; it is evident that the proximity of the complex zero pair to the disturbance poles (located on the imaginary axis) controls the notches' depth.

3. The stability properties of the closed loop are discussed in detail in [4]; the gain and phase margins are (12 dB, 40^o) in case I, (11 dB, 50^o) in II, and (10 dB, 60^o) in III.

TABLE 3. $|\theta(s)/T(s)|$ UNCANCELLED POLES AND ZEROS

Case	Poles	Zeros
I	-20±j62, -32±j39, -18±j18	-5±j67, -0.1, -215
II	-1±j63, -66±j70, -18±j18	-0.25±j63, -0.3, -401
III	-0.9±j63, -137±j141, -18±j18	-0.05±j63, -0.7, -7450

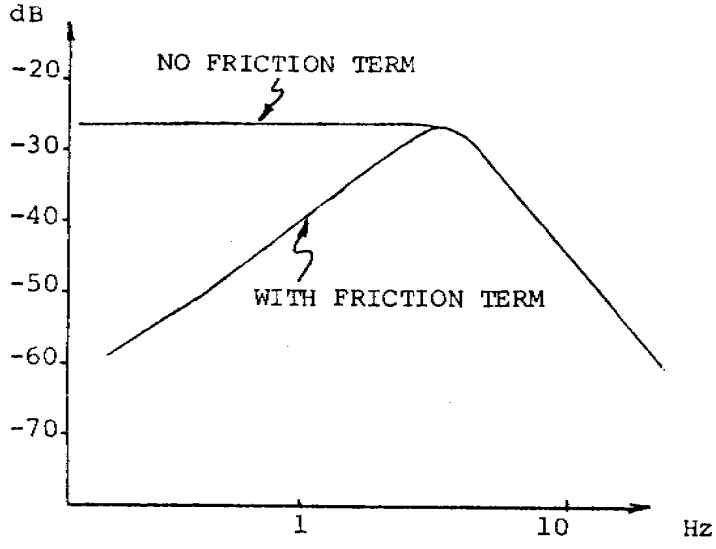


Figure 4. $\left|\frac{\theta}{T}(s)\right|$ — Recoil Inoperative.

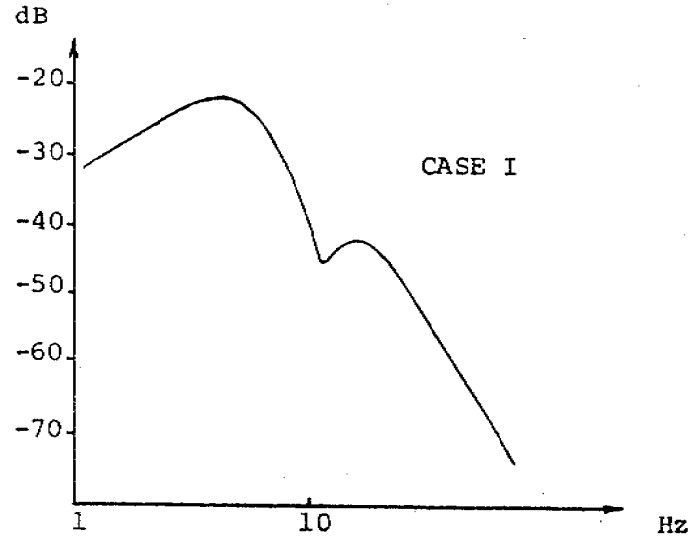


Figure 5. $\left|\frac{\theta}{T}(s)\right|$ With Recoil Disturbance — Case I.

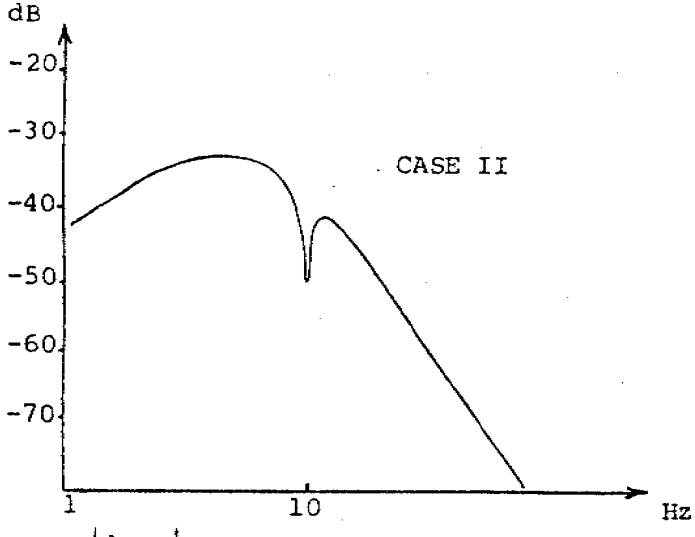


Figure 6. $\left|\frac{\theta}{T}(s)\right|$ With Recoil Disturbance — Case II.

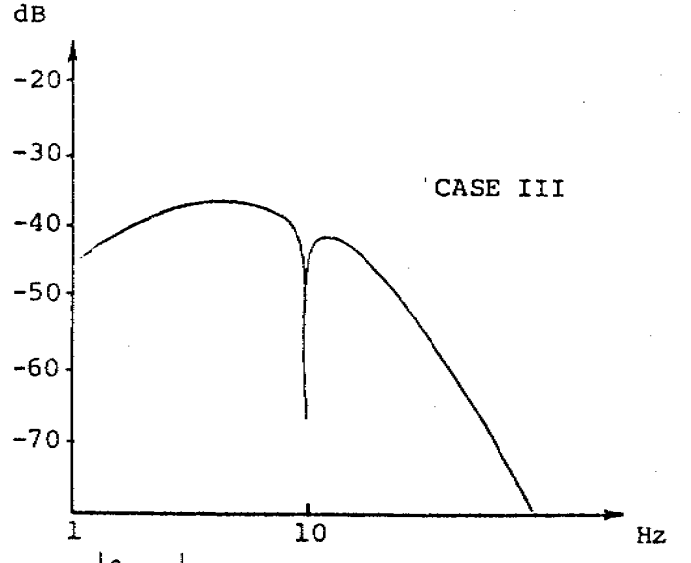


Figure 7. $\left|\frac{\theta}{T}(s)\right|$ With Recoil Disturbance — Case III.

4. MICROPROCESSOR HARDWARE DESCRIPTION

Microprocessor hardware/software development facilities used in implementing and evaluating the discrete linear quadratic control algorithms discussed in this paper are shown in Fig. 8. The host computer shown is an upgraded 8080-based MDS 220 development system with the memory expanded to 64K bytes RAM and 1.25 megabytes disk. In addition, there is a high-speed line printer, an 8-channel 12-bit A/D, a 4-channel 12-bit D/A and a PROM programmer for 2708, 2716, and 2732 PROMS (8, 16, and 32K bits per unit, respectively). Because the 8080 requires 1.2 milliseconds to perform a 32-bit floating point multiply, a means of speeding up computation was required in order to complete the control algorithm computation within the required 0.01 second sample time. Therefore, a SBC 310 high-speed mathematics board was added which does the same multiply in 85 milliseconds. However, this board must communicate with the CPU via the system bus in order to store and then load the required four byte data words. This process requires approximately 90 milliseconds. To minimize this excess overhead time, another SBC 310 high-speed mathematics board was added which permits one board to compute while the other is storing data.

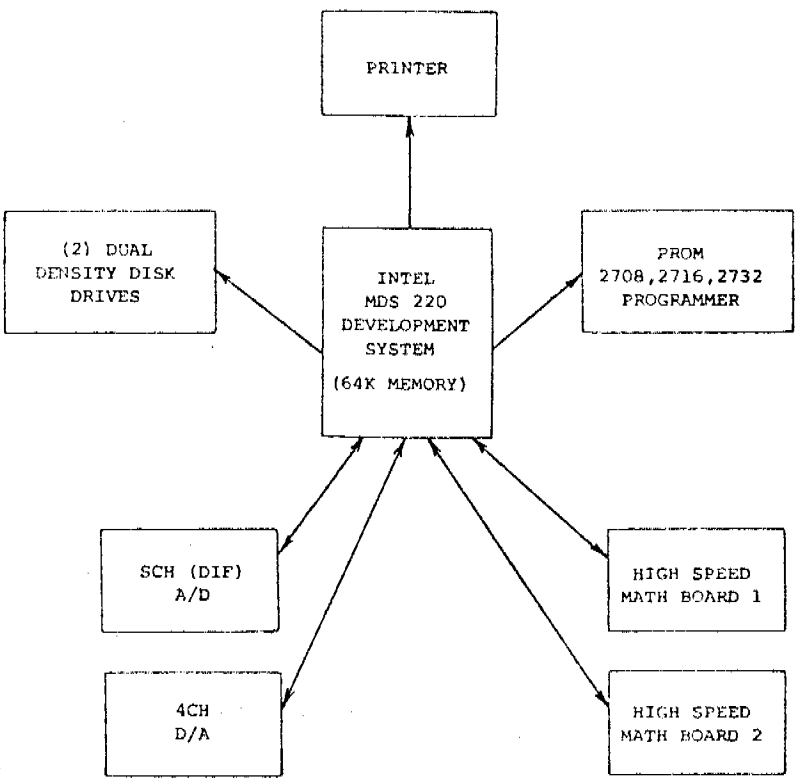


Figure 8. Hardware/Software Facilities.

A diagram of the inertia wheel interfaced with the MDS 220 Microprocessor Development System is shown in Fig. 9. The signal generator is used to drive Motor No. 2, which in turn generates disturbances to the wheel. The microcomputer system controls Motor No. 1 which stabilizes the wheel in the presence of disturbances induced by Motor No. 2.

Each preamp/power amp combination is capable of supplying a total of ±20 volts comfortably at stall current. The demodulator is a standard analog design. The output of this device has very low ripple; however, this repetitive noise is sufficient to present stability problems with some high bandwidth observer/controller designs. In order to minimize the effects of this measurement noise, a digital demodulation device was developed to replace the analog demodulator. A detailed discussion of this device will appear in a separate paper.

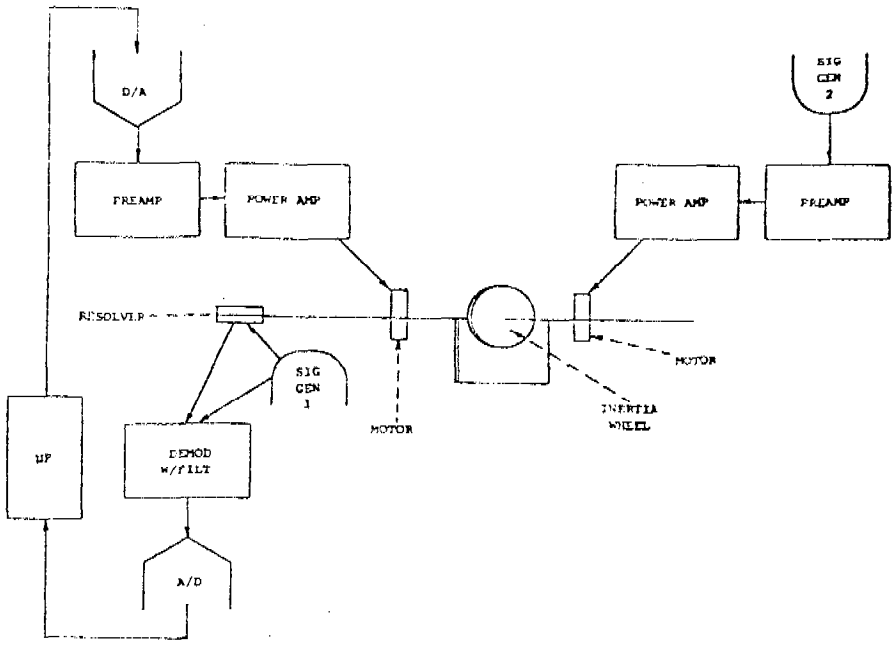


Figure 9. Inertia Wheel — Computer Interface.

5. SOFTWARE IMPLEMENTATION

The microcomputer program which implements the discrete LQ regulator algorithms is partitioned into two parts, as shown in Fig. 10. The first section, written in FORTRAN, converts decimal numbers to the 32-bit floating point format required by the high-speed mathematics boards. The second part is written in assembly language and executes the algorithm for the controller and the observers.

Originally, the basic design program (without the recoil disturbance) was written in FORTRAN. However, the program required 54 milliseconds to complete each iteration. Since the control law design requires that each interaction be completed within 10 milliseconds, recoding was necessary to speed up the computation. This was accomplished by recoding the algorithms in sequential assembly language statements, using macro definitions. This modification reduced the program execution time from 54 milliseconds to 4.5 milliseconds. The iteration time for the recoil torque observer design was reduced in similar fashion to 11.5 milliseconds. To further reduce the iteration time below 10 milliseconds, error check routines were deleted resulting in an iteration time of 8 milliseconds. Table 4 summarizes the execution times associated with each implementation. Further reductions in execution time require the use of a faster micro such as the 8086/8087.

TABLE 4. EXECUTION TIMES (FLPT)

	8080 Dev Sys (MDS-220) 2 MHz Clock		8086 (ISMC 86/12) 5 MHz Clock
	Software	HSMB	8087
32-Bit Mult	1.2 mS	85 uS OP + 90 uS OH 175 uS	19 uS OP + 27 uS OH 46 uS
Word lengths	32 Bit	32 Bit	32/64/80 Bit
	Fortran	Assembly	
Algorithm w/rate & friction observer	54 mS	4.5 mS	
Algorithm w/rate & torque observer		11.5 mS (8.5 w/o error checks)	

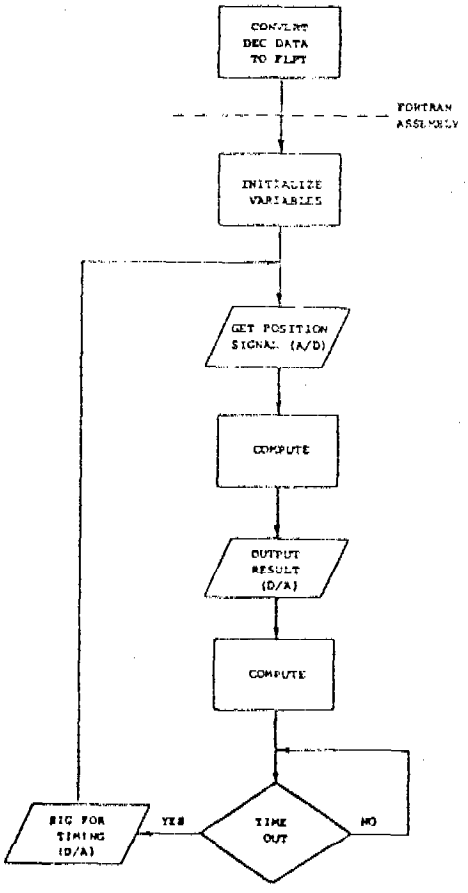


Figure 10. Algorithm Flowchart.

MACRO DEFINITIONS AND CALLS

(WAIT, PUT DATA INTO H.S.M. BOARD, COMPUTE)

PUT MACRO HSMB, WTERR, DATA1, DATA2, OPCODE

(WAIT, GET DATA FROM H.S.M. BOARD)

GET MACRO HSMB, WTERR, TMP

(WAIT FOR COMPLETION OF OPERATION (SUBMACRO))

WAIT MACRO WTERR

TERMS

HSMB	= HIGH-SPEED MATH BOARD NUMBER
WTERR	= WAIT ERROR
= NUL	= NOP
= 0	= WAIT TILL OPERATION COMPLETE
= 1	= DO 0, CORRECT FOR OVER-OR-UNDERFLOW BY SETTING EXP TO MAX
= 2	= DO 0, PRINT ERROR (IF OTHER THAN ABOVE) AND EXIT TO MONITOR
= 3	= DO 0, 1, AND 2 ABOVE
DATA 1	= ADDRESS FOR OPERAND
= NUL	= NO DATA
DATA 2	= ADDRESS FOR OPERATOR
= NUL	= DO NO COMPUTATION
TMP	= ADDRESS TO STORE DATA
= NUL	= GET NO DATA

Figure 11. Macro Definitions.

Since assembly language was needed for algorithm implementation, two user macros, PUT and GET and one submacro, WAIT, were defined. Figure 11 defines each macro and gives the expanding sequence for each parameter. Figure 11 also defines the individual terms comprising the macros. The flow charts for each macro are provided in Figs. 12, 13, and 14. It should be noted that these macro routines are not called like a subroutine, but are "expanded." When the macros were coded, all required variations were included in the routine or "defined." When the programmer references a defined macro, he types its name, followed by a list of parameters enabling selection of various parts of the routine and modifying the internal variables, thereby customizing the macro for that particular use. This expanded version is then inserted into the main program, by the assembler, where its reference existed. This permits programming time to be minimized while maintaining maximum execution time by utilizing sequential coding to eliminate subroutine call times.

When the PUT macro is expanded, the I/O port address and memory base address are first obtained for the selected board. The WAIT submacro is then expanded. If required, four bytes of data for each address, Data 1 and Data 2, are then stored in the board memory. Finally, if required, the type of operation to be performed is then transferred to the board via the port.

When the GET macro is expanded, the I/O port address and memory base address are first obtained for the selected board. The WAIT submacro is then expanded. Finally, if required, four bytes of data are loaded from the boards and stored at address TMP.

When the WAIT submacro is expanded within either macro PUT or GET, if a 0, 1, 2, or 3 is designated as the substitution for variable WTERR, execution is halted until the predesignated math board has completed its operation. If a 1 or 3 is designated, the CPU corrects for overflow (OF) or underflow (UF). Finally, if a 2 or 3 is designated, the error code is checked, and if other than an OF or UF error is found, the CPU prints the type and location of the error (program point counter) on the system console, then terminates execution.

The first routine, FDATC, was kept totally in FORTRAN. The function of this program is to convert numbers in standard decimal format to 32-bit floating point format. The program flow is shown in Fig. 15 and operates as follows. This FORTRAN routine calls an assembly routine (in the main program) which sets up a storage table for 32-bit (4-byte) words. Regaining control, FDATC then passes the starting address of each data word to another assembly subroutine for storage in the table. This procedure is then repeated for storing the constants.

Referring to Fig. 16, the input routine which controls the A/D converter is called ADIN. Since the wheel is moved only a short distance, the voltage differential to the A/D converter is small. To maximize resolution of the A/D, a programmable preamplifier is keyed by the program. If the programmer sets the SCLD flag to 0, the amplifier gain is 1. If the flag is 1, a gain of 8 is programmed and then rescaled back down in software to prevent saturation of the A/D. If a large wheel displacement occurs, a third option is available; i.e., SCLD = 3, which causes the A/D to sample once with a gain of one, choose the correct gain and then sample as before.

The second part of ADIN was written to try to decrease noise at the algorithm input. If flag AVG-0, then one sample is used for the data. If AVG≠0,

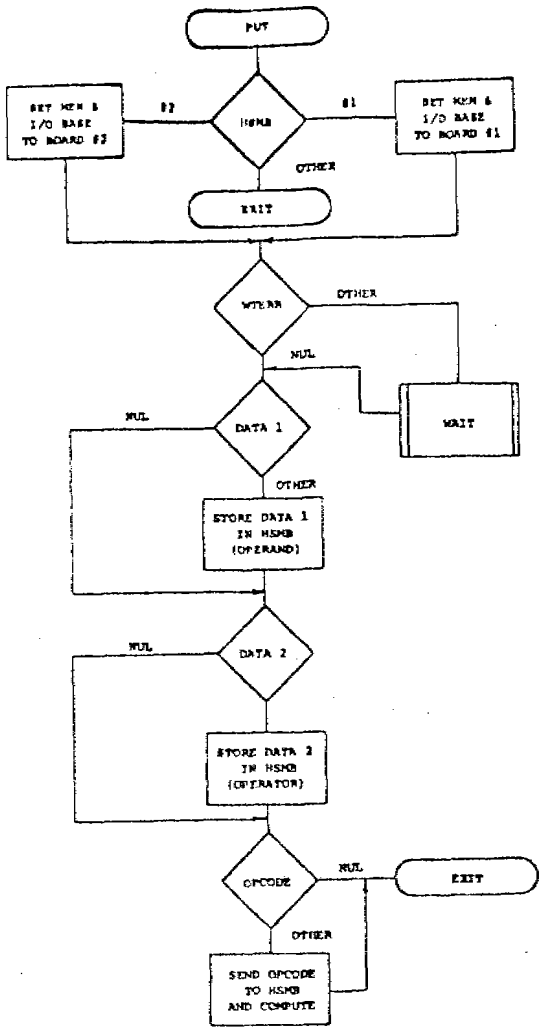


Figure 12. PUT Macro FlowChart.

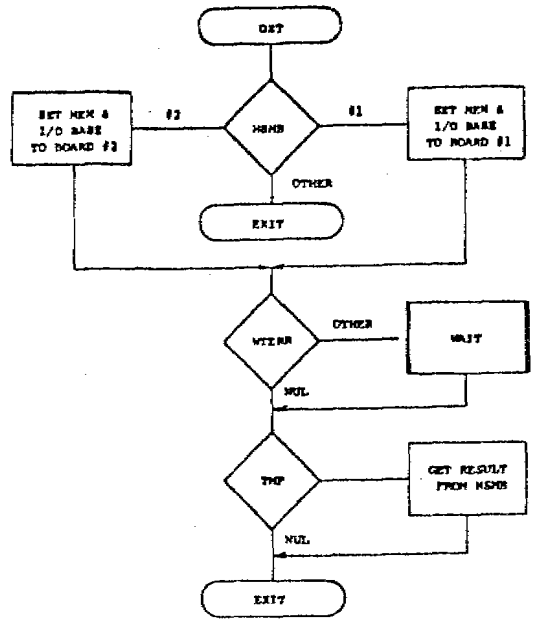


Figure 13. GET Macro Flowchart.

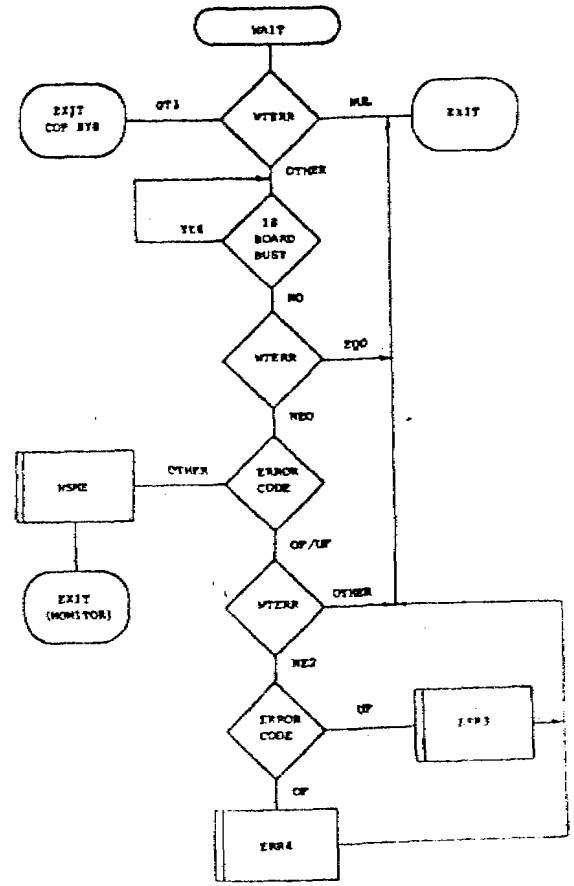


Figure 14. WAIT Macro Flowchart.

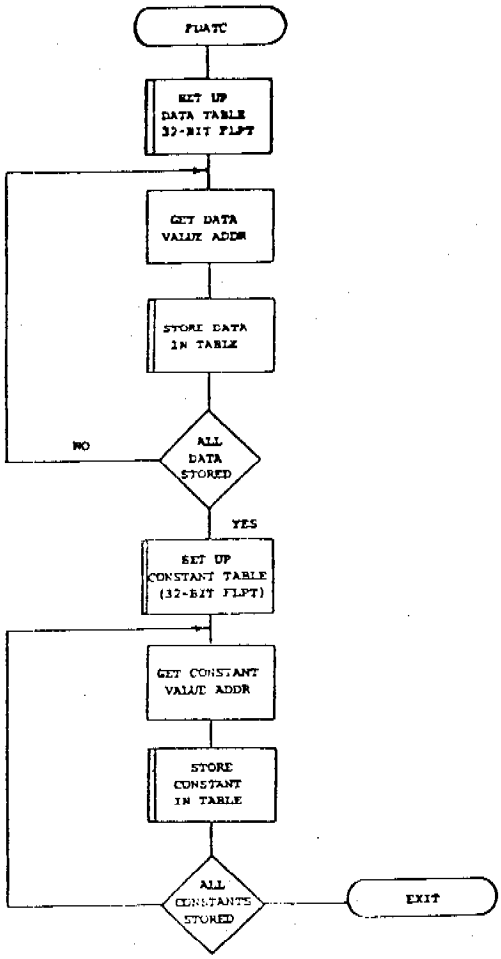


Figure 15. FDATC Flowchart.

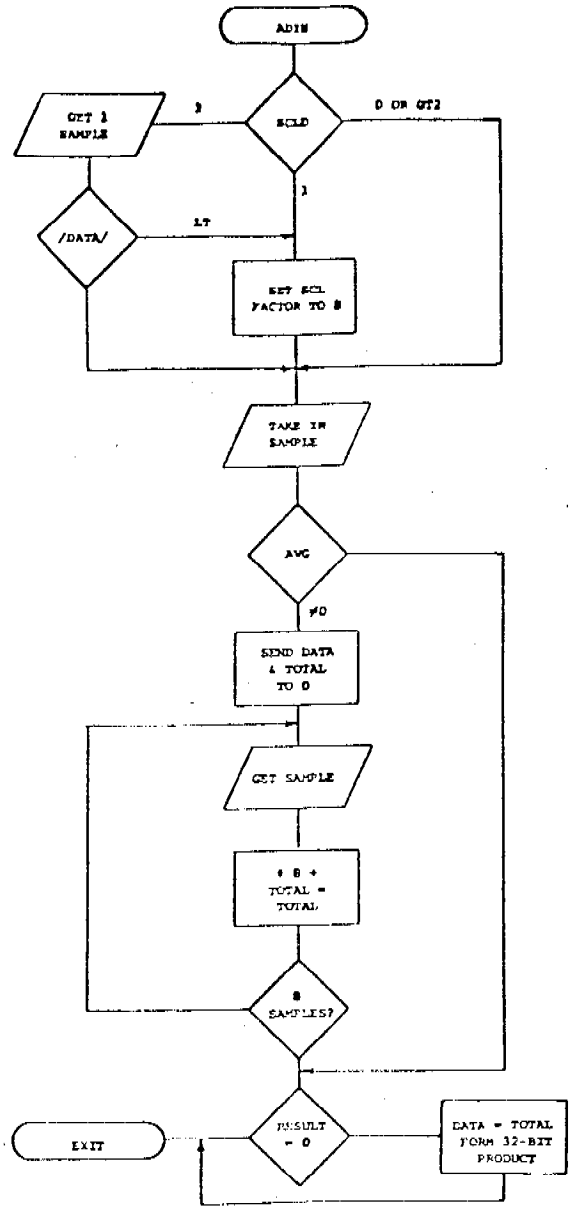


Figure 16. ADIM Flowchart.

then seven additional samples are taken quickly and averaged. This averaging routine takes about 1.5 milliseconds and even though this caused a slight phase shift, it was not great enough to effect algorithm performance. However, since no noticeable improvement in system response was observed using the averaging routine, it was deleted in the implementation of the recoil case design.

Finally, the last section of ADIN converts the 12-bit fixed-point input from the A/D, to 32-bit floating point. A problem which arises in this process is that when a 16-bit fixed-point zero (00 00) is converted to a 32-bit floating point, the high-speed math board does the conversion but flags an error. Therefore, before each conversion, if the data is zero, a 32-bit zero is forced as the return data.

To obtain the correct sampling rate (0.01 sec) the entire computation cycle must be completed in less than 0.01 sec followed by a delay routine. Figure 17 shows the two systems of delay routines used. The first, DLYS, counts down a 32-bit number to implement a software delay. DLY2, on the other hand, waits until the A/D is triggered forming a hardware delay. Both routines contain code to invert the polarity of a 5-volt square wave from a spare D/A channel for timing the full sampling rate; i.e., TIME = 1/2 period (square wave).

The three error routines are shown in Fig. 18. If the error is other than OF or UF, HSME prints out the type of error, where in the program the error occurred (point counter) and the exits to give control to the system monitor, terminating execution. Since the math boards have no extended precision temporary numbers, when an overflow occurs, the board computes the correct number, but subtracts BE_{16} from the exponent. To minimize this error, ERR3 sets the value to maximum.

ERR4 does the same but with an underflow error, since a BE_{16} was added to the exponent in this case. In addition, both ERR3 and and ERR4 output a narrow pulse to the square wave timer, a +10 volts for overflow and a -10 volts for underflow, to enable observation of these two errors.

In a continuing effort to minimize execution time, the program will eventually be run on an 8086 microprocessor with an 8087 coprocessor for floating point computation. The 8086 runs off a 5-megahertz clock and has a six statement cue for increased speed with consecutive statement coding (600 ns for minimum statement). With the 8087, a 32-bit floating point multiplication takes only 18 microseconds, and since it operates as a coprocessor, the overhead is only 26 microseconds. Also, the 8087 performs double precision 64-bit computation and has an intermediate temporary storage register, 80 bits wide, to eliminate the overflow and underflow problem.

6. PRELIMINARY TESTING RESULTS

As indicated in subsection 3.4, the torque-to-pointing angle transfer function, $|\theta(s)/T(s)|$, is of major interest in evaluating the design performance. Experimentally, it is convenient to obtain $y(s)/e_{v_2}(s)$ (see Fig. 1) as it (approximately) scales with $|\theta(s)/T(s)|$. (Because of the back emf effect it will not scale exactly.) Figures 19 through 22 show the Bode plots of $y(s)/e_{v_2}(s)$ for the two- and four-state observer designs:

1. Figure 19 shows the case in which the friction estimate is fed back (two state observer).

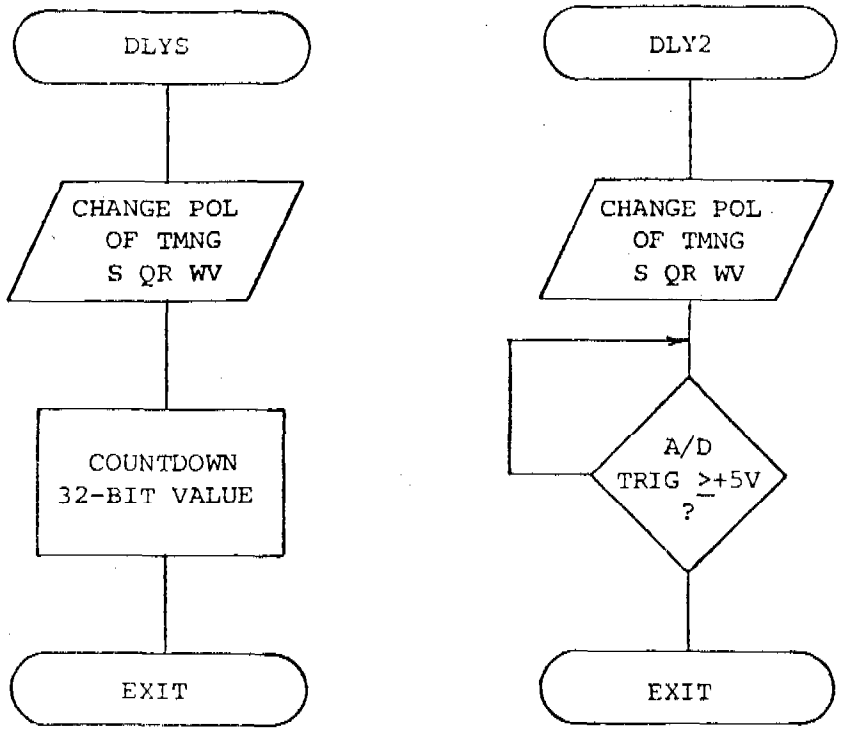


Figure 17. Delay Routines.

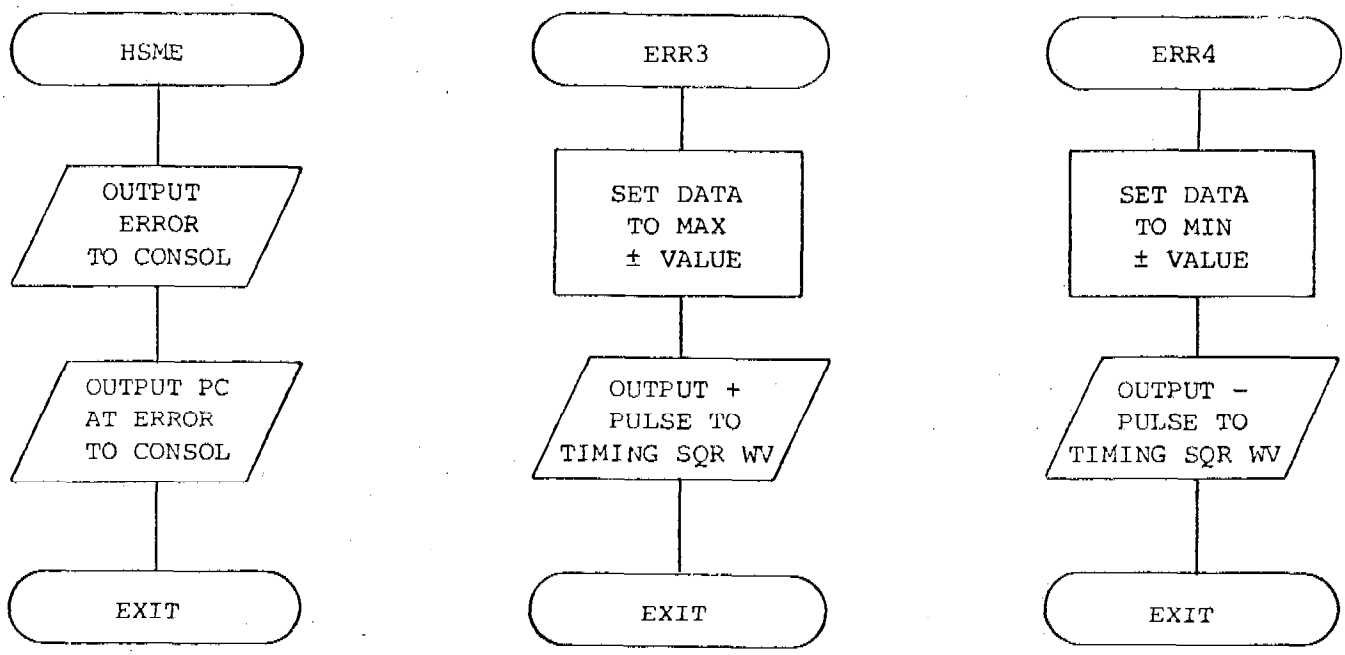


Figure 18. Error Routines.

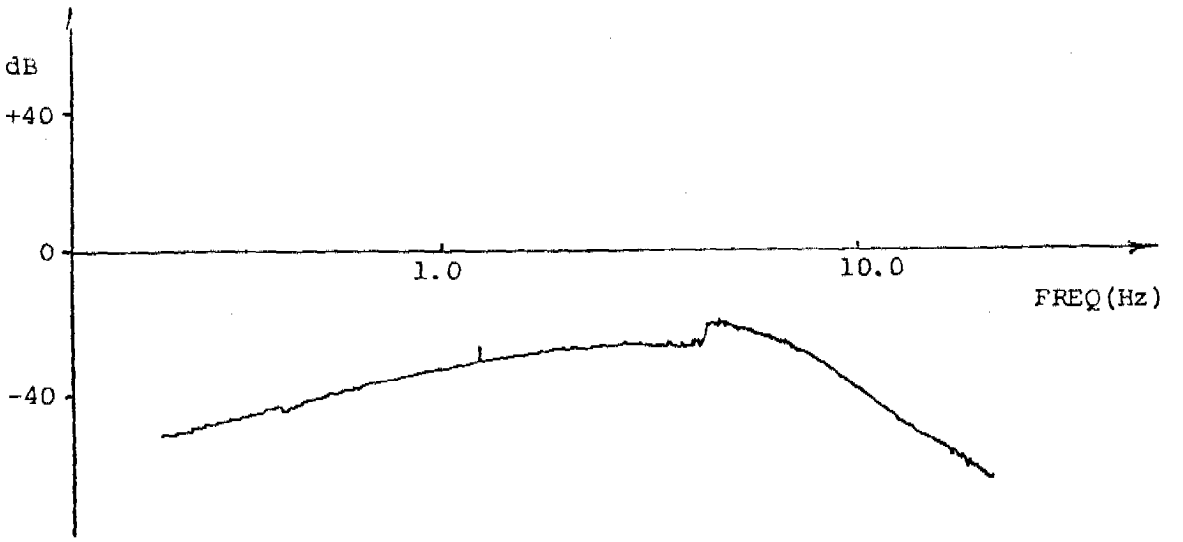


Figure 19. $\left|\frac{y}{e_{v_2}}(s)\right|$ Testing Result, With Friction Feedback.

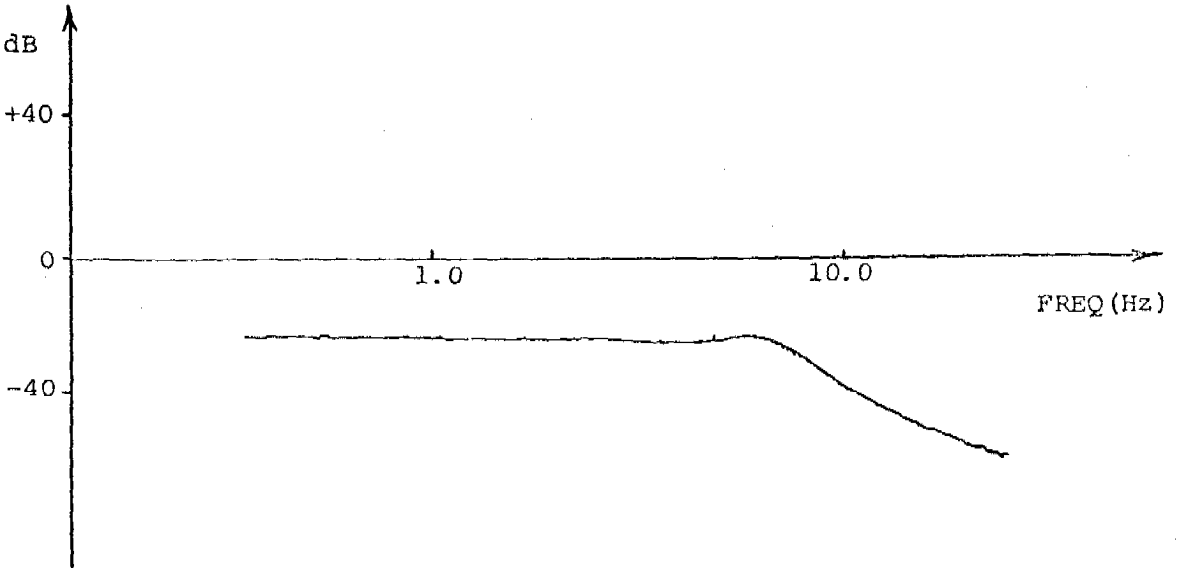


Figure 20. $\left|\frac{y}{e_{v_2}}(s)\right|$ Testing Result, Without Friction Feedback.

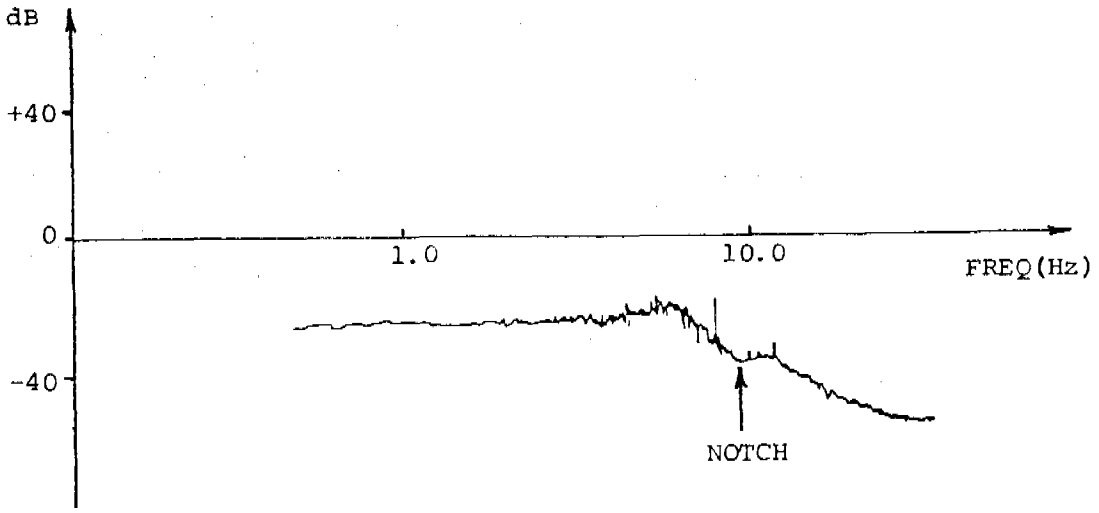


Figure 21. $\left|\frac{Y}{e_{v_2}}(s)\right|$ Testing Result, Recoil Case I, No Friction.

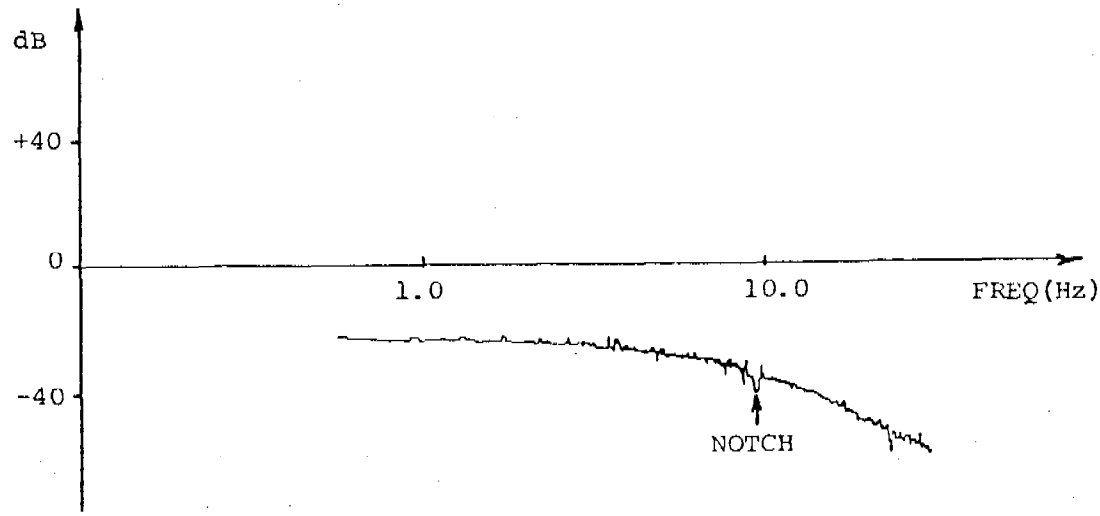


Figure 22. $\left|\frac{Y}{e_{v_2}}(s)\right|$ Testing Result, Recoil Case III, No Friction.

2. Figure 20 is the case where $\hat{f}$ is not used in the feedback path (two-state observer).

3. Figures 21 and 22 correspond to the four-state observer design (cases I and III of subsection 3.3.2, respectively) with the friction estimate excluded from the control law (one can detect the notches at 10 Hz).

It is evident that the disturbance rejection properties in these cases replicate the theoretical results (Fig. 4). By overlooking the Bode plots' imperfections due to the analyzer limitations, one can see that the friction feedback term effects the desired low frequency (up to ~5 Hz) gain droop. This property is well illustrated in Fig. 23. We excite the secondary motor with a sinusoidal signal at 1, 5, and 10 Hz, and show the inertia wheel responses in the two cases of the two-state observer. It is clear that the low frequency disturbance (1 Hz) is well suppressed when the friction term is fed back. It is also evident that the break frequency is somewhat above 5 Hz, as we still observer, at this frequency, a better disturbance suppression performance in the friction feedback case. At 10 Hz both designs exhibit the same response which indicates that we are beyond the break frequency.

Similar tests were performed on the four-state observer designs, but no pictures comparable to Fig. 23 are available at this time. It was observed, however, that the 10 Hz disturbance is further suppressed by a factor of ~2 when the recoil estimate is used in the feedback loop (Cases I through III).

7. CONCLUSIONS

In this paper, two discrete-time control algorithms were obtained. They were designed to stabilize a microcomputer-based servo control system (emulating a helicopter turret system) located in the Stabilization Research Laboratory facility at ARRADCOM. The algorithms include LQ-based digital control laws for precision stabilization in the presence of external torque disturbances, and reduced-order observers for disturbance and system state estimation. Both algorithms were successfully implemented and tested at the ARRADCOM facility and it is shown that the experimental results are in close agreement with the theoretical results.

8. REFERENCES

1. Yip, P.T. and N. Coleman, "An Adaptive Lead Prediction Algorithm for Maneuvering Target Engagement," Transactions of the 26th Conference of Army Mathematicians, ARO Report 81-1.

2. Coleman, N., N.K. Loh, D.H. Chyung, and K. Lee, "Application of Modern Control System," Proceedings of Tri-Service Control Conference, APG, Maryland, December 11-12, 1980.

3. Gully, S.W., "Inertial Platform Synthesis Employing Observers to Accomodate Disturbances and Coupling," Proceedings of the 1976 IEEE Conference on Decision and Control, pp. 65-77.

4. Korn, J., "Torque Disturbance Suppression and Stability Margins for a Gun Pointing System," work in progress.

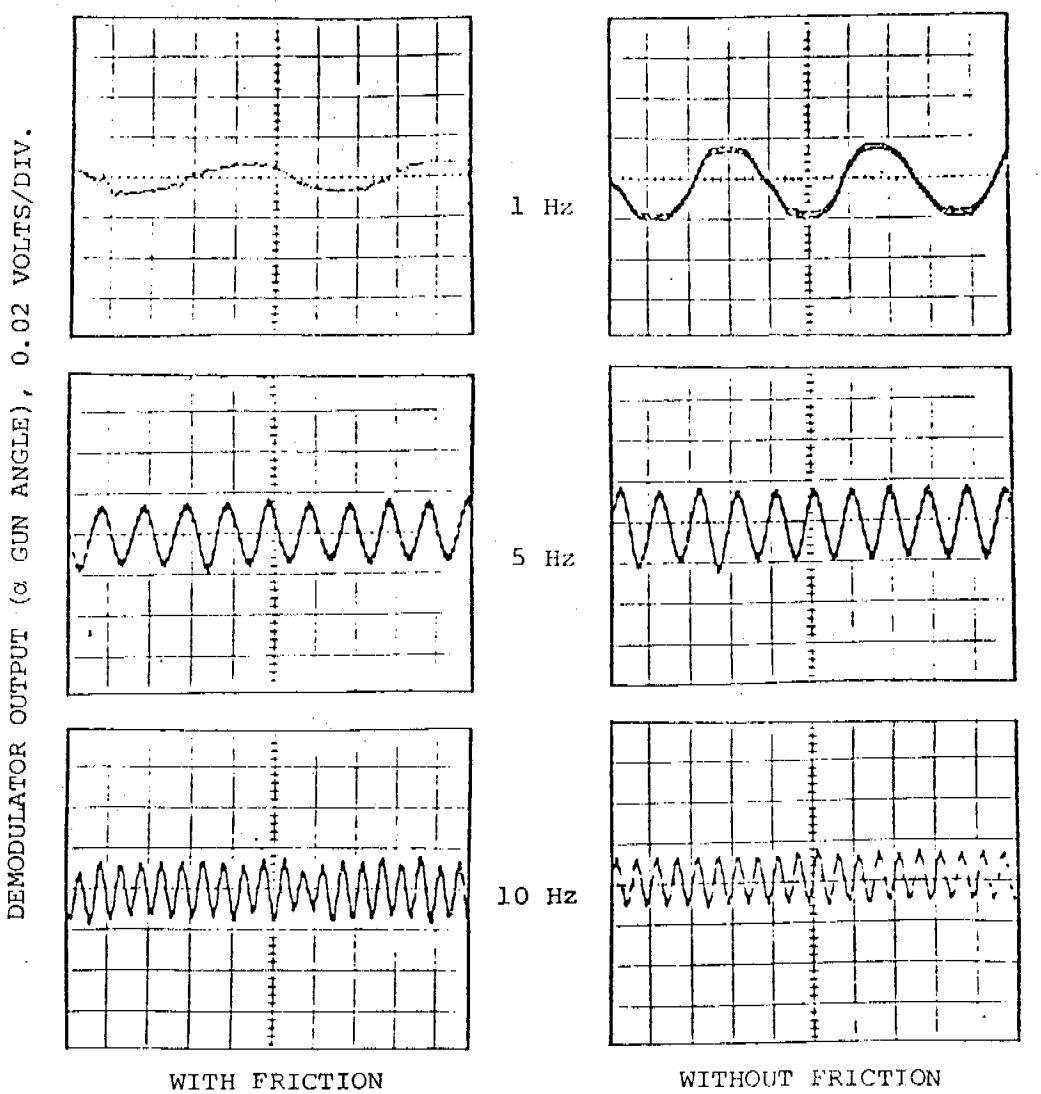


Figure 23. Disturbance Rejection With and Without Bias Feedback.

5. Dorato, P. and A. Levis, "Optimal Linear Regulator: The Discrete-Time Case," IEEE Transactions on Automatic Control, Volume AC-16, Number 6, December 1971.

6. Gully, S.W., "Synthesis of Observers for Linear Multivariable Control Systems," Ph.D. Dissertation, Polytechnic Institute of New York.

APPENDIX

1. DISCRETE-TIME CONTROL LAW FORMULATION

The process of obtaining an equivalent discrete-time control law from the continuous-time system is carried out in two steps [5]:

1. The discrete-time representation of the cost functional (Eq. 3-8) is

$$J(u_k) = \sum_{k=0}^{\infty} \underline{x}_k' Q_d \underline{x}_k + 2\underline{x}_k' S_d u_k + u_k' R_d u_k \qquad \text{(A-1)}$$

subject to the discretized system

$$\underline{x}_{k+1} = \Phi \underline{x}_k + \underline{\Gamma} u_k \qquad \text{(A-2)}$$

where

$$\Phi = \Phi(\Delta T) = e^{A\Delta T}$$

$$\underline{\Gamma} = \underline{\Gamma}(\Delta T) = \int_0^{\Delta T} \Phi(t)\underline{b}dt$$

$$Q_d = Q_d(\Delta T) = \int_0^{\Delta T} \Phi'(t) Q \Phi(t) dt \qquad \text{(A-3)}$$

$$\underline{S}_d = \underline{S}_d(\Delta T) = \int_0^{\Delta T} \Phi'(t) Q \underline{\Gamma}(t) dt$$

$$R_d = R_d(\Delta T) = \int_0^{\Delta T} (\underline{\Gamma}'(t)Q\underline{\Gamma}(t)+R)dt \qquad \text{(A-3)}$$

and ΔT = sampling interval.

2. To facilitate computations, the optimal control problem (Eq. A-1 and A-2) is reduced to an equivalent problem

$$J(u_k) = \sum_{k=0}^{\infty} \underline{x}_k' Q_{eq} \underline{x}_k + u_k' R_d u_k \qquad \text{(A-4)}$$

subject to

$$\underline{x}_{k+1} = \Phi_{eq}\underline{x}_k + \underline{\Gamma}u_k \qquad \text{(A-5)}$$

where

$$\Phi_{eq} = \Phi - \underline{\Gamma}R_d^{-1}\underline{S}_d'$$
$$Q_{eq} = Q_d - \underline{S}_d R_d^{-1}\underline{S}_d' \qquad \text{(A-6)}$$

2. DISCRETE-TIME REDUCED-ORDER OBSERVERS

Unlike the control law design, discrete-time reduced-order observers cannot be obtained by transforming the continuous-time into an equivalent discrete-time problem. It is required therefore, to carry out the observer design process from the discrete-time system (Eq. A-2) directly.

The estimator dynamics are constructed to estimate the n-m unmeasured states, where n is the system order, and m is the number of available measurements. We partition, therefore, the system (Eq. A-2) as

$$\begin{bmatrix} \underline{x}_{r,k+1} \\ \underline{x}_{p,k+1} \end{bmatrix} = \begin{bmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22} \end{bmatrix} \begin{bmatrix} \underline{x}_{r,k} \\ \underline{x}_{p,k} \end{bmatrix} + \begin{bmatrix} \Gamma_1 \\ \Gamma_2 \end{bmatrix} \underline{u}_k \qquad \underline{y}_k = [C_1 \quad C_2] \begin{bmatrix} \underline{x}_{r,k} \\ \underline{x}_{p,k} \end{bmatrix} \qquad \text{(A-7)}$$

where $\underline{x}_{r,k} \in R^{mx1}$, $\underline{x}_{p,k} \in R^{(n-m)x1}$ and C_1 is invertible. One may estimate these states by employing the reduced-order observer due to Gully [2], viz.,

$$\hat{\underline{x}}_{p,k} = T\underline{z}_k + H\underline{y}_k$$

$$\hat{\underline{x}}_{r,k} = C_1^{-1}\underline{y}_k - C_1^{-1}C_2\underline{x}_{p,k} \tag{A-8}$$

The observer state, $\underline{z}_k \in R^{(n-m)x1}$, propagates according to the equation,

$$\underline{z}_{k+1} = T^{-1}(\bar{\Phi}-H\bar{C})T\underline{z}_k + T^{-1}[(\bar{\Phi}-H\bar{C})H+\Phi_{21}C_1^{-1}-H(C_2\Phi_{21}+C_1\Phi_{11})C_1^{-1}]\underline{y}_k$$

$$+ T^{-1}[\Gamma_2-H(C_2\Gamma_2+C_1\Gamma_1)]\underline{u}_k \tag{A-9}$$

where

$$\bar{\Phi} = \Phi_{22} - \Phi_{21}C_1^{-1}C_2$$

$$\bar{C} = C_2(\Phi_{22} - \Phi_{21}C_1^{-1}C_2) + C_1\Phi_{12} - C_1\Phi_{11}C_1^{-1}C_2 \tag{A-10}$$

The selection of T and H completes the observer design. The transformation matrix T enables one to scale the observer gains, and should be chosen accordingly. The selection of the gain matrix H determines the observer dynamics (Eq. A-9), and is therefore, more subtle. It can be shown [3],[6] that choosing H is equivalent to solving an optimal control problem

$$J(\underline{v}_k) = \sum_{k=0}^{\infty} \underline{e}_k'\tilde{Q}\underline{e} + \underline{v}_k'\tilde{R}\underline{v}_k \qquad \tilde{Q} \geq 0 \quad , \quad \tilde{R} \geq 0 \tag{A-11}$$

subject to

$$\underline{e}_{k+1} = \bar{\Phi}'\underline{e}_k + \bar{C}'\underline{v}_k \quad . \tag{A-12}$$

The choice of $\tilde{Q}$ and $\tilde{R}$ is not as straightforward as in the control law problem, since strictly speaking, the observer state does not usually have physical interpretation. These matrices should rather be chosen for desired observer pole locations and gain magnitude considerations.

In summary, we conveniently rewrite the observer equations (A-8 through A-10) in a concise form, viz.,

$$\underline{z}_{k+1} = A_1\underline{z}_k + A_2\underline{y}_k + A_3\underline{u}_k$$

$$\underline{\hat{x}}_k = \begin{bmatrix} \underline{\hat{x}}_{r,k} \\ \underline{\hat{x}}_{p,k} \end{bmatrix} = A_z\underline{z}_k + A_y\underline{y}_k \qquad \text{(A-13)}$$

where the coefficients A_1, A_2, and A_3 can be identified from Eq. A-9, and

$$A_z = \begin{bmatrix} -C_1^{-1}C_2T \\ \hline T \end{bmatrix}, \quad A_y = \begin{bmatrix} C_1^{-1} - C_1^{-1}C_2H \\ \hline H \end{bmatrix}. \qquad \text{(A-14)}$$

DATA ANALYSIS OF INTRUSION SIGNATURES FOR PHYSICAL SECURITY APPLICATIONS WITH A MINICOMPUTER

WEN-WU SHEN
COUNTER INTRUSION LABORATORY
USA MERADCOM, FT. BELVOIR, VIRGINIA 22060

February 1981

ABSTRACT

A data acquisition and analysis system using a minicomputer is described. Effort of the system development, and its applications to signal processing and analysis is currently a part of a program on physical security research. The object of the research is directed toward development of target identification techniques to be implemented in digital signal processors (or intrusion detection sensors). The digital signal processor performs automatic decision-making in different environmental conditions as a distributed processing device for FIDS (Facilities Intrusion Detection System). The FIDS is a central processor monitoring security of an area or complex to be protected. Data from various intrusion detection devices such as Radio Frequency Motion Sensors (RFMS), Vibration Sensors, Passive Ultrasonic Sensors (PUS), Ultrasonic Motion Sensors (UMS), Passive Infrared Motion Sensors (PIMS), etc. have been partially analyzed. To be discussed here in the presentation will include the test data from RF motion sensors and vibration sensors.

1.0 Introduction

The Counter Intrusion Laboratory of Mobility Equipment Research and Development Command (MERADCOM) is currently engaged in the development of data acquisition and analysis equipment (DAAE). The DAAE will consist of the Data Acquisition Systems (DACS) and a Data Analysis System (DAN). The DACs will be microprocessor-based recording devices with software-control capability. The DACs will be used in the field to select, record, and store intrusion signals as well as non-intrusion false alarm stimuli data. The DAN is a Digital Equipment Corporation PDP11 minicomputer which will soon be upgraded. It has been used for data acquisition and analysis in the laboratory. The DAC which is being developed will consist of low-frequency, medium-frequency and high-frequency units. The DACs and the DAN will be operationally integrated to form an effective data acquisition and analysis system that will provide scientists and engineers with an efficient data-processing and computing tool for physical security RDT&E programs.

2.0 Intrusion Detection Sensors

The Counter Intrusion Laboratory has developed a number of sensors for the Facilities Intrusion Detection System (FIDS). The sensors included but were not limited to Balanced Magnetic Switch, Grid Wire Sensor, Duress Sensor, Contraband Sensor, Vibration Sensor, Passive Ultrasonic Sensor, Ultrasonic Motion Sensor, and Radio-Frequency Motion Sensor (or Microwave Motion Sensor), etc. Among those sensors, only the last four types of sensors required signal processing. Furthermore, the vibration sensor and the passive ultrasonic sensor are the passive devices, and their circuit designs were almost identical

except the difference in bandpass filter. However, a rejection filter (through a multiplier) can be optionally applied to the passive ultrasonic sensor to remove the energy emitted from an active device such as ultrasonic motion sensor if it is present.

The RF motion sensor and the ultrasonic motion sensor developed by the MERADCOM are active devices. They apply the Doppler Principle to detect the motion of an intruder. However, those sensors were designed primarily for indoor implementation. The received signals were usually complicated by environmental multipaths and were generally very complex.

3.0 Analytical Description of Intrusion Signatures

However, numerical generation of the intrusion signatures can be performed through computer simulation or synthesis for some types of sensors. Brief mathematical description was given here for the active intrusion detection devices, particularly for the RF motion sensors.

Let T(t) be the signal transmitted by an antenna or a transducer and the signal be represented as

$$T(t)=A_o\cos(2\pi f_o t+\phi_o) \tag{1}$$

where f_o is the carrier frequency, A_o the amplitude, and ϕ_o the phase. Let the information received by an antenna or a microphone be R(t). And then R(t) may be represented as

$$R(t)=\sum_{n=1}^{\infty} A_n\cos[2\pi(f_o\pm f_n)t+\phi_n] \tag{2}$$

where n=1, 2, ----∞ representing the multipath arrivals, A_n the amplitudes of the signals, f_n the Doppler shifted frequencies due to a moving object, and ϕ_n the multipath phases of the signal received.

Let T(t) and R(t) pass through a multiplier (mixer) as they do in the RFMS and UMS circuits. Then, the output from the mixer can be represented as

$$S(t)=\frac{A_o}{2}\sum_{n=1}^{\infty} A_n\cos[2\pi(2f_o\pm f_n)t+\phi_o+\phi_n]+\frac{A_o}{2}\sum_{n=1}^{\infty} A_n\cos[\pm 2\pi f_n t+\phi_n-\phi_o] \qquad (3)$$

On the right-hand side of eq.(3), the first summation consists of the high-frequency information and the second summation is primarily the very low-frequency components which contain the intrusion information. We are only interested in the low-frequency information only. Therefore, applying a low-pass filter, we have

$$S(t)=\frac{A_o}{2}\sum_{n=1}^{\infty} A_n\cos[\pm 2\pi f_n t+\phi_n-\phi_o] \qquad (4)$$

This is the information that is manageable by a digital computer.

Equation (4) serves as a fundamental description of the intrusion signatures for a number of sensors developed by the Counter Intrusion Laboratory. It may be integrated for the very much simplified cases. However, computer simulations can be performed to synthesize the signatures by using the equation. And the computer simulations of intrusion signals are useful for understanding the various signal characteristics and for extracting the features of the simulated data as well as the observed signatures.

4.0 Data Analysis For Intrusion Detection Sensors

Testing and evaluation of intrusion detection sensors have been conducted in the past. Two computer systems were used for the data acquisition and analysis: One was the real-time data acquisition system which was used in the field for digital recording; the other was used mainly for data analysis in the laboratory. Figure 1 shows the block diagram for data analysis system in the laboratory. The central processor was a DEC PDP11/05 minicomputer. The

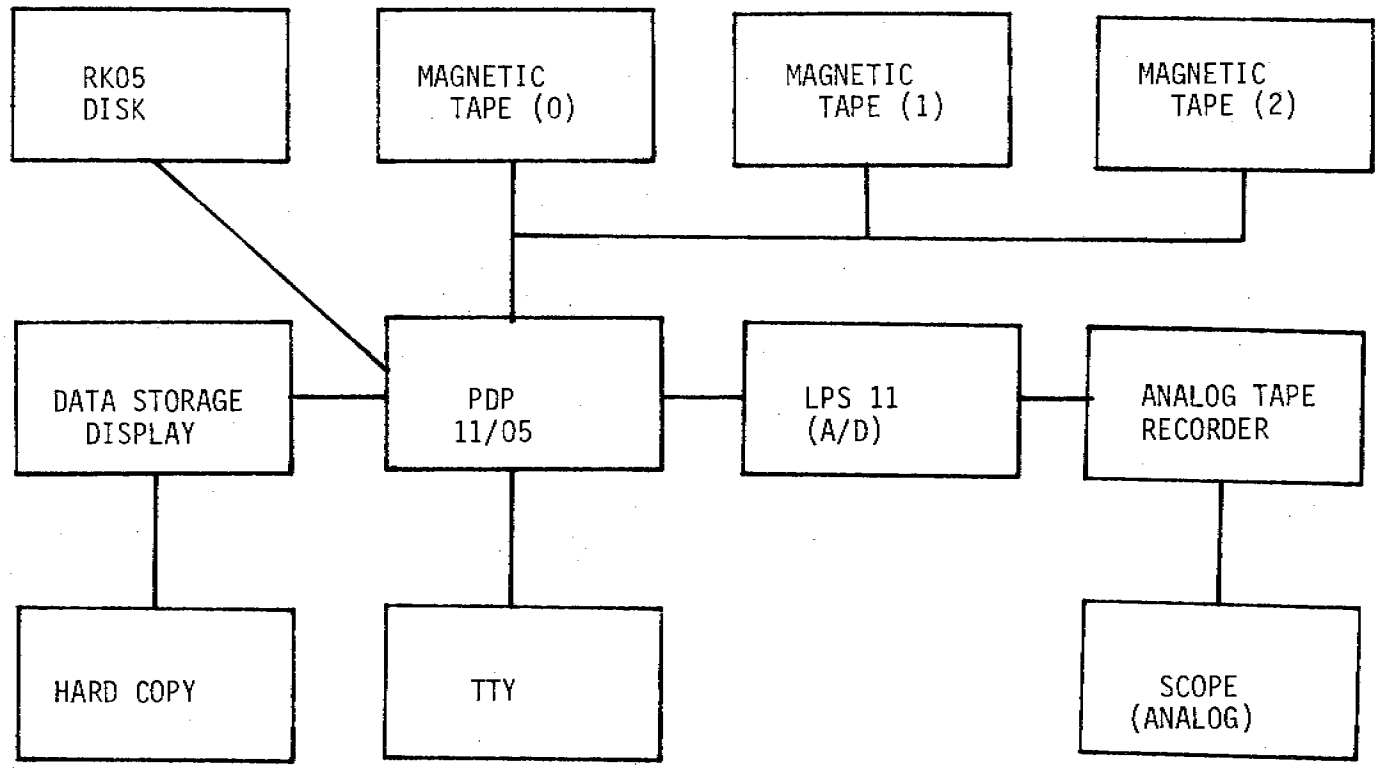


Figure 1. Block Diagram Of The Data Analysis System In The Laboratory.

peripherals included a RK05 disk, three magnetic tape drives, data storage and display, a hard-copy unit, and above all a LPS11 device. The LPS11 is an Analog-To-Digital Converter by which we analyzed the data from the analog tapes. The scope is for a quick view of analog signals.

Data which had been analyzed recently included the signatures from the passive ultrasonic sensors, ultrasonic motion sensors, RF motion sensors and vibration sensors. However, we would limit our discussion to the results from RF motion sensors and vibration sensors only.

4.1 RF Intrusion Signals

Testing of the RFMS was conducted in Bldg. 2093, Ft. Belvoir, VA. Figure 2 shows the configuration of the test building. Two RFMS transmitting antennas were mounted in one end of the building. In the other end near the office area were mounted two RFMS receiving antennas. The signals recorded were the sum of two receiving antennas.

Figure 3 shows the background data without stimuli. The RFMS uses a carrier frequency at 915 MHz. The data shown here has a sampling rate at 40Hz where the carrier frequency has been removed (i.e.eq. (4)). Trace A was the data which was taken when heating system was in the warm-up phase. And Trace B was recorded about 10 minutes after the Trace A. At that time, the door was rattling in the wind. The last trace of the time-series data was recorded in the general background in the winter where the heating air flow was there in the duct. The air flow caused vibration in the duct and the high-frequency components of the data was probably the phase fluctuations due to the RF reflection on the duct.

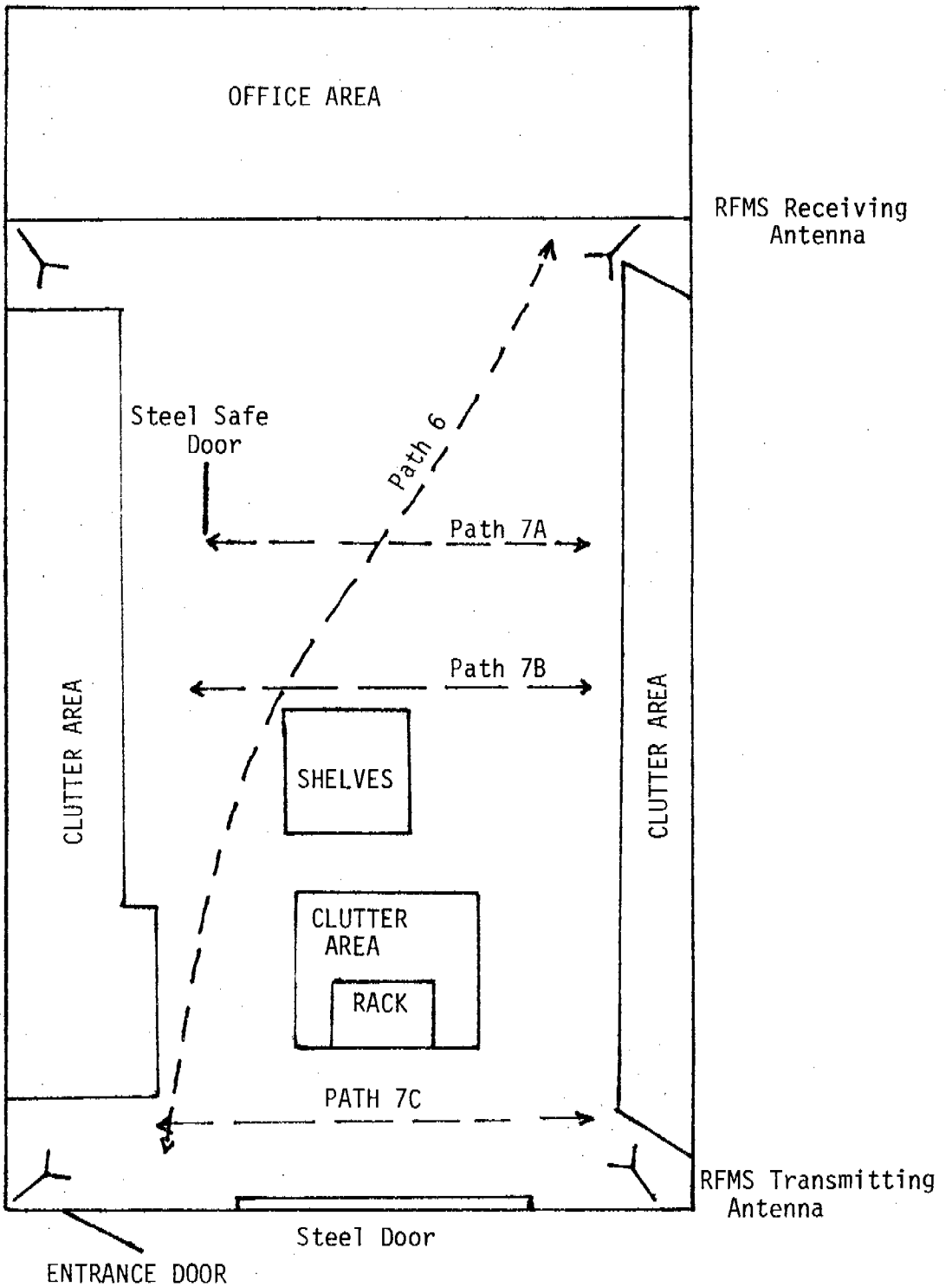


Figure 2. Configuration of Test Building (40' x 50')

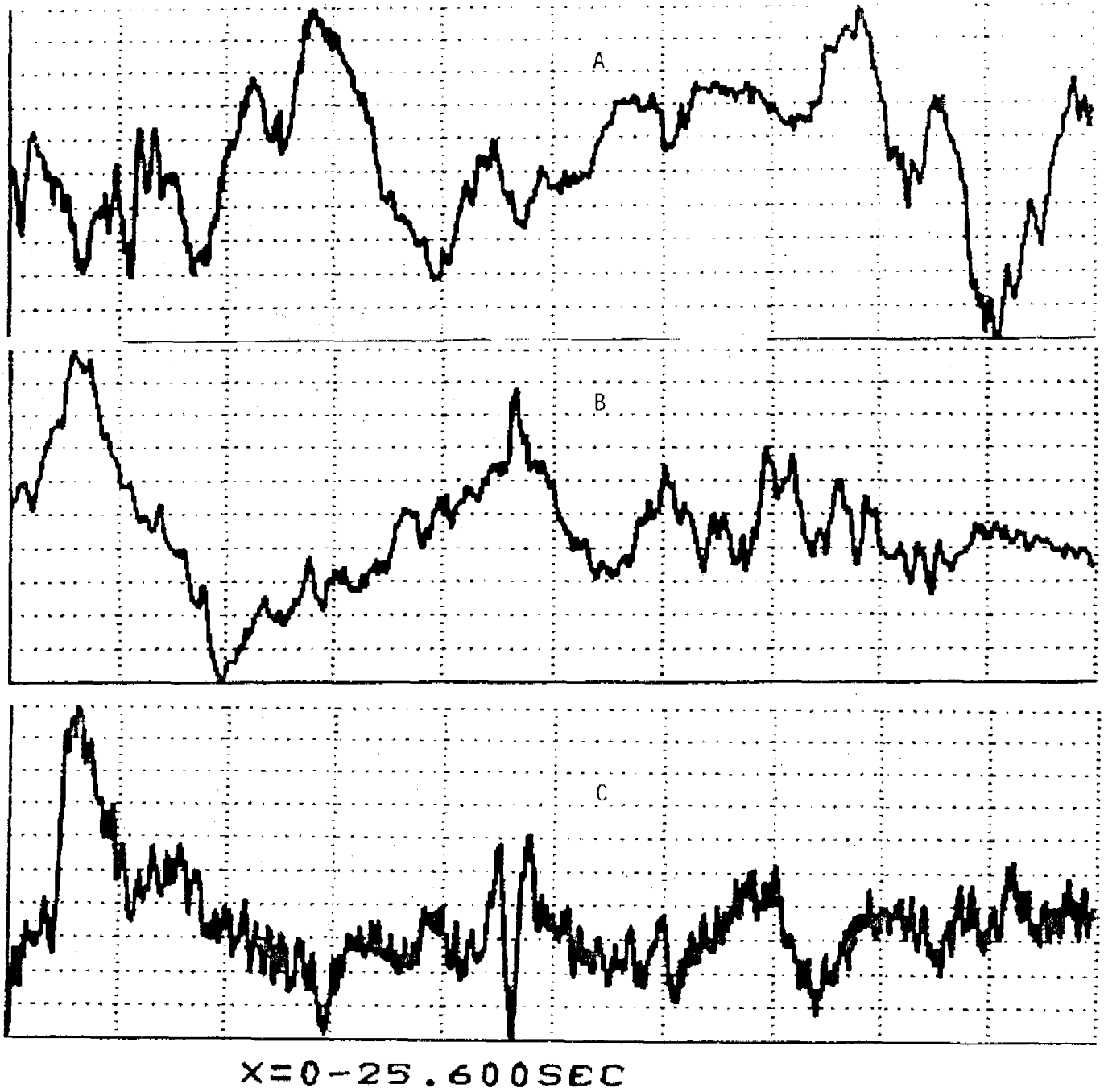


Figure 3. Background Data Without Stimuli
A. Heating System In Warm-up Condition
B. Door Rattling In The Wind
C. Heating Air Flow In The Duct

Figure 4 shows the time-and frequency-domain data in which one was shaking the steel door from outside. The purpose of the test was to simulate the windy conditions. The spectral lines here indicated the vibration characteristics of the steel door. Those lines would be shifted to the lower-frequency range and disappeared as time went by when one stopped the shaking.

A test was conducted by opening and closing the entrance door for a number of times. Figure 5 shows the time-series data for the test. The first trace illustrated the situation where one opened and closed the door normally and continuously for five times. Trace B of the data showed the testing where one opened and slammed shut the same door for five times. Because the door was open outside, the contribution to the time-domain features was mostly from the situations where the door had its surface closely parallel to the surface of the wall.

A number of walking or running tests were conducted where a man walked or ran in various paths. The dashed lines in Figure 2 indicated the paths for the walking/running tests in discussion. Figure 6 shows the time-domain data by the man walking or running in the Path 6. The first trace of the figure was the time-domain data where the man walked from the entrance door to the office area following the dashed-line path. Trace B was the signatures where the man walked in the opposite direction of Trace A. If the man walked exactly in the same path in the opposite direction with the same speed, we might expect that Trace B was the time reversal of Trace A. In any case, Trace C was the data taken when the man was running in the same path to-and-fro twice.

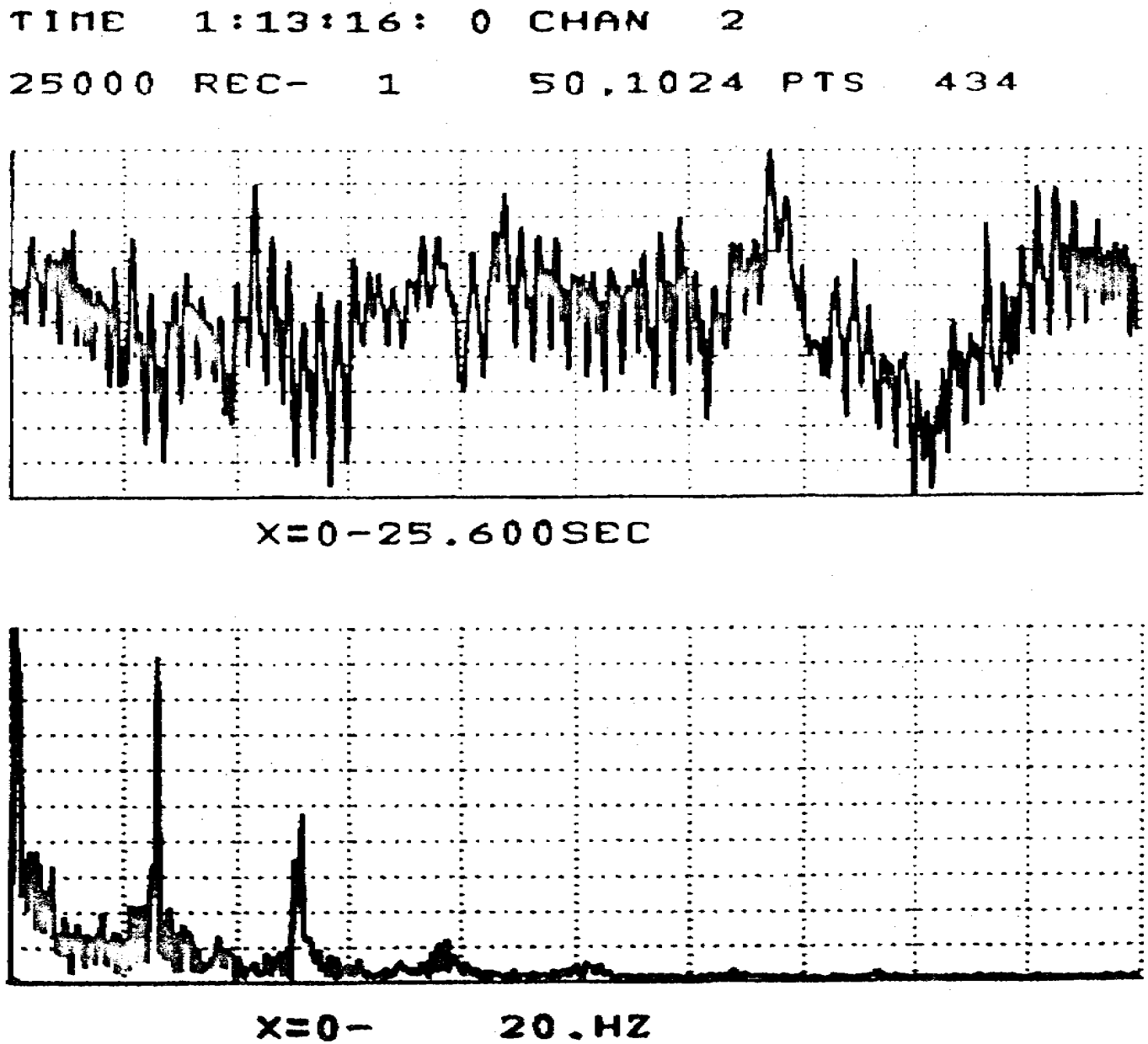


Figure 4. Data By Shaking the Steel Door From Outside

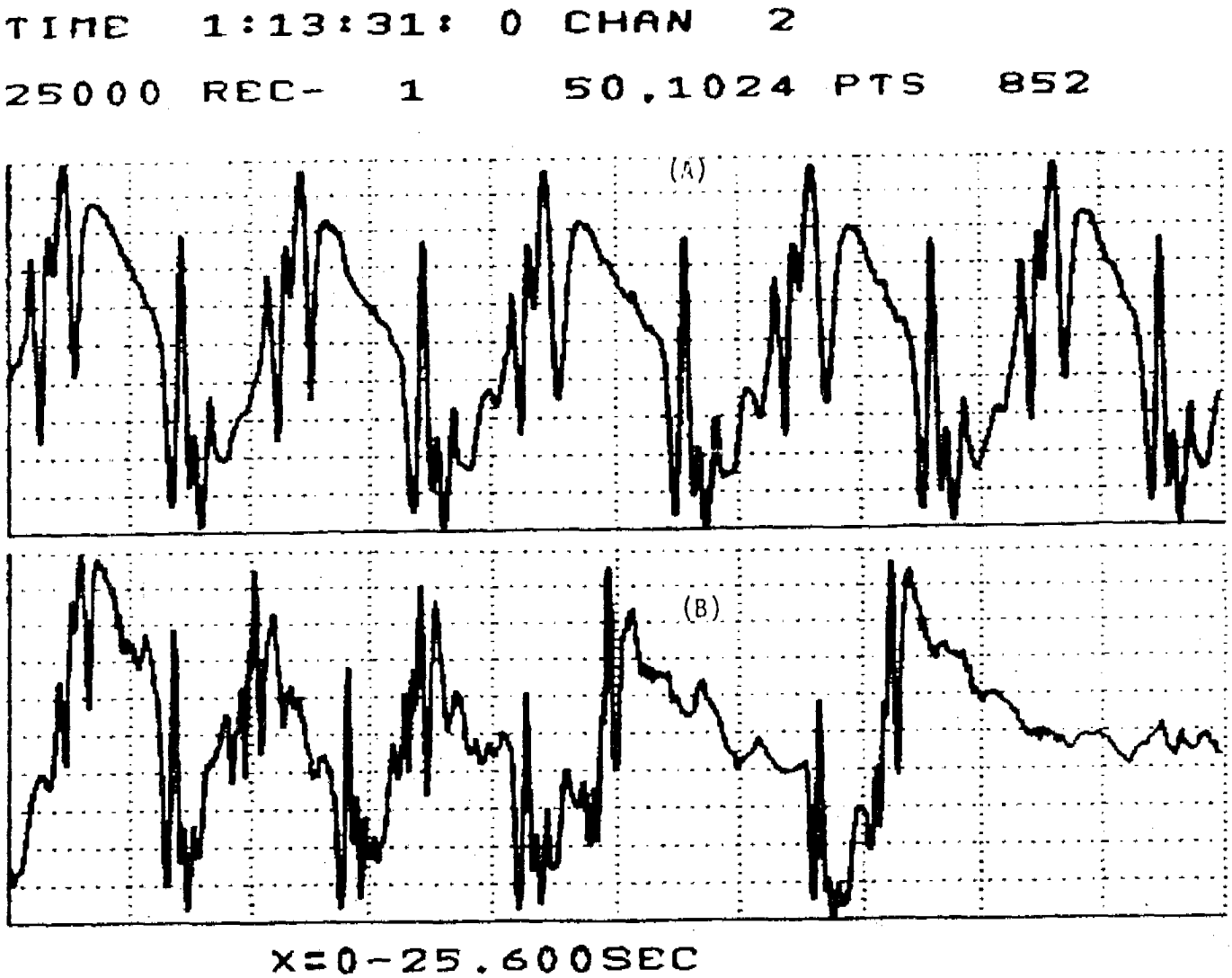


Figure 5. Data By Opening And Closing The Front Entrance Door.
(A) Normally And Continuously Five Times
(B) Opening And Slamming The Door Five Times

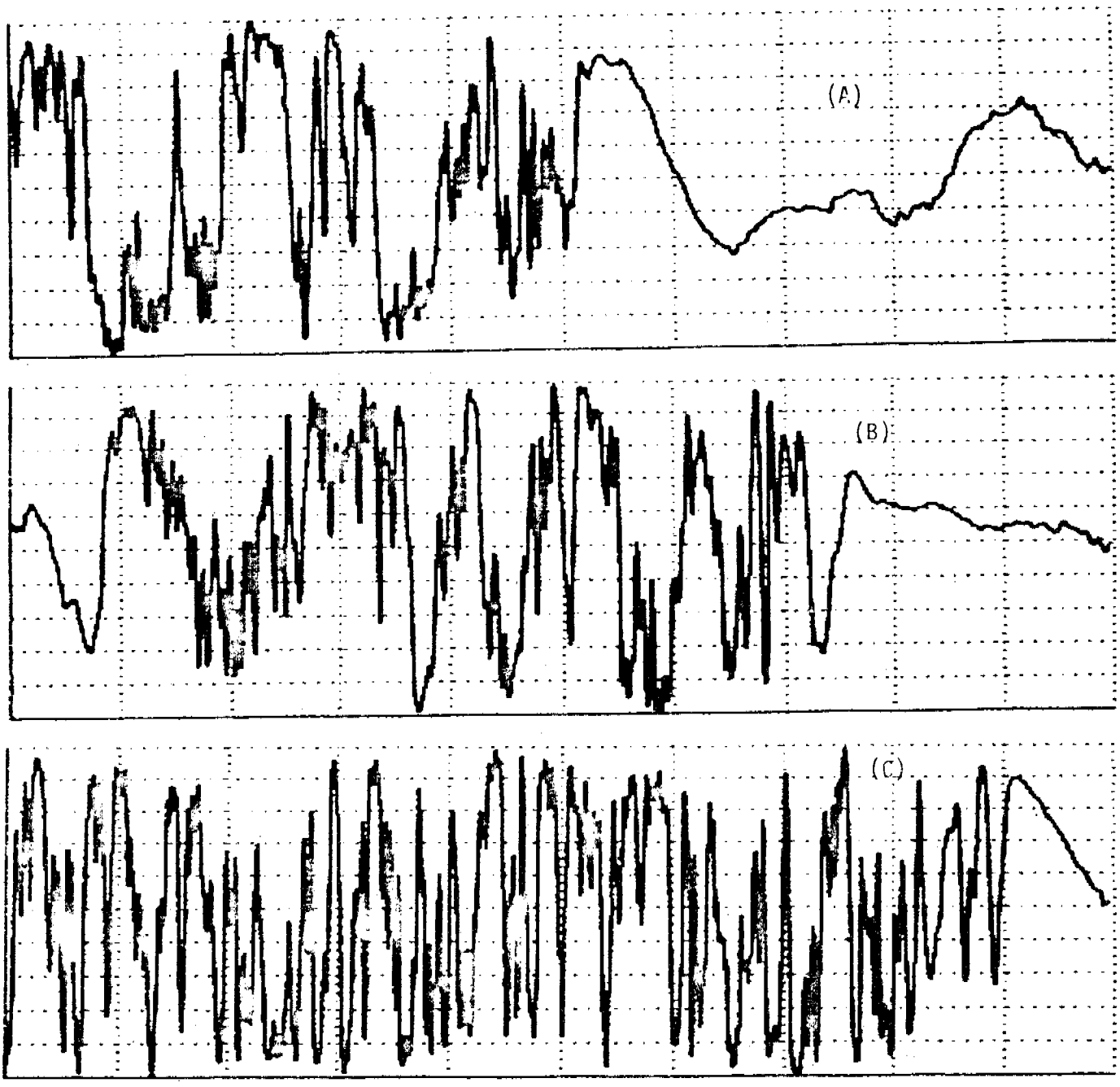


Figure 6. Data By Man Walking Or Running (Path 6 in Figure 2)
A. Walking From Entrance Door To The Office Area
B. Walking From The Office Area To The Entrance Door
C. Running From The Entrance Door To The Office Area; Then Back And Forth

The following figure (Figure 7) shows the situations where the man was walking from right to left and back twice in the building following the dashed line path 7A, 7B, and 7C shown in Figure 2. The complex time-domain features of the data suggested the extensive environmental reverberation, multipaths, and the Doppler spread of frequencies. Another factor which caused the complexity of time-domain feature was the fact that there were two emitters and two receivers used in the test.

The complete set of the data acquired in the RFMS test was compiled and presented in the Appendix. In the appendix, the time-domain data as well as its Fourier amplitude spectrum were illustrated. Discussions of the data analysis on RF sensors so far are limited to the signal processing with emphasis on data acquisition and display. Mathematical computations for parametrizations or for feature extractions have not been attempted yet. However, the time-domain features seem meaningful. To a certain extent, they can be understood and make sense visually.

4.2 Impact Response Analysis For The Naval Steel Lattice Vibration Sensor System

In 1978, the BDM Corporation under a contract with MERADCOM conducted a data acquisition effort for the steel lattice vibration sensor system at the Naval Weapons Station in Yorktown, Virginia. One of the objectives for the test was to collect the vibration sensor system response and the performance information for analysis. Two of the naval ammunition magazines were used for the test. The magazines were earth-covered arch-type structures of traditional design and were made entirely of reinforced concrete.

Figure 8 shows the diagram of the steel lattice network and the sensor configuration. The lattice network was typical there in the naval installation.

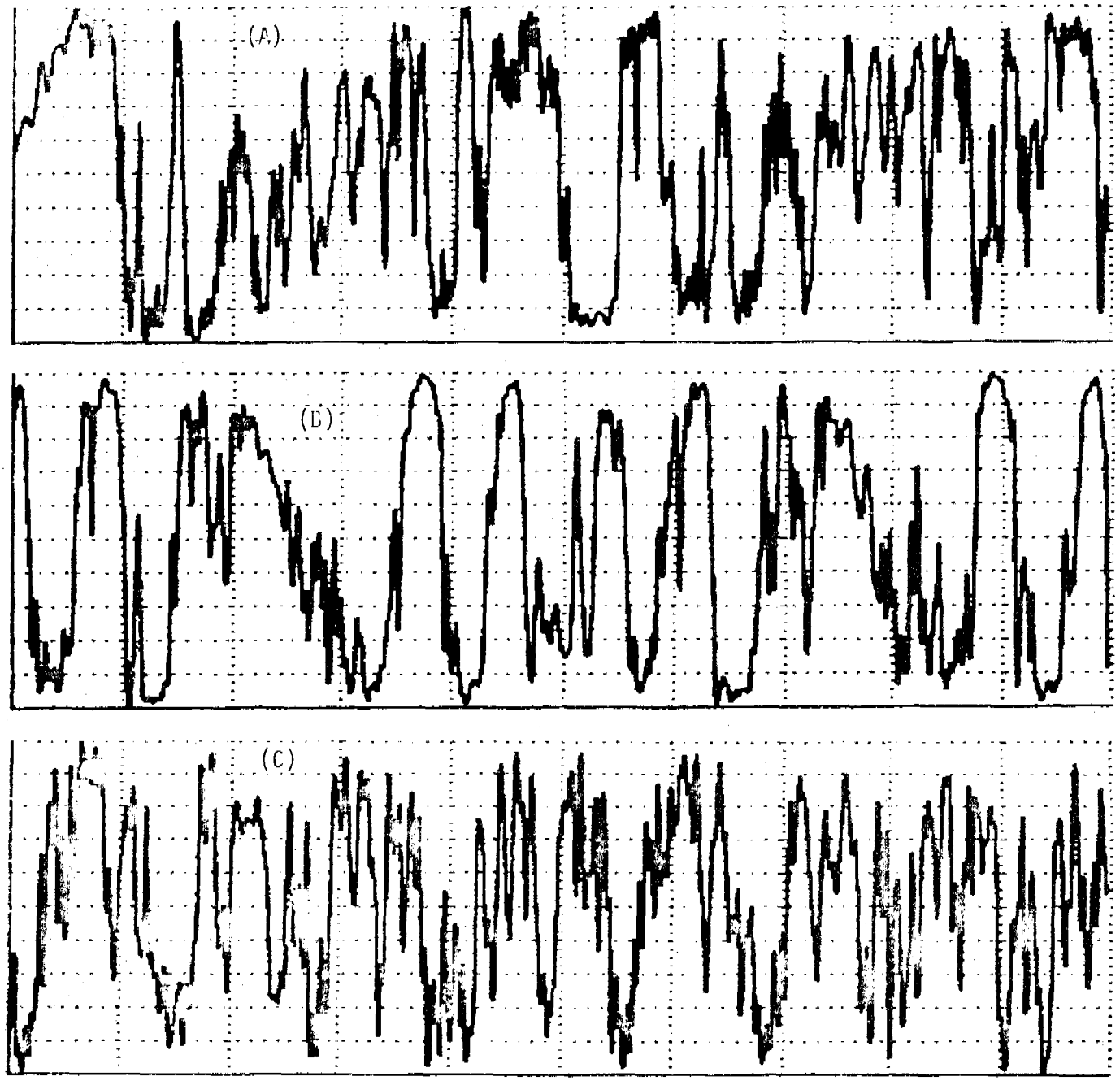
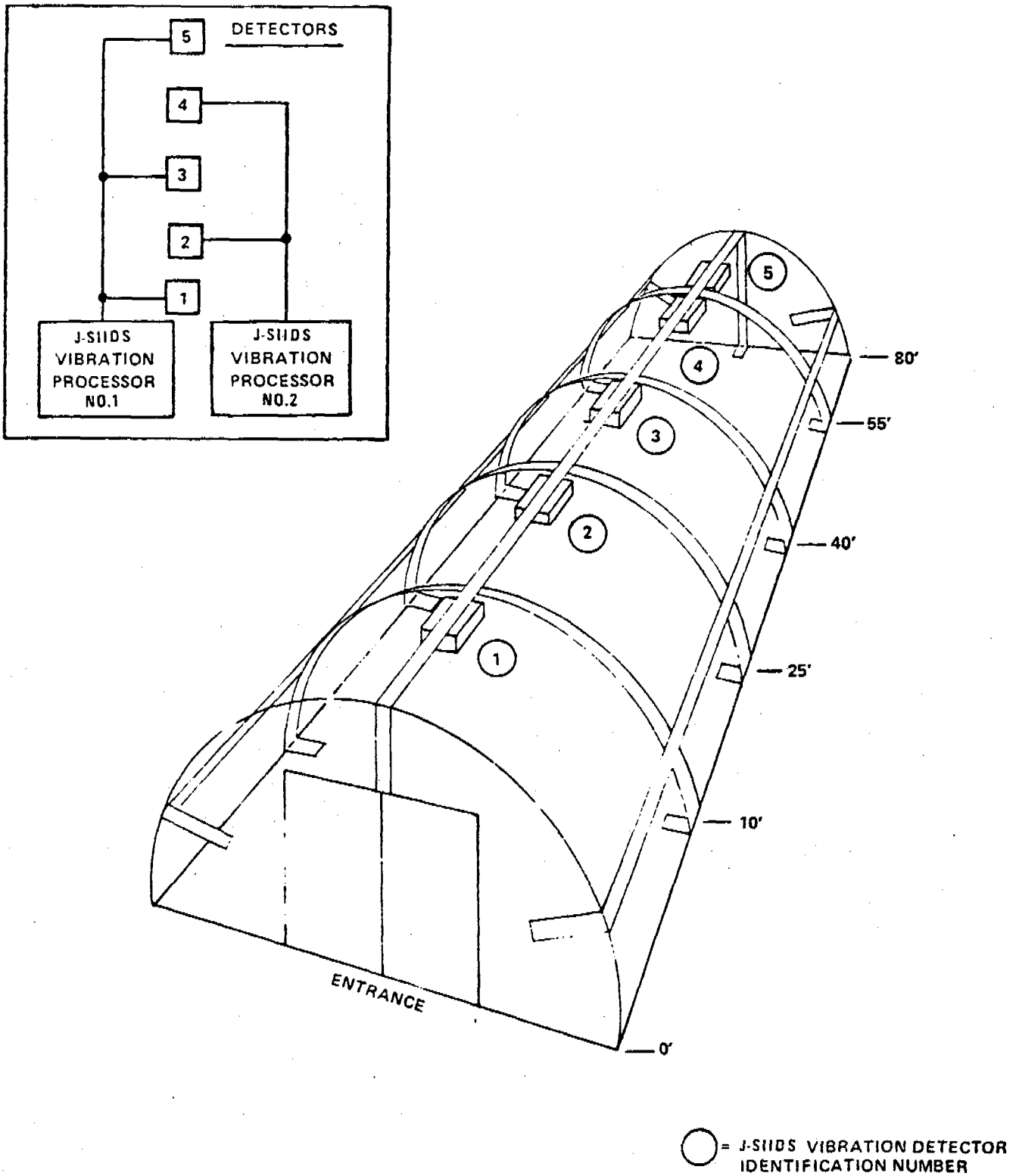


Figure 7. Data By Man Walking (Paths 7A, 7B, 7C In Figure 2)



1896A/79W

Figure 8. J-SIIDS Vibration Sensor Configuration

The bunker was covered with earth at the ground level. Five detectors of the J-SIIDS (Joint-Services Interior Intrusion Detection System) vibration sensors were mounted on the central longitudinal bar as the signal receivers which were shown in the figure. In the upper corner on the left was the configuration of detectors which showed that the outputs from the detectors were connected into two separate processors. Analog data was taken at the input to the processor and at the bandpass filter output in the processor. Therefore, four-channels of data were recorded in the analog tape in which two (ch. 7 and 6) were wide-band input and the other two (ch. 4 and 5) were noted as processed channels.

Figure 9 shows the block diagram of signal processing for the test data from the vibration sensors. The analog data was digitized and stored on the magnetic tapes. In order to acquire the digital data with the required 40KHz sampling rate for analysis, the Ampex recorder was played back with a factor of ¼ of the recording speed. With the sampling rate, a substantial amount of data was written on the tapes. To find the signals, we screened the raw digital tapes with an envelope detector and printed the locations of the signals. On the basis of printer output, we displayed the signal data and the noise data as well on the CRT. The desired data was written on the output tape for later analysis.

Figure 10 shows a display of the data from a thumper signal and its Fourier amplitude spectrum. The signal was generated by applying a thumper on the concrete surface inside the bunker. The upper trace is the time-domain data and the lower part is its frequency-domain spectrum.

The object of the present analysis on the vibration sensor data from the Yorktown test was to find the optimum detection bandwidth for the impact response data from the bunkers. The other purpose was to verify the J-SIIDS vibration sensor's bandpass characteristic. In the lower part of Figure 9 on

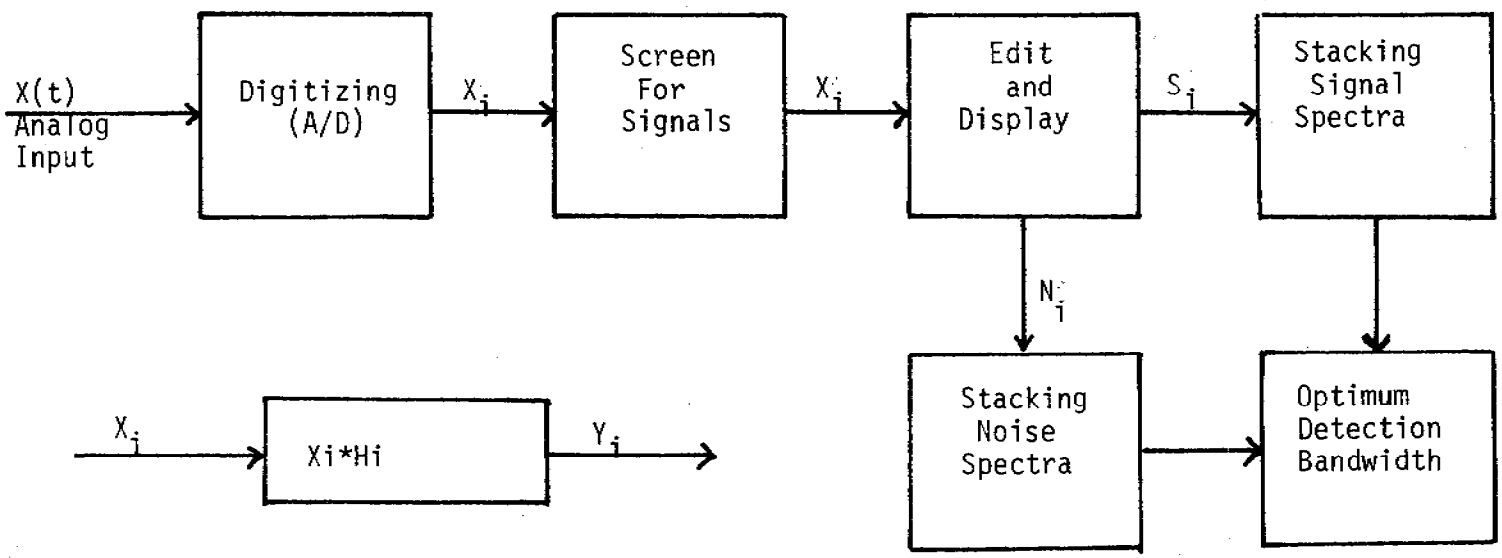


Figure 9. Block Diagram of Signal Processing For The Test Data From The Vibration Sensors

the left, the diagram shows that the input data X_i passes through a convolution filter H_i and yields the output Y_i. Then, the purpose of the present work was to find H_i so as to maximize the signal-to-noise ratio for the output Y_i. In order to accomplish the purpose, we stacked the signal spectra and the noise spectra separately from the data ensemble which have been edited on the tape. Finally, we computed the filter H_i in the frequency domain.

The vibration sensor test resulted in a large volume of analog data where a number of stimuli was used in the test. However, presented here are limited to the results of two stimuli, namely thumper and rotohammer. Furthermore, all four channels of the recorded data were processed for impact response analysis and for verification of bandpass filter characteristics. Partial accomplishment of the analysis had been presented in a technical report in September 1980 and the remaining part will soon be presented in a separate report. In the current presentation, only results from Channels 7 and 4 (i. e., processor #2 input and output) will be discussed.

4.2.1 Thumper:

Figure 11 shows the average power spectrum from the edited noise records. The noise spectrum indicated the A.C. power-source contamination in the data. Figure 12 shows the average signal amplitude spectrum from the thumper on the concrete surface. It showed two significant passbands of energy: one was in the range below 6 KHz and the other was in the 10-15 KHz bandwidth. The low-frequency band was in the normal audible range and the higher-frequency band was used for detection purpose so that false alarms can be reduced or eliminated. For the optimum computation, Figure 13 shows the frequency domain response of the filter which maximized the signal-to-noise ratio for the detection circuit. There was a 9-point smoothing similar to the Hamming Window applied to the

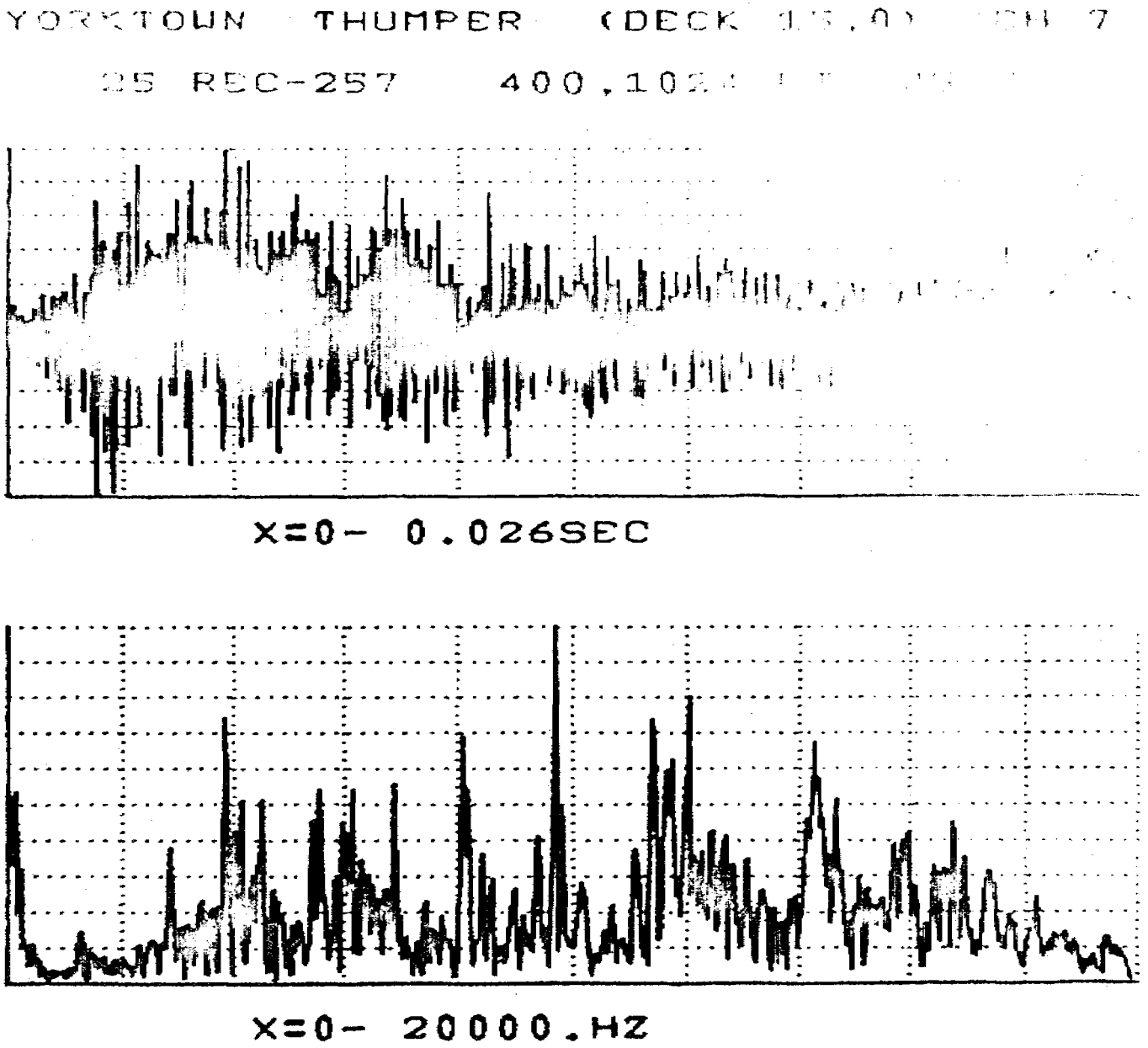


Figure 10. Display of A Thumper Signal And its Fourier Amplitude Spectrum

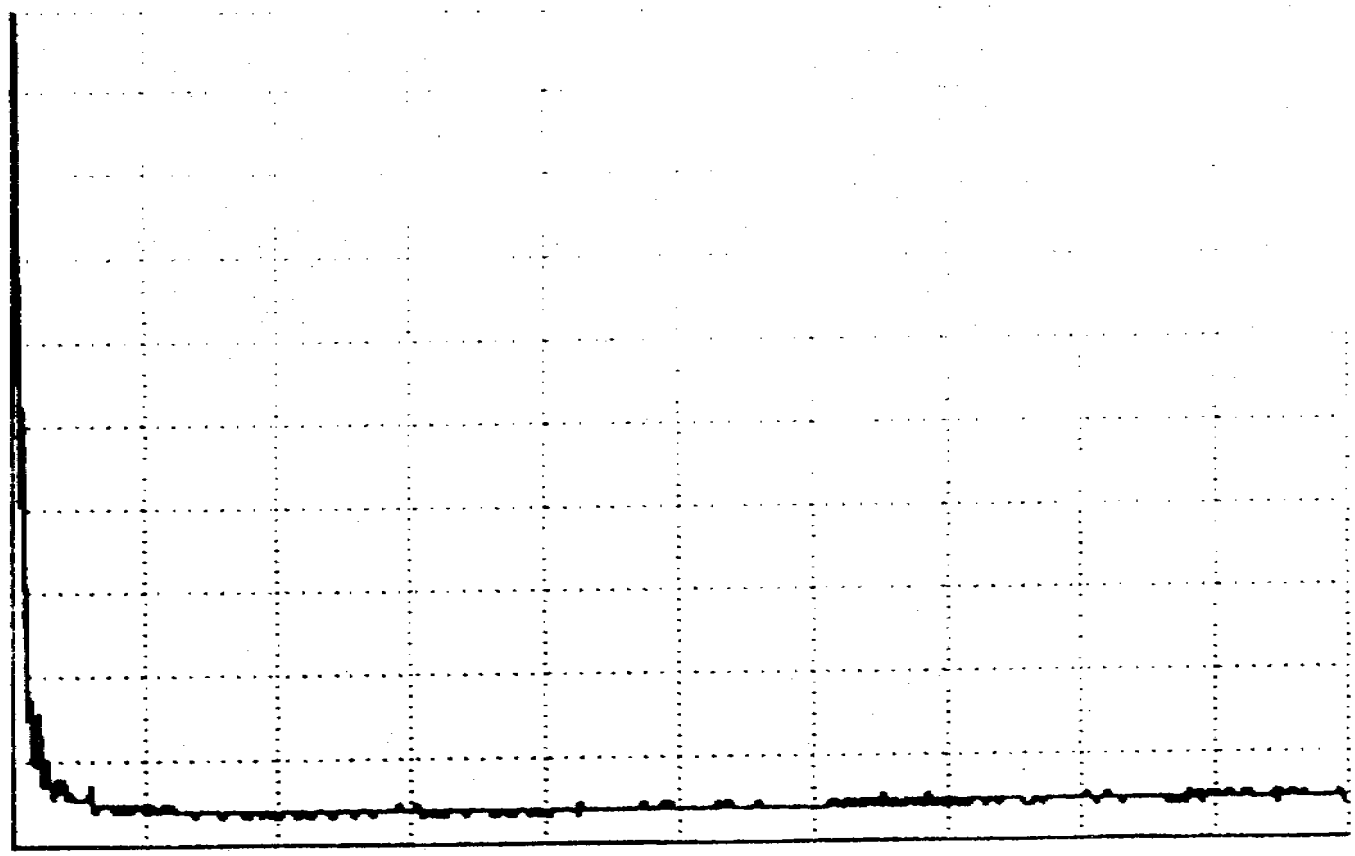


Figure 11. Average Noise Fourier Amplitude Spectrum (Thumper)

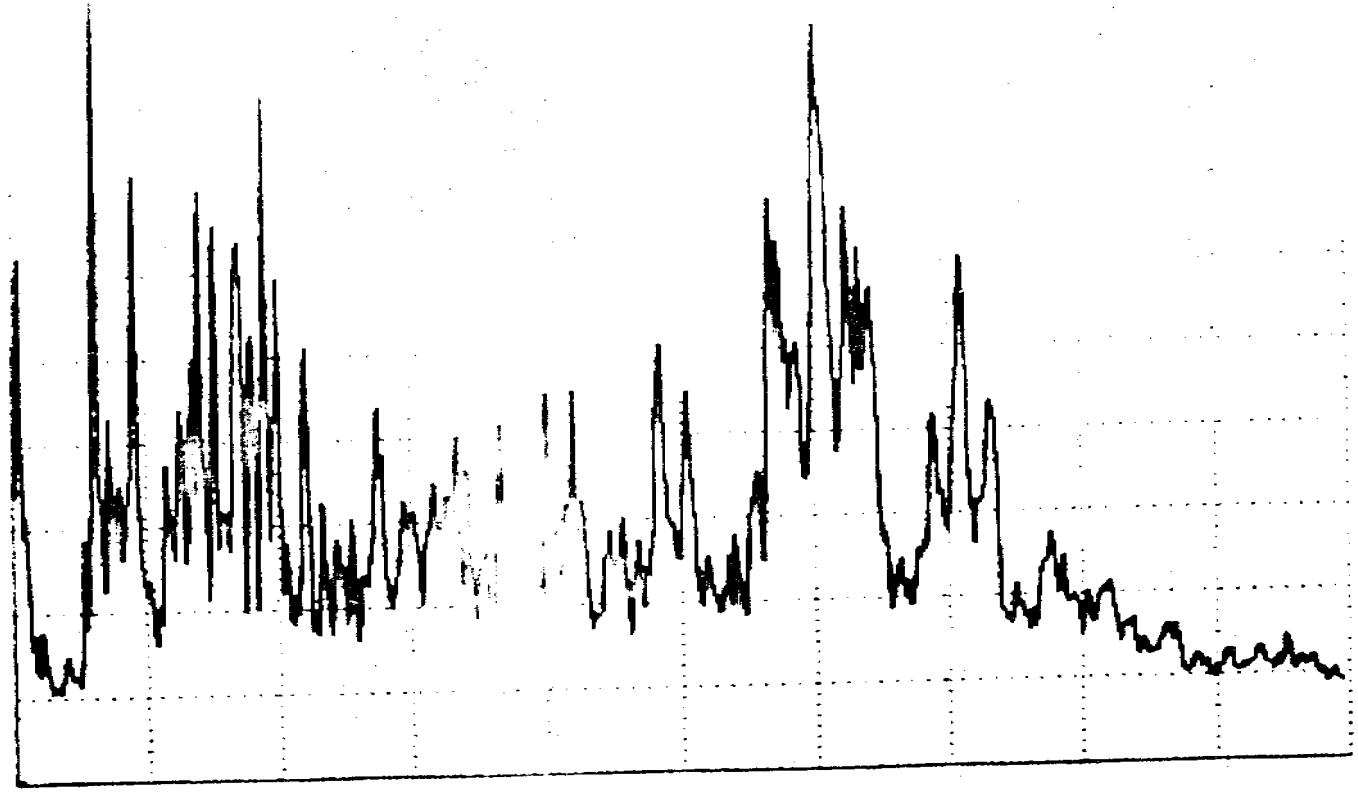


Figure 12. Average Fourier Amplitude Spectrum For Thumper Signals

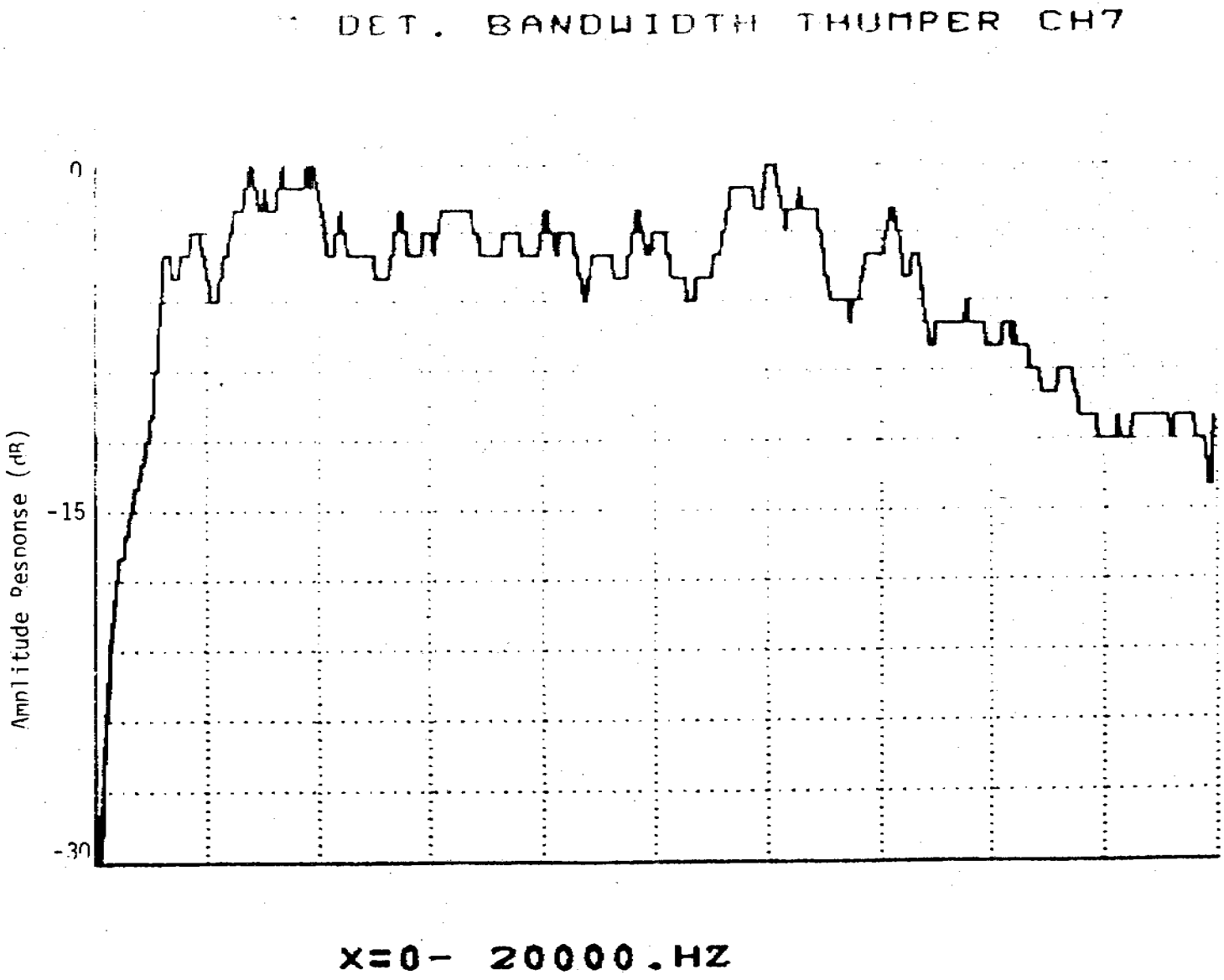


Figure 13. Optimum Detection Response For Processor 2 Input (Thumper)

response curve. The 6-dB response of the optimum detection bandwidth in the higher frequency range was about from 10 KHz to 15 KHz. Figure 14 shows the same analysis for the Channel-4 data.

4.2.2. Roto Hammer

The same analysis was done and presented for the signals using the roto hammer as stimulus. Figure 15 presents the average frequency-domain amplitude spectrum for the noise records taken during the roto hammer test. The noise samples might include the data while the rotohammer was running in the air. The rotohammer ran continuously for a period of time at each test point during the test. Figure 16 shows the average Fourier amplitude spectrum for the rotohammer signals. There are two apparent important bands of energy in the spectrum: one at the low frequency range below 4 KHz and the other at the 8-12 KHz passband. (Note: For this part of the data, the digitization was done after the Ampex recorder was being used in the field for some time. As compared with Figure 13, we thought that the speed of the recorder might not be very accurate and it might be slowed down somewhat while the data was being digitized). For the results of optimum detection analysis, Figure 17 shows the response curve in the frequency domain for the processor 2 input (channel 7, wideband). There was no smoothing applied to the response curve. Figure 18 shows the response curve where smoothing has been done. Figure 19 shows the computation for the processed channel (ch. 4). Compared with Figure 14, the response band was shifted to the lower frequency range, possibly due to the cause of the Ampex recorder as noted earlier.

5.0 Summary and Discussion

5.1 Summary

The current data acquisition and analysis system for the physical security RDT&E has been briefly presented. The intrusion signatures from RF motion sensors

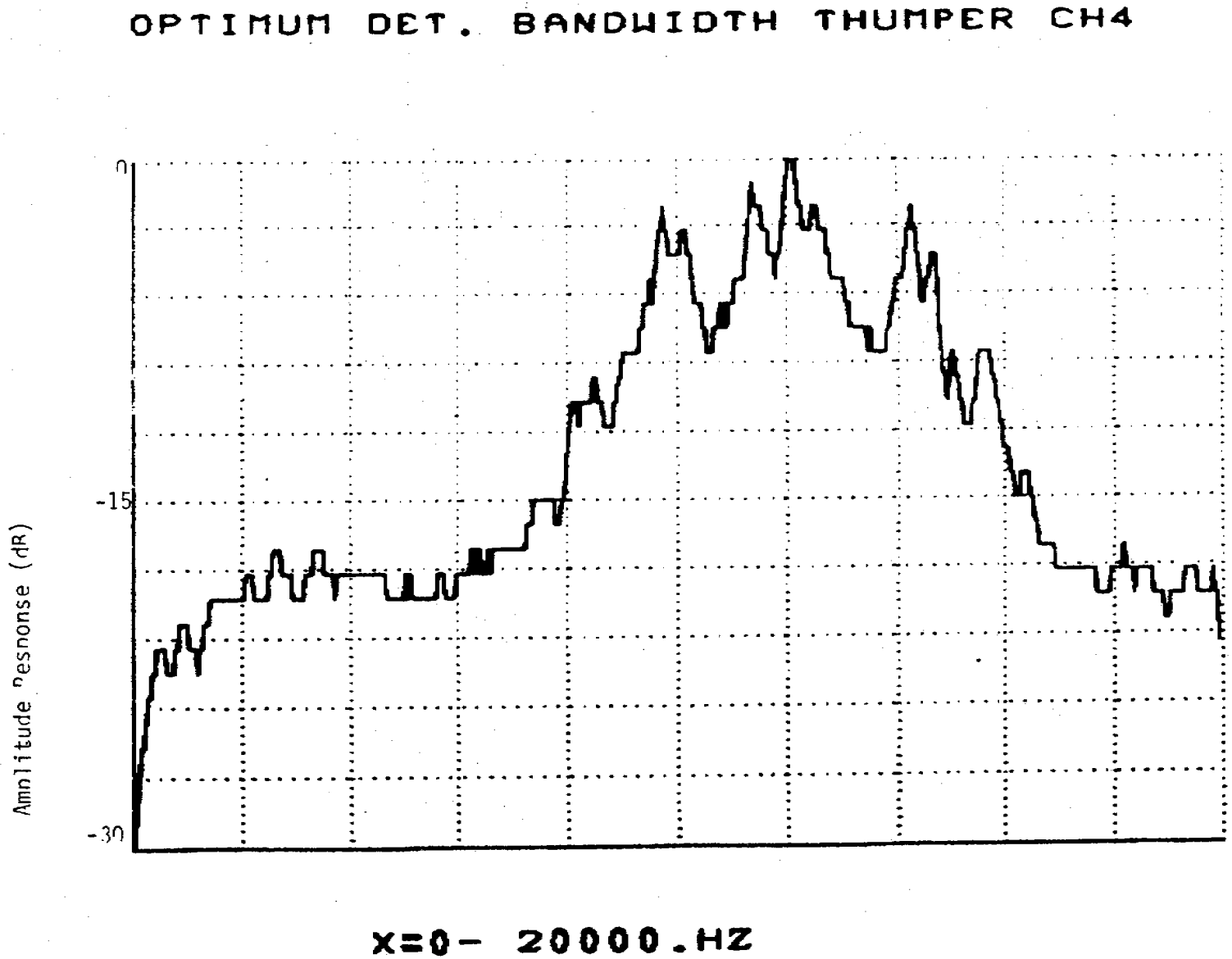


Figure 14. Optimum Detection Response For Processor 2 Output (Thumper)

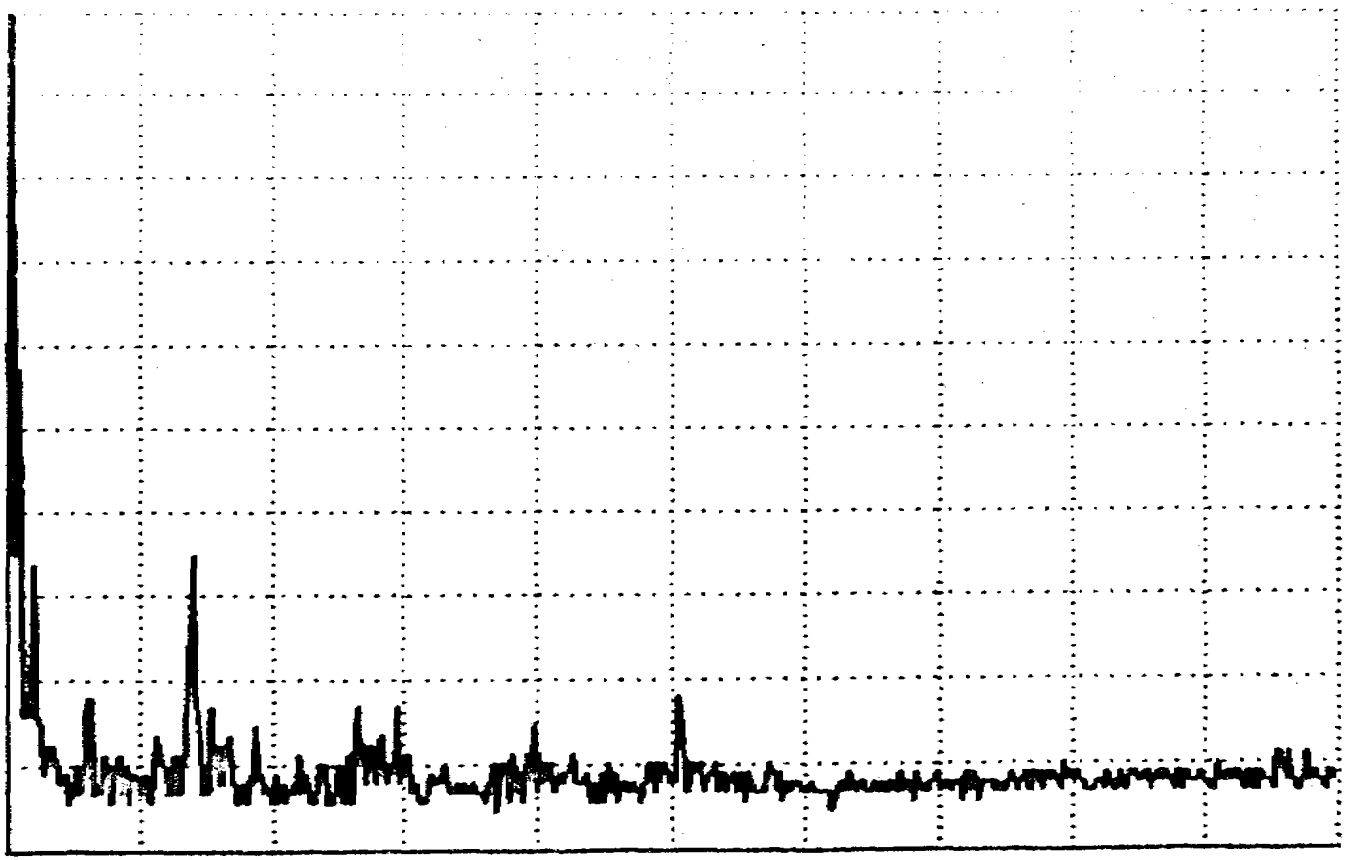


Figure 15. Average Fourier Amplitude Spectrum For Noise Samples (Rotohammer)

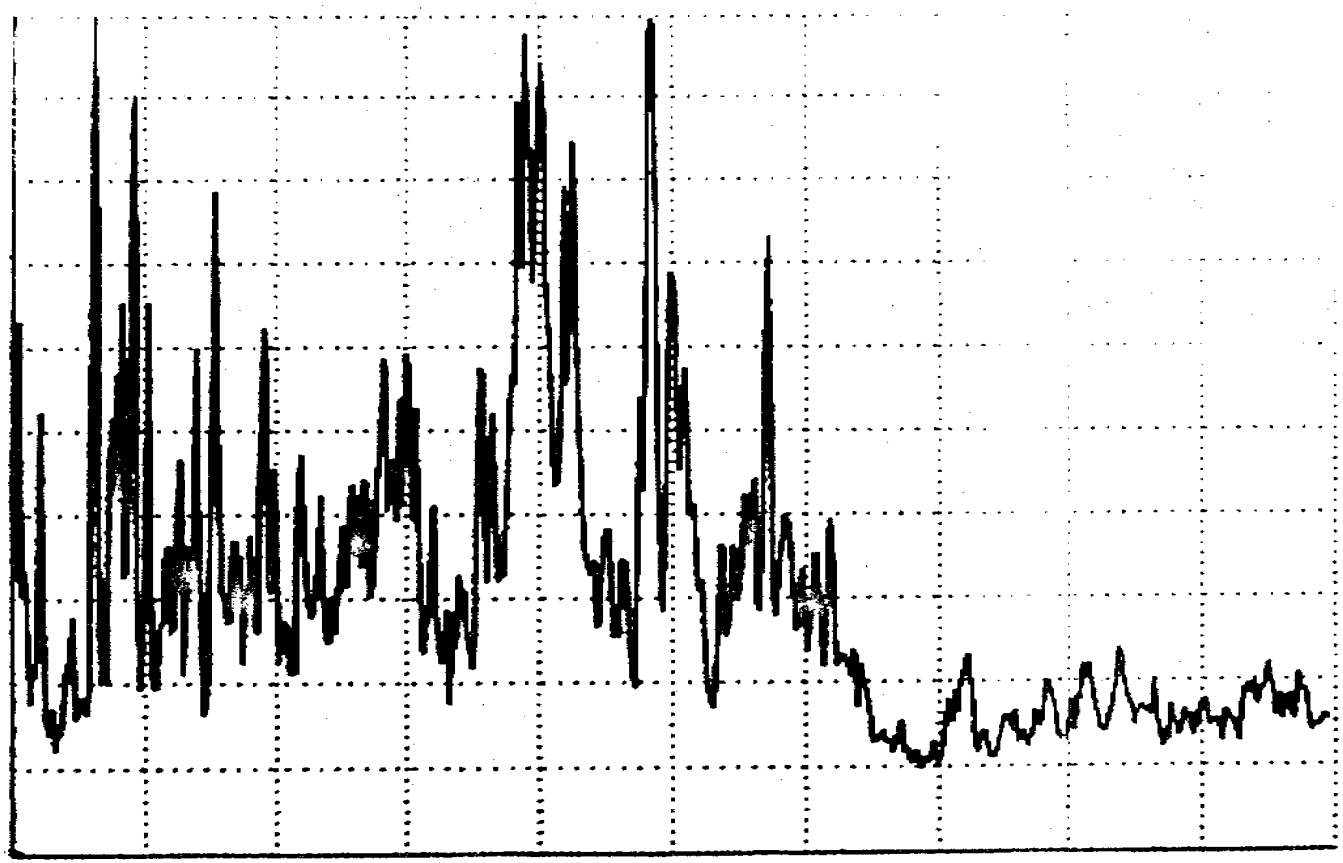


Figure 16. Average Fourier Amplitude Spectrum For The Potohammer Signals

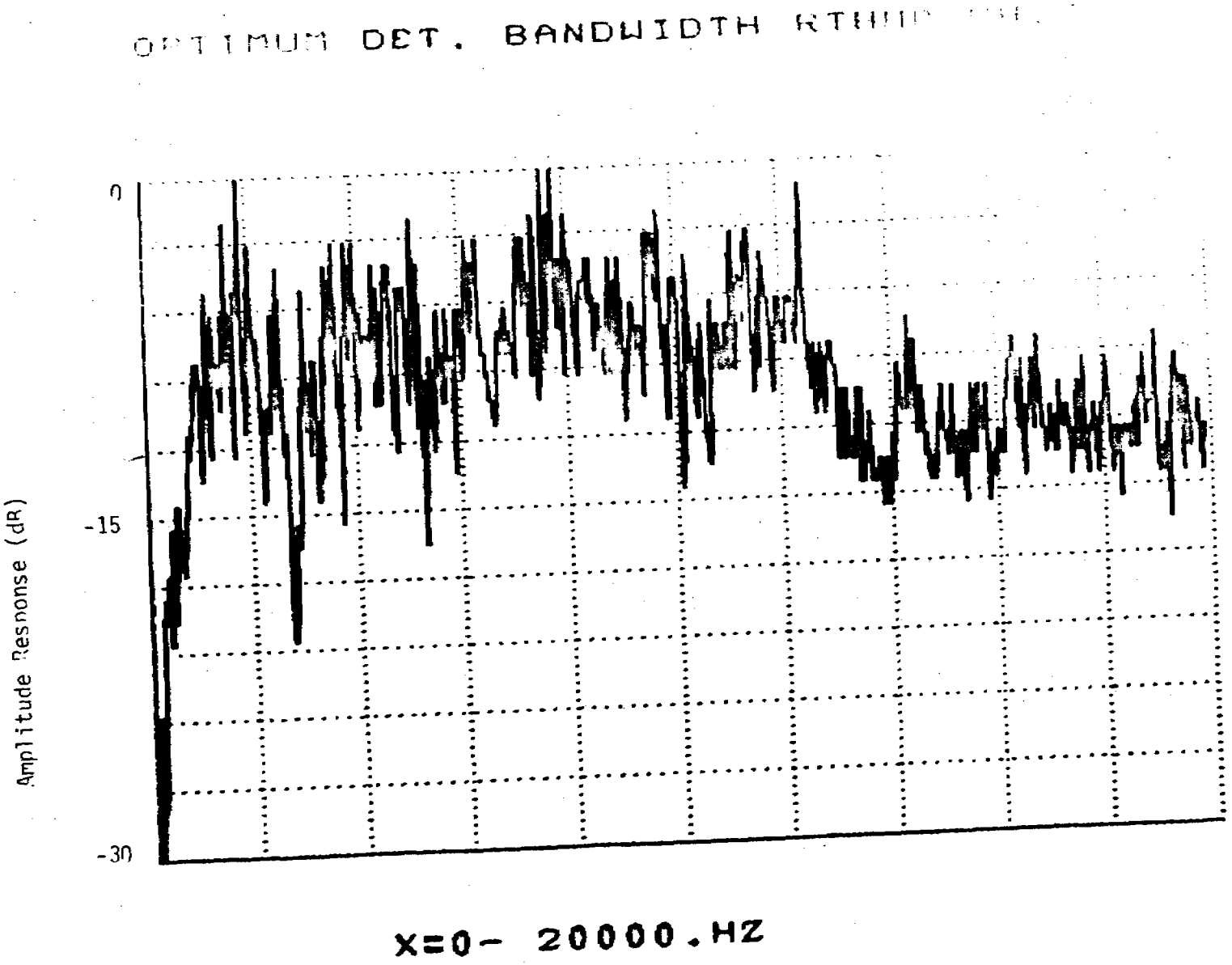


Figure 17. Optimum Detection Response For Processor 2 Input (Roto Hammer)

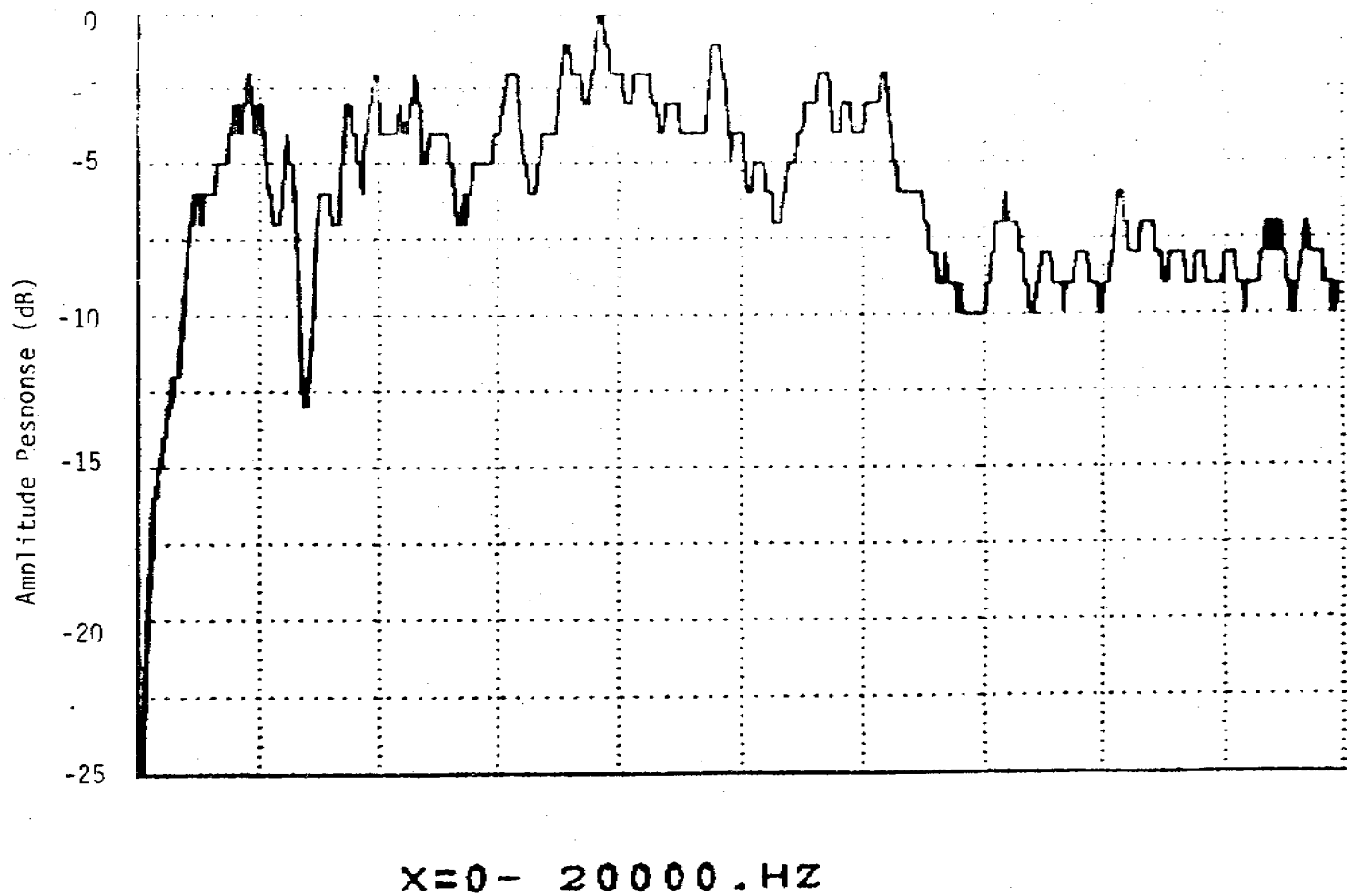


Figure 18. Optimum Detection Response For Potohammer
(Smoothed Version of Figure 17)

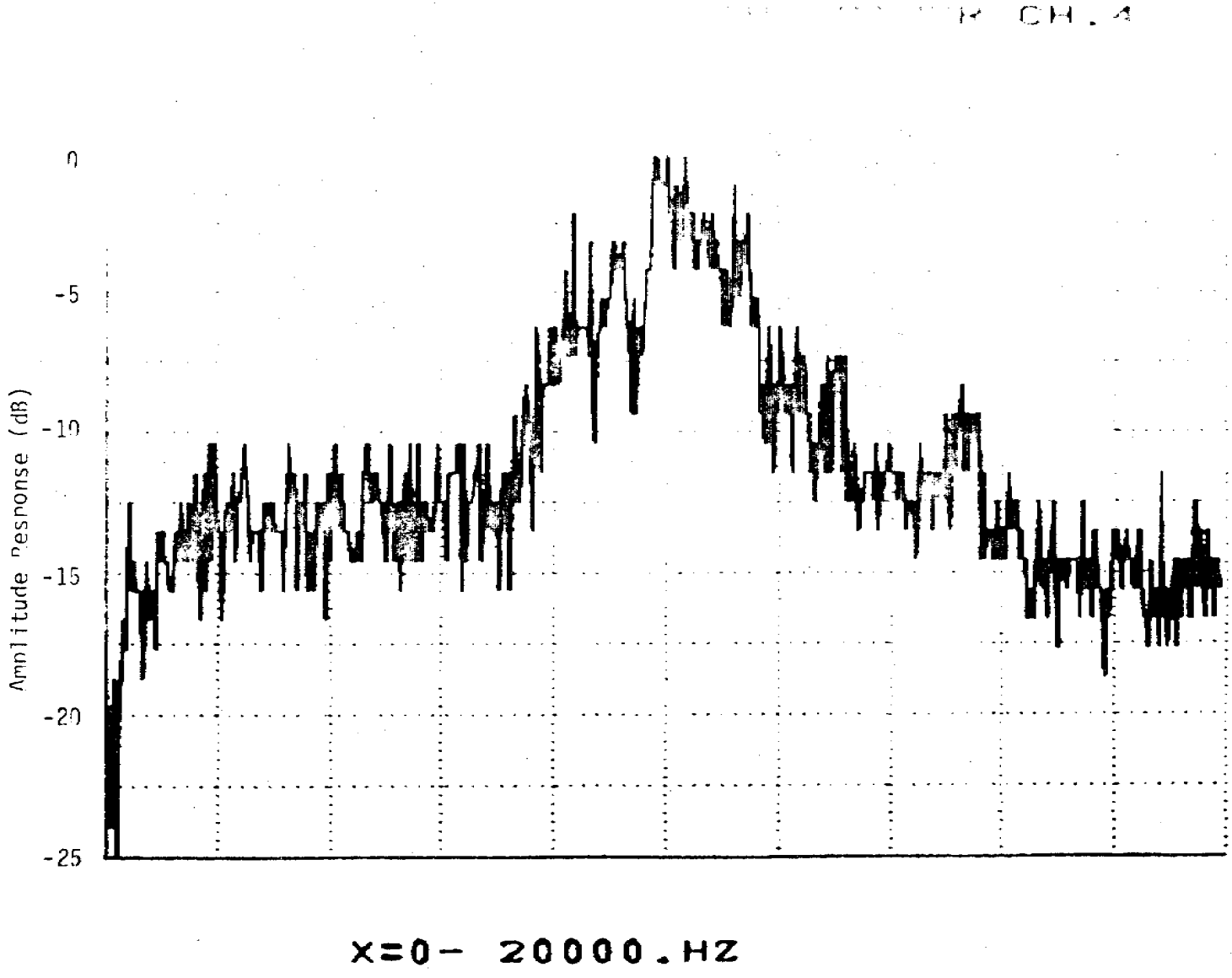


Figure 19. Optimum Detection Response For Processor 2 Output (Rotohammer)

and vibration sensors have been discussed. The time-domain RF sensor data appeared to be comprehensible. In addition to envelope detection of intrusions, detections by applying the time-domain features seem feasible and can be attempted. Parametrizations using the frequency-domain amplitudes may also be an important area for feature extractions.

The impact response data from concrete surface suggested two important frequency bands of energy concentration: one was in the low frequency range below 6 KHz and the other in the 10 KHz - 15 KHz range. The former passband is in the normal audible range and is not suitable for application in detecting the forced intrusion. However, the latter passband is useful for detecting the forced intrusion in the structure of the earth-covered concrete where the false alarms can be reduced to a minimum.

5.2 Discussion

The present DEC PDP 11/05 minicomputer is about adequate for data acquisition and display with some computing capability. The difficulty in performing extensive computing using the current system lies in the fact that its memory capacity is only 8K words. However, the system will soon be upgraded a medium range PDP 11 system with additional peripherals. We expect that the computing and data-handling capabilities of the data analysis system will be enhanced significantly.

The time-domain features from the RF motion sensor is comprehensible for some cases. Time-domain feature study can be done by simplifying the transmitter and receiver components for better understanding and gradually by increasing the elements of transmitters and/or receivers to the configuration for the practical applications such as those shown in the test.

Data from vibration sensor test in Yorktown, Virginia was very voluminous. Various types of data in different conditions were taken and the data qualities were also varied from one case to the other. The processed channel which was designed for detection in the vibration sensor was in agreement with the impact response data from the earth-covered concrete.

APPENDIX

This appendix compiled the data for the RF motion sensor test at Building 2093. A total of 21 tests was conducted. Presented here are the displays of the time-domain amplitudes (1024 points) and the frequency-domain spectra and the frequency-domain spectra. The right-most number in the second line was the maximum count of the time-domain amplitudes. For example, in Figure A-1, the maximum amplitude was 249, which was relatively lower as compared with those of the walking tests, say 1027 in Figure A-10.

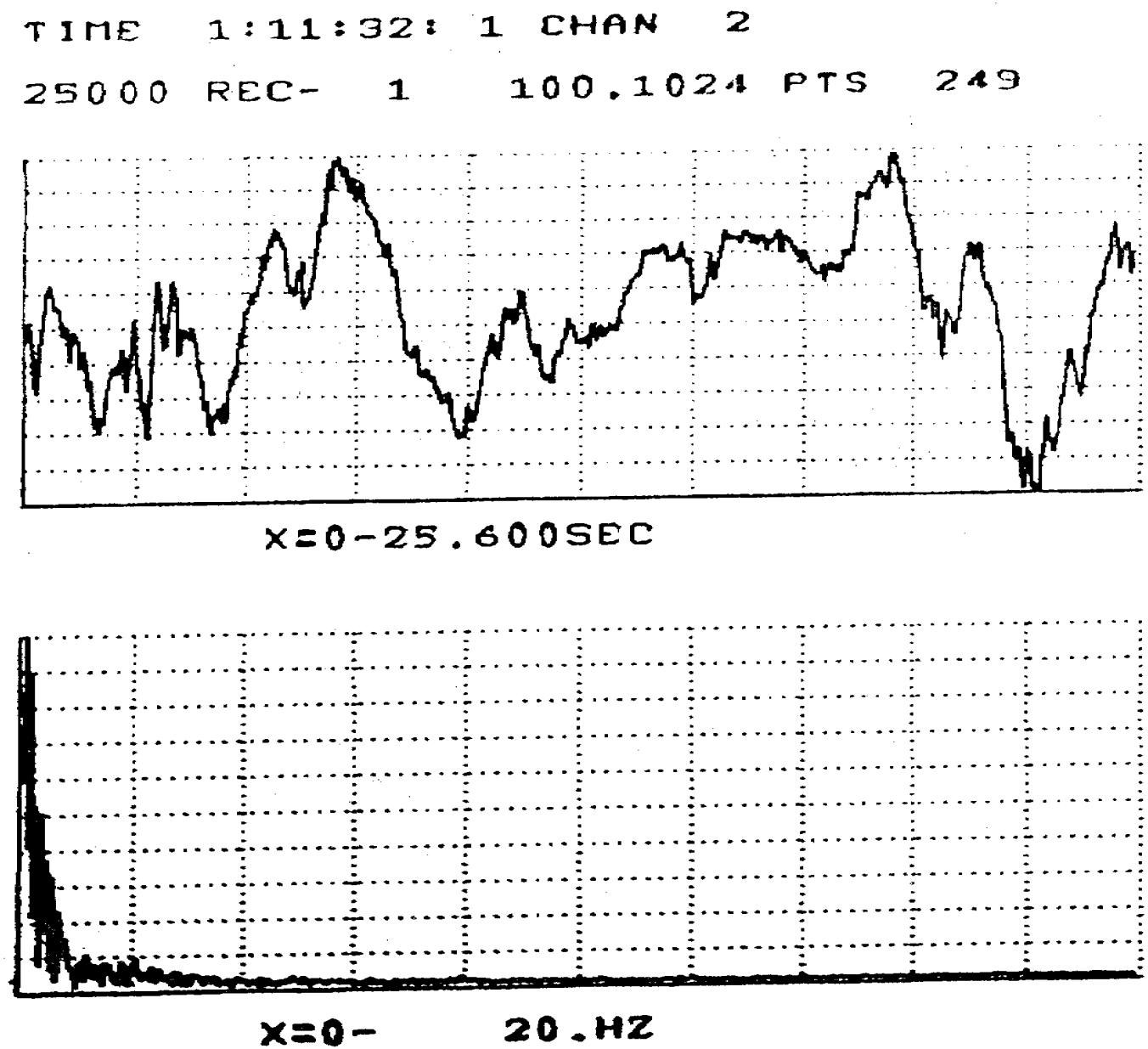


Figure A-1. Background-Heating System in Warm-up Condition

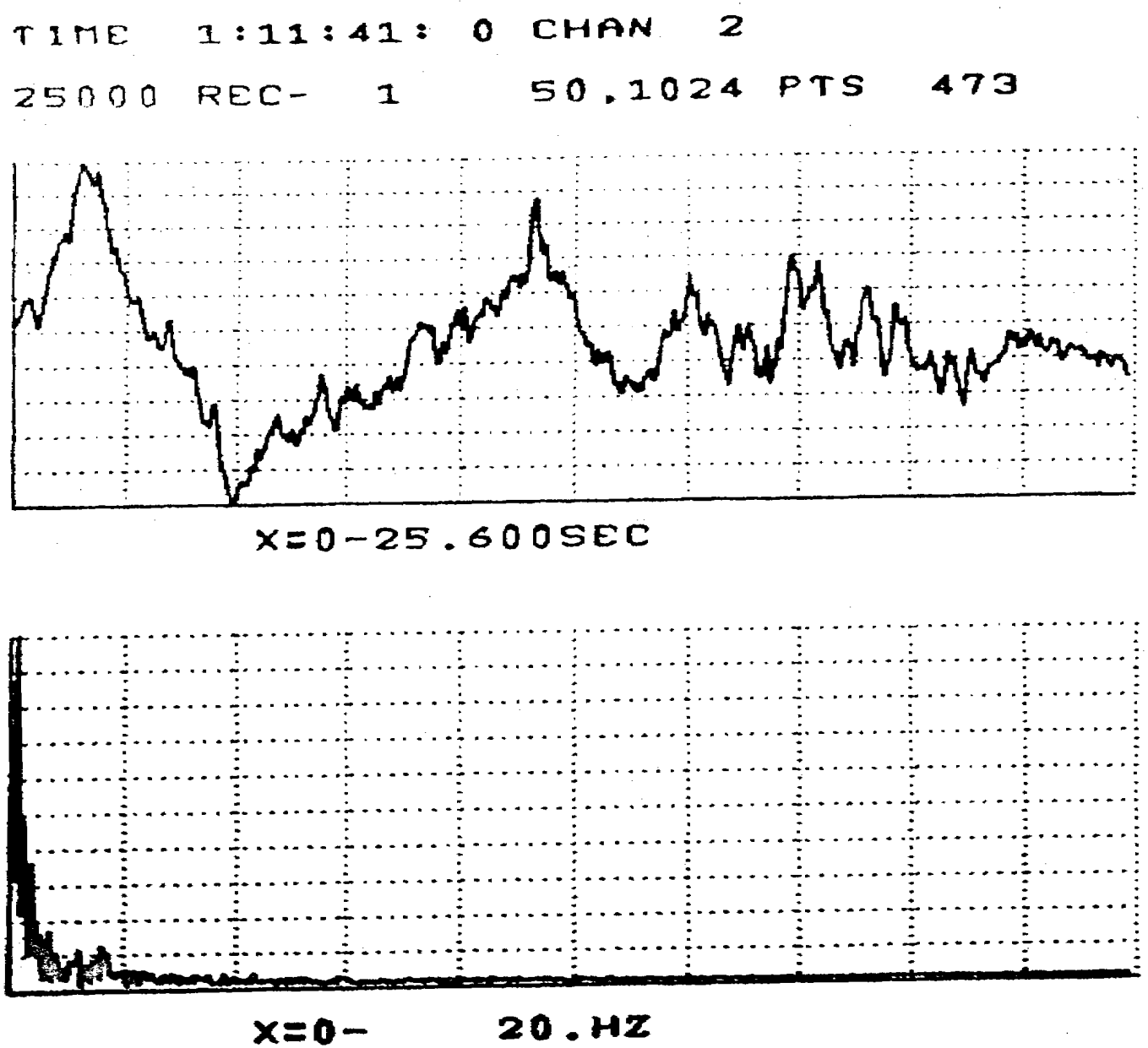


Figure A-2. Background - No Stimuli (Large Door Rattling In the Wind)

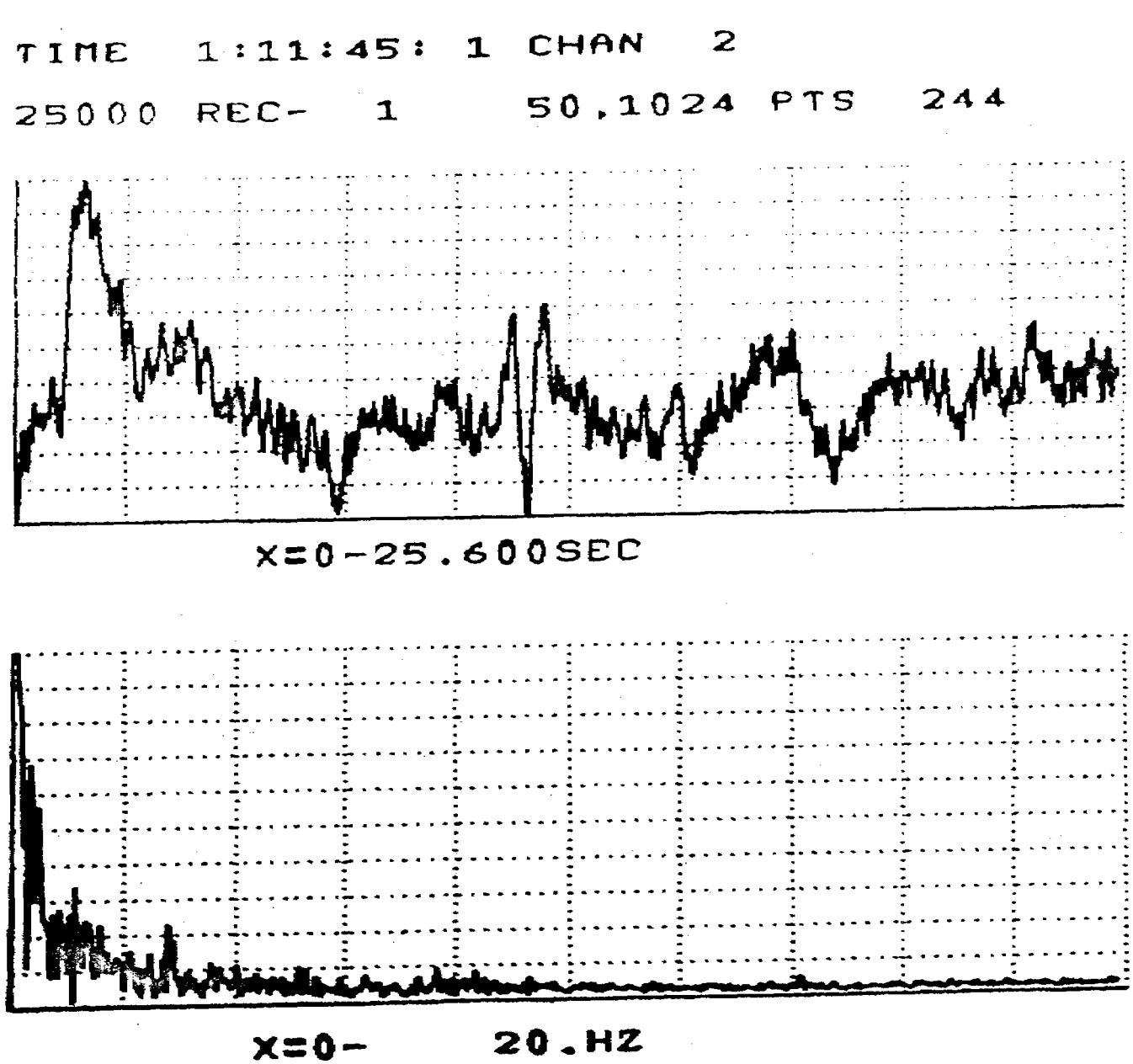


Figure A-2. Background - Heating Air Flow On Duct

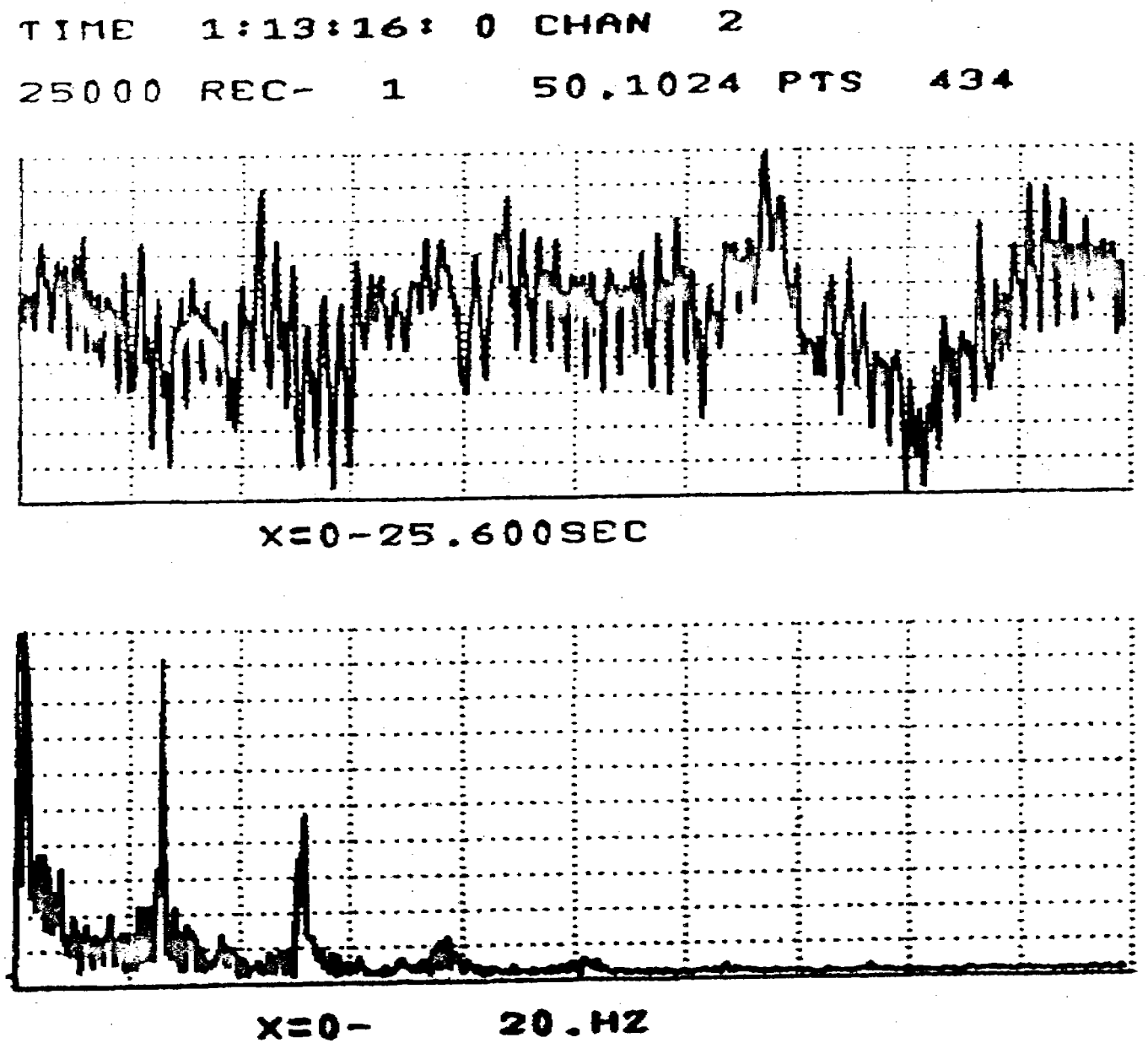


Figure A-4. Shaking Large Steel Door From Outside

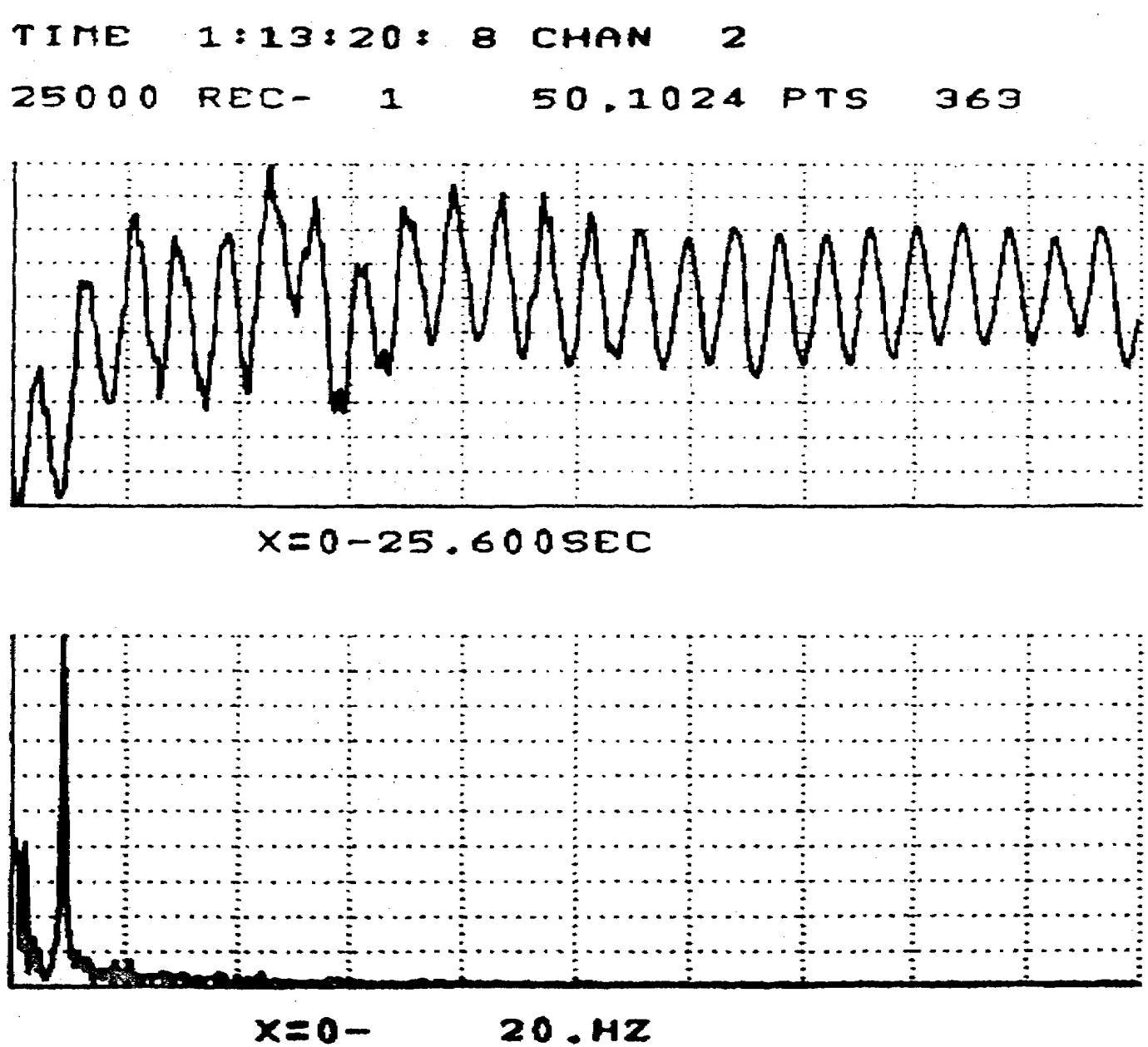


Figure A-5. Background - No Stimuli

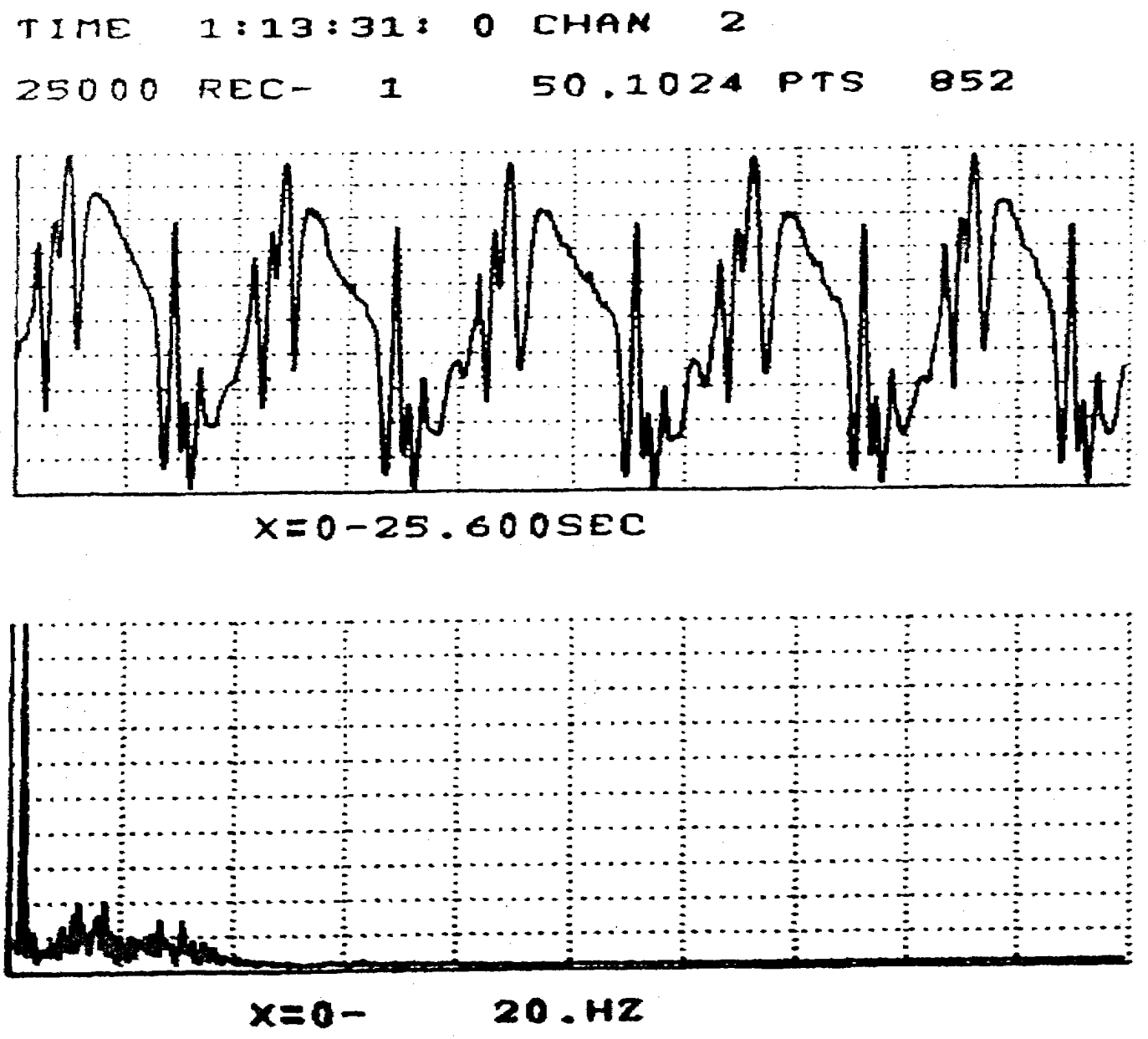


Figure A-6. Opening and Closing The Entrance Door Normally For Five Times

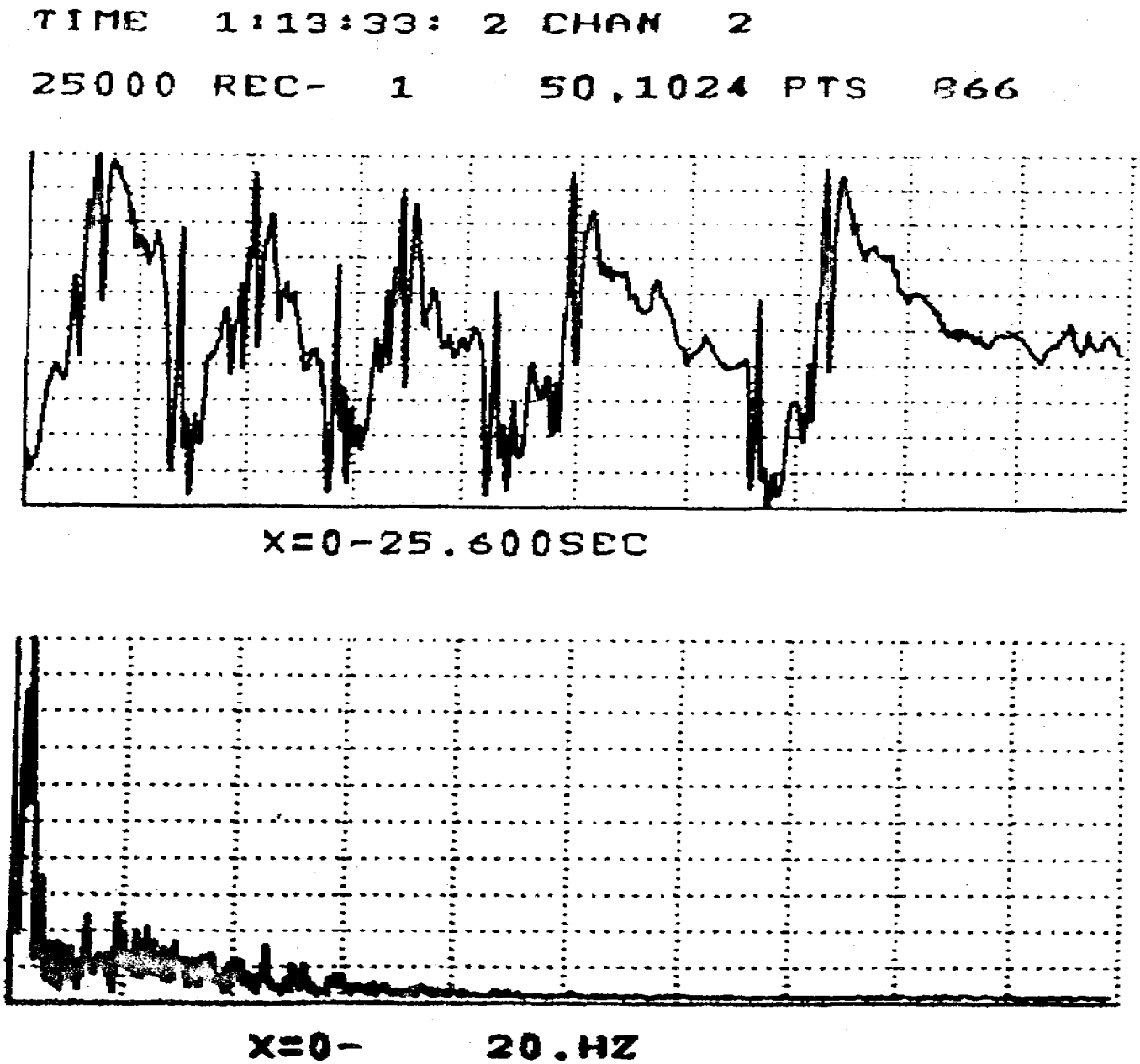


Figure A-7. Opening And Slamming Shut The Entrance Door Five Times

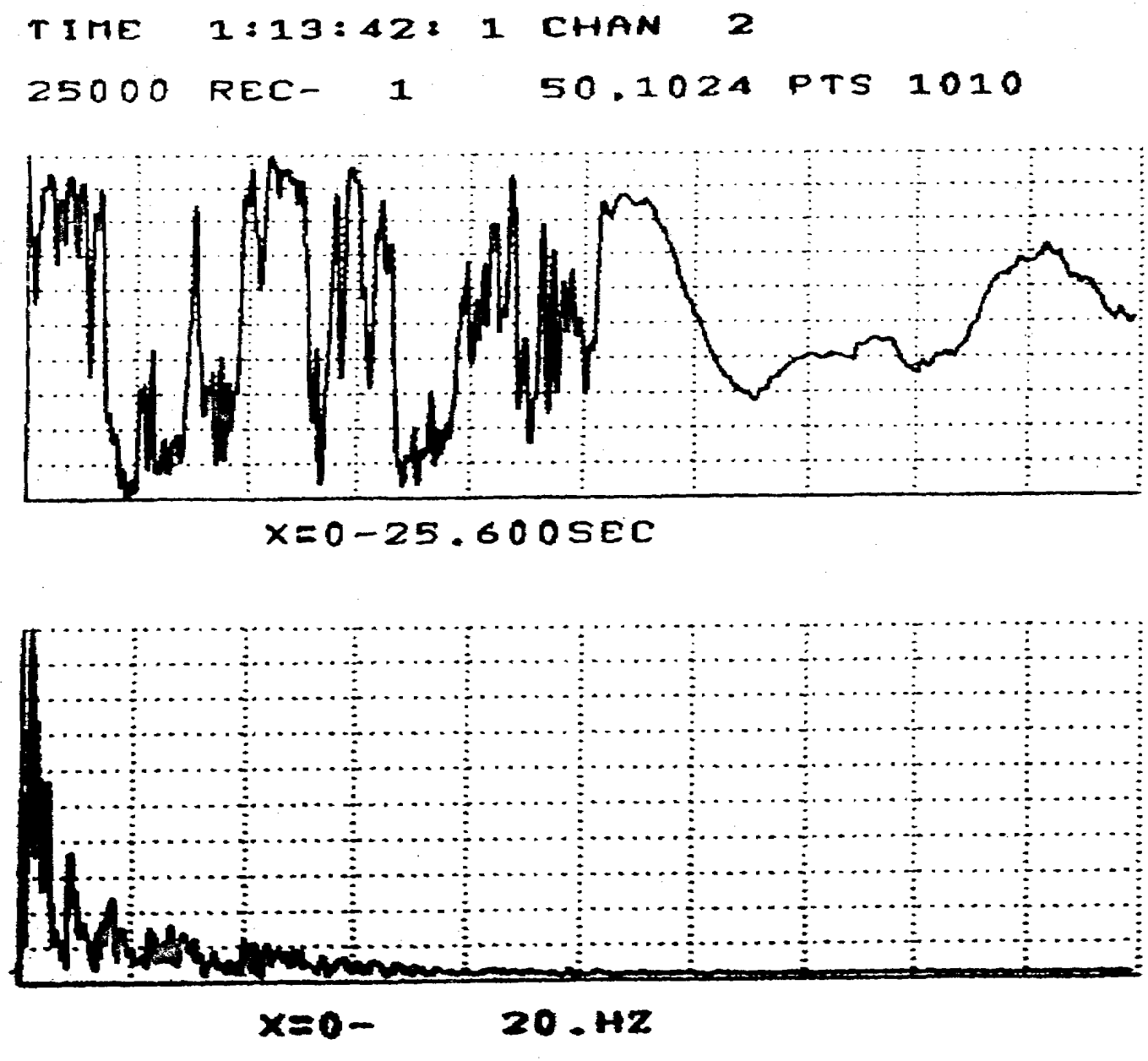


Figure A-8. Man Walking (From Entrance Door To The Office Area. See Figure 2, Path 6)

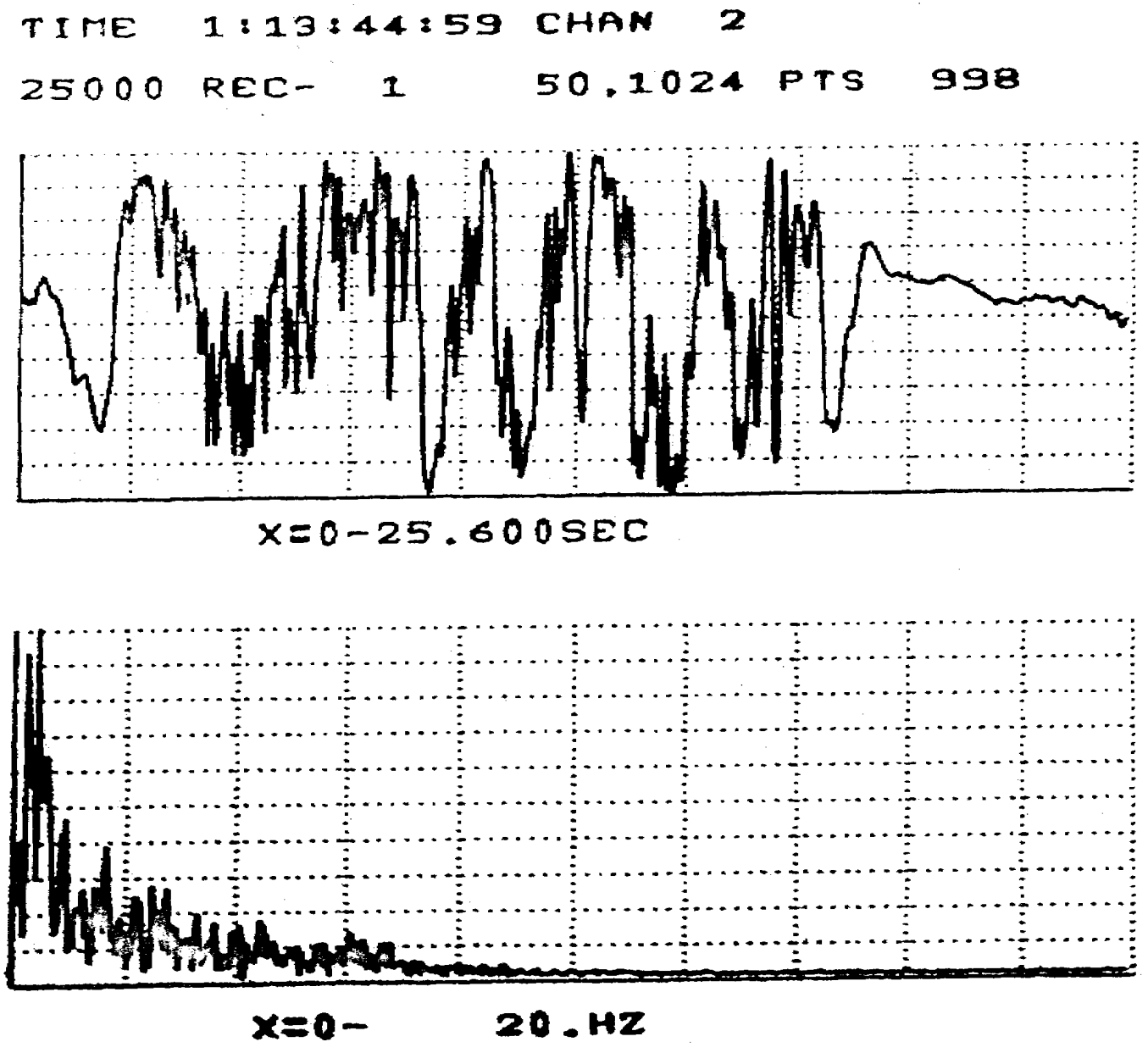


Figure A-9. Man Walking (Opposite To That Of Figure A-8)

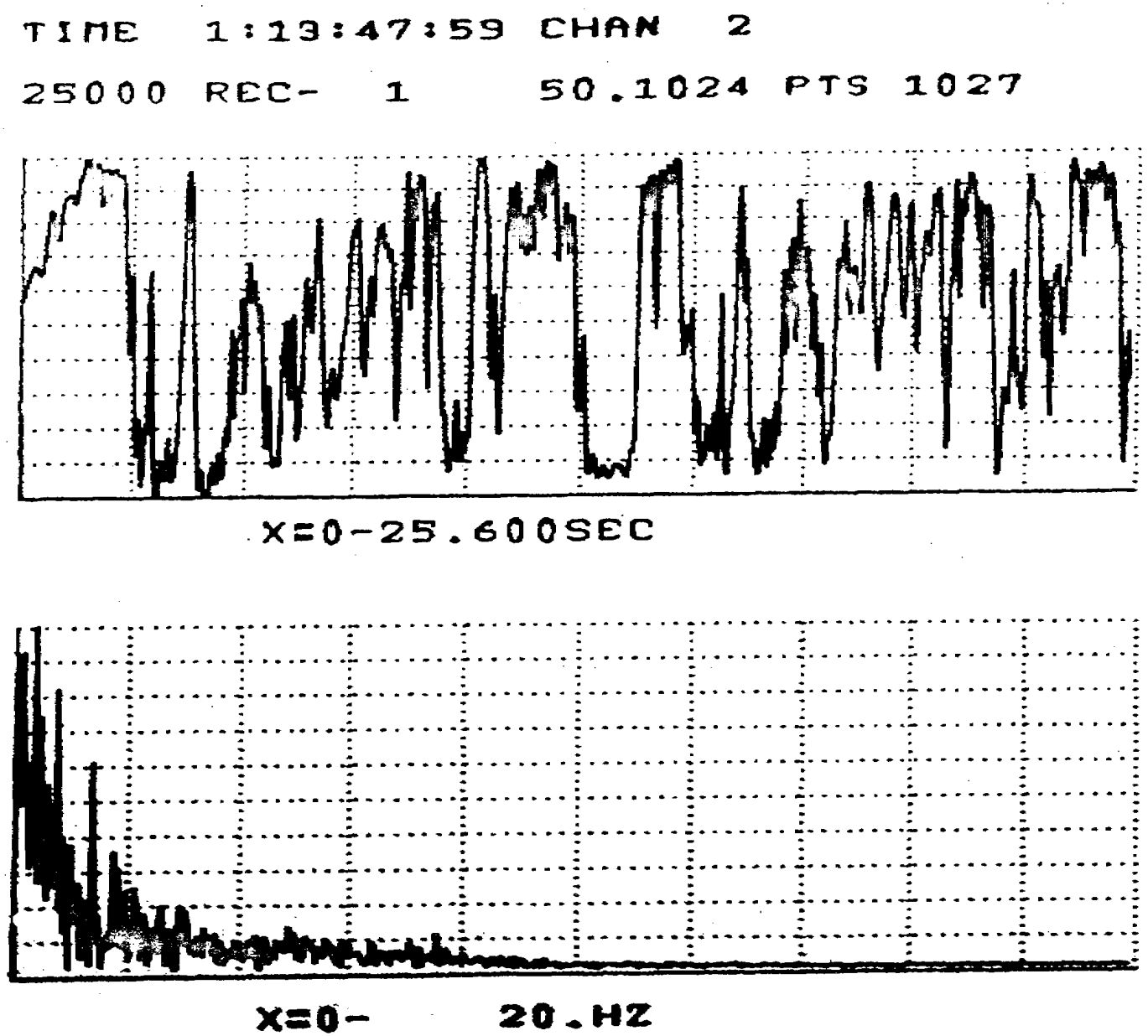


Figure A-10. Man Walking (See Figure 2, Path 7A)

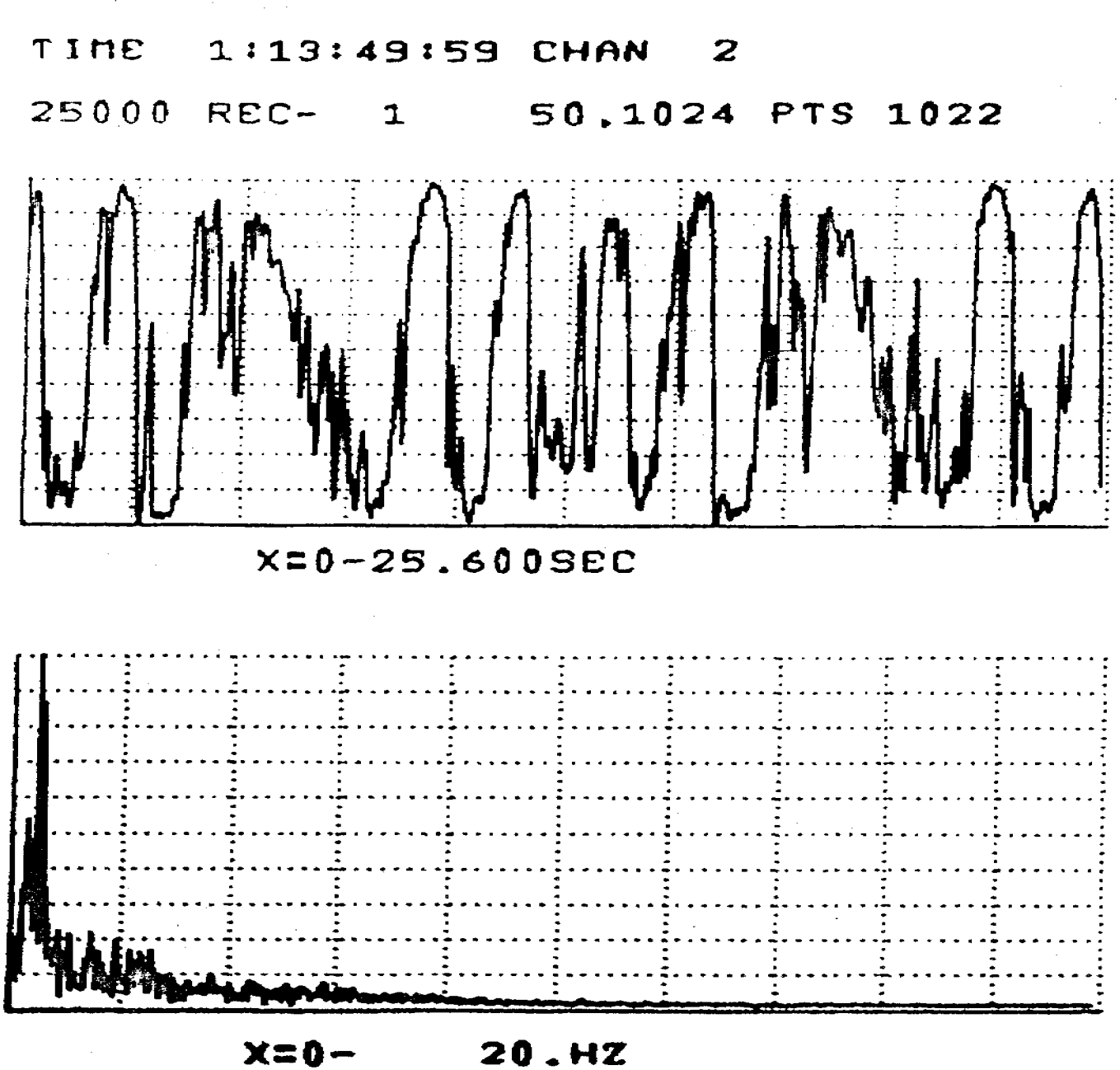


Figure A-11. Man Walking (See Figure 2, Path 7B)

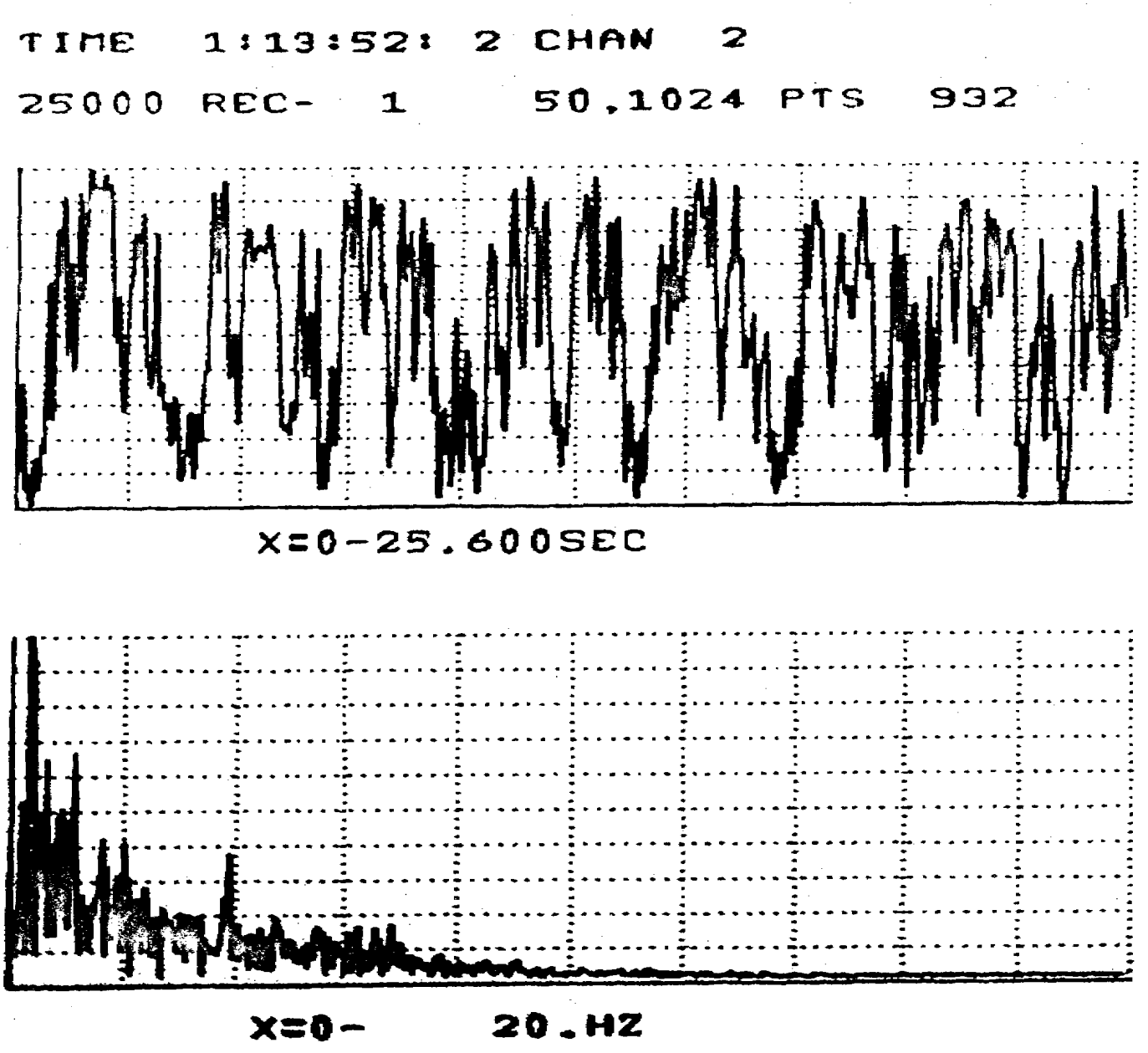


Figure A-12. Man Walking (See Figure 2, Path 7C)

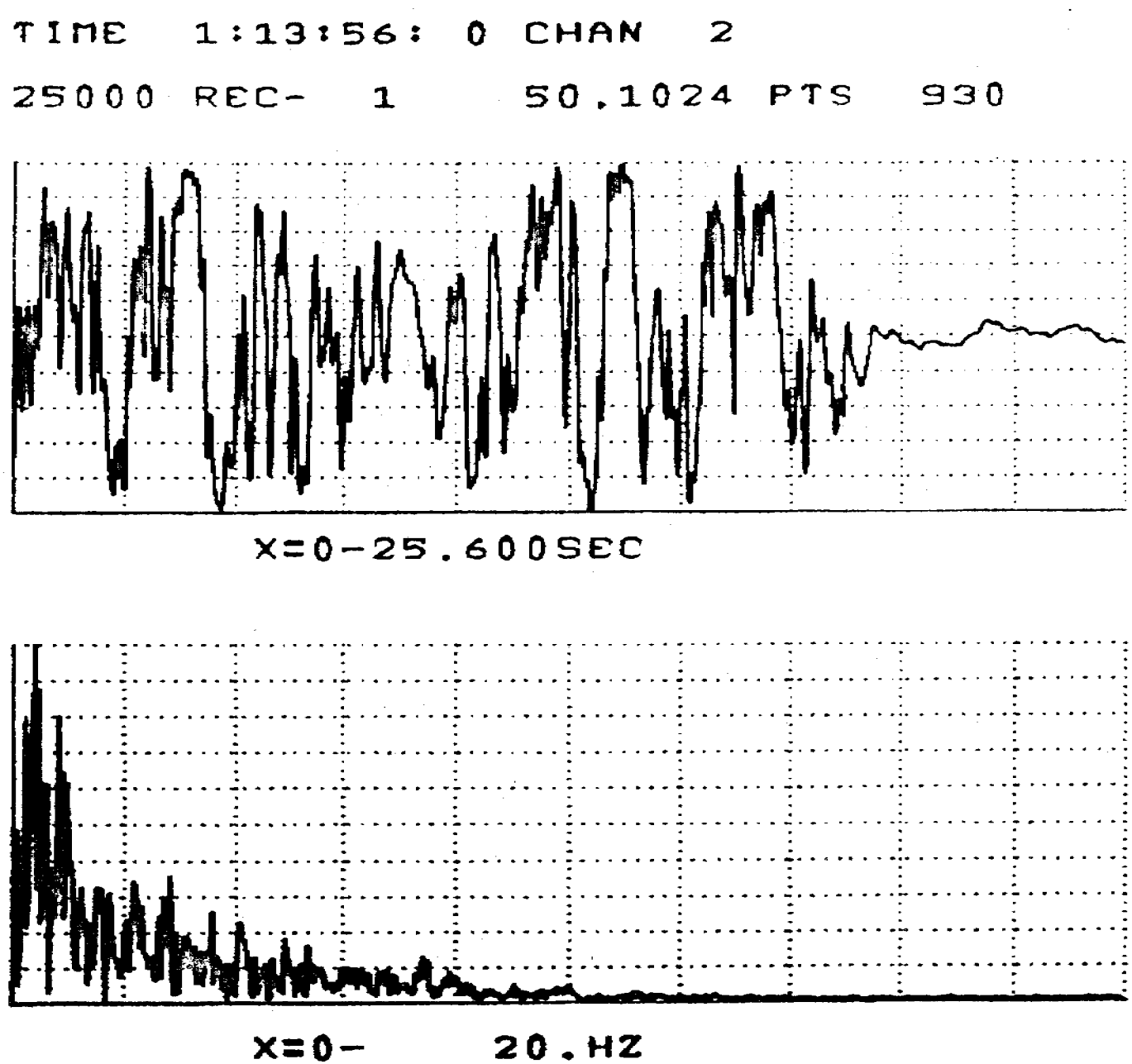


Figure A-13. Man Walking (From Right To Left, 45° To The Central Line And Back)

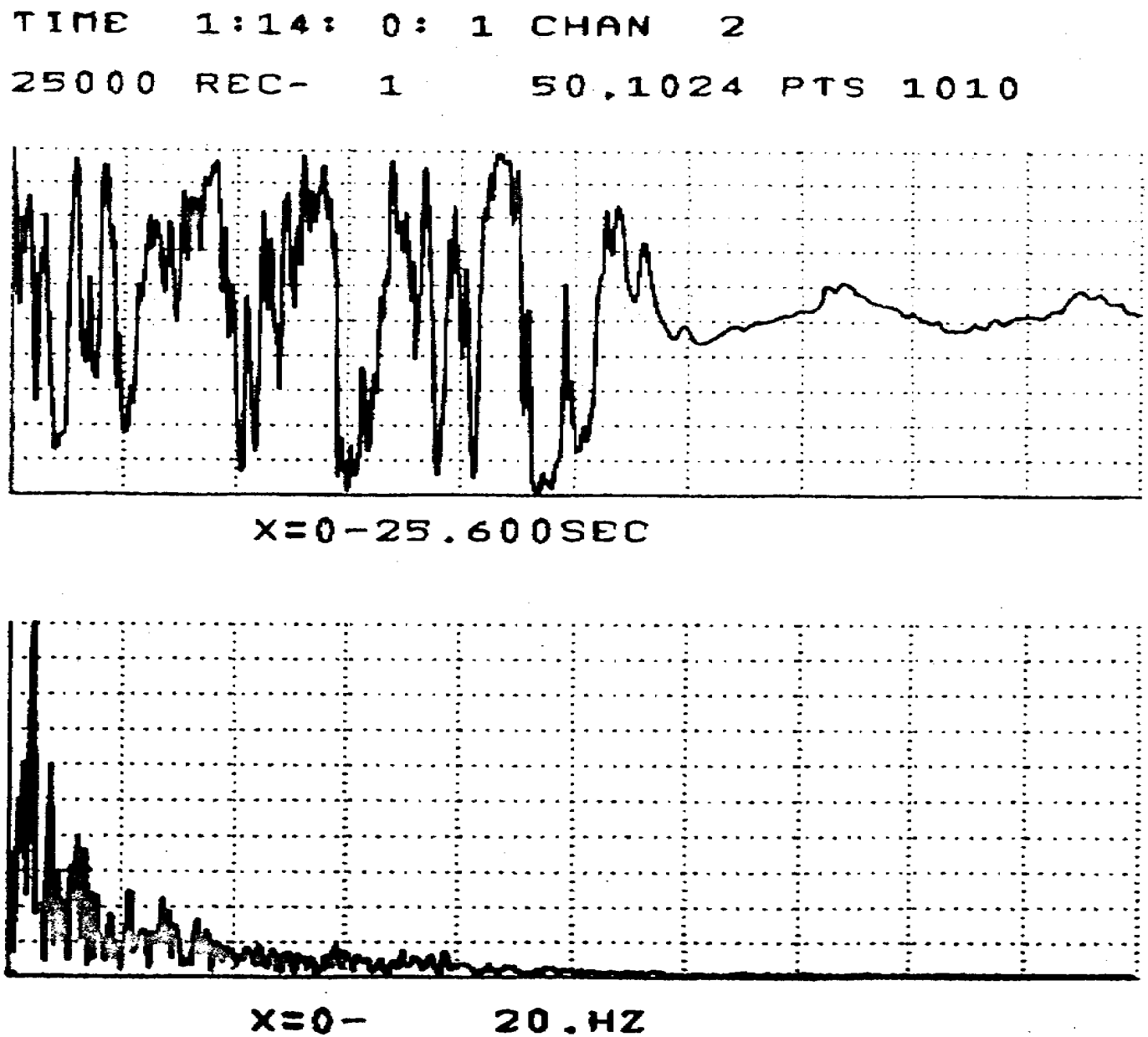


Figure A-14. Man Walking (From Right To Left, 45° To The Central Line And Back)

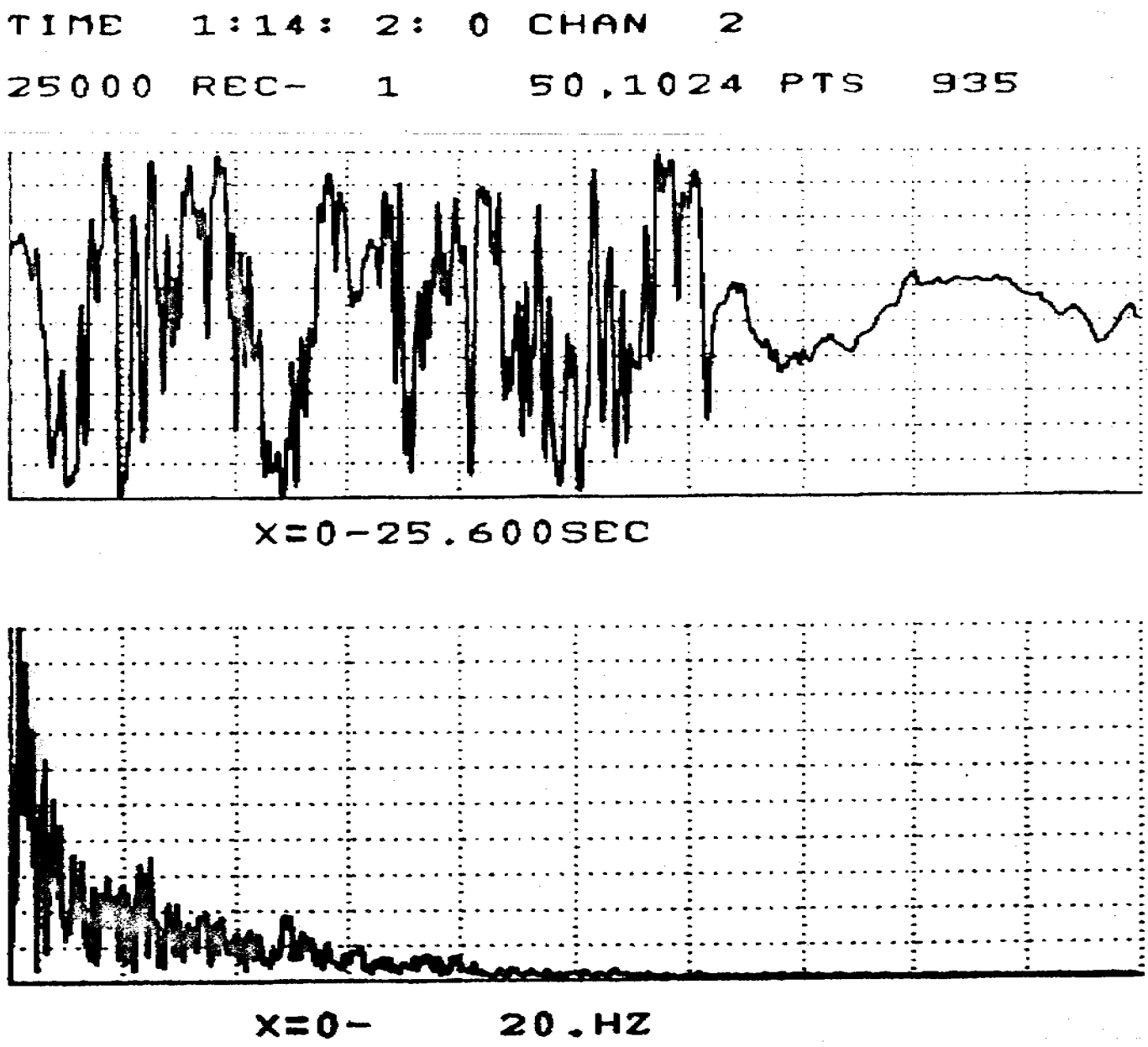


Figure A-15. Man Walking (From Right To Left, 45° To The Central Line And Back)

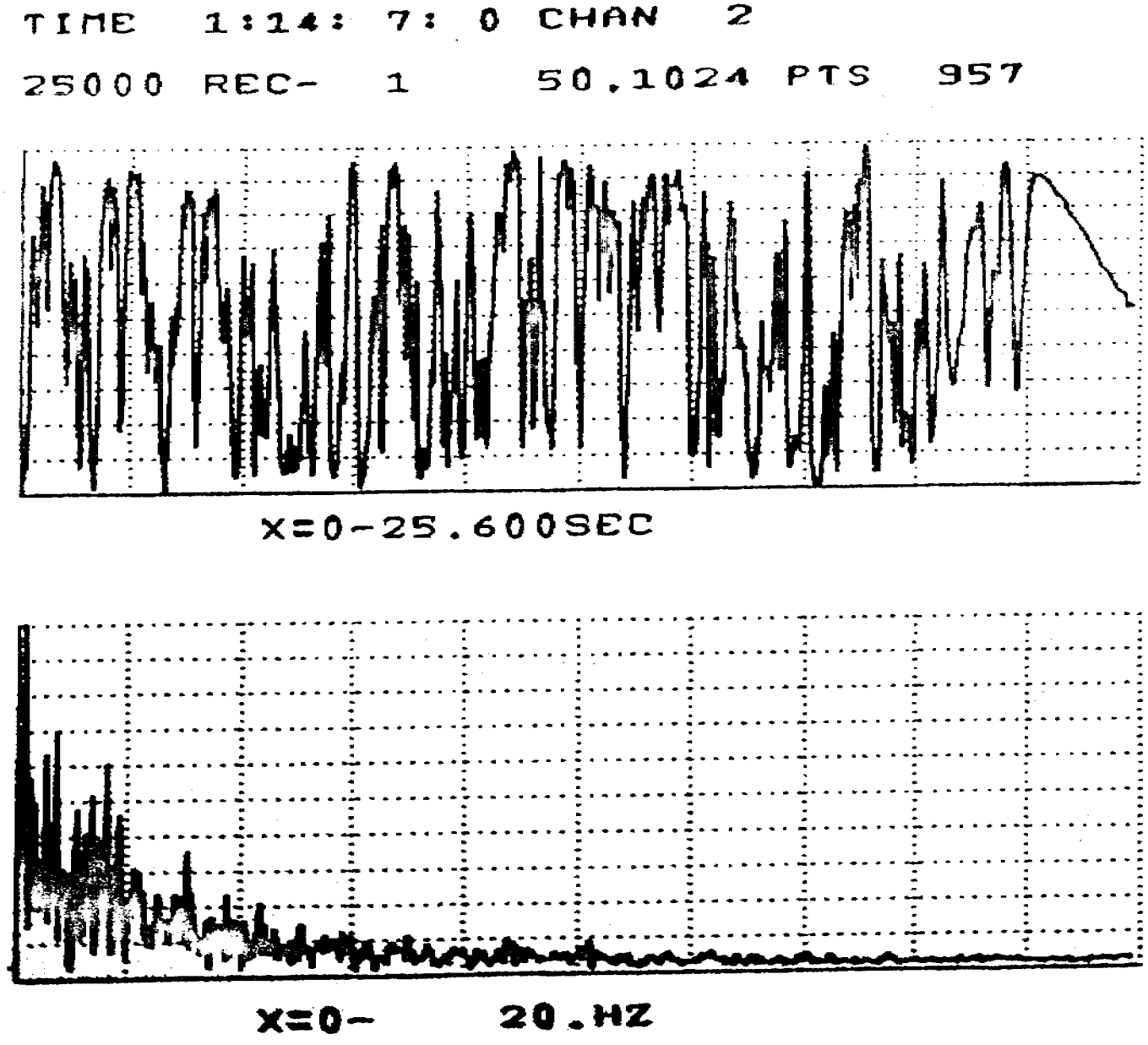


Figure A-16. Man Running (From Entrance Door To The Office Area And Back, See Figure 2, Path 6)

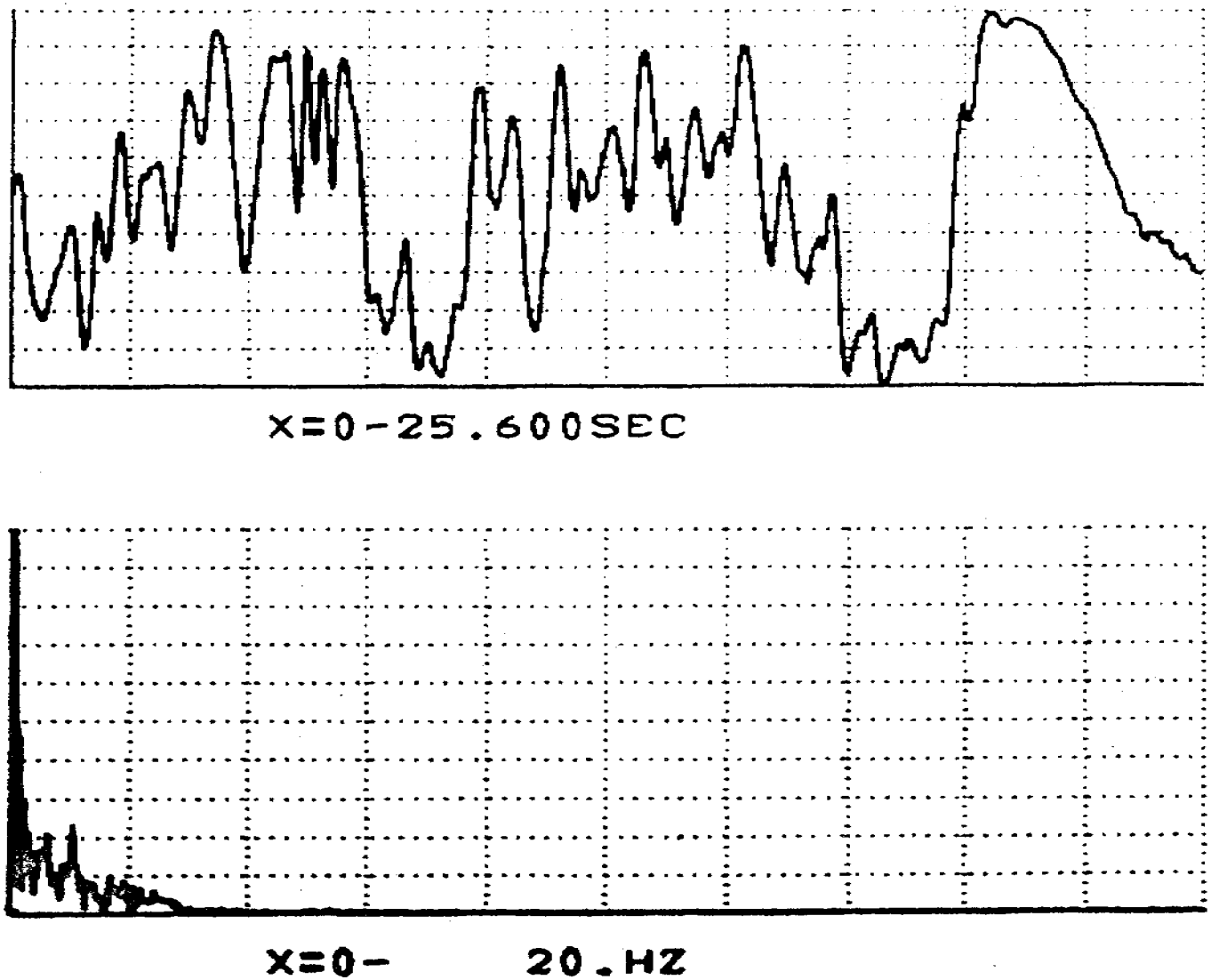


Figure A-17. Dolly With Four Wheels, Boxes Piled To Five Feet High, Pulled Across Floor By String

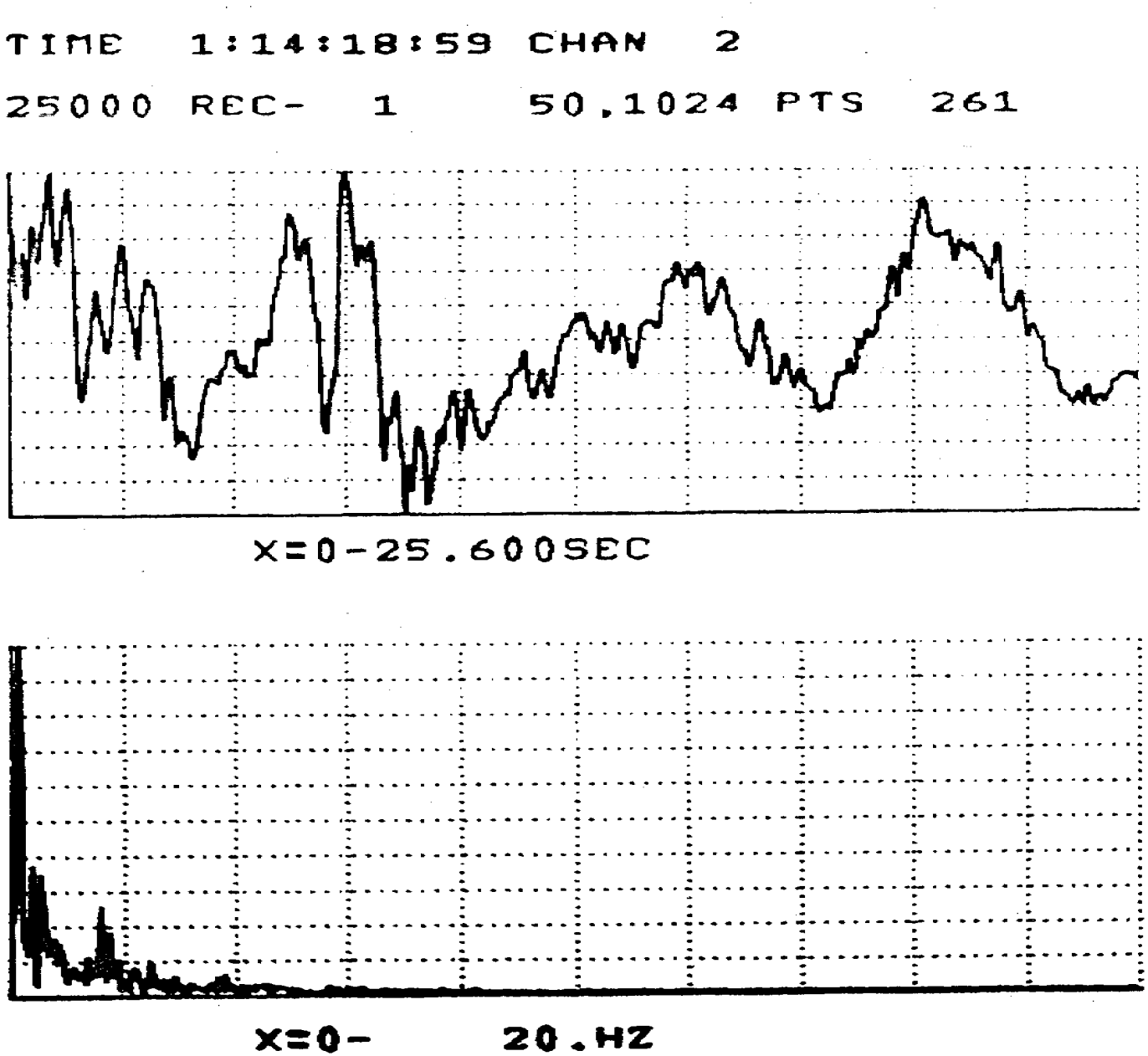


Figure A-18. Steel Ball 7½" Air Rolled Slowly.

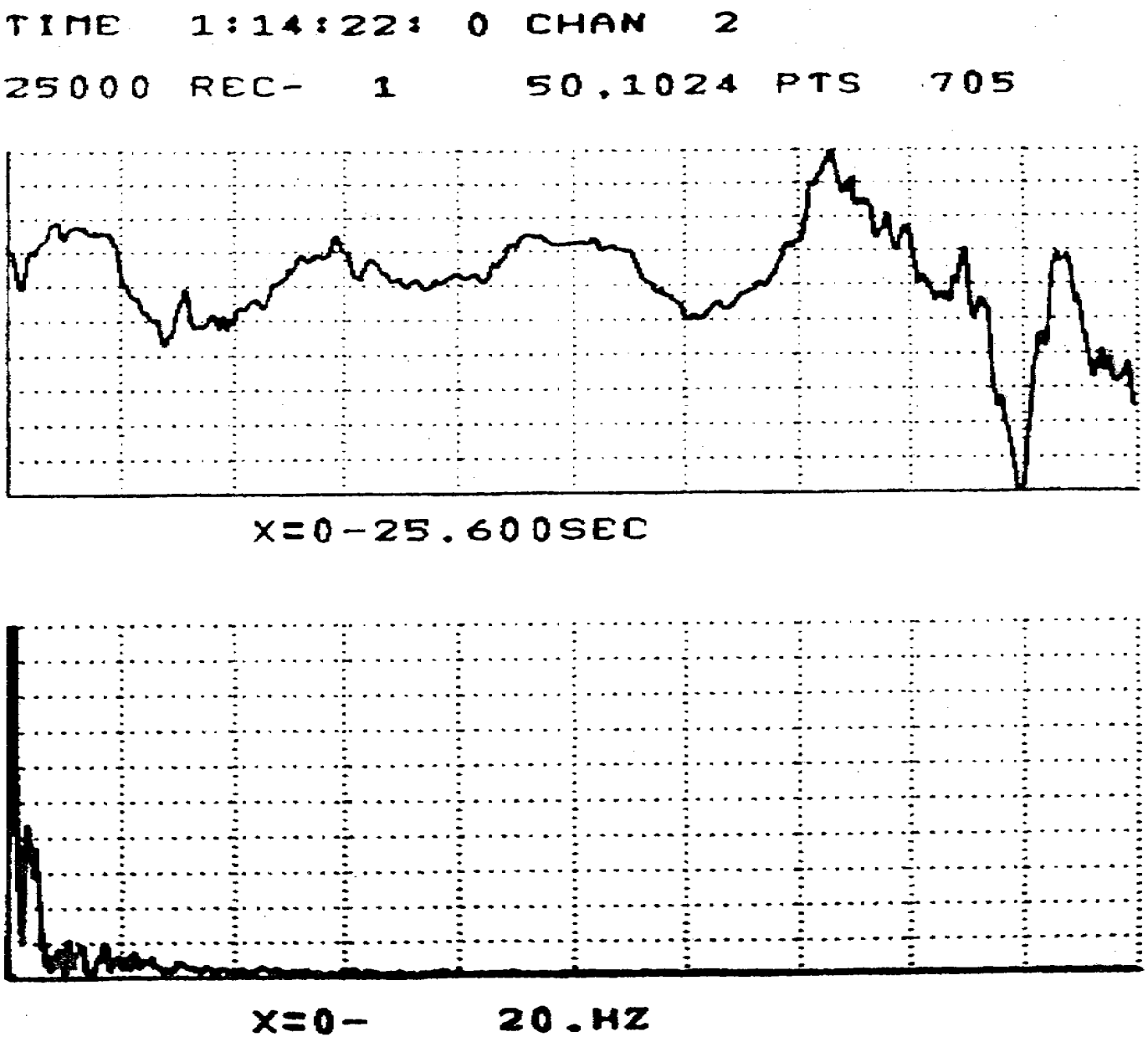


Figure A-19. Steel Ball 7¼ Air Rolled Slowly (Opposite To That Of Figure A-18)

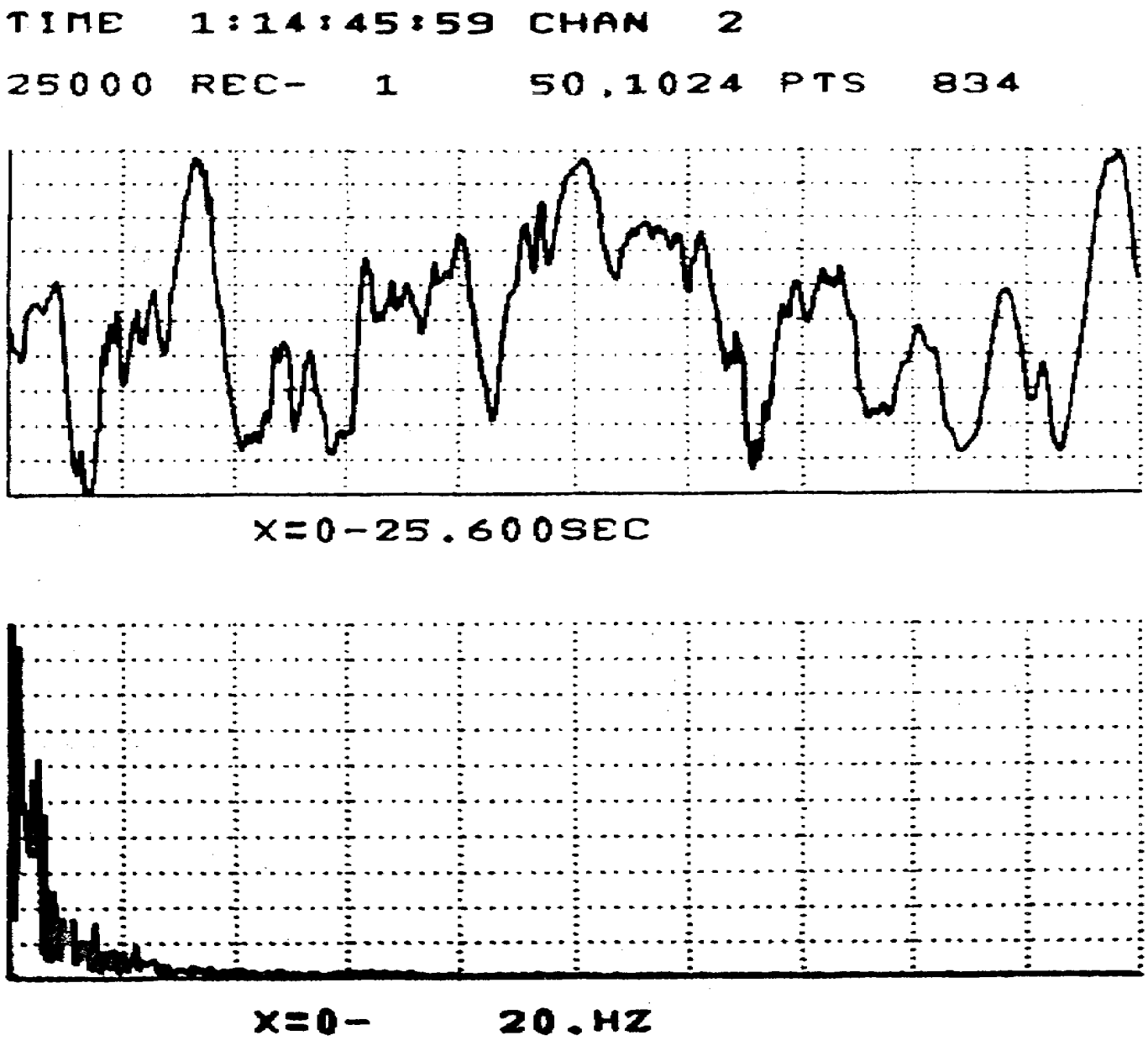


Figure 20. Man Walking Slowly To The Office Area Along The Center Line

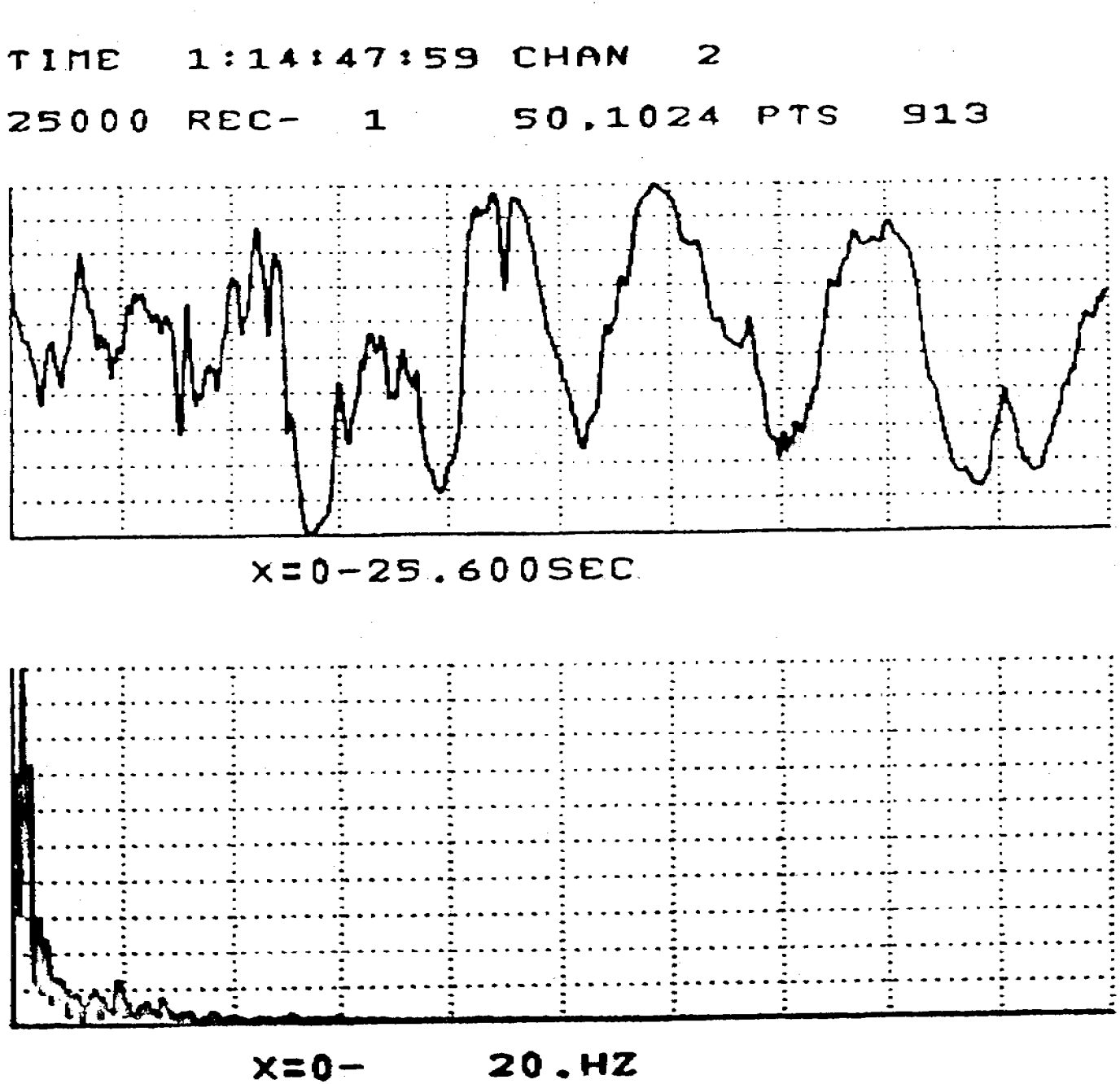


Figure A-21. Man Walking Slowly From Left To Right At The Center Of The Building (Perpendicular To That Of Figure 20)

NUMERICAL SOLUTION TO BEAM VIBRATIONS UNDER A MOVING COUPLE

Julian J. Wu
U.S. Army Armament Research and Development Command
Large Caliber Weapon Systems Laboratory
Benet Weapons Laboratory
Watervliet, NY 12189

ABSTRACT. The finite element solution formulation in time- and space-coordinates is extended to beam vibrations effected by a moving couple. This problem has direct application to gun motions analysis with an unbalanced moving projectile. The moving load, instead of being a time-dependent Dirac delta function as for the case of a moving concentered force, is now the derivative of this Dirac delta function. This singular function does not present any difficulty due to the variational process employed. This solution procedure is described together with results of beam motions subjected to a couple moving with various speeds.

1. INTRODUCTION. In a previous report [1], this writer presented a finite element-variational formulation which discretizes the spatial and time variable in the same manner. The method was applied to a problem of beam motion subjected to moving concentrated forces. Results were shown to be in excellent agreement with known solutions. this same formulation is now applied to the problem of a couple, i.e., a concentrated bending moment.

A recent investigation by S. H. Chu [2] on the interacting forces between a projectile and the cannon tube indicates that the couple produced by the eccentricity of the projectile as it moves down the tube may be of such a magnitude that its effect on the tube motion becomes significant. It is then important that the problem associated with moving moments can be analyzed adequately. The purpose of this note is to present the modification necessary to the previous formulations so that the solutions of a beam motion problem under a moving bending moment can be obtained routinely. Results of a cantilevered beam subject to such a load are also presented.

2. DIFFERENTIAL EQUATION AND NONDIMENSIONALIZATION. Consider a Euler-Bernoulli beam subjected to a moving couple M. The equation differential can be written as

$$EIy'''' + \rho A\ddot{y} = -M\delta'(x-\bar{x}) \qquad (1)$$

where $y(x,t)$ denotes the beam deflection as a function of spatial coordinate x and time t. E, I, A, ρ denote elastic modulus, second moment of inertia area and material density respectively. A dirac function is denoted by δ, $\bar{x} = \bar{x}(t)$ is the location of M, a prime ($'$) denotes differentiation with respect to x and a dot ($\dot{}$), differentiation with respect to t. Note that the right hand side of Eq. (1) has a dimension of force due to the fact that $\delta'(x-\bar{x}) = d/dx\ \delta(x-\bar{x})$ has a dimension of $(\text{length})^{-1}$.

Introducing nondimensional quantities

$$\hat{y} = y/\ell \ , \quad \hat{x} = x/\ell \ , \quad \hat{t} = t/T \ , \tag{2}$$

where ℓ is the length of the beam and T is a finite time, within $0 \leq t \leq T$, the problem is of interest, Eq. (1) can be written as

$$y'''' + \gamma^2 \ddot{y} = - Q\bar{\delta}'(x-\bar{x}) \tag{3}$$

The hats (^) have been omitted in Eq. (3) and

$$\gamma = \frac{c}{T}$$

$$Q = \frac{M\ell}{EI} \tag{4}$$

with

$$c^2 = \frac{\rho A \ell^4}{EI}$$

Boundary conditions associated with Eqs. (1) or (2) will now be introduced in conjunction of a variational problem. Consider

$$\delta I = 0 \tag{5a}$$

with

$$I = \int_0^1 \int_0^1 [y''y^{*''} - \gamma^2 \ddot{y}y^* + Q\bar{\delta}(x-\bar{x})y^*]dxdt$$

$$+ \int_0^1 dt\{k_1 y(0,t)y^*(0,t) + k_2 y'(0,t)y^{*'}(0,t)$$

$$+ k_3 y(1,t)y^*(1,t) + k_4 y'(1,t)y^{*'}(1,t)\}$$

$$+ \gamma^2 \int_0^1 dx\{k_5[y(x,0) - Y(x)]y^*(x,1)\} \tag{5b}$$

where $y^*(x,t)$ is the adjoint variable of $y(x,t)$. If one takes the first variation of I considering $y(x,t)$ to be fixed:

$$(\delta I)_{\delta y=0} = 0 \tag{5a'}$$

and consider δy^* to be completely arbitrary, it is easy to see that Eqs. (5) are equivalent to the differential Eq. (3) and the following boundary and initial conditions.

$$\left.\begin{aligned} &y'''(0,t) + k_1 y(0,t) = 0 \\ &y''(0,t) - k_2 y'(0,t) = 0 \\ &y'''(1,t) - k_3 y(1,t) = 0 \\ &y''(1,t) + k_4 y'(1,t) = 0 \end{aligned}\right\} \quad 0 \leq t \leq 1 \qquad (6a)$$

$$\dot{y}(x,0) = 0$$

and

$$\dot{y}(x,1) - k_5[y(x,0) - Y(x)] = 0 \qquad 0 \leq x \leq 1 \qquad (6b)$$

Taking appropriate values for k_1, k_2, k_3, and k_4, problems with a wide range of boundary conditions can be realized. The initial conditions in Eqs. (6b) are that the beam has zero initial velocity, and, if one takes k_5 to be ∞ (or larger number compared with unity),

$$y(x,0) = Y(x)$$

The meaning for cases where k_5 is not so need not be our concern here.

To derive the finite element matrix equations, one begins with Eq. (5a') and write

$$(\delta I)_{\delta y=0} = 0 \qquad (7a)$$

$$= \int_0^1 \int_0^1 [y''\delta y^{*''} - \gamma^2 \dot{y}\delta\dot{y}^* + Q\bar{\delta}'(x-\bar{x})\delta y^*]dxdt$$

$$+ \int_0^1 dt[k_1 y(0,t)\delta y^*(0,t) + k_2 y'(0,t)\delta y^{*'}(0,t)$$

$$+ k_3 y(1,t)\delta y^*(1,t) + k_4 y'(1,t)\delta y^{*'}(1,t)]$$

$$+ \gamma^2 \int_0^1 dx\{k_5[y(x,0) - Y(x)]\delta y^*(x,1)\} \qquad (7b)$$

Introducing element local variables

$$\begin{aligned} \xi &= \xi^{(i)} = Kx-i+1 \\ \eta &= \eta^{(i)} = Lt-j+1 \end{aligned} \tag{8a}$$

or

$$\begin{aligned} x &= \frac{1}{K}(\xi+i-1) \\ t &= \frac{1}{L}(\xi+j-1) \end{aligned} \tag{8b}$$

where K is the number of divisions in x and L, in t. (A typical grid scheme is shown in Figure 1). Equation (7b) can now be written as

$$\sum_{i=1}^{K}\sum_{j=1}^{L}\int_0^1\int_0^1\left[\frac{K^3}{L}\,y''(ij)\delta y^{*''}(ij)-\frac{\gamma^2 L}{K}\,y(ij)\delta y^{*}(ij)\right]d\xi d\eta$$

$$+\sum_{j=1}^{L}\int_0^1 d\eta\left[\frac{k_1}{L}\,y(ij)(0,\eta)\delta y^{*}(ij)(0,\eta)+k_2\frac{K^2}{L}\,y'(ij)(0,\eta)\delta y^{*'}(ij)(0,\eta)\right.$$

$$+\sum_{i=1}^{K}\int_0^1\frac{d\xi}{K}\left[\gamma^2 k_5\left(y_{(ij)}(\xi,0)\delta y^{*}_{(ij)}(\xi,1)\right)\right]$$

$$=-\sum_{i=1}^{K}\sum_{j=1}^{L}\frac{Q}{L}\int_0^1\int_0^1\bar{\delta}'(x-\bar{x})\delta y^{*}(ij)(\xi,\eta)d\xi d\eta$$

$$+\sum_{i=1}^{K}\frac{\gamma^2 k_5}{K}\int_0^1 d\xi\left[Y_{(i)}(\xi)\delta y^{*}_{(iL)}(\xi,1)\right] \tag{9}$$

The shape function vector is now introduced. Let

$$y(ij)(\xi,\eta)=\underset{\sim}{a}^{T}(\xi,\eta)Y(ij)$$

$$y^{*}(ij)(\xi,\eta)=\underset{\sim}{a}^{T}(\xi,\eta)\underset{\sim}{Y}^{*}(ij)=\underset{\sim}{Y}^{*T}(ij)\underset{\sim}{a}(\xi,\eta) \tag{10}$$

Equation (9) then becomes

$$\sum_{i=1}^{K}\sum_{j=1}^{L} \delta Y^{*T}(ij) \left\{\frac{K^3}{L} \underset{\sim}{A} - \frac{\gamma^2 L}{K} \underset{\sim}{B}\right\} \underset{\sim}{Y}(ij)$$

$$+ \sum_{i=1}^{L} \delta Y^{*T}(ij) \left\{\frac{k_1}{L} \underset{\sim}{B}_1 + \frac{k_2 K^2}{L} \underset{\sim}{B}_2\right\} \underset{\sim}{Y}(ij)$$

$$+ \sum_{i=1}^{L} \delta Y^{*T}(Kj) \left\{\frac{k_3}{L} \underset{\sim}{B}_3 + \frac{k_4 K^2}{L} \underset{\sim}{B}_4\right\} \underset{\sim}{Y}(ij)$$

$$+ \sum_{i=1}^{K} \delta Y^{*T}(iL) \left\{\frac{\gamma^2 k_5}{K} \underset{\sim}{B}_5\right\} \underset{\sim}{Y}(iL)$$

$$= \sum_{i=1}^{K}\sum_{j=1}^{L} \delta Y^{*T}(ij) \frac{Q}{L} F(ij) + \sum_{i=1}^{K} \delta Y^{*T}(iL) \frac{\gamma^2 k_5}{K} \underset{\sim}{G}(i) \qquad (11)$$

where, as it can be seen easily, that

$$\underset{\sim}{A} = \int_0^1 \int_0^1 \underset{\sim}{a}_{,\xi\xi}\, \underset{\sim}{a}^T_{,\xi\xi}\, d\xi d\eta$$

$$\underset{\sim}{B} = \int_0^1 \int_0^1 \underset{\sim}{a}_{,\eta}\, \underset{\sim}{a}^T_{,\eta}\, d\xi d\eta$$

$$\underset{\sim}{B}_1 = \int_0^1 \underset{\sim}{a}(0,\eta)\underset{\sim}{a}^T(0,\eta)d\eta \quad , \quad \underset{\sim}{B}_2 = \int_0^1 \underset{\sim}{a}_{,\xi}(0,\eta)\underset{\sim}{a}^T_{,\xi}(0,\eta)d\eta \qquad (12)$$

$$\underset{\sim}{B}_3 = \int_0^1 \underset{\sim}{a}(1,\eta)\underset{\sim}{a}^T(1,\eta)d\eta \quad , \quad \underset{\sim}{B}_4 = \int_0^1 \underset{\sim}{a}_{,\xi}(1,\eta)\underset{\sim}{a}^T_{,\xi}(1,\eta)d\eta$$

$$\underset{\sim}{B}_5 = \int_0^1 \underset{\sim}{a}(\xi,1)\underset{\sim}{a}^T(\xi,0)d\xi$$

and

$$\underset{\sim}{F}(ij) = \int_0^1 \int_0^1 a(\xi,\eta)\bar{\delta}'(ij)(\xi-\bar{\xi})d\xi d\eta \quad , \quad \underset{\sim}{G}(i) = \int_0^1 \underset{\sim}{a}(\xi,1)Y(i)(\xi)d\xi$$

where

$$\bar{\delta}'_{(ij)}(\xi-\bar{\xi}) = \frac{d}{dx}\,\bar{\delta}_{(ij)}(\xi-\bar{\xi})$$

is the local version of the function $\bar{\delta}(x-\bar{x})$ appeared in Eq. (9). The specific form of $\delta(y)(\xi-\xi)$ will be described later in a paragraph prior to Eqs. (18).

Now Eq. (11) can be assembled in a global matrix equation

$$\underset{\sim}{\delta Y^*}^T \underset{\sim}{K}\ \underset{\sim}{Y} = \underset{\sim}{\delta Y^*}\ \underset{\sim}{F} \tag{13}$$

By virtue of the fact that δY^* is not subjected to any constrained conditions, one has

$$\underset{\sim}{K}\ \underset{\sim}{Y} = \underset{\sim}{F} \tag{14}$$

which can be solved routinely. Numerical results of several problems in this class will be presented in a later section.

3. FORCE VECTOR DUE TO A MOVING COUPLE. We shall describe here the procedures involved to arrive at the force vector contributed by a moving couple. This force vector has appeared in Eq. (12) as

$$\underset{\sim}{F}_{(ij)} = \int_0^1 \int_0^1 \underset{\sim}{a}(\xi,\eta)\bar{\delta}'_{(ij)}(\xi-\bar{\xi})d\xi d\eta \tag{15a}$$

Perform integration-by-parts once. Equation (15a) can be written as

$$\underset{\sim}{F}_{(ij)} = -\int_0^1 \int_0^1 \underset{\sim}{a}_{,\xi}(\xi,\eta)\bar{\delta}_{(ij)}(\xi-\bar{\xi})d\xi d\eta \tag{15b}$$

The shape function $\underset{\sim}{a}(\xi,\eta)$ is a vector of 16 in dimension. In the present formulation we have chosen the form:

$$a_k(\xi,\eta) = \bar{b}_i(\xi)\bar{b}_j(\eta) \quad , \qquad \begin{matrix} k = 1,2,3,\ldots 16 \\ i,j = 1,2,3,4 \end{matrix} \tag{16a}$$

and

$$a_{k,\xi}(\xi,\eta) = \bar{b}_i'(\xi)\bar{b}_j(\xi) \tag{16b}$$

The relations between k and i,j are given in Table I. These are the consequences of the choice of the shape function such that $Y_{(ij)}$, the generalized coordinates of the (ij)th element, represent the displacement, slope, velocity, and angular velocity at the local nodal points. Thus

$$\bar{b}_i(\xi) = \sum_{p=1}^{4} \bar{b}_{ip}\xi^{p-1} \quad ; \quad \bar{b}_i'(\xi) = \sum_{p=1}^{4} \bar{b}'_{ip}\xi^{p-1} \tag{17}$$

The values of $\bar{b}_{ip}$ are given in Tables II and III.

TABLE I. RELATIONSHIP BETWEEN (i,j) AND k IN EQUATION (16)

k	(i,j)	k	(i,j)
1	(1,1)	9	(1,3)
2	(2,1)	10	(2,3)
3	(1,2)	11	(1,4)
4	(2,2)	12	(2,4)
5	(3,1)	13	(3,3)
6	(4,1)	14	(4,3)
7	(3,2)	15	(3,4)
8	(4,2)	16	(4,4)

TABLE II. VALUES OF $\bar{b}_{ip}$ IN EQUATION (17)

i \ p	1	2	3	4
1	1	0	-3	1
2	0	1	-2	1
3	0	0	3	-2
4	0	0	-1	1

TABLE III. VALUES OF $\bar{b}'_{ip}$ IN EQUATION (17)

i \ p	1	2	3	4
1	0	-6	6	0
2	1	-4	3	0
3	0	6	-6	0
4	0	-2	3	0

Now, let us consider $\bar{\delta}_{(ij)}(\bar{\xi}-\bar{\xi})$. This "function" represents the effect of the Dirac delta function $\bar{\delta}(\bar{x}-\bar{x})$ on the (ij)th element. If the curve of travel $\bar{x} = \bar{x}(t)$ does not go through the element (i,j), $\bar{\delta}_{(ij)}(\bar{\xi}-\bar{\xi}) = 0$. If it passes through that element, one has

$$\bar{\delta}_{(ij)}(\bar{\xi}-\bar{\xi}) = \bar{\delta}(\bar{x}-\bar{x}) = K\bar{\delta}(\bar{\xi}-\bar{\xi}) \tag{18a}$$

with

$$\bar{\xi} = \bar{\xi}(\eta) \tag{18b}$$

The function $\bar{\xi}(\eta)$ is derived from $\bar{x} = \bar{x}(t)$. For example, if the force moves with a constant velocity, one has

$$\bar{x} = \bar{x}(t) = vt \tag{19a}$$

it follows from Eqs. (8) that

$$\bar{\xi} = \bar{\xi}(\eta) = -i+1 + \frac{vK}{L}(\eta+j-1) \tag{19b}$$

With Eqs. (16), (17), (18), and (19), one writes (15) as

$$F_{(ij)k} = K\int_0^1 \int_0^1 a_{k,\xi}(\xi,\eta)\bar{\delta}(\bar{\xi}-\bar{\xi})d\xi d\eta \tag{20a}$$

$$F_{(ij)k} = K\int_0^1 \int_0^1 \bar{b}'_{p}\bar{b}_{jq}\,\xi^{p-1}\eta^{q-1}\bar{\delta}(\bar{\xi}-\bar{\xi})d\xi d\eta \tag{20b}$$

Equation (20) can then be evaluated easily once the exact form of $\bar{\xi}$ is written. For example, if $\bar{\xi} = \eta$, Eq. (20) reduces to

$$F_{(ij)k} = \sum_{p=1}^{4}\sum_{q=1}^{4} k\,\bar{b}'_{ip}\,\bar{b}_{iq}\int_0^1 \xi^{p+q-2}d\xi$$

$$= \sum_{p=1}^{4}\sum_{q=1}^{4} \frac{k\bar{b}'_{ip}\bar{b}_{jq}}{p+q-1} \tag{21}$$

TABLE IV. DEFLECTION $y(x,t)/\ell$ OF A CANTILEVERED BEAM UNDER A MOVING CONCENTRATED MOMENT ($T = 10^{10}$ sec.)

t/T \ x/ℓ	0	0.25	0.50	0.75	1.00
0.	0.	0.	0.	0.	0.
0.25	0.	.03125	.09375	0.15625	0.21875
0.50	0.	.03125	.12500	0.25000	0.37500
0.75	0.	.03125	.12500	0.28125	0.46875
1.00	0.	.03125	.12500	0.28125	0.50000

TABLE V. DEFLECTION $y'(x,t)/\ell$ OF A CANTILEVERED BEAM UNDER A MOVING CONCENTRATED MOMENT ($T = 10^{10}$ sec.)

t/T \ x/ℓ	0	0.25	0.50	0.75	1.00
0.	0.0000	0.0000	0.0000	0.0000	0.0000
0.25	(0.0072)	0.2366	0.2481	0.2496	0.2500
0.50	(0.0021)	0.2481	0.4856	0.4981	0.5000
0.75	0.0005	0.2496	0.4981	0.7366	0.7500
1.00	0.0000	0.2505	0.5021	0.7572	1.0000

TABLE VI. DEFLECTION $y(x,t)/\ell$ OF A CANTILEVERED BEAM UNDER A MOVING CONCENTRATED MOMENT

(T = 10.0 sec.)

x/ℓ t/T	0.	0.125	0.250	0.375	0.500	0.625	0.750	0.875	1.000
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
0.25	0.0000	0.0051	0.0313	0.0564	0.0891	0.1134	0.1453	0.1700	0.2021
0.50	0.0000	-0.0023	0.0333	0.0484	0.1126	0.1488	0.2711	0.2560	0.3256
0.75	0.0000	-0.0319	0.0462	-0.0054	0.0966	0.0686	0.1920	0.1983	0.3320
1.00	0.0000	-0.4458	0.3255	-0.6155	0.1702	-0.7785	-0.0488	-0.8189	-0.0162

TABLE VII. DEFLECTION $y(x,t)/\ell$ OF A CANTILEVERED BEAM UNDER A MOVING CONCENTRATED MOMENT

(T = 1.0 sec.)

x/ℓ t/T	0.	0.125	0.250	0.375	0.500	0.625	0.750	0.875	1.000
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
0.25	0.0000	0.0028	0.0131	0.0238	0.0283	0.0277	0.0233	0.0168	0.0095
0.50	0.0000	0.0029	0.0148	0.0370	0.0709	0.1075	0.1396	0.1688	0.1969
0.75	0.0000	0.0052	0.0304	0.0776	0.1473	0.2307	0.3263	0.4253	0.5223
1.00	0.0000	0.0289	-0.0109	0.0734	0.2351	0.3871	0.5186	0.6218	0.7262

TABLE VIII. DEFLECTION $y(x,t)/\ell$ OF A CANTILEVERED BEAM UNDER A MOVING CONCENTRATED MOMENT

(T = 10.0 sec.)

$y(x,t)/\ell$ [x 10^{-1}]

x/ℓ t/T	0.	0.125	0.250	0.375	0.500	0.625	0.750	0.875	1.000
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
0.25	0.0000	0.0051	0.0445	0.0354	-0.0089	-0.0103	0.0037	0.0032	0.0058
0.50	0.0000	-0.0056	0.0260	0.0437	0.0970	0.0760	-0.0291	-0.0388	0.0558
0.75	0.0000	-0.0115	0.0564	-0.0019	0.0296	0.1196	0.1806	0.0559	-0.2438
1.00	0.0000	-0.1861	0.2165	-0.1210	0.2015	0.0879	0.0013	-0.0888	0.3299

4. NUMERICAL DEMONSTRATIONS. Some numerical results obtained will now be presented. Let us consider a cantilevered beam subjected to a unit moving couple with a constant velocity

$$v = \frac{\ell}{T}$$

As T varies from ∞ to 0, the velocity varies from 0 to ∞.

It will be helpful to compare v with some reference velocity which is a characteristic of the given beam. It is known that for a cantilevered beam, the first mode of vibration has a frequency (see, for example, [3])

$$f_1 = \frac{\omega}{2\pi} = \frac{1}{2\pi}\left(\frac{1.875^2}{c}\right) = \frac{0.560}{c} \quad \text{(cycles per seconds)}$$

and the period,

$$T_1 = 1.786\ c$$

where

$$c^2 = \frac{\rho A \ell^4}{EI}$$

Consider the vibration as standing waves. They travel at a speed

$$v_1 = 2\ell f_1 = \frac{1.12\ell}{c}$$

Hence, the relative velocity

$$\bar{v} = \frac{v}{v_1} = \frac{T_1}{2T} = 0.893\ \frac{c}{T}$$

We shall take c = 1.0 for the moving force problems. Thus, f_1 = 0.560 Hz. T_1 = 1.786 sec. and

$$\bar{v} = 0.893/T$$

Using grid schemes of 4x4 (i.e., four segments in spatial and four in time coordinates) and 8x4. Tables IV through VIII show the beam deflections (and slopes) as the concentrated moment Q = 1.0 moves from the left to the right end. Since we have defined T as the time required for the load to travel from one end to another, t = 0.5T, for example, indicates the point in time when the load is at the midspan of the beam.

In Tables IV and V, T is set to 10^{10} sec. which is extremely large compared with the beam characteristic time of T_1 = 1.786 sec. The solution should reduce to the static problem. This is certainly the case as shown in these two tables. These results are obtained using a grid scheme of 4x4.

For results shown in Tables VI through VIII an 8x4 grid scheme has been used. The beam deflections for T = 10, 1.0, and 0.1 seconds are shown in Table VI, VII, and VIII respectively.

Finally, these deflection curves are also plotted in Figures 2 through 10. From these figures and the tabulated results, one observes that while some of the results are extremely good, others are changing so rapidly with respect to time or space variable that an assessment on their accuracy is very difficult. Hence, further investigations on numerical convergence of these data is necessary.

REFERENCES

1. J. J. Wu, "Beam Motions Under Moving Loads Solved By Finite Element Method Consistent in Spatial and Time Coordinates," Technical Report No. ARLCB-TR-80046, U.S. Army Armament Research & Development Command, Large Caliber Weapon Systems Laboratory, Benet Weapons Laboratory, Watervliet, NY, November 1980.

2. S. H. Chu, "In-Bore Motion Analysis of the 155 MM XM712 Projectile When Fired in the M198 Howitzer," Proceedings of the Army Symposium on Solid Mechanics, AMMRC MS 80-4, Army Materials and Mechanics Research Center, Watertown, MA, pp. 270-288, 1980.

3. K. N. Tong, Theory of Mechanical Vibration, John Wiley, New York, 1960, p. 257; p. 256.

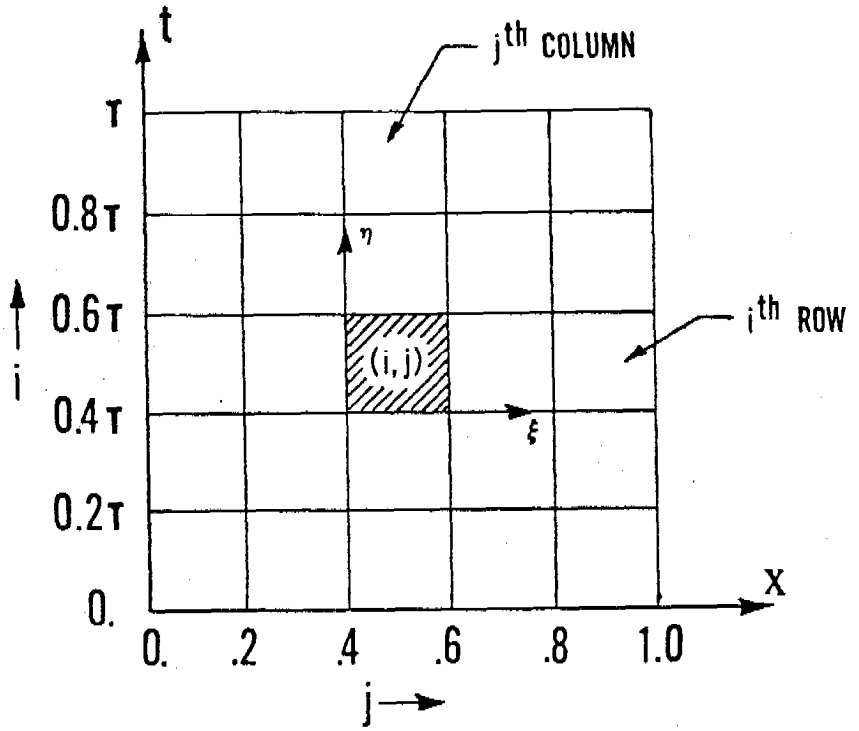


Figure 1. A Typical Finite Element Grid Scheme Showing the $(i,j)^{th}$ Element and the Global, Local Coordinates.

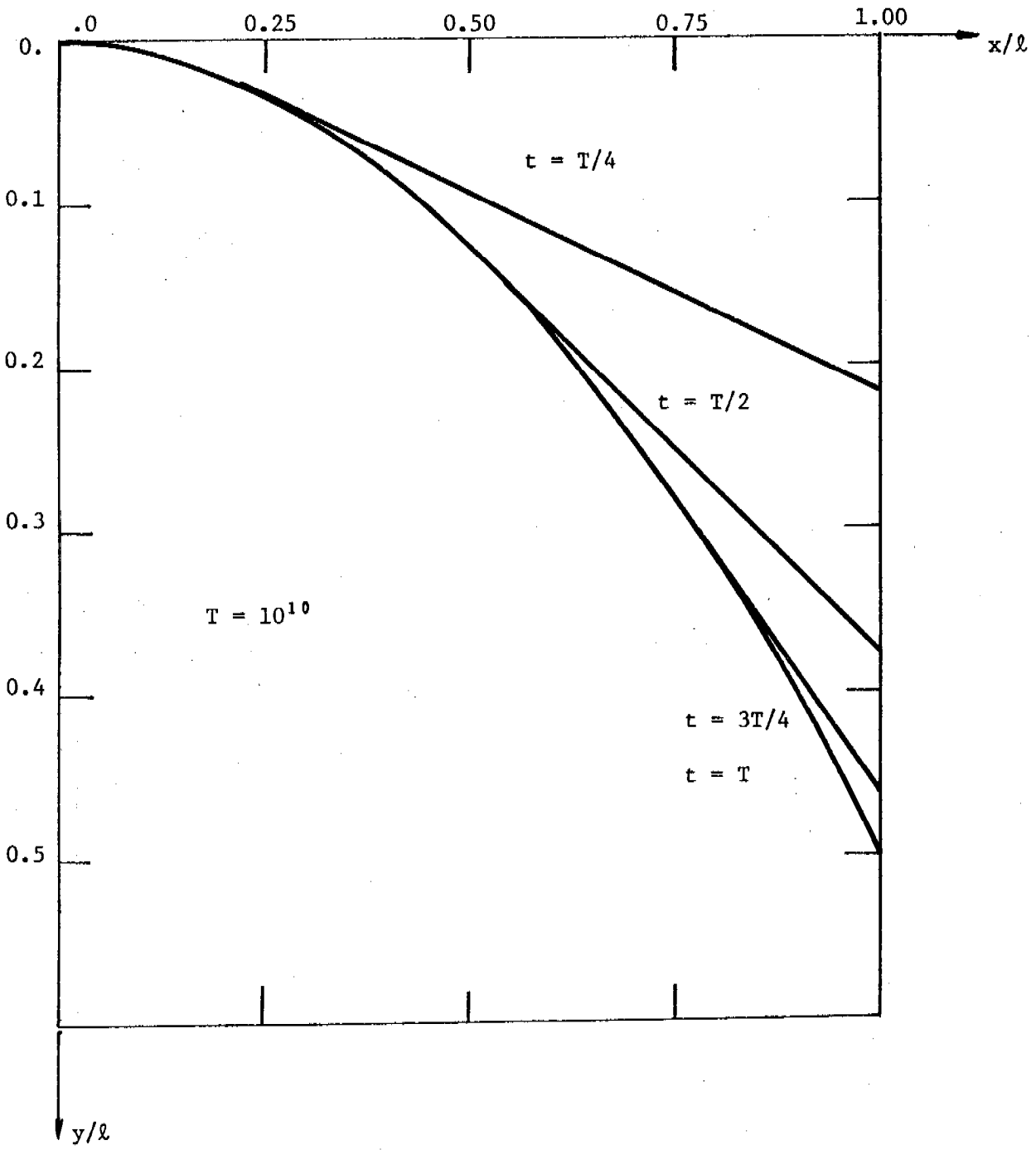


Figure 2. Deflection(s) of a Cantilevered Beam Under a Moving Couple.

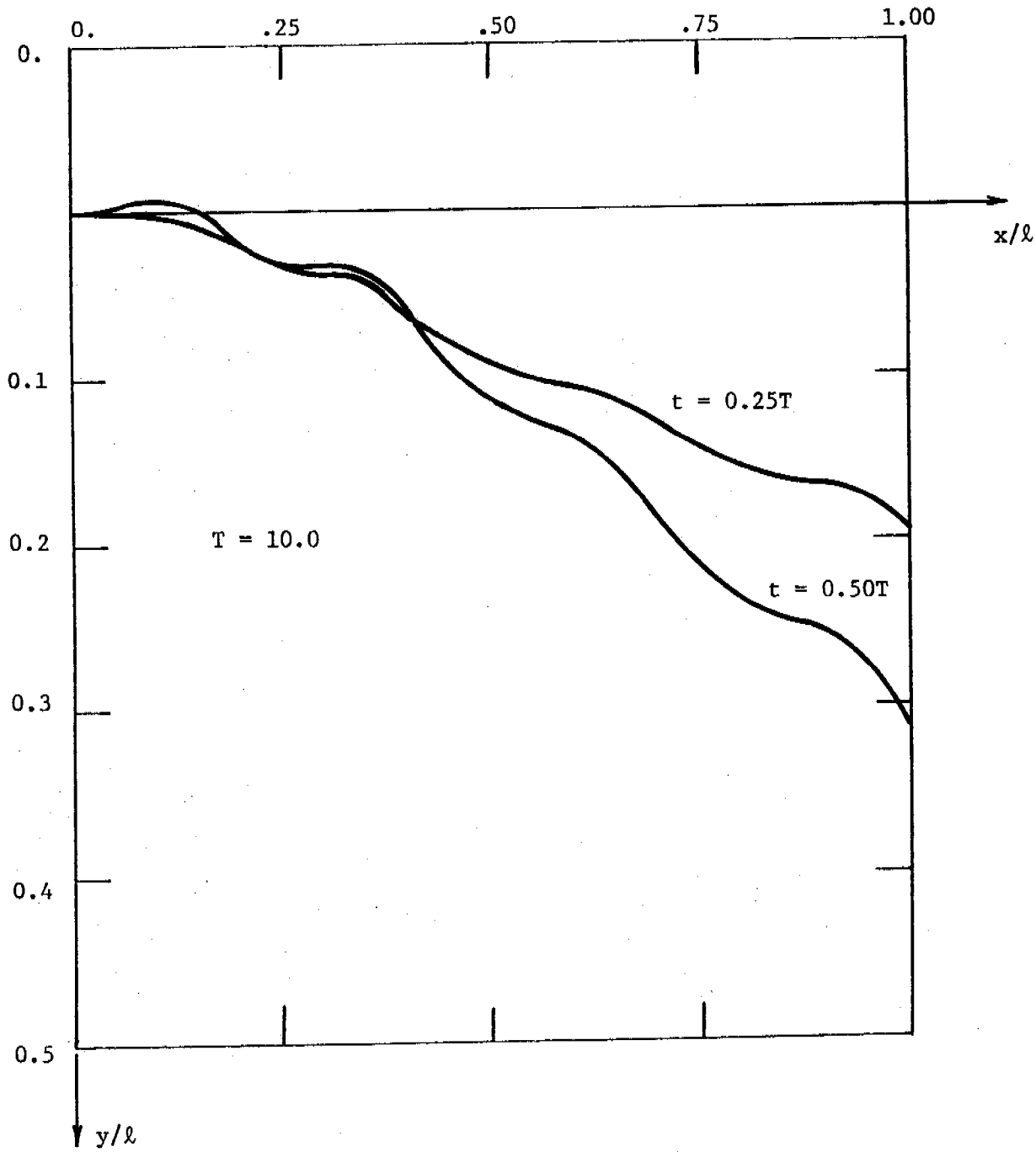


Figure 3. Deflection(s) of a Cantilevered Beam Under a Moving Couple.

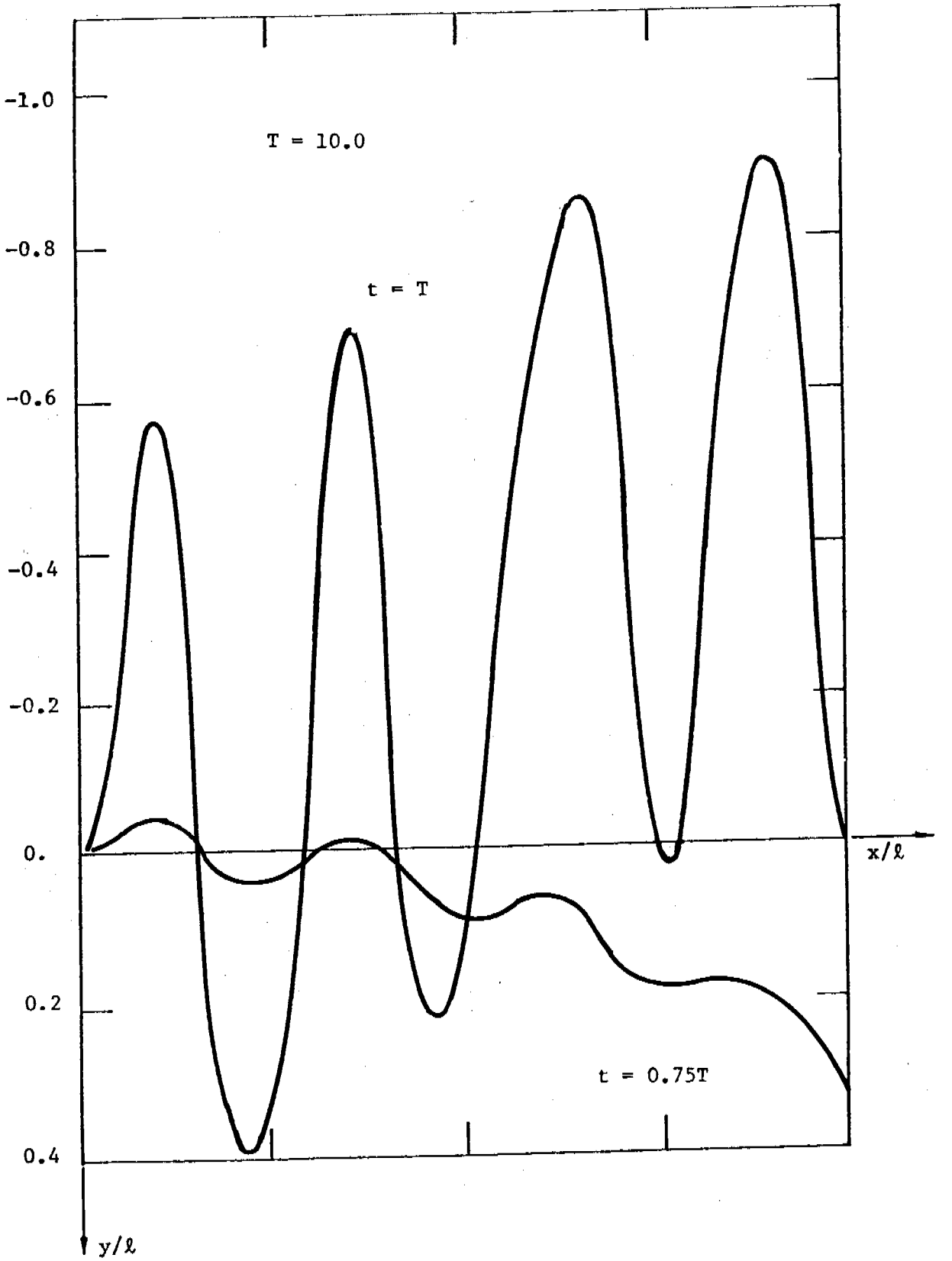


Figure 4. Deflection(s) of a Cantilevered Beam Under a Moving Couple.

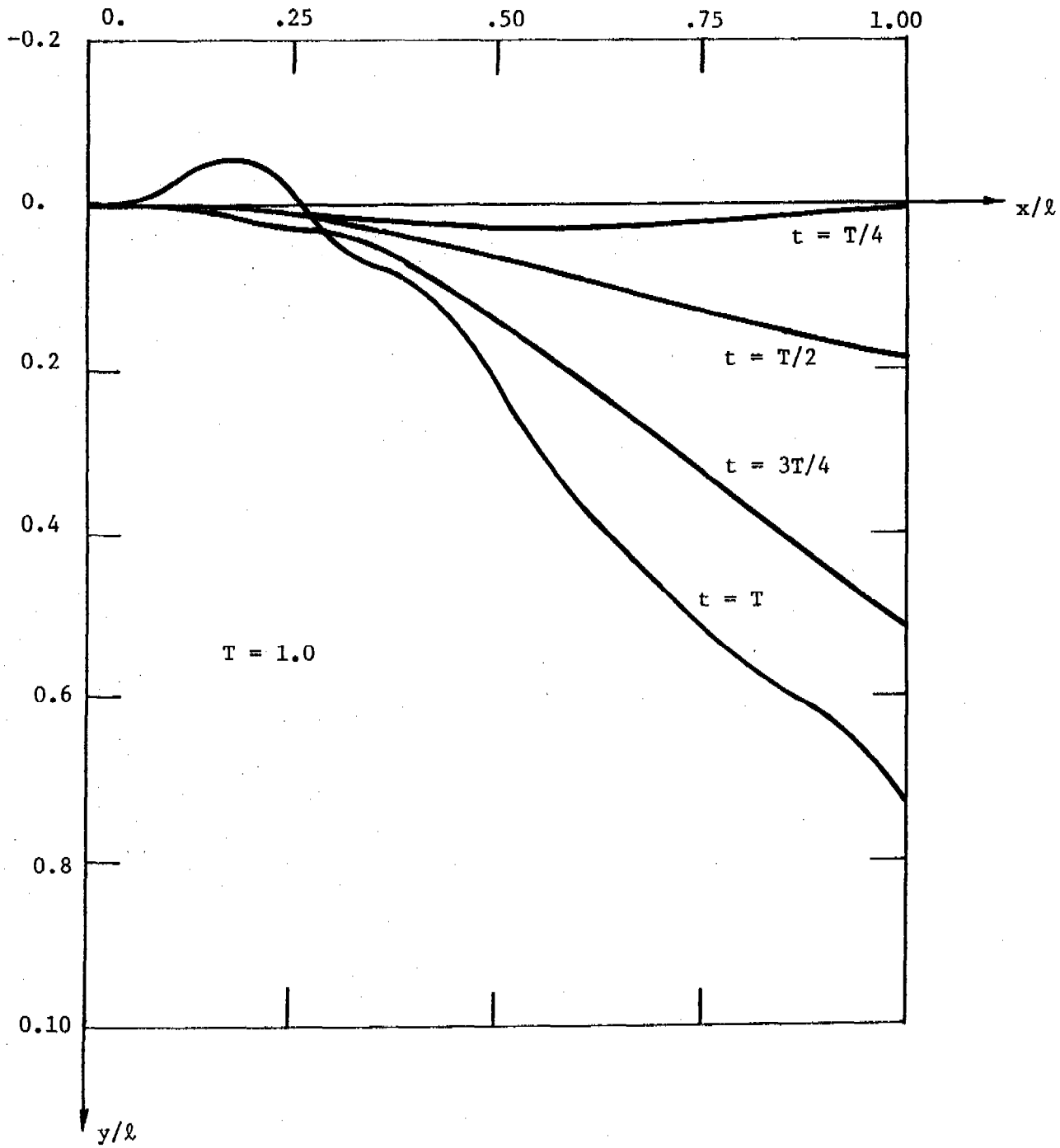


Figure 5. Deflection(s) of a Cantilevered Beam Under a Moving Couple.

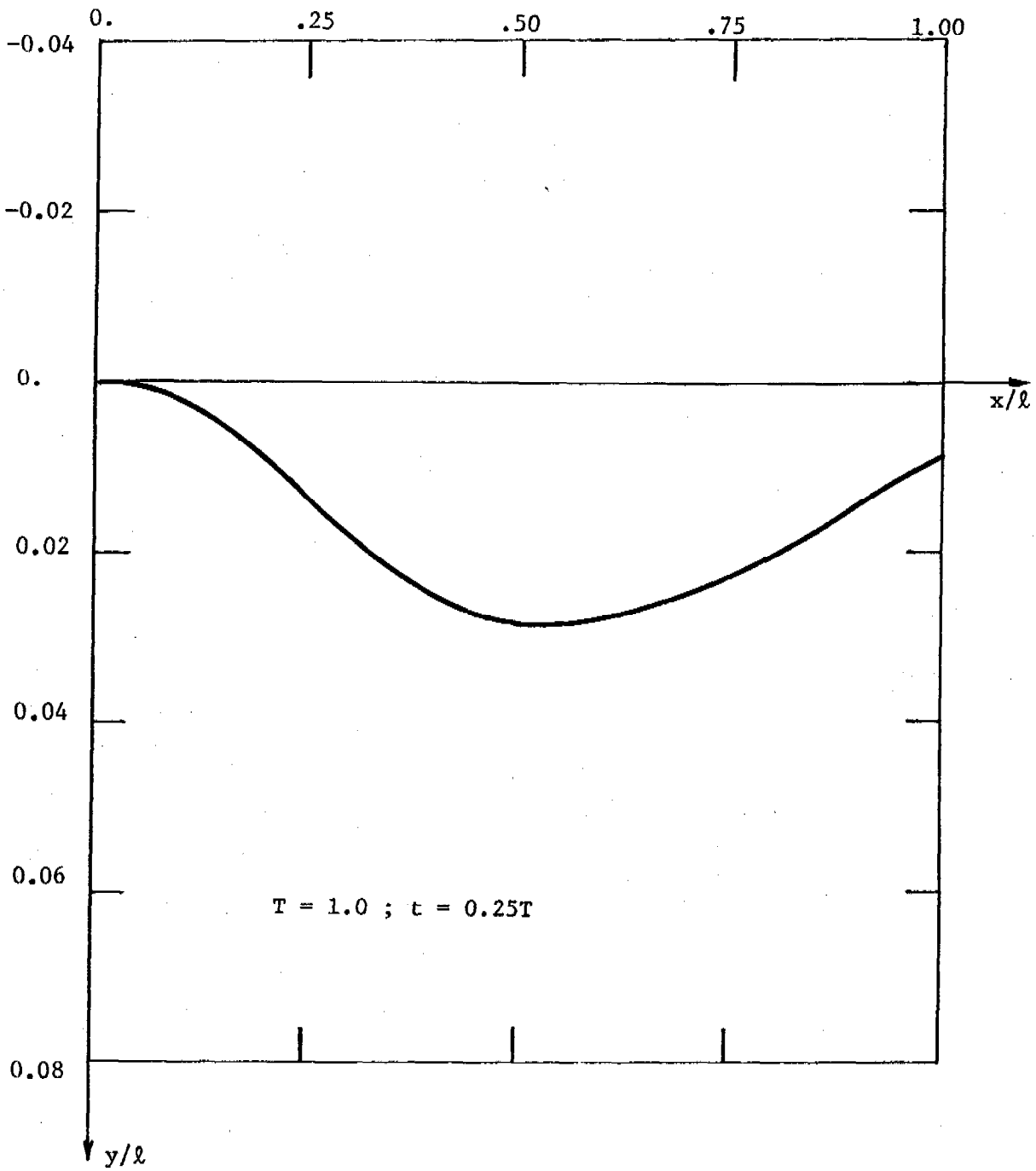


Figure 6. Deflection(s) of a Cantilevered Beam Under a Moving Couple.

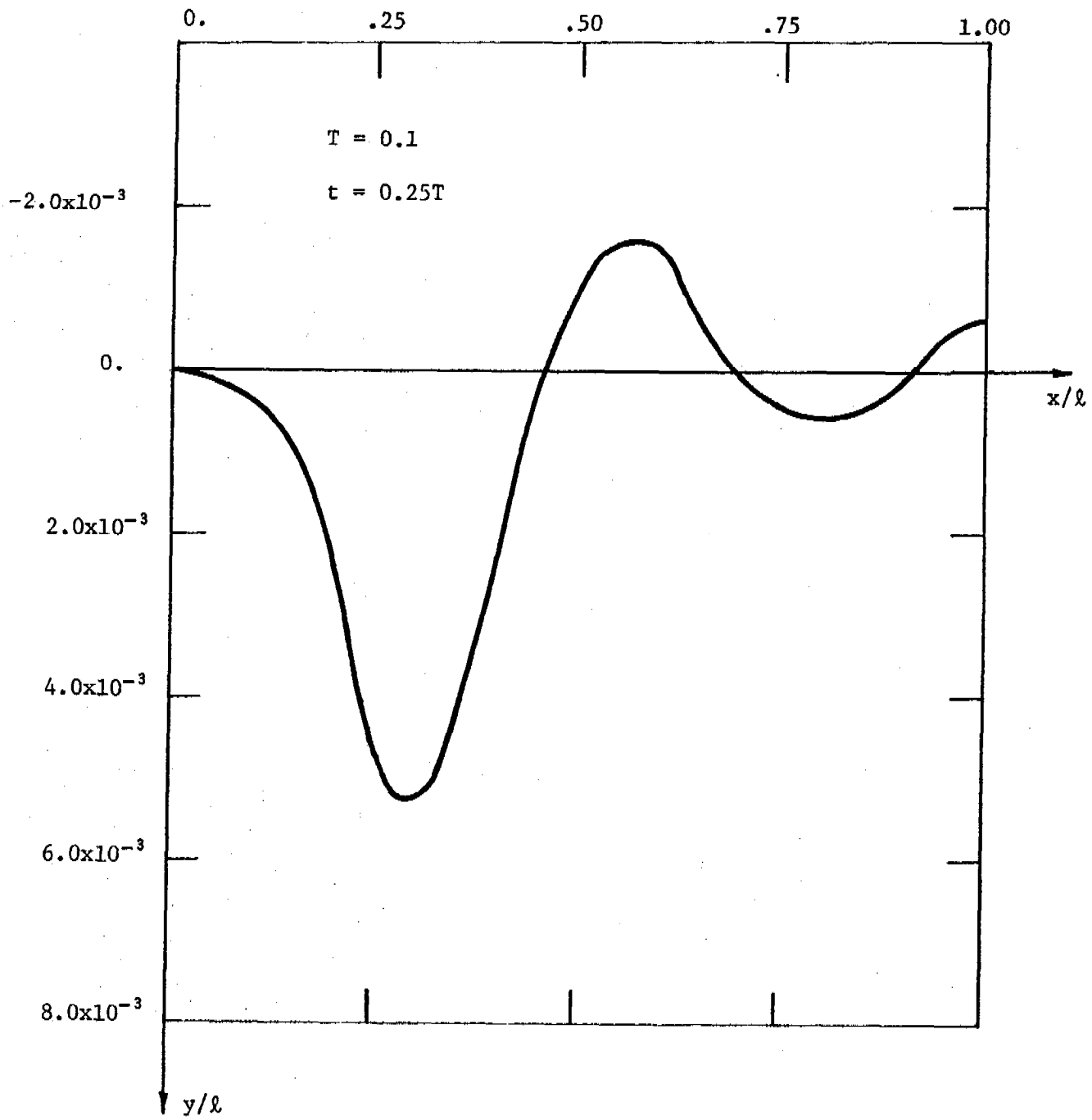


Figure 7. Deflection(s) of a Cantilevered Beam Under a Moving Couple.

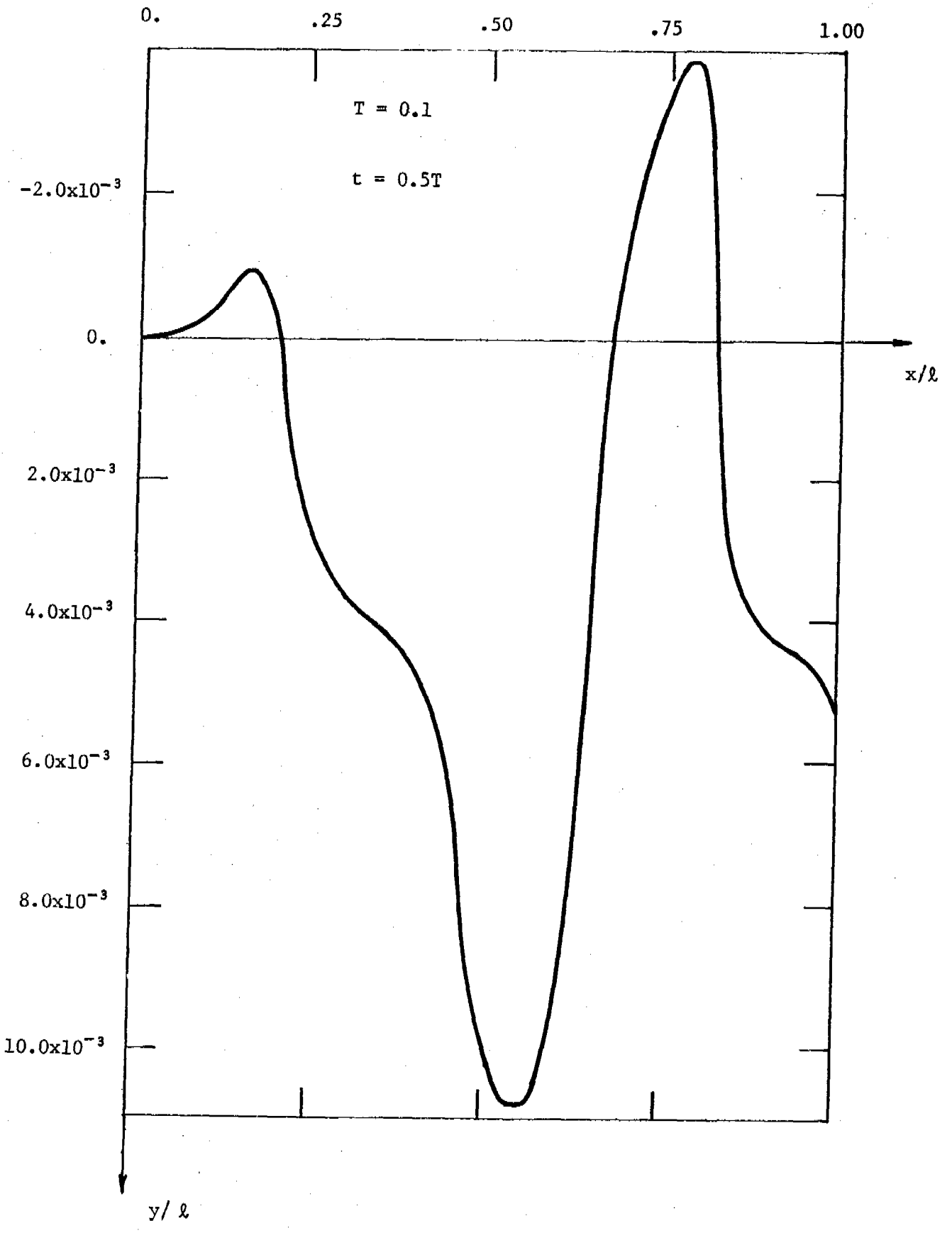


Figure 8. Deflection(s) of a Cantilevered Beam Under a Moving Couple.

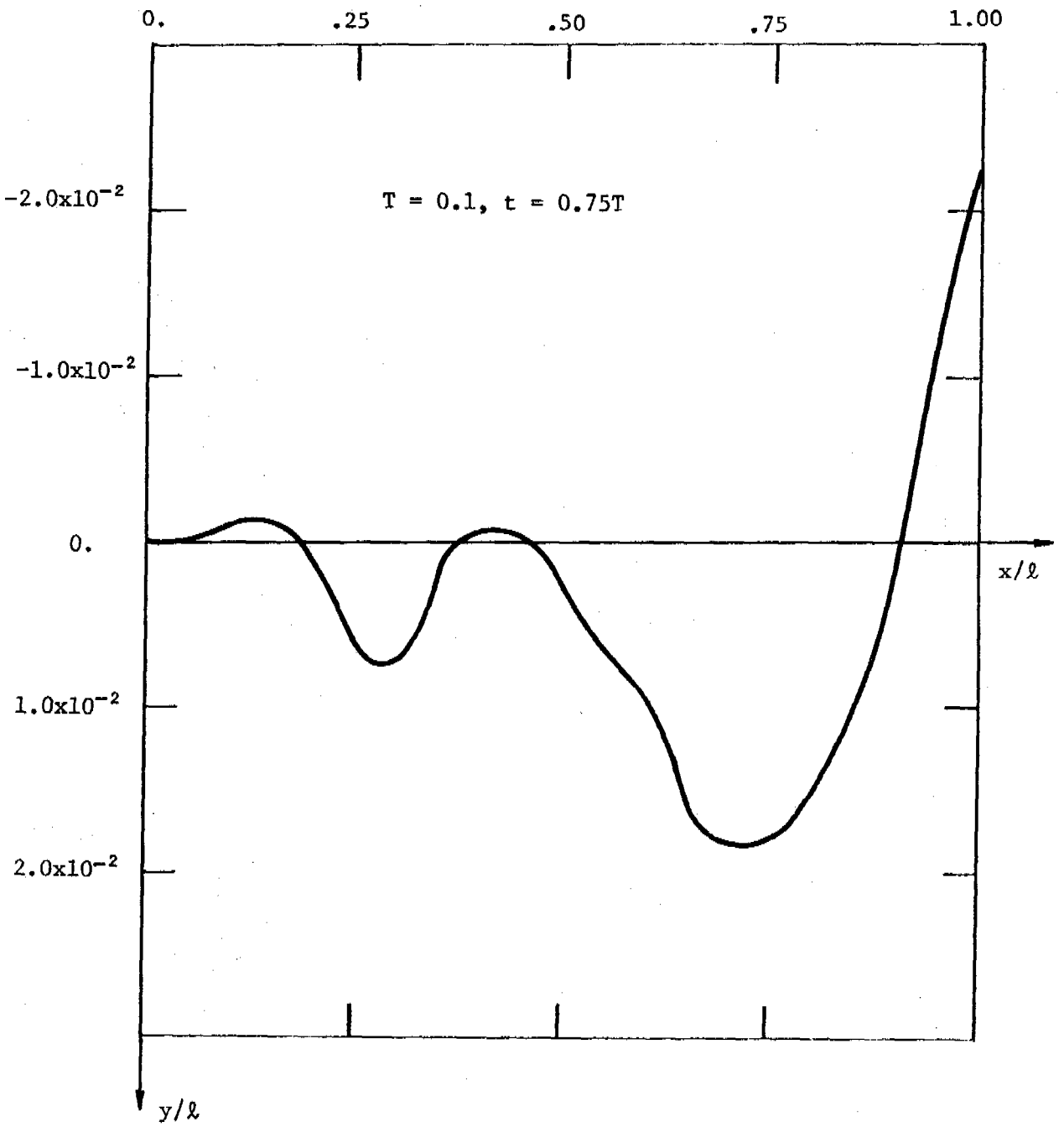


Figure 9. Deflection(s) of a Cantilevered Beam Under a Moving Couple.

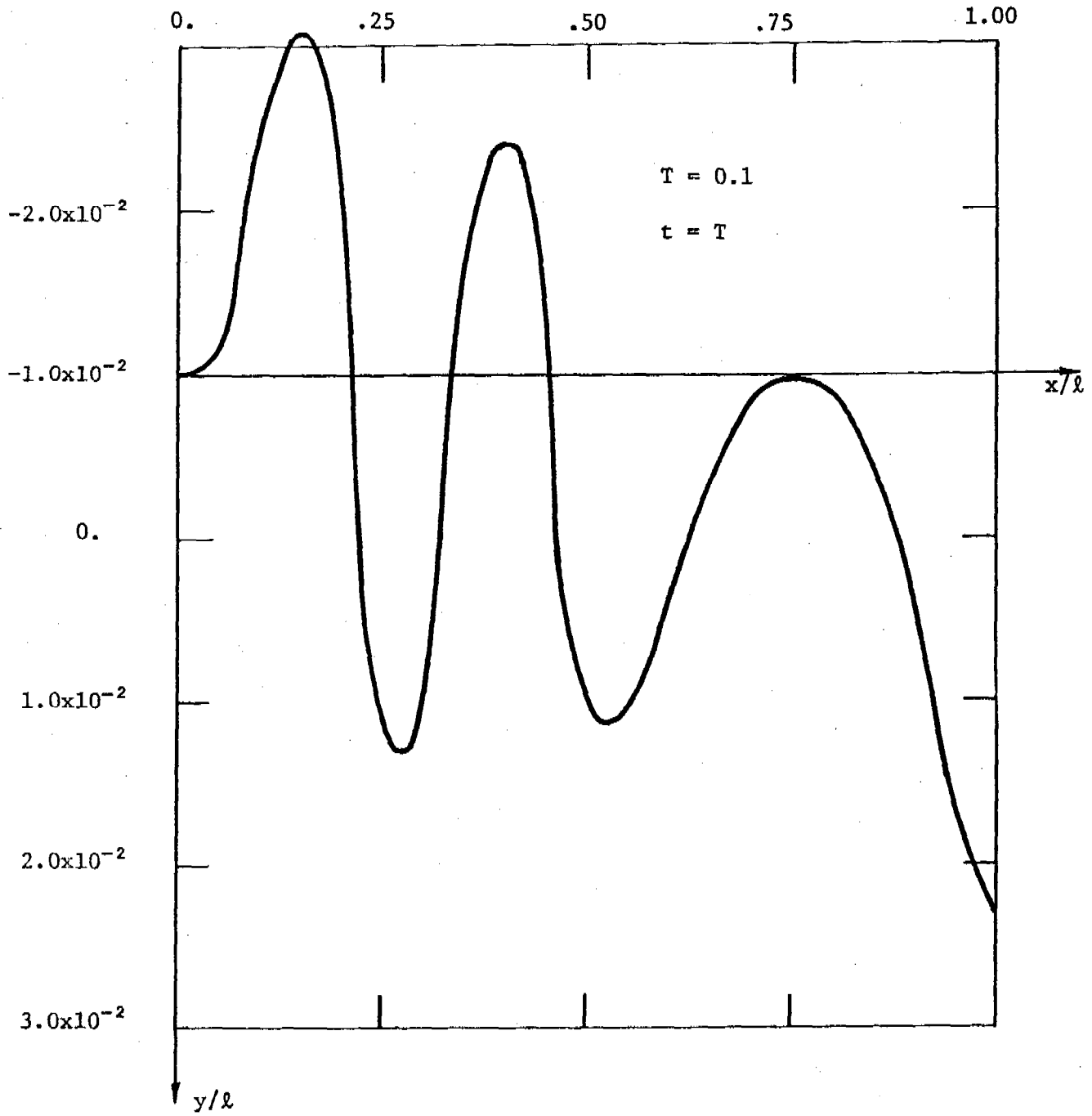


Figure 10. Deflection(s) of a Cantilevered Beam Under a Moving Couple.

ANALYTICAL SOLUTIONS IN NUMERICAL ANALYSIS

J L Harris
US Army Missile Command
US Army Missile Laboratory
Redstone Arsenal, Alabama 35898

ABSTRACT. This paper will deal with the useage of exact solutions to differential equations as a means to decrease the computational burden associated with guided missile flight simulation. In this type of simulation, the governing differential equations are typically solved for the highest order derivitive, then numerically solved by forward integration over time, using some appropriate integration method and a time increment which must be chosen sufficiently small for representation of the various system responses. Many of the system responses can be represented by linear differential equations, the solutions to which are well known. By computing the exact solution to the equation, the time increment (dt) can be chosen as large as the dynamics of the inputs to the response function will allow, rather than being governed by the dynamics of the response function itself.

In general this will result in a computer time reduction, although there may be instances where this is not true. An example will be presented.

1. INTRODUCTION. The stimulation to write this paper has grown out of several years of guided missile flight simulation on the part of the author. Most of this simulation was done digitally. The typical problem involves the generation of a missile trajectory as it flies toward its target, which in general is also moving. Fig. 1 illustrates this. This simulation has been undertaken to determine miss distance relative to the target (aim point), probability of target kill, maximum intercept range, altitude, and other performance limiting conditions, such as missile control system requirements.

2. SIMULATION APPROACH. The general approach to digital simulation of guided missile motion is to arrange the various differential equations which govern the instantaneous motion into expressions which define the highest derivative for each variable which must be represented. These equations may be partial differential equations, and they may be non linear with non constant coefficients. They are then numerically integrated by one of the many numerical integration schemes available. The intent of this paper is not to treat numerical integration schemes, rather it is to suggest that occassionally there are differential equations embedded in the representation which can be solved analytically and to propose that these be solved analytically instead of numerically. Table 1 presents the six equations for missile motion. These are always present. Control equations, etc., vary from missile to missile.

3. A SIMPLIFIED EXAMPLE. Figure 2 shows a basic block diagram for a simulation representation. The situation is driven by the difference in target and missile position, expressed in inertial space, taking round earth

considerations into account if range/altitude are sufficiently large. The difference in target and missile positions are used to calculate the orientation in inertial space of a line connecting the missile and target. Missile orientation is referenced to inertial space, and the seeker may be referenced to missile orientation or to inertial space, in either case the seeker pointing error is the difference between the seeker centerline and the line of sight angles. The seeker error is used to drive the seeker in a direction to reduce the error, and to command the missile to maneuver as needed to produce target intercept. This command produces a control deflection of some form (fin deflection, thrust deflection, etc.). The airframe responds to this, and accelerations are produced, which are integrated to update missile velocity and position, thus closing the loop.

4. A DETAILED EXAMPLE. The seeker functional representation from Figure 2 is shown with an additional level of detail added in Figure 3. It is assumed that the error is multiplied by a constant, K, then input to the seeker drive as a rate command, $\dot{\theta}_c$. It is assumed that the seeker's response to this command can be represented by a first order differential equation (first order response with time constant τ). This is still a considerable simplification of course. The commanded rate will be constant throughout the interval between data samples. The seeker rate and position are uniquely defined by the solution of the differential equation and insertion of the appropriate time as shown in Figure 3. Obviously the time of interest is the time of taking a new data sample. This solution, shown in Figure 3, involves an exponential which is typically time consuming to evaluate, but for a constant sampling interval and a constant time constant it needs to be evaluated only once, at problem initiation. Figure 3 also shows the numerical burden of calculating this response numerically with a very simple algorithm. It can be seen by inspection that the numerical burden is exactly equal for the two solutions, per step. But to achieve acceptable accuracy it would typically be required to break the sampling interval into sub-intervals for the case of numerical integration, even if a sophisticated integration algorithm were used. The number of subintervals would depend on the ratio of the sampling interval to the time constant.

For the analytical, i.e., exact, solution there is no error, obviously.

5. ON APPLICATION. Most of the so-called detail which we insert into missile simulations consists of response functions (transfer functions) which represent differential equations with constant coefficients, the solutions to which are well known and catalogged. For the sake of computation speed these should not be treated the same as the other differential equations which cannot be solved exactly. The analytical solutions should be taken advantage of.

In the sampled-data seeker case chosen for illustration the application is straight forward. Moving into the control system, not shown in detail, the situation is more complicated because the data flow becomes analog, and the concept of "sampling interval" is lost. Still, to do this simulation digitally some time interval (or intervals) is chosen by which to advance the solution. Use can still be made of exact solutions in many cases.

REPRESENATIVE PROBLEM
MISSILE FLIGHT SIMULATION

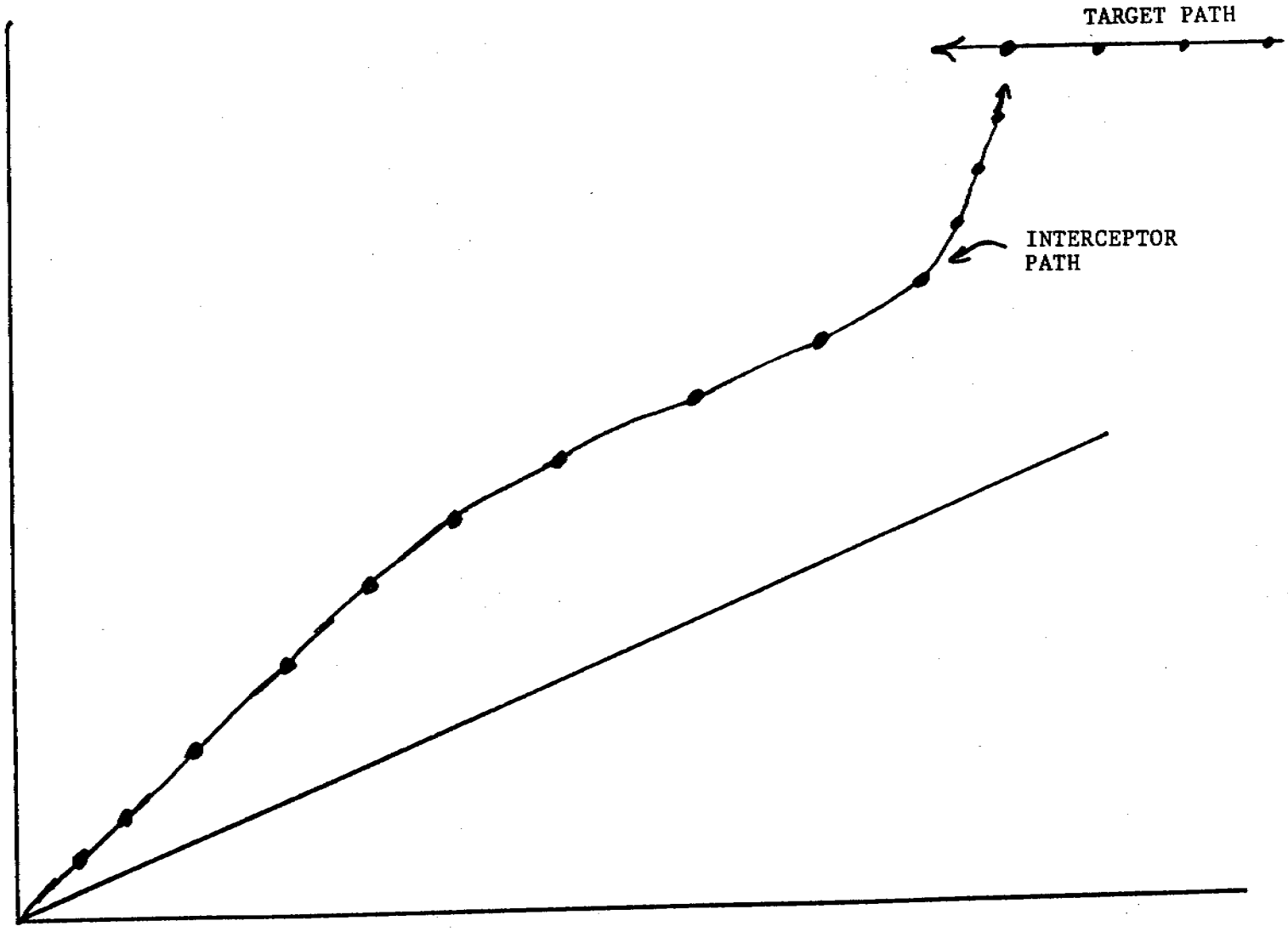


FIGURE 1

TABLE 1

DIGITAL SIMULATION APPROACH

THE GOVERNING EQUATIONS FOR MISSILE MOTION CANNOT BE SOLVED EXACTLY. THEY CAN ONLY BE NUMERICALLY APPROXIMATED

MISSILE MOTION EQUATIONS

$$\dot{U} = \left(T_{CL} - qS\,C_A\right)/m + G_{x'} - (WQ - VR)$$

$$\dot{V} = \left(T_y + qS\,C_y\right)/m + G_{y'} - UR + WP$$

$$\dot{W} = \left(T_P + qS\,C_z\right)/m + G_{z'} + UQ - VP$$

$$\dot{P} = \left[qSD\left(C_r - \frac{PD}{2v}C_{\ell p}\right) - P\dot{I}_x\right]/I_x$$

$$\dot{Q} = \left[qSD\left(C_{mz} - \frac{QD}{2v}C_{mq}\right) - Q\dot{I}_z - PR\left(I_x - I_z\right) + T_P\ell_P\right]/I_y$$

$$\dot{R} = \left[qSD\left(C_{my} - \frac{RD}{2v}C_{mr}\right) - R\dot{I}_z + PQ\left(I_x - I_y\right) - T_y\ell_y\right]/I_z$$

FUNCTIONAL REPRESENTATION
PASSIVE HOMING

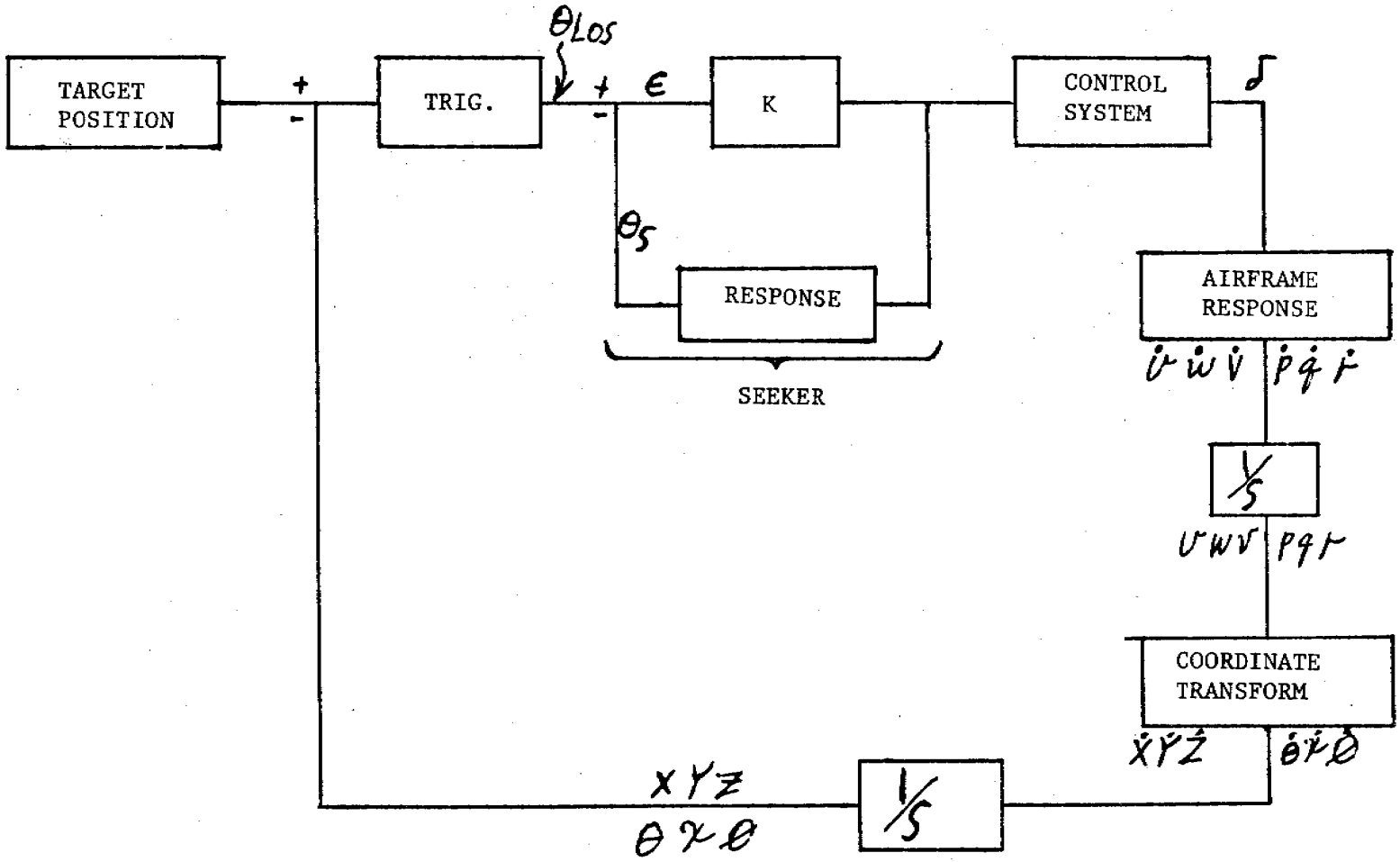
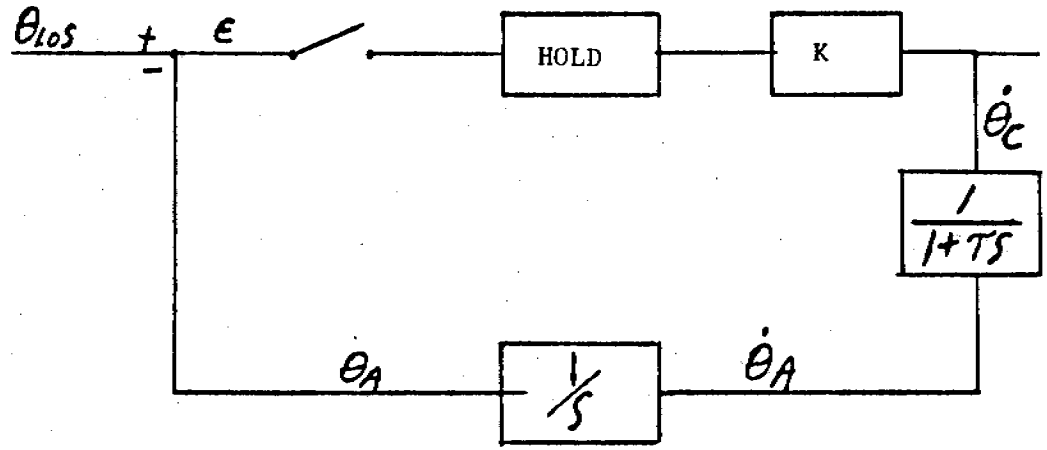


FIGURE 2

SAMPLED DATA SEEKER



$$\tau\ddot{\theta}_A + \dot{\theta}_A = \dot{\theta}_C \quad , \quad \ddot{\theta}_A = (\dot{\theta}_C - \dot{\theta}_A)/\tau$$

THIS DIFFERENTIAL EQUATION NEED NOT BE SOLVED NUMERICALLY

ANALYTICALLY

$$\theta_A = \tau(\dot{\theta}_C - \dot{\theta}_{0A})(e^{-t/\tau} - 1) + \dot{\theta}_C t + \theta_{0A}$$

$$\theta_{n+1}^A = \tau(\dot{\theta}_n^C - \dot{\theta}_n^A)(e^{-\Delta t/\tau} - 1) + \dot{\theta}_n^C \Delta t + \theta_n^A$$

$$\dot{\theta}_{n+1}^A = \dot{\theta}_n^C - e^{-\Delta t/\tau}(\dot{\theta}_n^C - \dot{\theta}_n^A)$$

NUMERICALLY
EULER INTEGRATION

$$\theta_{n+1}^A = \frac{\Delta t^2}{2\tau}(\dot{\theta}_n^C - \dot{\theta}_n^A) + \dot{\theta}_n^A \Delta t + \theta_n^A$$

$$\dot{\theta}_{n+1}^A = \frac{\Delta t}{\tau}(\dot{\theta}_n^C - \dot{\theta}_n^A) + \dot{\theta}_n^A$$

THE COMPUTATIONAL LOAD IS EQUAL, FOR THIS CASE

FIGURE 3

A STRATEGY FOR INTEGRATING PROGRAM TESTING AND ANALYSIS

Leon Osterweil
University of Colorado
Boulder, Colorado

ABSTRACT. This paper presents a view of how the techniques of static analysis and dynamic program testing can be combined and integrated into a tool supported methodology which smoothly incorporates the best features of each. The paper is composed of two major components. The first is more general and descriptive. In it the central importance of dynamic testing by means of programmer generated assertions is stressed first, and some remarks about tool support for assertion testing are made. Various weaknesses of dynamic testing are then remarked upon, motivating the desirability of using static analysis as well. The general characteristics of static analysis, and especially data flow analysis are described next. Static analysis is then described as a technique for making certain kinds of dynamic testing more efficient and trustworthy. Symbolic execution and formal verification are presented next and described as logical and important components of the integrated system being described. The second component of the paper deals with TOOLPACK, an integrated ensemble of tools of the types described in the first component. The architecture and high level design of TOOLPACK are described, and some implementation plans are presented.

This work supported by ARO grant DAAG 29-80-C-0094, National Science Foundation grants MCS8000017, MCS77-02194, and Department of Energy grant DE-AC02-80ER10718.

I. INTRODUCTION. Software engineering is a discipline which has recently been experiencing a period of considerable but unstructured growth. It now shows signs of embarking upon a phase of coordination and consolidation. There has been a large amount of work devoted to the development of softwre engineering tools. This seems to be particularly promising work, as tools are vehicles for capturing software engineering concepts in a way which is tangible and useful to software practitioners. Through well-implemented tools, desirable policies can be promulgated and enforced throughout a project, in a way which increases the coordination and efficiency of that project.

In the past, the quality of tools produced has been spotty. Worse, however, the goals of most tools and the domains of their efficacy have rarely been clearly enunciated. As a consequence, it has been difficult for the community of software practitioners to select tools appropriate for facilitating work on the specific tasks comprising their software development activities. Thus specification of the goals and domains of efficacy of a tool should be an important part of its documentation. The availability of such specifications should enable practitioners to intelligently select and configure a set of tools into an environment capable of supporting specific software production activities.

In this paper we propose a generic configuration of tool capabilities and an approach to building an integrated system of such tools, called TOOLPACK.

II. CLASS ONE - DYNAMIC TESTING AND ANALYSIS TOOLS. The terms dynamic testing and dynamic analysis, as used here, are intended to describe most of the systems known as execution monitors, software monitors and dynamic debugging systems ([Balz 69], [Fair 75], [Stuc 75] and [Gris 70]).

In dynamic testing systems, a comprehensive record of a single execution of a program is built. This record -- the execution history -- is usually obtained by instrumenting the source program with code whose purpose is to capture information about the progress of the execution. Most such systems implant monitoring code after each statement of the program. This code captures such information as the number of the statement just executed, the names of those variables whose values had been altered by executing the statement, the new values of these variables, and the outcome of any tests performed by the statement. The execution history is saved in a file so that after the execution terminates it can be perused by the tester. This perusal is usually facilitated by the production of summary tables and statistics such as statement execution frequency histograms, and variable evolution trees.

Despite the existence of such tables and statistics, it is often quite difficult for a human tester to detect the source or even the presence of errors in the execution. Hence, many dynamic testing systems also monitor each statement execution checking for such error conditions as division by zero and out-of-bounds array references. The monitors implanted are usually programmed to automatically issue error messages immediately upon detecting such conditions in order to avoid having the errors concealed by the bulk of a large execution history.

Some of this can be exemplified with the aid of a simple minded program. Figure 1 shows a program whose purpose is to produce the areas of rectangles and triangles having integer dimensions, when the dimensions are given as input. The program, a procedure called area, is divided into two major functional portions. One function, implemented by procedure lookup, returns the area of the triangle or rectangle by using a table lookup. The two dimensions input for the object are used as the first two indices into the table, a three-dimensional array, A. If the area of a rectangle is desired, the value 1 must be input with the dimensions, a value 2 indicates the area of a triangle is desired. A value 0 causes the lookup loop to terminate. The value 1 or 2 is used as the third indexing coordinate into array, A.

Array A is initialized by the second functional portion of the program implemented by the procedure init. This procedure initializes A in a somewhat indirect way, perhaps

motivated by an interest in eliminating the need for multiplications.

In Figure 2 we see the same program augmented by the code necessary to monitor for two types of errors -- division by zero and out of bounds array reference. This monitor-augmented program is typical of the code which would be generated automatically by a straightforward dynamic test tool. The monitors are positioned so as to assure that any occurrence of either of the two errors will be detected immediately before it would occur in the actual execution of the program. To a human observer it is obvious that many of these probes are redundant. We shall be very much concerned with studying the forms of automated analysis necessary to suppress such probes.

Some systems ([Fair 75], [Stuc 75]) additionally allow the human program tester to create his own monitors and direct their implantation anywhere within the program.

The greatest power of these systems is derived from the possibility of using them to determine whether a program execution is proceeding as intended. The intent of the program is captured by sets of assertions about the desired and correct relation between values of program variables. These assertions may be specified to be of local or global validity. The dynamic testing system creates and places monitors as necessary to determine whether the program is behaving in accordance with asserted intent as execution proceeds.

Figure 3 shows how the example program might be annotated with assertions. These assertions are designed to capture the intent of the program and explicitly state certain non-trivial error conditions, to which this program seems particularly vulnerable. Figure 4 shows how the code of Figure 1 might be augmented in order to test dynamically for adherence to or violation of the assertions shown in Figure 3. It should be clear from this example that dynamic assertion verification offers the possibility of very meaningful and powerful testing. With this technique, the tester can in a convenient notation specify the precise desired functional behavior of the program (presumably by drawing upon the program's design and requirements specifications). Every execution is then tirelessly monitored for adherence to these specifications. This sort of testing obviously can focus on the most meaningful aspects of the program far more sharply than the more mechanical approaches involving monitoring only for violations of certain standards such as zero division or array bounds violation.

From the preceding discussion it can be seen that dynamic testing is a powerful technique for detecting the

presence of errors. Because its results are applicable only to a single execution, however, it cannot be used in any obvious way to effectively demonstrate the absence of errors. Thus, it is not by itself an appropriate technique for verification (i.e., the process of showing that a program necessarily behaves as intended). Furthermore, although the assertions used for dynamic verification may themselves be valuable documentation of intent, dynamic testing does not itself create useful documentation of the nature of the program itself. Finally it is important to observe that the benefits of dynamic testing can only be derived as the result of heavy expenditures of machine storage and execution time.

III. CLASS TWO - STATIC ANALYSIS TOOLS. In the category of static analysis tools, we include all programs and systems which infer results about the nature of a program from consideration and analysis of a complete model of some aspect of the program. An important characteristic of such tools is that they do not necessitate execution of the subject program yet infer results applicable to all possible executions.

A very straightforward example of such a tool is a syntax analyzer. With this tool the individual statements of a program are examined one at a time. At the end of this scan it is possible to infer that the program is free of syntactic errors.

A more interesting example is a tool such as FACES [Rama 75] or RXVP [Mill 74] which performs a variety of more sophisticated error scans. These tools both, for example, perform a scan to determine whether all procedure invocations are correctly matched to the corresponding definitions. The lengths of corresponding argument and parameter lists are compared, and the corresponding individual parameters and arguments are also compared for type and dimensionality agreement. By comparing every procedure invocation with its corresponding definition in this way it is possible to assure that the program is free of any possibility of such a mismatch error. Note that this analysis requires no program execution, yet produces a result applicable to all possible executions. This sort of analysis, requiring a comparison of combinations of statements, can also be used to demonstrate that a program is free of such defects as illegal type conversions, confusion of array dimensionality, superfluous labels and missing or uninvoked procedures.

Data flow analysis is a still more sophisticated form of static analysis which is based upon consideration of sequences of events occurring along the various paths through a program. As such it is capable of more powerful analytic results than combinational scans such as those just described. The DAVE System [Oste 76], [Fosd 76] is a good example of such a tool. This system examines all paths originating from the start of a FORTRAN program and is capable of determining that no path, when executed, will cause a reference to an uninitialized variable. DAVE also examines all paths originating from a variable definition and is capable of determining whether or not there is a subsequent reference to the variable. A definition not subsequently referenced is called a "dead" definition. Hence DAVE is also capable of showing that a Fortran program is free of dead variable definitions.

Data flow analysis is based upon examination of a flow graph model of the subject program. The flow graph of every program unit is created and its nodes are annotated with descriptions of the uses of all variables at all nodes. Nodes representing procedure invocations cannot be annotated in this way immediately. Figure 5 shows the collection of three annotated flowgraphs which would be created to represent the variable usage by the statements of the example program of Figure 1. Procedures such as init and lookup which invoke no others are completely annotated. For such procedures a data flow analyzer like DAVE would determine the presence or absence of uninitialized variable references and dead variable definitions. This can be done by using data flow analysis algorithms such as LIVE and AVAIL [Hech 75] to efficiently determine the usage patterns of the program variables along the paths leading into or out of a program node. Having done this, it is possible to complete the data flow analysis of the main program. The details of this procedure can be found in [Fosd 76].

In summary we have seen that static analysis can be used to determine the presence or absence of certain classes of errors and to produce certain kinds of program documentation. Hence it is useful as an adjunct to a testing procedure and offers weak verification capabilities. It is also useful in supplying limited forms of documentation (e.g., the input/output behavior or a procedure's parameters and global variables). There is currently ongoing research which indicates that static analysis, particularly data flow analysis, can be used to both verify and test for wider classes of errors, as well as to produce additional forms of documentation (e.g., [Tayl 80]).

Of particular interest to us here is the possibility of using static data flow analysis to suppress certain of the probes generated by dynamic assertion verification tools as part of a comprehensive test procedure. As noted earlier, many of these probes generated by dynamic test aids are redundant. Their presence adds to the size and execution time of a test run yet has no diagnostic value. Hence an automatic procedure which removes them makes testing more efficient. It also serves to focus attention on the importance of exercising the remaining probes. Sometimes it is possible to remove all the probes generated by an assertion or single error criterion. In this case, it has been de facto demonstrated that the error being tested for cannot occur, and this aspect of the program's behavior has been verified. This perspective shows how testing and verification activities can be coordinated with each other.

For a specific example of this, let us examine the program in Figure 2. We will demonstrate how the three static analysis approaches - line-by-line, combinational and data

flow - can remove progressively more error probes. It is perhaps illuminating to observe that what is being contemplated here is actually code optimization in the classical sense (e.g., see [Alle 76], [Scha 73]. We are attempting to identify and remove redundant code in some cases and to move code to more advantageous positions in other cases. Even the techniques employed are directly derivative from optimization techniques.

A straightforward line-by-line scan of the program in Figure 2 will suffice to remove several test probes. Clearly the inequality tests in statements E2, E6, and E9 must always be true. Hence no more sophisticated analysis is needed to justify the removal of these probes.

A combinational examination of contiguous sequences of tests can eliminate other probes. For example, E4 and E7 contain identical tests, without any intervening flow of control or test variable alteration. Hence one of the tests can be removed. Similarly, either E10 or E13 can be removed, and either E11 or E14 can be removed. This sort of probe removal is based upon analysis that is quite similar to "peephole optimization" [Scha 73].

Additional probe removal can be justified by data flow analysis arguments. This analysis could be used to remove the test probes at E4 and E7, as well as the probes at E19 and E22. It should be noted that this analysis is more powerful than the combinational analysis outlined above, and thus capable of justifying the removal of the probes named earlier. Some insight into procedures for some removal of such probes can be found in [Oste 77] and [Boll 79].

Static analysis can also be used to justify the deletion of certain probes inserted in response to assertions. Note that assertion A1 in Figure 3 expands to probe statements P1,1; P1,2; P1,3; P1,4; and P1,5. Assertion A4 also expands to 5 probes in the program in Figure 4. All of these probes could be avoided if a static scan were used first to determine which (if any) of the procedure parameters were used as outputs by the procedure.

In this case static analysis can be used to remove all probes resulting from an assertion. Hence verification of the assertion can be achieved. On the other hand, we saw that many, but not all, of the subscript range checking probes can be removed by static analysis. We shall shortly show that some additional probes can be removed by using symbolic execution and constraint solving.

We have thus shown that there are significant assertion types and error categories which can be completely verified through static analysis. It seems important to determine

which other assertion types and error categories give rise to probes which can be partially or totally removed by static analysis. This is currently an open research area. It is clear, however, that assertions of functional equality such as A2 and A3 are beyond easy verification by static analysis. Furthermore, the removal of subscript range test probes involving functions of test variables (e.g., 1 <= J-1 <= 20 in E8) seems to require either a set of special case static analyses or a different more general form of analysis. We discuss such a different type of analysis next.

IV. CLASS THREE - SYMBOLIC EXECUTION TOOLS. By symbolic execution, we mean the process of computing the values of a program's variables as functions which represent the sequence of operations carried out as execution is traced along a specific path through the program. If the path symbolically executed is a path from a procedure start node to an output statement, then the symbolic execution will show the functions by which all of the output values are computed. The only unknowns in these functions will be the input values (either parameters in the case of an invoked procedure or read-in values when a main program is being symbolically executed).

Thus for example, suppose we symbolically execute the path 1, 2, 32, 3, 4, 5, 6, 7, 8, 9, 10, 11, 10, 11 in the program shown in Figure 1. At node 8 the value of i will be given by "1", and the value of A(1,1,1) will also be given by "1". After node 10 has been executed the first time, the value of j will be given by "2", A(1,2,1) will be given by "1 + 1". The next time node 10 is symbolically executed j will be "3" and A(1,3,1) will be "1 + 1 +1." If the path 8, 9, 10, 11, 10 is symbolically executed, then when node 8 is reached the value of i will be an unknown and hence represented by "i". The value of A(i,1,1) will likewise be represented by "i". When node 10 is reached for the first time j will receive the value "2" and A(i,2,1) will receive the value "i + i." Similarly, the next time node 10 is reached j will receive the value "3" and A(i,3,1) will receive the value "i + i + i".

A small number of symbolic execution tools has been built [Howd 78], [King 76], [Clar 76]. These tools mechanize the creation of the formulas and maintain incremental symbol tables. They employ formula simplification heuristics in an attempt to forestall the growth in size of the generated formulas and foster recognition of the underlying functional relations. (It should be noted, however, that these simplifiers do not take roundoff error into account and, therefore, may misrepresent the actual function computed by a sequence of floating-point computations). Hence a symbolic execution tool would report the value of A(i, 3, 1) after two iterations of the loop at node 9 to be "3 * i."

The foregoing discussion strongly indicates that symbolic execution is an excellent technique for documenting a program. Symbolic traces provide documentation of the actual functioning of a program along any specific path. In order to use symbolic execution as a technique for testing and verification however, it is necessary to augment the technique with a constraint solving capability.

In order to clarify this, let us begin by observing that the above described functional behavior occurs only

when the given path is executed. In general, however, a given program can execute an (often infinite) variety of paths, depending upon the program's input values. The conditions under which a given path is executed can often be determined by symbolic execution and constraint solution. Consider the program given in Figure 1, as represented by the flowgraph in Figure 5. Each edge of the flowgraph can be labeled by a predicate describing the conditions under which the edge will be traversed. Thus for example, the edge (7,8) is labeled "h > 1", the edge (9,10) is labeled "b > 2", (5,6) is labeled "h < 20" and edge (11,10) is labeled "j < b" (note that node 11 is assumed to represent the loop incrementation and termination test operations). Sequential control flow edges such as (8,9) and (10,11) are labeled by the predicate "true." Now clearly a given path will be executed if and only if all of the predicates attached to all of the path edges are satisfied. Unfortunately, a simple textual scan will express these constraints only in terms of the variables within the statements. Thus the constraints will in general not show their underlying interrelations. If the constraints are expressed in terms of the formulas derived through symbolic execution of the path, then a set of constraints all expressed in terms of the program's input values is obtained. Any solution of this set of constraints is a set of input values sufficient to force execution of the given path.

It is important to observe that some constraint systems are unsatisfiable, indicating that the path spawning them is unexecutable. We shall make important use of this shortly. No less important is the observation that the problem of determining a solution to an arbitrary system of constraints is in general unsolvable. Hence we must not expect that this potentially useful capability can be infallibly implemented.

Experimentation has indicated, however, that for an important class of programs the constraints actually generated are quite tractable [Clar 76].

Testing and verification capabilities can be achieved by attempting to solve constraints embodying error conditions and statements of intent. Thus, for example, if we create a predicate constraining the subscript i to be "i < 1" at statement 8, we are specifying an out-of-bounds array reference error. This constraint is clearly inconsistent with the constraint "i > 1" attached to edge (7,8). Hence it is impossible for the first array subscript at statement 8 to be below bounds. Hence we have shown that one of the tests generated in Figure 2 is superfluous. A symbolic execution of a path from node 1 through node 8 will similarly show that testing i against 20 is superfluous for that path. The dynamic test for that error condition can be safely

removed if it is shown that all paths through node 8 must create constraints inconsistent with "i > 20." In this example that is the case because procedure init does not alter the value of h and init is always invoked with h = 20. These facts can be inferred from static analysis. Hence a combination of static analysis, symbolic execution and constraint solution can be used to eliminate statement E1 of Figure 2. Similar arguments can be used to eliminate statements E4, E7, E5, E8, E10, E11, E12, E13, E14, E15, E19 and E22.

Statements E8 and E15 are particularly interesting. It could be argued that static analysis is sufficient to eliminate these subscript checking probes as well. The subscripts being checked here, however, are functions of program variables. Surely static analysis rules could be devised for each of these situations, but other rules would have to be devised for other common occurrences. The result would be an inelegant mass of special procedures. A symbolic trace, on the other hand, easily shows all functional relations, and readily expresses the needed range checking tests directly in terms of the input values. Thus the symbolic execution/constraint solving approach provides an elegant technique which avoids the need for the inelegant special-cases approach.

It is important to note that we have analytically justified the removal of virtually all subscript checking probes from the program in Figure 2. In particular, all probes inserted to check the subscripts of statements 8, 10 and 17 can be removed. Hence we have verified that these statements correctly reference array A.

Although statement E16 is a probe for a different error (division by zero) it should be apparent that the analytic technique just described can be used to show that the test embodied in E16 is also unnecessary. This error condition is expressed as the constraint "xk=0." This will be inconsistent with any constraint set arising from symbolic execution of a path through node 14. Yet static analysis will show that node 14 must always be executed prior to node E16. Hence it is verified that the division in statement 18 is always well defined.

Probes E17, E18, E20 and E21 cannot be removed, however. In fact symbolic execution of a path such as 34, 35, 36, 21, 22, 23, 24, 25 yields only the following constraints:[1]

---

[1]The notation (i) should be read as "the ith value taken as input, to this path." Hence in this case (3) means "the third value read in."

③ ≠ 0 (from edge (35,36))
③ = 1 (from edge (24,25))

Thus clearly when statement 25 is encountered ③ is constrained to be 1, but ① and ② are subject to no constraints. An out-of-bounds subscript error at statement 25 could be simulated by any of the constraints i<1, i>20, j<1, or j>20. After symbolic execution these become ① <1, ① >20, ② <1 and ② >20. None of those is consistent with the constraints generated by consideration of path edges. Hence a solution such as

① = 0
② = 21
③ = 1

can clearly force execution of an array subscript reference error at statement 25. Thus we see that the symbolic execution/constraint solving technique is a powerful testing aid. It should be noted that the ATTEST system [Clar 76] implements most of the capabilities just described.

Perhaps the most important use of symbolic execution/constraint solution is as a technique for verifying assertions of functional relations between program variables. At the end of the previous section it was noted that verification of assertions such as A2, A3, A5 and A6 is beyond the power of the static analyzers which had been presented. We saw that static analysis is quite adept at inferring all the possible sequences of events which might arise during execution of a program, and that by comparing these with specifications of correct and incorrect sequences, testing and verification capabilities are obtained. When the statements of correct behavior are couched as predicates involving program variables, however, symbolic execution/constraint solution is most useful. This is not surprising, as symbolic execution is a technique for tracing and manipulating the functional relations between program variables.

We have already discussed the fact that the subscript references at statements 25 and 27 may cause array bounds violations. This was determined by using symbolic execution/constraint solution to demonstrate that probes P5,1 and P6,1 are not inconsistent with path induced constraints. Thus they cannot safely be removed and assertions A5 and A6 cannot be verified.

On the other hand, these techniques can help verify the correctness of assertions A2 and A3. By using symbolic execution for the path 10, 11, 10, we obtain the relation

$$A(i,j,1) = A(i,j-1,1) + i$$

Viewing this as a recurrence relation whose initial condition is given by

$$A(i,1,1) = i$$

we can obtain the analytic solution

$$A(i,j,1) = j * i$$

from the theory of finite difference equations. This relation is exactly the one asserted by A2. Hence this assertion is analytically verified and need not be dynamically verified. Clearly, this capability rested heavily upon being able to draw on results from finite mathematics. Cheatham has created a tool with impressive inferential capabilities of this sort [Chea 78], although the problem of determining the closed form of a recurrence is in general intractable. Also required here is the ability to recognize when two formulas are equivalent. This problem is likewise intractible in general.

Additional pitfalls of demonstrating functional equivalence are demonstrated by assertion A3. Here we easily see that symbolic execution will establish that after statement 17

$$A(i,j,2) = A(i,j,1)/2.0$$

This is mathematically equivalent to the equation

$$A(i,j,2) = 0.5*A(i,j,1),$$

and is readily recognized as being equivalent. Because of the peculiarities of floating point hardware, however, the two formulas

$$A(i,j,1)/2.0 \text{ and } 0.5*A(i,j,1)$$

will often evaluate to different values. Hence the results of symbolic verification and dynamic verification may differ.

Despite these various limitations we are encouraged to believe that symbolic execution/constraint solution can be used to yield impressive documentation, testing and verification capabilities. Perhaps these limitations can be put in better perspective by observing that symbolic execution and constraint solution are the basic techniques used in formal verification or so called "proof of correctness" ([Elsp 72], [Lond 75], [Hant 76]).

In formal verification the intent of a program must be captured totally by assertions imbedded according to the dictates of a criterion such as the Floyd Method of

Inductive Assertions [Floy 67]. The correctness verification is established by symbolically executing all code sequences lying between consecutive assertions and showing that the results obtained are consistent with the bounding assertions. The consistency demonstration is generally attempted by using predicate calculus theorem provers rather than constraint solvers as discussed here.

It is crucial to observe, that these theorem provers are subject to the same theoretical limitations discussed earlier. The undecidability of the First Order Predicate Calculus makes it impossible to be sure whether a theorem is true or false. Hence we cannot be guaranteed of an answer to the question of whether a symbolic execution will yield results consistent with its bounding assertions. Furthermore, the symbolic execution may make simplifications and transformations of real formulas which do not recreate the functioning of floating point hardware. These and similar limitations of formal verification have long been acknowledged. Yet still formal verification is rightly regarded as a useful technique capable of increasing one's confidence in the functional soundness of a program. This is exactly the sense in which the symbolic execution/constraint solution technique just discussed should be considered worthwhile.

In fact, this technique is of more worth to a practitioner than formal verification, because of its flexibility. As already observed, formal verification requires a complete, exhaustive statement of a program's intent. The technique just described focuses on attempting to justify or disprove the validity of individual assertions. This gives the practitioner the ability to probe various individual aspects of his program as he may desire. From this perspective we view formal verification as the logical, orderly culmination of a process of verifying progressively more complete assertion sets. This culmination is rarely reached due to its prohibitive costs.

V. A STRATEGY FOR INTEGRATING TOOL CAPABILITIES. In this section we propose some ways in which the preceding classes of tools can be combined to address important software implementation objectives. It seems that in creating software the overriding goal is to create a product which demonstrably meets its current objectives and shows promise of being adaptable to meet foreseeable changes in the objectives. Much research and experimentation has been devoted to studying how to achieve this goal, and much is yet to be understood. From this past work, however, a view of the software development activity can be safely advanced. A possible diagram of this view of the software production activity is shown in Figure 6.

From this diagram it is clear that the activity should be greatly facilitated by automated aids to documentation, testing and verification. The preceding sections have provided a basis for seeing how such automated aids can be fashioned from a coalition of static analysis, symbolic execution and dynamic testing aids. We now propose some details.

A. Documentation

A complete set of program documentation must fully describe the structure and functioning of the program. Clearly such a set must describe a wide variety of aspects of the program. At present it seems that certain of these items of description must inevitably be supplied by humans. The previous sections of the paper have shown, however, that some documentation can be generated by tools. This documentation is, moreover, probably more reliably and cheaply done by such tools. In addition, if some documentation is done by tools, the remaining documentation is likely to be done more carefully by humans, thereby suggesting the possibility of greater quality and reliability.

Earlier sections of this paper suggest that static analysis tools should be used first to create such documentation as cross reference tables, variable evolution trees, and input/output descriptions of individual variables and procedures. Symbolic execution tools can be used next to create descriptions of the functional effects of executing various paths through the code. With constraint solution, a complete input/output characterization of the code could be obtained. Performance characteristics can be measured and documented with the aid of a dynamic testing tool. It is proposed that all this documentation be stored in a central data base, forming a skeleton of the complete documentation. Editors and interactive systems might be used to gather from humans such things as text descriptions of variables and procedures.

Each of the three tool classes produces a different kind of documentation. The types of documentation are only loosely related, hence the order of application of the tools can be dictated by the importance of each to the particular project. It is important to be aware, however, that static analysis is relatively inexpensive, symbolic execution is relatively expensive, constraint solution is usually quite expensive, and dynamic testing can be quite expensive if extensive elaborate test runs are done.

B. Testing

In a tool-assisted testing activity, the order of application of the tools is important. We have seen that tools can be used to focus the testing effort on paths and situations which appear to be more error prone. This is done by elimination of probes which were created to test for common programming errors and for adherence to explicit assertions. We saw that many probes can be removed by application of progressively stronger (and more costly) static analysis. Some remaining probes may be removed as a result of symbolic execution/constraint solution. We saw that these probes are likely to be the more substantive ones, monitoring for adherence to asserted functional intent. Their removal constitutes significant verification, but it can be expected that the cost of this will be relatively high. Hence symbolic execution should probably be employed cautiously or not at all as a test aid.

Finally, a dynamic test tool should be used to gather definite information about the existence and sources of error in the program. As already noted, testing can only show the presence of error in a test case, and even a simple program may have an infinite number of possible test cases. Hence the tool aided procedure just outlined has added importance in that it helps suggest test cases - namely those designed to exercise probes not analytically removed.

We have seen that testing and verification can be closely related activities. It is important to remember, however, that they do differ, most noticeably in their goals and placement in the software production process. Testing is the process of looking for errors. It should be viewed as an activity which occurs frequently during code production. Verification is the process of demonstrating the absence of errors. As such it should not be undertaken until and unless testing has failed to uncover errors. Thus it is a less frequent, more critical process, usually warranting greater expense and thoroughness. Our earlier discussion has shown specific ways in which verification results can be obtained as outgrowths of testing activities. We have also seen, however, that some activities provide good verification results but are likely to be relatively

costly. Because verification is a less frequent, more critical activity the extra cost may well be warranted.

C. Verification

A verification activity should start out like the testing activity just described. The first step is to suppress error testing probes and probes resulting from assertions. Static analysis can be used to suppress some probes, but the most significant probes probably can be removed only by symbolic execution. Verification is achieved on an assertion-by-assertion basis only when all probes generated by a single assertion have been removed. In this way stronger more complete verification can be obtained incrementally at greater cost and effort. Complete formal verification can be attempted if desired as the culmination of this process.

A final word should be said about the need for both verification and testing. It has been observed that testing cannot demonstrate the absence of errors. Hence verification should be attempted. We have also observed that the verification process has its own risks. The most important risk is that an assertion verification attempt may end inconclusively because of the failure to determine the consistency of constraints or the truth of a theorem. As already noted, this does not necessarily signify the falsity of the assertion, just that the verification attempt ended inconclusively. Another important risk is that the verification may be successful but rely implicitly upon false assumptions about the semantics of language constructs. As an example of this, we saw that symbolic executors generally make incorrect simplifying assumptions about the functioning of floating point hardware. As a result even a complete formal verification of program correctness may not completely rule out the possibility of an execution-time error. Hence it seems that both testing and verification should be considered techniques for raising the confidence of project personnel in the software product. Each is capable of bolstering confidence in its own way, and neither should be employed to the exclusion of the other.

VI. THE ARCHITECTURE AND DESIGN OF TOOLPACK. We now briefly summarize progress to date (March 1981) in designing a specific configuration of tools to meet many of the objectives just described. This tool configuration, named TOOLPACK[2] has as its objective the conveyance of strong comprehensive tool support to programmers who are writing, testing, transporting or analyzing mathematical software. Hence it must provide strong support for documentation, testing, and verification, as well as such code creation activities as editing.

It was decided that it would be prudent to address some specific needs of this well-established community as a prelude to attempting to address the general needs of a more general community, because of the lack of experience in building and studying such such large configurations of tools. It is anticipated that experience with systems such as TOOLPACK will eventually lead to the establishment of guidelines for production of other, perhaps more general, tool configurations.

The following summary is extracted from [Oste 81], wherein additional details can be found.

Before commencing with description of the design, it is important to enunciate the following basic assumptions:

1. The mathematical software whose writing, testing, and analysis is to be supported by TOOLPACK is to be written in a dialect of Fortran 77, which shall be carefully chosen to span the needs of as broad and numerous a user community as is practical.

2. TOOLPACK is to be designed to provide cost effective support for the production by up to 3 programmers of programs whose length is up to 5000 lines of source text. TOOLPACK may be less effective in supporting larger projects.

3. TOOLPACK is to be designed to provide cost effective support for the analysis and transporting of programs

---

[2]TOOLPACK is a cooperative project involving researchers at Argonne National Laboratories, Bell Telephone Laboratories, International Mathematical and Statistical Libraries, Inc., Jet Propulsion Laboratory, Numerical Algorithms Group, Ltd., Purdue University, University of California at Santa Barbara and University of Colorado. The project is being funded by the Dept. of Energy and the National Science Foundation, as well as the participating institutions.

whose length is up to 10,000 lines of source text. TOOLPACK may be less effective in supporting larger projects.

4. TOOLPACK will support users working in either batch or interactive mode, but may offer stronger more flexible support to interactive users.

A. Overview

A primary motivating goal of the design proposed here is that user support be supplied in as direct and painless a fashion as is feasible. In particular, the design attempts to relieve the user of having to understand the natures and idiosyncrasies of individual TOOLPACK tools. It also relieves the user of the burden of having to combine or coordinate these tools. Instead the design encourages the user to express his needs in terms of the requirements of his own software job. The TOOLPACK support system is designed to then ascertain which tools are necessary, properly configure those tools, and present the results of using the tools to the user in a convenient form.

The design encourages the user to think of TOOLPACK as an energetic, reasonably bright assistant, capable of answering questions, performing menial but onerous tasks and storing and retrieving important bodies of data. The aim of this is to make humans more effective in creating, documenting, testing and verifying program code.

In order to reach this view, the user should think of TOOLPACK as a vehicle for establishing and maintaining a file system containing all information important to the user, and using that file system to both furnish input to needed tools and capture the output of those tools. Clearly, such a file system is potentially quite large and is to contain a diversity of stored entities. Source code modules would certainly reside in the file system, but so would such more arcane entities as token lists, and flowgraph annotations. In order to keep TOOLPACK's user image as straightforward as possible this design proposes that most file system management be done automatically and internally to the TOOLPACK system, out of the sight and sphere of responsibility of the user. The user, in addition is to be encouraged to have access to only a relatively small number of files - only those such as source code modules and test data sets which are of direct concern to him. The user may create, delete, alter and rename these entities. More important, however, the user may manipulate these entities with a set of commands which selectively and automatically configure and actuate the TOOLPACK tool ensemble. The commands are designed to be easy to understand and use. They borrow heavily on the terminology used by a programmer in creating and testing code, and conceal the sometimes

considerable tool mechanisms needed to effect the results desired by the user.

B. User Visible File System Entities

In order to encourage and facilitate the preceding view of TOOLPACK, the system will support the naming, storage, retrieval, editing and manipulation of the following classes of entities, which should be considered to be the basic objects of TOOLPACK:

1. Program units:

A TOOLPACK program unit (PU) is the same as a Fortran program unit, except that TOOLPACK will require a number of representations of the program unit other than the source code (e.g., the corresponding token list and parse tree). The identity, significance, and utilization of these other representations are to be made transparent to the casual user. They will be managed automatically by the TOOLPACK system. On the other hand, they will be accessible and usable by more expert users through published standard naming conventions and accessing functions.

2. Execution Units:

Any set of TOOLPACK program units which the user chooses to designate, can be grouped into a TOOLPACK execution unit (EU). Other execution units may also be named as constituents of an execution unit, as long as no circularity is implied by such definitions. Ordinarily it is expected that an execution unit will be a body of code which is to be tested as part of the incremental construction process. Hence an execution unit might be a set of newly coded program units and a test harness. It is, however, not unreasonable (and indeed potentially quite useful) to consider a subprogram library to be an execution unit as well. Here, too, a TOOLPACK execution unit will consist of more than just source text, but the user will not need to be aware of the existence of any such additional entities.

An execution unit may also include optionally specifiable transformation specifications in order to enable users to painlessly apply canonical transformations to their code. This will facilitate such functions as porting of code and coding in higher level pseudolanguages and languages such as RATFOR [Kern 75] and EFL [Feld 78]. The preferred syntax for specifying the attachment of such transformation specifications to an EU has not yet been decided upon. It does seem clear, however, that this specification would be straightforward to accomplish if the TOOLPACK command language were functional in structure.

3. Test Data Collections:

A TOOLPACK test data collection (TDC) is a collection of test data sets to be used in exercising one or more TOOLPACK execution units. A test data collection may consist of one or more sets of the complete input data needed to drive the execution of some EU. Each test input data set may also have associated with it a specification of the output which is expected in response to processing of the specified input.

4. Options Packets:

A TOOLPACK options packet (OP) is a set of directives specifying which of the many anticipated options are to be in force for a particular invocation of one of the TOOLPACK tools. We see, for example, the need for Test Option Packets (TOP's) to specify dynamic testing probe insertion options.

The reason for defining these four entities as being basic to TOOLPACK is that they seem to facilitate the key processes of creating, transporting, documenting, testing, and verifying program code by giving the user considerable power and very broad flexibility in specifying how these activities are to be done. This design is intended to make it straightforward for the user to manipulate programs and to designate any body of code as the object of documentation testing and verification; and to make it easy for the user to select various degrees of rigor and thoroughness in analyzing and testing that code by exercising it with test data sets selected from the file system. This can perhaps best be seen by introducing the TOOLPACK command set and indicating how it is to be used to manipulate these named data entities.

C. The TOOLPACK Command Language

As indicated earlier, the exact syntax for the TOOLPACK command language has not been established and is still under study. A decision on a specific syntax will be made in the near future, and is likely to reflect our current predisposition towards functional notation.

Currently we are in a position to specify much of the semantic content of this language. In the following sections we name generically and characterize generally the major primitive functional capabilities currently anticipated.

The proposed TOOLPACK functional primitive set seems to divide logically into four parts: file system management primitives, edit (synthesis) primitives, tool application

(analysis) primitives, and perusal primitives. In the following subsections, the needed primitives will be discussed individually. Specific names and syntax are attached where necessary only as an aid to the discussions, rather than as a concrete proposal.

1. Data Base Manipulation Primitives

a. NEW entity

Invocation of this primitive results in the creation within the TOOLPACK file system of a specific entity, named as an argument to the primitive, which is either a PU, EU, TDC, or OP, as specified by the user. It is proposed that a TOOLPACK entity name be qualifiable by a structured qualification scheme facilitating the process of keeping backup versions, formatted versions, and transformed versions of code, and variously instrumented versions of EU's.

In addition, to simplify creating the named entity within the TOOLPACK data base, it is proposed that the NEW invocation also automatically invoke a special purpose tutorial/editor to assist the user in creating the desired entity. This is currently under discussion and not a firm part of the design. In the case of a NEW PU, that would be a text editor or Fortran intelligent editor. Special purpose editors might also be built to support the creation of NEW EU's OP's and TDC's as well.

b. OLD entity

Invocation of this primitive will cause the retrieval of the named entity and invocation of the editor appropriate for the type of the named entity.

c. DELETE entity

Invocation of this primitive results in the named entity being marked for deletion by the TOOLPACK file system.

d. REPLACE

Invocation of this primitive results in the entity currently being edited being stored back in the TOOLPACK file system. Ordinarily, the edited source image supplants the unedited source image and any currently stored images which have been derived from the source are automatically deleted by the TOOLPACK file management system. If, however, a new entity name is specified as an argument to this function, then the currently stored entity is left untouched, and a new entity is created and initialized to consist of the newly edited source text.

e. RENAME

Invocation of this primitive simply changes the name of an entity.

2. Edit (synthesis) primitives

EDIT entity

Invocation of this functional primitive would have basically the same effect as that of the OLD primitive. The appropriate editing capability would be summoned, and the named entity would be retrieved from the file system and readied for manipulation.

It is important to point out that this latter operation is expected to require a considerable amount of care and sophistication if it is to be done effectively in the general case. This is because of the TOOLPACK philosophy of considering some of the file system entiities to be accessed and manipulated individually. As a result it is conceivable that a user might access and alter a PU's parse tree, but never access the source text (or vice versa). In such a case it would be necessary for the TOOLPACK file management capability to be aware of the fact that inconsistency had been introduced between these two PU versions. Although this inconsistency may be tolerable for a while, a strategy for recognizing how and when to remedy it will have to be evolved. For example, the PU's source text might not need to be altered to become consistent with an altered parse tree until and unless the user were to attempt to access the source text (e.g., through OLD or EDIT).

More complicated situations may arise in carrying out operations on EU's. It is expected that users will be able to edit the source text of an entire EU by being given access to the source text of each of its comprising PU's. A strategy must be evolved, however, for dealing with the impacts of changes thereby made to the source texts of PU's incorporated into several different EU's. Likewise, a strategy is needed for correctly and efficiently effecting the execution of an EU which has previously been executed, but which contains some constituent PU's which have been EDIT'ed in the interim.

In solving such problems we intend to be guided to good solutions by the strategy successfully employed in the MAKE capability [Feld 79].

It is important to note that all of these sorts of problems could be addressed expediently, but inefficiently, by adopting a strategy of retaining little in the file system and purging entities from it upon any suspicion that

they might become inconsistent. This strategy could be adopted for early releases of TOOLPACK while more efficient strategies are evolved and tested.

3. Tool Invocation (Analysis) Primitives

These primitives invoke the functions which are at the heart of the reason for the TOOLPACK project - namely the facilitation of documentation, testing and verification. Consequently, great pains are being taken to make them easy to understand and use. In an important sense, the rest of the TOOLPACK primitive set has been designed so as to make these tool invocation primitives straightforward.

a. FORMAT entity [option packet]

Invocation of this primitive causes a named program unit to be taken as input to the TOOLPACK formatting tool. The resulting output text will supplant the original source text, and any derived images of the original source text will be deleted, unless a new entity name is also specified. In case a new entity name is supplied, the output of the formatter will be named with this new name and stored in the TOOLPACK file system as source text. It is expected that option packets will be specifiable in order to facilitate user selection from among many formatting options.

b. STRUCTURE entity [option packet]

Invocation of this primitive has the same effect as invocation of the FORMAT command, except that the TOOLPACK structurer is invoked instead of the formatter.

c. ANALYZE entity [option packet]

Invocation of this primitive results in the static analysis of the entity named. If the entity is a program unit, then single unit analysis will be performed. If the entity is an execution unit, then each program unit will be analyzed individually and integration analysis will also be performed.

An options packet may be specified by the user. This packet will enable the user to specify a level of thoroughness which will cause analysis to go as far as the lexical level, the syntactic level, the static semantic level or the data flow level. If this specification is omitted, the TOOLPACK system will select a default option (probably full data flow analysis).

The results of this analysis will be placed into an entity-attribute-relational data base which will then be available for perusal by a browsing subsystem to be

described subsequently, or for use as the basis for report generation tools whose goal would be the creation of superior documentation.

It should be clear that invocation of the ANALYZE command will effect the marshalling and configuration of a considerable assortment of tools and tool fragments. In addition, the stronger forms of analysis will necessitate the use of a number of intermediate images of the source text (e.g., parse tree, flowgraph, callgraph). As stated earlier, an important design criterion was that these maneuverings and the materialization of these intermediate images be concealed from the user and made the responsibility of the TOOLPACK system.

d. EXECUTE TEST EUname, TDCname, OPname

Invocation of this primitive results in the dynamic test execution of a collection of test data sets by a specified execution unit. The test data sets comprising the test data collection "TDCname" are fed into the execution module derived from the execution unit "EUname" one at a time, with the results of each execution being used to build an execution history data base. This data base also would be used to supply answers to user-posed questions as well as reports needed for documentation purposes.

The user may optionally specify a test options packet whose purpose is to select and specify which of the numerous execution monitoring options are to be employed during the test runs. The power and flexibility of the dynamic test monitoring system is to be considerable (see [Feib 81]). This is deemed to be necessary, but is also considered to be a serious problem, in that a casual or novice user may be intimidated by the variety of available choices. Hence it is proposed that a set of standard Test Option Packets (TOP's) be prepared by the builders of the dynamic test monitoring system and stored permanently in the TOOLPACK file system. Users could select from among these, tailor them to individual needs by using the TOP editor, or create their own TOP's from scratch. One of the standard TOP's would be configured to be the default TOP, enabling the user to do useful dynamic testing without needing to specify any TOP.

The actions occurring in response to an EXECUTE TEST invocation are expected to be extensive and complex. Here too, every effort has been made to conceal this complexity from the user while still offering considerable flexibility to construct testing situations and have them carried out with minimal expense through extensive reuse of intermediate data objects and entities.

D. Perusal Primitives

TOOLPACK will ultimately contain tools to facilitate the examination of the various entities in the TOOLPACK file system. This document has already described various special purpose editors, part of whose purpose will be to facilitate examination of the user-named file system entities (e.g., the PU source text, EU's, OP's and TDC's).

A different sort of tool is desirable for use in perusing the output of the static analysis and dynamic testing tools. As already noted, these tools will produce as output sets of analytic and diagnostic packets which could profitably be viewed as relational data bases. Tools for effectively browsing these data bases could be specifically constructed to efficiently scan these data bases for answers to expected queries. Existing text editors will probably serve as primitive forerunners of these tools in early releases of TOOLPACK.

Although it is probable that there will eventually be different browsers for browsing the static analysis and dynamic testing data bases, it is expected that they will both be invoked by the same command:

BROWSE[databasename]

The databasename is one which will be automatically created by the TOOLPACK system by a straightforward naming algorithm. For example, the data base produced by test run # n of TDC t applying TOP p to EU e would perhaps be named e/p/t/nDB. After each test run the user would be supplied this name and the size of the data base itself and offered the opportunity to SAVE the data base. SAVE'd data bases would then be available for subsequent BROWSE'ing.

The data base name would be optional in the BROWSE command. When omitted the data base last generated would be assumed.

The BROWSE command processor would determine from the data base name the type of data base to be BROWSE'd (static or dynamic) and invoke the necessary browsing tool.

E. An illustration of how the TOOLPACK architecture might be used to support the process of constructing a program

The following diagram is inserted here in an attempt to demonstrate that the TOOLPACK system architecture, as presented here, is capable of satisfying the requirements to which the TOOLPACK group has addressed its efforts. Thus the diagram is intended to show that individuals attempting

to perform code creation, as outlined earlier in this document, can be significantly aided and supported by the TOOLPACK system architecture as described. The diagram depicts what is thought to be a reasonable procedure for code creation. Hence the fact that it seems to be strongly supported by the proposed architecture is taken to be encouraging.

It is not claimed here that this activity diagram is a paradigm of "proper procedure." Hence readers who perceive or pursue this task in a different way should not feel that TOOLPACK disapproves of them or will not support them. Rather, such readers are strongly encouraged to determine whether the TOOLPACK system will be useful to them in supporting their activities.

In particular it should be noted that no symbolic execution or formal verification capabilities are currently proposed for inclusion in TOOLPACK, nor are they included in the activity diagram. This reflects the perception that mathematical software writers currently go about their work without these capabilities. As discussed earlier, these capabilities are regarded as being of great potential value and importance. Hence it is expected that they may be included in future releases of TOOLPACK.

Comments and Elaboration on Major Activities

1. Create regimen of test cases and required outcomes:

An editor is used to create these TDC's; results are stored in the data base for subsequent use, each TDC is indexed by a name supplied by the user.

2. Compose new source text:

A text editor is used for source code creation. The editor may be language dumb, or may incorporate various types of language awareness - e.g., may parse input, accepting only syntactically correct source, and outputting parse tree and/or token list; may automatically do some polishing as well. The output of this process may be PU source text, token list, parse tree or some combination of the three. Whatever the output, it is to be stored in the central file system indexed by PU name and version id., perhaps supplied by the user. The user may also define EU's as sets of PU versions and transformation specifications and assign these EU's names, thereby creating other file system entities.

3. Polish and/or structure text:

The user may at this point wish to polish and or structure source text created. There is to be an automatic purge of unpolished version of the PU from file system, unless the

user directs that the polished version be saved under new version name.

4. Perform static analysis:

The user requests "ANALYZE" and specifies a level of thoroughness for analysis and an EU (by name). New EU's may be defined at this point. Single unit and integration analysis is done - lexical, syntactic, static semantic and data flow - at user option. A data base of analytic results is created for browsing by means of the BROWSE subsystem.

5. Set up test runs:

The user creates TOP's, specifying types and thoroughness of dynamic monitoring. He may modify or create new TDC's here as well. This is basically an editing activity. The user must create new TDC's or access TDC's created in activity 1; an interactive editor would be useful here; a source text editor may be used to inject new assertions in the source text; a TOP editor may be used to build and modify various TOP's.

6. Run dynamic test(s):

The user specifies a sequence of test runs as a sequence of triples (EU,TOP,TDC) of named database entities; test runs are made and results go into relational data bases for perusal by the BROWSE processor.

This involves automatic instrumentation, compilation, link editing (including fetching of run-time libraries to support monitoring) creating data bases of results, creating and presenting to user of requested results.

7. Browse source text and test execution results:

This involves use of a query system and information management system to help the user identify and understand errors well enough to fix them.

VII. OVERVIEW OF TOOLPACK IMPLEMENTATION APPROACH. The preceding section was devoted to a presentation of a user's view of the TOOLPACK system. The purpose of this section is to suggest an interior, or implementor's, view of the TOOLPACK system. This section is not purported to be a complete design specification. It is offered, rather, in support of the contention that the TOOLPACK system, as presented, is eminently implementable with existing knowledge and technology. Hence the reader should feel comfortable in considering the merits of the proposed set of capabilities freed, to some extent, of worries about its realizability.

A. The File System

Clearly the primary feature of the proposed TOOLPACK system is the central file system of information about the subject program. The user is encouraged to think and plan his work in terms of it, and the functional tools all draw their input from it and place their output into it.

This file system is to be initialized with the start of a project and remain and grow throughout the lifetime of the project. There is no reason why 2-3 users may not all access this file system although we will make the implicit assumption that it is accessed by one user at a time in a non destructive way.

The TOOLPACK system itself will manage the file system primarily by means of a tree structured directory system and a modular set of file accessing and updating primitives. TOOLPACK files will not correspond directly to host machine files, but will rather be mapped onto segments of one or more large host system files. The TOOLPACK file accessing and updating capabilities will effect this segmentation and operate directly upon these large host system files. The object of this approach is to reduce the overhead of dealing directly and depending too heavily upon host file systems. An implementation of such a set of I/O capabilities (called PIOS) has been written in portable Fortran [Hans 80a]. Experience with this system has shown that this approach can be pursued without unacceptable loss of speed and efficiency. Thus, this system is expected to be used at least as a guide to an effective modularization of capabilities, and will perhaps, be incorporated into TOOLPACK in toto.

A tree structured file directory system (PDS) has also been written in portable Fortran [Hans 80b]. This is also quite appealing as at least a model of effective functionality and modularization, and perhaps as a body of code to be directly incorporated into TOOLPACK. It offers the added feature of being designed for ready interfacing with PIOS, thereby forming a portable file directory and accessing

mechanism. This tandem will go far towards implementing the TOOLPACK file system.

B. The Virtual Aspect of the File System and the Retention/ Replacement Module

A stated design objective for TOOLPACK is that it run effectively on a wide range of machines effectively utilizing larger amounts of storage when and if they can be made available. One way in which large amounts of storage can be effectively utilized is to store all derived and intermediate entities for possible future reuse. Storage economies can be gained by refusing to store those entities and instead regenerating them as needed. The strategy for retaining or regenerating these entities must be adjustable and transparent. It is highly desirable that both the end user and the tool ensemble always be safe in assuming that any needed named entities and derived images will always be available. Thus it is necessary that the TOOLPACK file management system assume the responsibility for either retrieving these items directly or having them created or regenerated (in case storage exigencies precipitated their deletion by TOOLPACK).

Perhaps the workings of this virtual file system scheme can best be understood through an example. Suppose one of the functional tools (e.g., the static analyzer) needed access to the parse tree of a particular version of a particular PU, let us call it SUBR/VER. The tool would request (and subsequently receive) the parse tree through a subroutine call such as

```
CALL DBFTCH('SUBR/VER/P', ARRAY, LEN)
```

where ARRAY is the name of the array within the tool which is to receive the parse tree, and LEN is a specification of the length of ARRAY, included in this invocation to guard against inadvertent array overflow.

Subroutine DBFTCH would then use 'SUBR/VER/P' as an index into the TOOLPACK file system directory in order to look up the internal designation of the TOOLPACK file containing the parse of SUBR/VER. Should there actually be such a TOOLPACK file, there would also be a length specifier for it. The length specifier would be used to determine whether the invoking tool's array was large enough to hold the parse. If so DBFTCH would need only to invoke a file I/O primitive to read the indicated TOOLPACK file into the invoking tool's array.

If the directory contained no entry for the parse tree, DBFTCH would need to see that a parse was created. Guidance for this process would come from an internal table

specifying how the various TOOLPACK derived images are to be derived from each other. This table would be essentially a directed acyclic graph (DAG) with the nodes representing the various file system entity types, and the edges representing processing capabilities. In particular, an edge would represent the processing capability needed to produce the entity at its head from the entity at its tail. It is worth noting that the production of some entities (e.g., an annotated flowgraph) might require that more than one process acting on more than one file system entity.

In any case DBFTCH would, from this table, produce an ordered list of the processes and entities which would be needed to produce the requested entity by traversing the dependency DAG. DBFTCH would then proceed down this list looking to see which entities are already stored in the file system. Using this information DBFTCH would then invoke in the correct order only those processes needed to produce the desired entity.

Returning to our example, DBFTCH would look up 'P' in the dependency DAG, which would then show that a parse tree is derived from a token list by a parser and a token list is derived from a source string by a lexical analyzer (lexer). Thus DBFTCH would next check for the existence in the file system of the token list for SUBR/VER. If it is present DBFTCH will invoke the parser producing the required parse tree. If the token list is absent, DBFTCH will first invoke the lexical analyzer to produce a token list from the source text, and then invoke the parser. If the source text should not be in the file system, an error message would be passed on to the user.

This virtual file system scheme could be stretched even farther. Although it is currently anticipated that the file system will hold in explicit form any formattings and structuring of a given piece of source text, this is not necessary. Such versions could be recreated by the file management system only when needed by following a procedure such as just outlined. Even whole static analysis or dynamic testing data bases could be regenerated in this way. This gives the file management system the flexibility to purge large files to regenerate storage while still retaining the ability to recreate these files when necessary.

This feature should prove particularly useful in hosting the TOOLPACK system on smaller storage machines. Here it may be necessary to permanently store only source text. Under these circumstances all derived images and intermediate entities will be routinely purged, requiring that they be recreated whenever needed. This will result in extra computation time to meet the user's request, but seems a very reasonable trade for the lack of storage.

In order for this scheme to work best the strategy for deciding which entities to delete and when to delete them must be carefully determined. There appears to be little experience in devising such strategies. Hence we must expect that a lengthy period of experimentation, observation and adjustment will be necessary. Thus our replacement/retention strategy will be encapsulated in a module to facilitate such experimentation and adjustment.

C. The Command Language Interpreter

As noted earlier, the TOOLPACK command language syntax has yet to be decided upon. Nevertheless, it is reasonable at this stage of design to sketch the architecture of the processor which must effect the execution of TOOLPACK commands.

This processor - the TOOLPACK command interpreter - will probably consist of three phases: command syntactic analysis, command decomposition into TOOLPACK functions, and sequential TOOLPACK function invocation.

Of the three, the first, syntactic analysis, should be the most straightforward. Once a command syntax is agreed to, a parser generator should suffice for the production of a parser capable of rendering commands into trees of command tokens. Straightforward though this may appear, it seems important to isolate syntax analysis in a separate phase in order to facilitate change. It is recognized that users' reactions to TOOLPACK may be strongly influenced by the perceived friendliness and ease of use of the command language itself. It thus seems important to enable changes in the language when and if experience indicates they are desirable. This will clearly be facilitated by using a parser-generator-created parser as the first phase of the command interpreter.

The second phase, command decomposition, will be more complex entailing 1) the selection of the standard template of TOOLPACK functions indicated by the command and 2) the elaboration of this template as indicated to be necessary by option selections, file system status, and reporting and contingency handling directives. In particular, it is expected that the semantics of each TOOLPACK command will be defined at least generally by a standard sequence of TOOLPACK functional processing steps to be performed by individual tool fragments. The flow of data structures through these fragments will be effected by the definition, creation and accessing of entities within the file system. Thus, the construction of specific file system primitive invocations will also be the responsibility of this second phase. In view of the preceding discussions of the virtual file system concept, it is clear that the regeneration and updating of

file system entities may also be entailed during command elaboration.

The end product of this phase is expected to be a sequential file of TOOLPACK directives describing in detail all steps needed to be carried out by TOOLPACK tool fragments in order to effect the specified command in the exact context of the current state of the TOOLPACK file system. As such this phase might well be viewed as a pseudocompilation into a machine independent intermediate code.

The final phase is the actual interpretation process. Here tool fragments and file system accessing primitives are invoked in the indicated sequential order, with allowances being made for alteration of sequencing due to errors or other contingencies.

D. Major TOOLPACK Functional Commands

As an aid to understanding how major TOOLPACK functional capabilities are to be fashioned out of smaller tool fragments, we now include diagrams indicating the way in which we propose to effect the implementation of two major functional commands.

1. ANALYZE (Static Analysis)

Figure 8 is a diagram showing how execution of the ANALYZE command will be effected by the operation of TOOLPACK tool fragments on file system entities. Some of these entities will be created by these fragments, but others (such as the token list and symbol table) should be thought of as perhaps having been created by the execution of other TOOLPACK commands.

File system entities are identifiable as being shown in squared boxes. Tool fragments are shown in circles or ovals. It should be noted that most of the indicated tool fragments have been built in at least prototype form as part of the DAVE project at the University of Colorado.

2. EXECUTE TEST (Dynamic Testing)

Figure 9 shows how execution of the EXECUTE TEST command will be implemented by tool fragments and functional capabilities (such as compilation and loading) to be borrowed from the host operating system and environment. It has been suggested that the testing capability as proposed in the NEWTON report [Feib 81] is too general to be comfortably thought of as a single functional capability. Hence the subdivision of NEWTON into smaller tool fragments and the introduction of several more sharply named and focussed capabilities is being studied.

3. Symbolic Execution

No symbolic execution capability is currently being planned for inclusion in early releases of TOOLPACK. This decision is influenced by the relatively high cost of creating this capability and its unfamiliarity to the TOOLPACK target user community. Both of these considerations are expected to change with time, encouraging the eventual inclusion of symbolic execution within TOOLPACK. This will be facilitated by the establishment in early TOOLPACK releases of a core set of tool fragments upon which a symbolic execution capability can be built at a later date.

VIII. SUMMARY.

Considerable experience with isolated individual tools has led us to believe that comprehensive collections of tools are possible and desirable. Logical tool integration strategies are now perceptible and are also reasonable as objects of experimental study. The TOOLPACK architecture is one such strategy. A system of this sort is under design and will be built and studied.

REFERENCES

[Alle 76] F. E. Allen and J. Cocke, "A Program Data Flow Analysis Procedure," CACM, 19, pp. 137-147 (March 1976).

[Balz 69] R. M. Balzer, "EXDAMS: Extendable Debugging and Monitoring System," Proc. AFIPS 1969 Spring Joint Computer Conference 34 AFIPS Press, Montvale, N. J.

[Boll 79] L. A. Bollacker, "Detecting Unexecutable Paths Through Program Flow Graphs," Masters Thesis, Dept. of Comp. Sci., Univ. of Colorado at Boulder, 1979.

[Chea 78] T. E. Cheatham, Jr. and D. Washington, "Program Loop Analysis by Solving First Order Recurrence Relations," Harvard Univ. Center for Research in Computing Technology, TR-13-78.

[Clar 76] L. A. Clarke, "A System to Generate Test Data and Symbolically Execute Programs," IEEE Transactions on Software Engineering, SE-2 pp. 215-222 (Sept. 1976).

[Elsp 72] B. Elspas, K. N. Levitt, R. J. Waldinger and A. Waksman, "An Assessment of Techniques for Proving Program Correctness," ACM Computing Surveys 4 pp. 97-147 (June 1972).

[Fair 75] R. E. Fairley, "An Experimental Program Testing Facility," Proc. First National Conf. on Software Eng., IEEE Cat. #75CH0992-8C pp. 47-55 (1975).

[Feib 81] J. Feiber, R. N. Taylor, L. J. Osterweil, "Newton--A Dynamic Program Analysis Tool Capabilities Specification," Tech. Report #CU-CS-200-81, Dept. of Computer Science, University of Colo., Boulder, Colo.

[Feld 78] Stuart I. Feldman, "The Programming Language EFL," Bell Laboratories Computer Science Technical Report #78.

[Feld 79] Stuart I. Feldman, "Make--A Program for Maintaining Computer Programs," Software Practice and Experience 9 (April 1979) pp. 255-265.

[Floy 67] R. W. Floyd, "Assigning Meanings to Programs," in Mathematical Aspects of Computer Science 19 J. T. Schwartz (ed.) Amer. Math. Soc. Providence, R.I. pp. 19-32 (1967).

[Fosd 76] L. D. Fosdick and L. J. Osterweil, "Data Flow Analysis in Software Reliability," ACM Computing Surveys 8 pp. 305-330 (Sept. 1976).

[Gris 70] R. Grishman, "The Debugging System AIDS," AFIPS 1970 Spring Joint Computer Conf., 36 AFIPS Press, Montvale, N. J. pp. 59-64.

[Hans 80a] D. R. Hanson, "A Portable File Directory System," Software Practice and Experience 10 (August 1980), pp. 623-634.

[Hans 80b] D. R. Hanson, "The Portable I/O System PIOS," University of Arizona, Department of Computer Science, Tech. Report #80-6a (April 1980, revised December 1980).

[Hant 76] S. L. Hantler and J. C. King, "An Introduction to Proving the Correctness of Programs," ACM Computing Surveys 8 pp. 331-354 (Sept. 1976).

[Hech 75] M. L. Hecht and J. D. Ullman, "A Simple Algorithm for Global Data Flow Analysis Problems," SIAM J. Computing 4 pp. 519-532 (Dec. 1975).

[Howd 78] W. E. Howden, "DISSECT - A Symbolic Evaluation and Program Testing System," IEEE Trans. on Software Eng., SE-4 pp. 70-73 (Jan. 1978).

[Kern 75] B. W. Kernighan, "Ratfor--A Preprocessor for a Rational Fortran," Bell Laboratories Computing Science Technical Report #55.

[King 76] J. C. King, "Symbolic Execution and Program Testing," CACM 19 pp. 385-394 (July 1976).

[Lond 75] R. L. London, "A View of Program Verification," 1975 International Conf. on Reliable Software, IEEE Cat. #75-CHO940-7CSR pp. 534-545 (1975).

[Mill 74] E. F. Miller, Jr., "RXVP, Fortran Automated Verification System," Program Validation Project, General Research Corp., Santa Barbara, Calif. (Oct. 1974).

[Oste 76] L. J. Osterweil and L. D. Fosdick, "DAVE - A Validation, Error Detection, and Documentation System for FORTRAN Programs," Software - Practice and Experience 6 pp. 473-486 (Sept. 1976).

[Oste 77] L. J. Osterweil, "The Detection of Unexecutable Program Paths Through Static Data Flow Analysis," Proceedings COMPSAC 77, IEEE Cat. #77CH1291-4C, pp. 406-413 (1977).

[Oste 81] L. J. Osterweil, "Draft TOOLPACK Architectural Design,"technical memorandum, University of Colorado at Boulder, Dept. of Computer Science, March 1981.

[Rama 75] C. V. Ramamoorthy and S.-B. F. Ho, "Testing Large Software With Automated Software Evaluation Systems," IEEE Transactions on Software Engineering SE-1 pp. 46-58 (March 1975).

[Scha 73] M. Schaeffer, A Mathematical Theory of Global Program Optimization, Prentice-Hall, Englewood Cliffs, N. J. 1973.

[Stuc 75] L. G. Stucki and G. L. Foshee, "New Assertion Concepts in Self-Metric Software," Proceedings 1975 International Conference on Reliable Software, IEEE Cat. #75-CHO940-7CSR pp. 59-71.

[Tayl 80] R. N. Taylor and L. J. Osterweil, "Anomaly Detection in concurrent Software by Static Data Flow Analysis," IEEE Trans. on Software Eng. SE-6, pp. 265-278 (May 1980).

```
1       PROCEDURE AREAS;
2       DECLARE REAL A(20,20,2), INTEGER P1, P2, P3;
3          PROCEDURE INIT (H,B);
4          DECLARE INTEGER H, B, I, J, K, REAL XK;
5          IF H > 20 THEN ERROR STOP;
6          IF B > 20 THEN ERROR STOP;
7          DO FOR I = 1 TO H;
8             A(I, 1, 1) = I;
9             DO FOR J = 2 TO B;
10               A(I, J, 1) = A(I, J-1, 1) + I;
11               END;
12            END;
13         K = 2;
14         XK = 2.0;
15         DO FOR I = 1 TO H;
16            DO FOR J = 1 TO B;
17               A(I, J, K) = A(I, J, K-1) / XK;
18               END;
19            END;
20         END;
21         PROCEDURE LOOKUP (I, J, K);
22         DECLARE INTEGER I, J, K;
23         CASE;
24,25         K = 1: PRINT "AREA OF" I, J "RECTANGLE IS"
              A(I, J, K);
26,27         K = 2: PRINT "AREA OF" I, J "TRIANGLE IS"
              A(I, J, K);
```

```
28,29          ELSE: PRINT "PARAMETER ERROR:  K = " K;
30             END;
31          END;
32       CALL INIT (20,20);
33       LOOP FOREVER;
34          READ P1, P2, P3;
35          IF P3 = 0 THEN STOP;
36                    ELSE CALL LOOKUP (P1, P2, P3);
37          END;
38       END;
```

FIGURE 1: An example program

```
1       PROCEDURE AREAS;
2       DECLARE REAL A(20,20,2), INTEGER P1, P2, P3;
3          PROCEDURE INIT (H,B);
4          DECLARE INTEGER H, B, I, J, K, REAL XK;
5          IF H > 20 THEN ERROR STOP;
6          IF B > 20 THEN ERROR STOP;
7          DO FOR I = 1 TO H;
E1            IF ~(1 <= I <= 20) THEN SUBSCRIPT RANGE ERROR;
E2            IF ~(1 <= 1 <= 20) THEN SUBSCRIPT RANGE ERROR;
E3            IF ~(1 <= 1 <= 2) THEN SUBSCRIPT RANGE ERROR;
8             A(I, 1, 1) = I;
9             DO FOR J = 2 TO B;
E4               IF ~(1 <= I <= 20) THEN SUBSCRIPT RANGE
                 ERROR;
E5               IF ~(1 <= J <= 20) THEN SUBSCRIPT RANGE
                 ERROR;
E6               IF ~(1 <= 1 <= 2) THEN SUBSCRIPT RANGE
                 ERROR;
E7               IF ~(1 <= I <= 20) THEN SUBSCRIPT RANGE
                 ERROR;
E8               IF ~(1 <= J-1 <= 20) THEN SUBSCRIPT RANGE
                 ERROR;
E9               IF ~(1 <= 1 <= 2) THEN SUBSCRIPT RANGE
                 ERROR;
10               A(I, J, 1) = A(I, J-1, 1) + I;
11               END;
12            END;
13         K = 2;
14         XK = 2.0;
```

```
15          DO FOR I = 1 TO H;
16             DO FOR J = 1 TO B;
E10               IF ~(1 <= I <=  20) THEN SUBSCRIPT RANGE
                  ERROR;
E11               IF ~(1 <= J <=  20) THEN SUBSCRIPT RANGE
                  ERROR;
E12               IF ~(1 <= K  <=  2) THEN SUBSCRIPT RANGE
                  ERROR;
E13               IF ~(1 <= I <=  20) THEN SUBSCRIPT RANGE
                  ERROR;
E14               IF ~(1 <= J <=  20) THEN SUBSCRIPT RANGE
                  ERROR;
E15               IF ~(1 <= K-1 <= 2) THEN SUBSCRIPT RANGE
                  ERROR;
E16               IF XK = 0 THEN ZERODIVIDE ERROR;
17                A(I, J, K) = A(I, J, K-1) / XK;
18                END;
19             END;
20          END;
21          PROCEDURE LOOKUP (I, J, K);
22          DECLARE INTEGER I, J, K;
23          CASE;
24             K = 1:
E17               IF ~(1 <= I <=  20) THEN SUBSCRIPT RANGE
                  ERROR;
E18               IF ~(1 <= J <=  20) THEN SUBSCRIPT RANGE
                  ERROR;
E19               IF ~(1 <= K  <=  2) THEN SUBSCRIPT RANGE
                  ERROR;
25                PRINT "AREA OF" I, J "RECTANGLE IS"
                  A(I, J, K);
```

```
26              K = 2:
E20                IF ~(1 <= I <=  20) THEN SUBSCRIPT RANGE
                   ERROR;
E21                IF ~(1 <= J <=  20) THEN SUBSCRIPT RANGE
                   ERROR;
E22                IF ~(1 <= K  <=  2) THEN SUBSCRIPT RANGE
                   ERROR;
27                 PRINT "AREA OF" I, J "TRIANGLE IS"
                   A(I, J, K);
28,29           ELSE: PRINT "PARAMETER ERROR:  K = " K;
30              END;
31    END;
32       CALL INIT (20,20);
33       LOOP FOREVER;
34          READ P1, P2, P3;
35          IF P3 = 0 THEN STOP;
36                    ELSE CALL LOOKUP (P1 P2, P3);
37          END;
38       END;
```

FIGURE 2

The program of Figure 1, with probes for zero-divide and subscript range errors inserted. The probes shown are those which would be inserted by a naive dynamic test tool and have statement numbers preceeded by the letter "E".

```
1       PROCEDURE AREAS;
2       DECLARE REAL A(20,20,2), INTEGER P1, P2, P3;
3          PROCEDURE INIT (H, B);
A1         ASSERT NO SIDE-EFFECTS
4          DECLARE INTEGER H, B, I, J, K, REAL XK;
5          IF H > 20 THEN ERROR STOP;
6          IF B > 20 THEN ERROR STOP;
7          DO FOR I = 1 TO H;
8             A(I, 1, 1) = I;
9             DO FOR J = 2 TO B;
10               A(I, J, 1) = A(I, J-1, 1) + I;
A2               ASSERT A(I, J, 1) = I*J;
11               END;
12            END;
13         K = 2;
14         XK = 2.0;
15         DO FOR I = 1 TO H;
16            DO FOR J = 1 TO B;
17               A(I, J, K) = A(I, J, K-1) / XK;
A3               ASSERT A(I, J, 2) = 0.5 * A(I, J, 1);
18               END;
19            END;
20         END;
21         PROCEDURE LOOKUP (I, J, K);
A4         ASSERT NO SIDE-EFFECTS;
22         DECLARE INTEGER I, J, K;
```

```
A5          ASSERT 1 <= I <= 20;
A6          ASSERT 1 <= J <= 20;
23          CASE;
24,25          K = 1: PRINT "AREA OF" I,  J   "RECTANGLE  IS"
               A(I, J, K);
26,27          K = 2: PRINT "AREA OF" I, J "TRIANGLE IS"
               A(I, J, K);
28,29          ELSE: PRINT "PARAMETER ERROR;  K = " K;
30             END;
31          END;
32       CALL INIT (20,20);
33       LOOP FOREVER;
34          READ P1, P2, P3;
35          IF P3 = 0 THEN STOP;
36                    ELSE CALL LOOKUP (P1, P2, P3);
37          END;
38       END;
```

FIGURE 3

The Program of Figure 1 as it might be augmented by assertions capturing the intent of the code.

```
1       PROCEDURE AREAS;
2       DECLARE REAL A(20,20,2), INTEGER P1, P2, P3;
3          PROCEDURE INIT (H, B);
4          DELCARE INTEGER H, B, I, J, K, REAL XK;
P1,1       DECLARE INTEGER HTEMP, BTEMP;
P1,2       HTEMP = H;
P1,3       BTEMP = B;
5          IF H > 20 THEN ERROR STOP;
6          IF B > 20 THEN ERROR STOP;
7          DO FOR I = 1 TO H;
8             A(I, 1, 1) = I;
9             DO FOR J = 2 TO B;
10               A(I, J, 1) = A(I, J-1, 1) + I;
P2,1             IF A(I, J, 1)  ~= I * J THEN PRINT  "ASSER-
                 TION VIOLATION AFTER STATEMENT 10"~
                 A(I, J, 1), I, J;
11               END;
12            END;
13         K = 2;
14         XK = 2.0;
15         DO FOR I = 1 TO H;
16            DO FOR J = 1 TO B;
17               A(I, J, K) = A(I, J, K-1) /XK;
P3,1             IF A(I, J, 2) ~= 0.5  *  A(I,  J,  1)  THEN
                 PRINT  "ASSERTION VIOLATION AFTER STATEMENT
                 17" A(I, J, 2), I, J;
18               END;
19            END;
```

```
P1,4        IF H  ~= HTEMP THEN PRINT "SIDE EFFECTS VIOLATION
            FOR H" H, HTEMP;

P1,5        IF B  ~= BTEMP THEN PRINT "SIDE EFFECTS VIOLATION
            FOR B" B, BTEMP;

20          END;

21          PROCEDURE LOOKUP (I, J, K);

22          DECLARE INTEGER I, J, K;

P4,1        DECLARE INTEGER ITEMP, JTEMP, KTEMP;

P4,2        ITEMP = I;

P4,3        JTEMP = J;

P4,4        KTEMP = K;

P5,1        IF ~(1 <= I <= 20) THEN PRINT  "ASSERTION VIOLA-
            TION AFTER STATEMENT 22" I;

P6,1        IF ~(1 <= J <= 20) THEN PRINT  "ASSERTION VIOLA-
            TION AFTER STATEMENT 22" J;

23          CASE;

24,25          K = 1: PRINT "AREA OF"  I,  J  "RECTANGLE  IS"
               A(I, J, K);

26,27          K = 2: PRINT "AREA OF" I, J "TRIANGLE IS"
               A(I, J, K);

28,29          ELSE: PRINT "PARAMETER ERROR:  K = "K;

30             END;

P4,5        IF I ~= ITEMP THEN PRINT "SIDE EFFECTS  VIOLATION
            FOR I" I, ITEMP;

P4,6        IF J ~= JTEMP THEN PRINT "SIDE EFFECTS  VIOLATION
            FOR J" J, JTEMP;

P4,7        IF K  ~= KTEMP THEN PRINT "SIDE EFFECTS VIOLATION
            FOR K" K, KTEMP;

31          END;

32       CALL INIT (20,20);

33       LOOP FOREVER;
```

```
34          READ P1, P2, P3;
35          IF P3 = 0 THEN STOP;
36                    ELSE CALL LOOKUP (P1, P2, P3);
37          END;
38      END;
```

FIGURE 4

The Program of Figure 1 as it might be augmented by probes inserted by an assertion checking tool in response to the assertions shown in Figure 3. The inserted probes are denoted by line numbers beginning with P. Line number PI,J is attached to the Jth statement generated as a result of assertion AI in Figure 3.

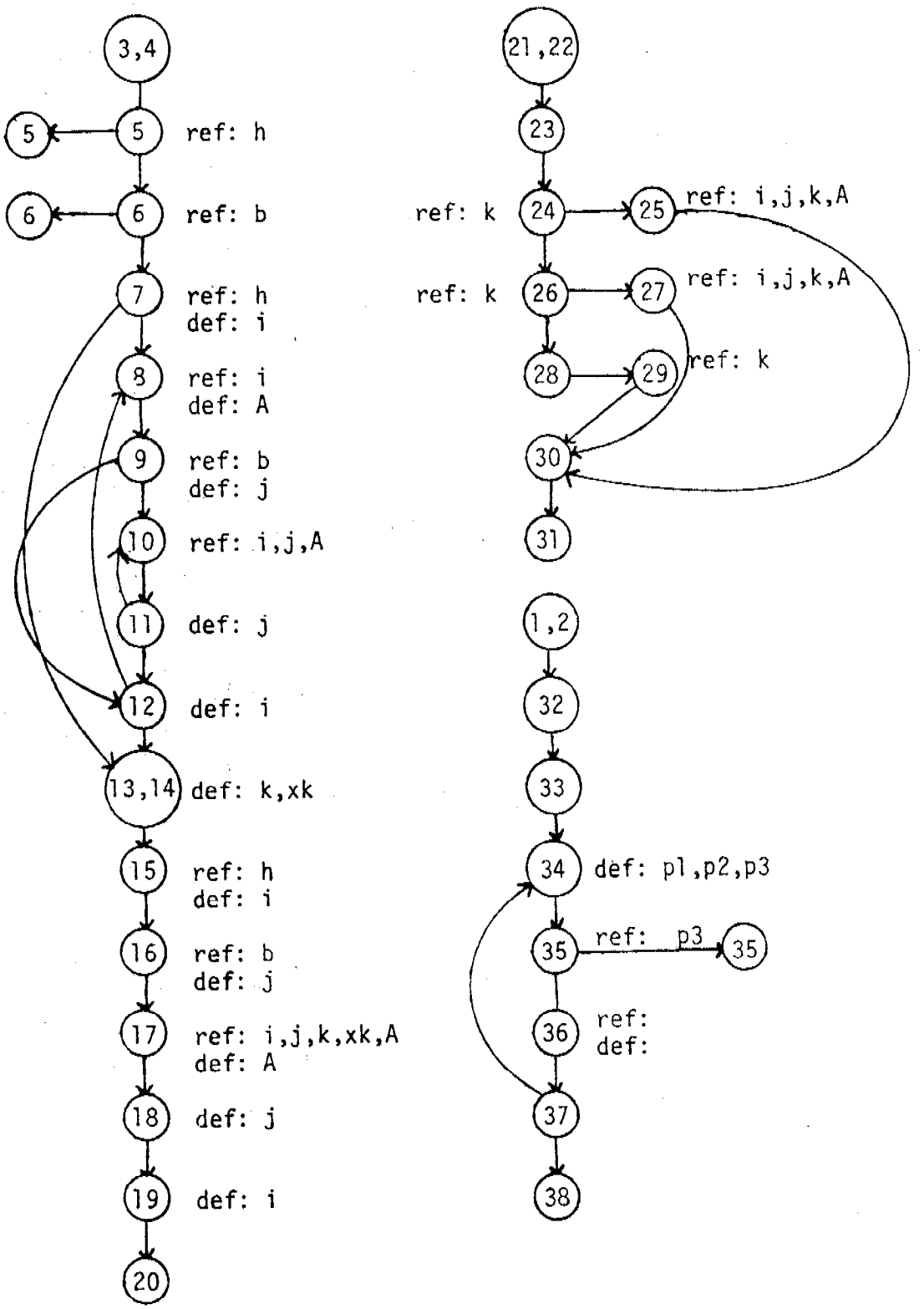


Figure 5

The flowgraphs of the three procedures in the example program of Fig. 1. The nodes are numbered by the statement of Figure 1. For each node, the program variables which are defined there and referenced there are listed. Note that node 36 represents a procedure invocation with variables as arguments. Thus the ref and def lists cannot be completed.

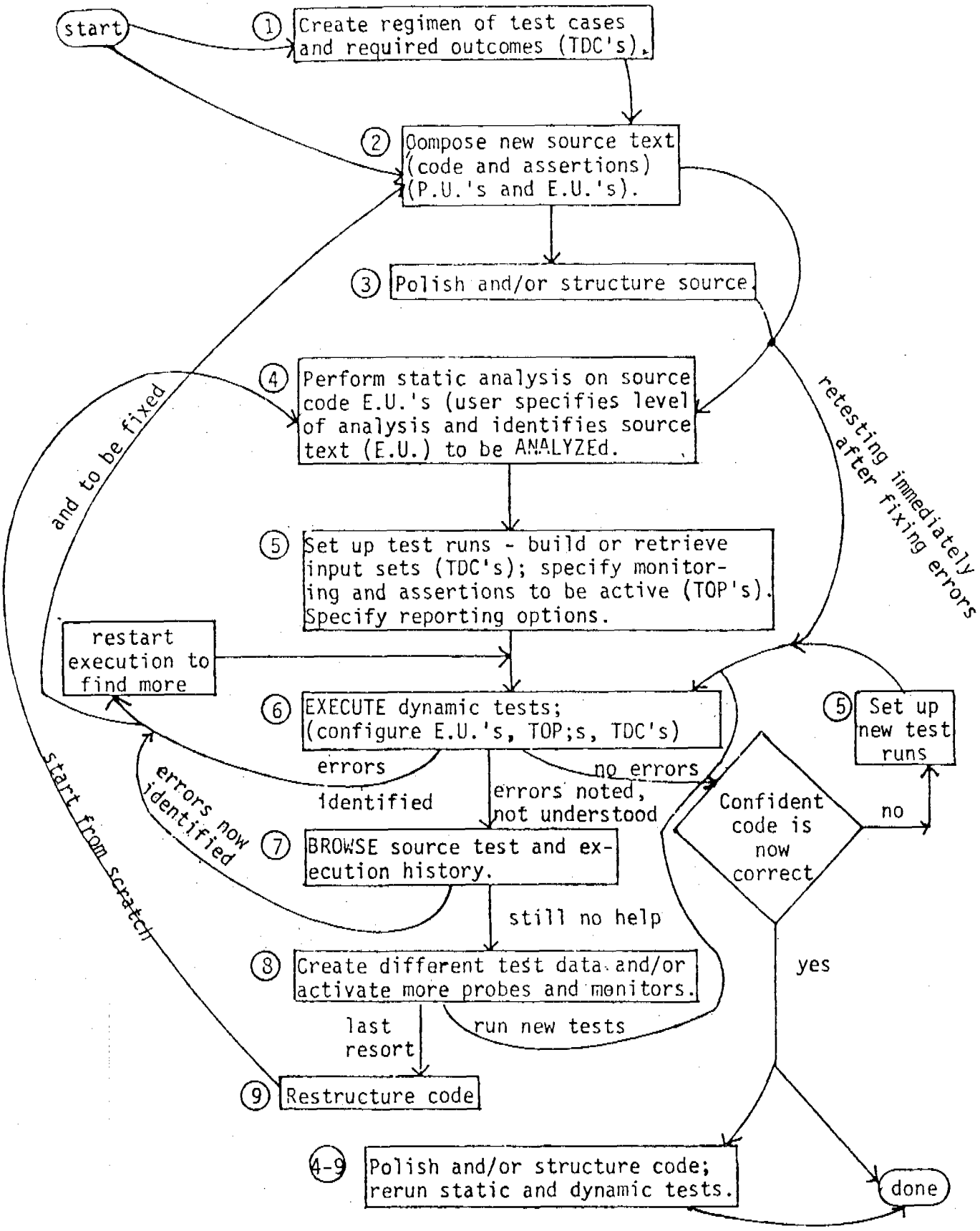

start
1
Create regimen of test cases and required outcomes (TDC's).
2
Compose new source text (code and assertions) (P.U.'s and E.U.'s).
3
Polish and/or structure source.
4
Perform static analysis on source code E.U.'s (user specifies level of analysis and identifies source text (E.U.) to be ANALYZEd.
5
Set up test runs - build or retrieve input sets (TDC's); specify monitoring and assertions to be active (TOP's). Specify reporting options.
restart execution to find more
6
EXECUTE dynamic tests; (configure E.U.'s, TOP;s, TDC's)
errors identified
errors now identified
no errors
errors noted, not understood
7
BROWSE source test and execution history.
still no help
8
Create different test data and/or activate more probes and monitors.
last resort
run new tests
9
Restructure code
4-9
Polish and/or structure code; rerun static and dynamic tests.
Confident code is now correct
no
5
Set up new test runs
yes
done
retesting immediately after fixing errors
and to be fixed
start from scratch

Figure 6

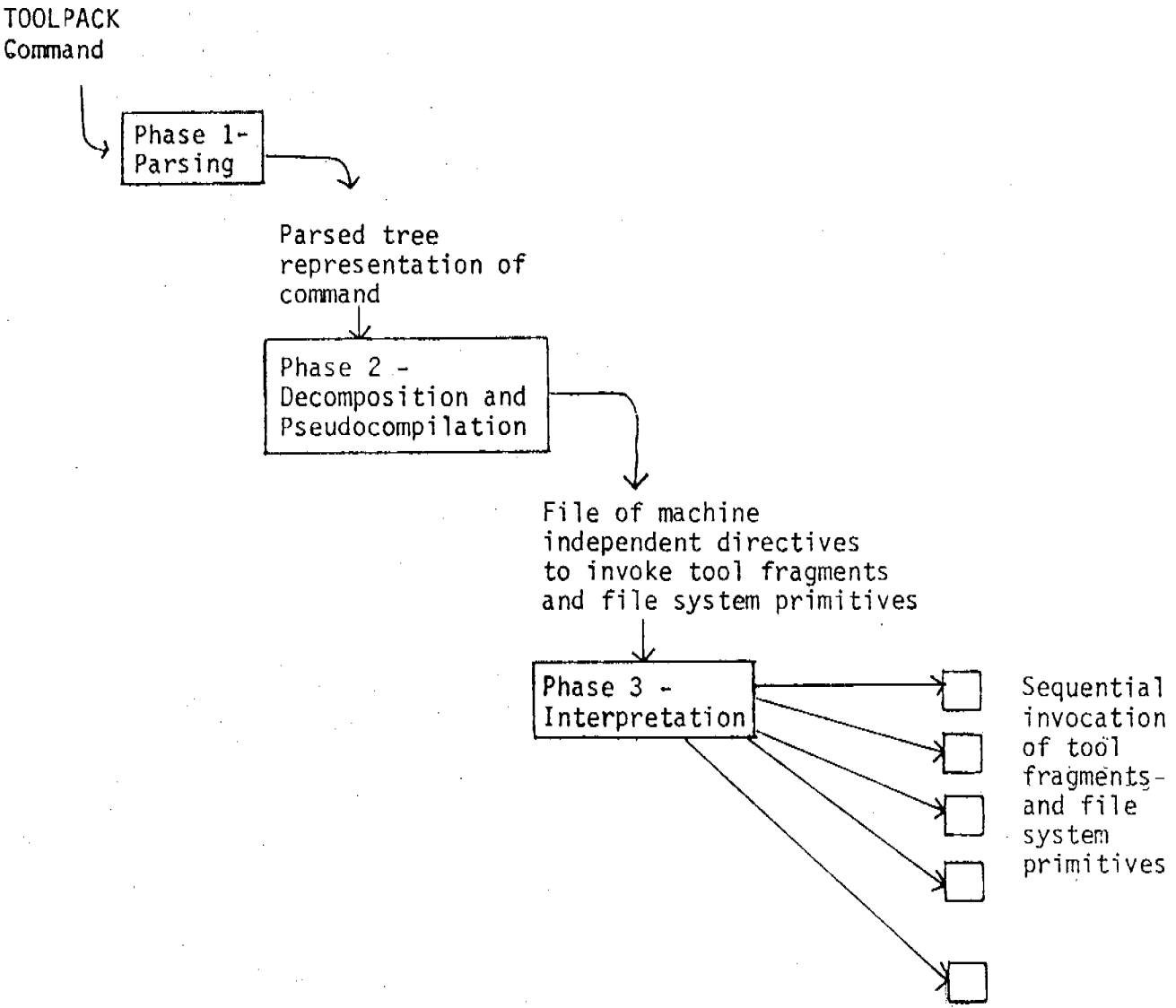


Figure 7. Schematic representation of the TOOLPACK command interpreter

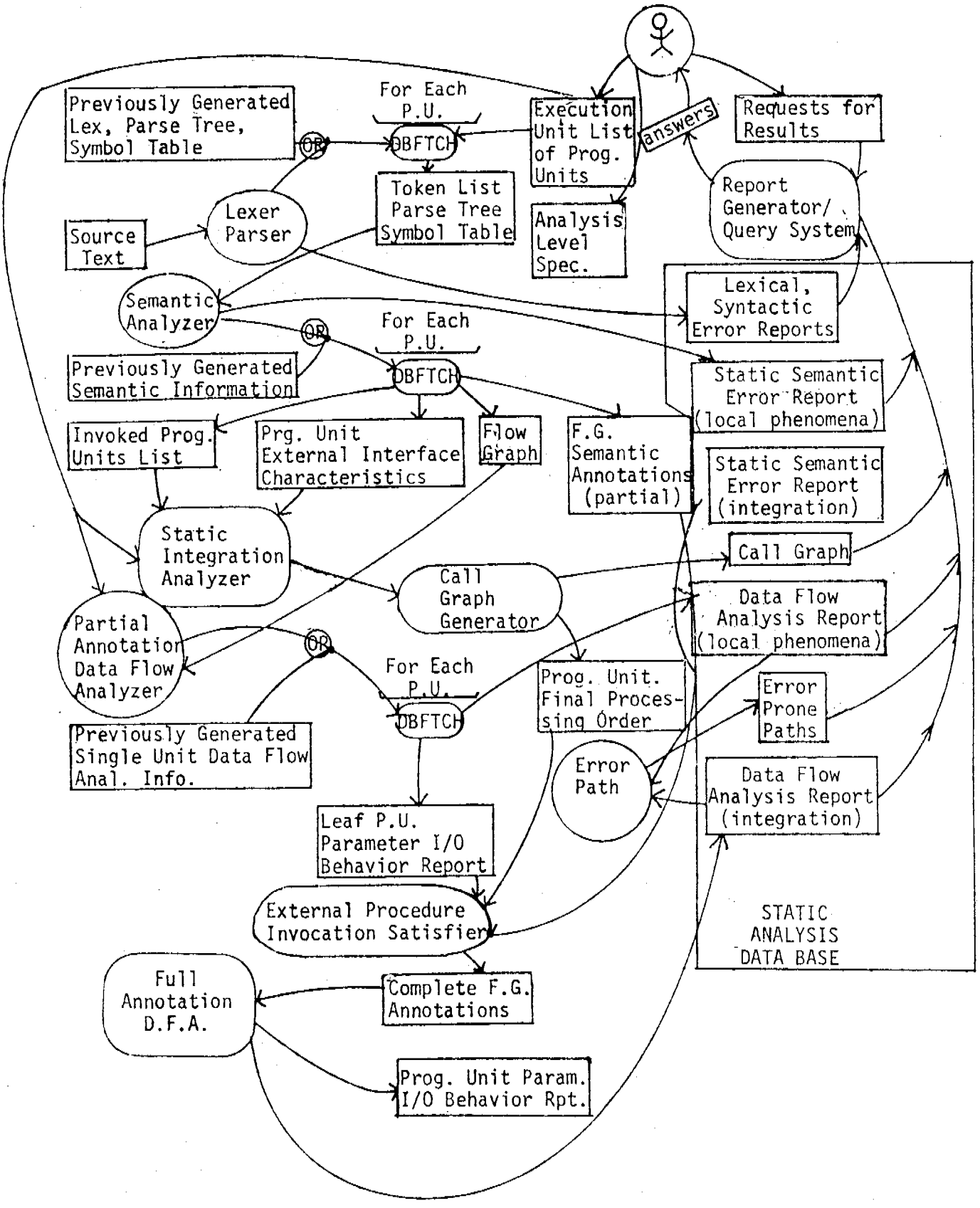


Figure 8: Objects in boxes represent data entities, objects in circles represent functional capabilities.

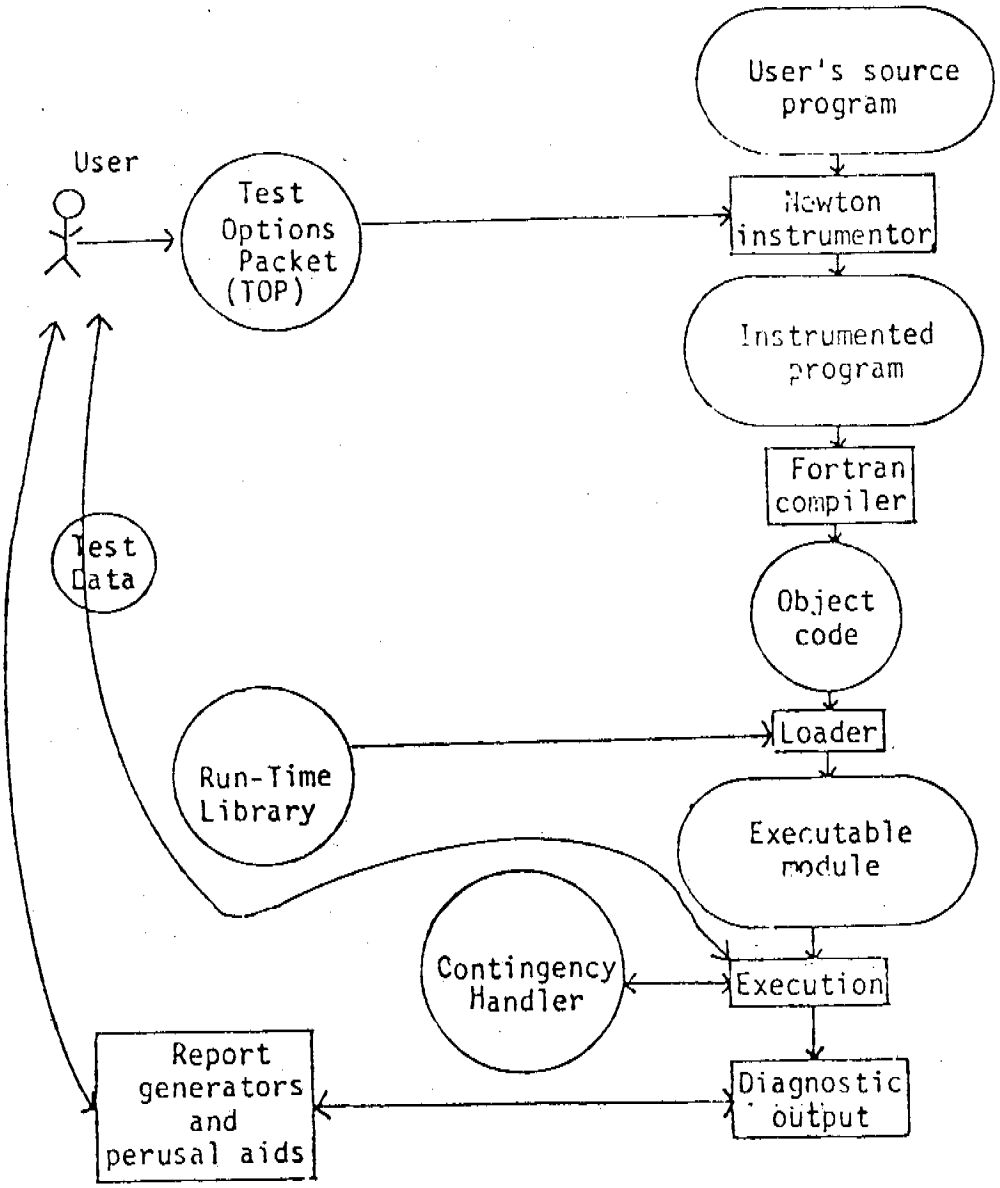


Figure 9. Use of Dynamic Test Tools

A MODIFIED KROENIG-PENNEY MODEL

Francis E. Council, Jr.[1]

ABSTRACT. The Kroenig-Penney model of a crystal has been modified such that it can be related to specific materials. One electron Green's functions have been used to introduce ionization potentials and thus gain specificity. This approach permits polyvalent materials to be studied with applications to the theory of elasticity where change of the internal energy of a crystal is considered as a result of deformations.

I. INTRODUCTION. The binding energy of a solid is considered as resulting from the alteration of the valence electron wave functions and the ion core wave functions. Since the core electrons shield each other, the primary cause of the binding energy is the valence electrons. There are several ways of describing the valence electrons; Frohlich's [1] approach, as well as that of many others not listed, was to divide a crystal of the metal into polyhedrons. Each polyhedron was associated with one atom and it was assumed that there was little, if any, overlap of the wave functions of the electrons of one polyhedron or cell with the electrons of another cell. This approach ignores the long range Coulombic interactions between ion-cores and the valence electrons since the boundary condition is that the normal derivative of the wave function is equal to zero at the boundary of a cell.

If a polyvalent atom is being considered, then the wave function is usually approximated as the result of an average potential; Raimes [2,3] is an example of this. Sachs [4] showed how the original Wigner-Seitz model with only one electron in a cell can be abandoned by assuming a uniform charge distribution. The free electron approximation was the approach that Wigner and Seitz [5,6] and many others have used, whereas the tight binding approximation was used by Mott and Jones [7] in which a valence electron is affected to a much greater degree by the Coulombic potential of the parent ion than by neighbors. Both the tight binding approximation and the free electron approach have an applicability, particularly for the alkali metals. A possible reason that both methods, free electron and tight binding, are useful is that the Fermi surfaces for these metals are approximately spherical such that pressure changes do not markedly affect a Fermi surface although Bardeen [8] has indicated that there is a fifteen percent difference between the experimental and theoretical values of the compressibility of lithium.

In view of the preceding statements, some other approach should be used if the properties of a polyvalent material are to be described properly. Also, Nazieres and Pines [9] have indicated that the kinetic energy and the potential play roughly comparable roles in determining electron behavior, implying that neither the tight binding nor free electron approximation is adequate for a proper description of the valence electrons. One way of combining these two descriptions is to use Bloch functions with a Kroenig-Penney model of the crystal being modified, such that both the Coulombic and repulsive potentials are included.

[1]Formerly with Management Information Systems, Directorate Army Mobility Equipment Research and Development Command, Fort Belvoir, VA. Dr. Council is presently with Vitro Laboratories, Inc., Silver Spring. MD.

II. THE TIGHT BINDING APPROXIMATION WAVE FUNCTION. One way of developing wave functions that are suitable for the tight binding approximation is to use one electron Green's functions. Start first with the wave function in the Schroedinger representation,

$$H\,\langle \mathbf{r}|t\rangle = i\hbar\frac{\partial}{\partial t}\langle \mathbf{r}|t\rangle \tag{1}$$

with an associated Green's function,

$$(i\hbar\frac{\partial}{\partial t} - H \pm i\varepsilon)G_{\pm}(\mathbf{r},\mathbf{r}';t,t') = \delta(\mathbf{r}-\mathbf{r}')\delta(t-t'). \tag{2}$$

The Green's function is a function of two spatial variables, $\mathbf{r}$ and $\mathbf{r}'$; two time variables, t and t'; and a parameter ε, introduced in order that the passage of time may be considered in both a forward and reverse direction. The Green's function takes the wave function at one position and time to some other position and time,

$$\langle \mathbf{r}|t\rangle = \begin{cases} i\hbar\int G_{-}(\mathbf{r},\mathbf{r}';t,t')\langle \mathbf{r}'|t'\rangle d\mathbf{r}' & \text{for } t > t' \\ i\hbar\int G_{+}(\mathbf{r},\mathbf{r}';t,t')\langle \mathbf{r}'|t'\rangle d\mathbf{r}' & \text{for } t < t' \end{cases} \tag{3}$$

If the Hamiltonian is independent of time, then a Fourier transform of the time variable gives

$$G_{\pm}(\mathbf{r},\mathbf{r}';t,t') = \frac{1}{(2\pi\hbar)^{\frac{1}{2}}}\int dE\; G_{\pm}(\mathbf{r},\mathbf{r}';E)\; e^{-\frac{iE}{\hbar}(t-t')} \tag{4}$$

and is a solution of the equation,

$$(E - H \mp i\varepsilon)G\pm(\mathbf{r},\mathbf{r}';t,t') = \delta(\mathbf{r}-\mathbf{r}') \; . \tag{5}$$

Since the Green's function is not translationally invariant, if a Fourier transform is performed on the Green's function with respect to the spatial variables, then both variables must be transformed. Consequently,

$$G_{\pm}(\mathbf{p},\mathbf{p}';E) = \langle \mathbf{p}|G_{\pm}|\mathbf{p}'\rangle = \frac{1}{(2\pi\hbar)^3}\iint d\mathbf{r}\; d\mathbf{r}'\; e^{-i(\mathbf{p}\cdot\mathbf{r} - \mathbf{p}'\cdot\mathbf{r}')/\hbar}G\pm(\mathbf{r},\mathbf{r}';E) \tag{6}$$

Now return to equation (5), pre-and post multiply by $\langle p|$ and $|p'\rangle$ respectively, and insert a unity expression, $|p''\rangle\langle p''|$, with the result

$$\langle \mathbf{p}|E - H \mp i\varepsilon|\mathbf{p}''\rangle\langle \mathbf{p}''|G_{\pm}|\mathbf{p}'\rangle = \delta(\mathbf{p}-\mathbf{p}') \tag{7}$$

The kinetic energy of equation (7) is expressed as

$$\langle \mathbf{p}|\hbar^2\nabla^2/2m|\mathbf{p}'\rangle = \frac{p^2}{2m}\,\delta(\mathbf{p}-\mathbf{p}') \tag{8}$$

While if a tight binding approximation is being considered, then the potential can be introduced as a Dirac delta function potential. By means of this technique, the potential energy for each of the valence electrons can be expressed in terms of the various ionization potentials. If the Dirac delta function potential is

$$V = -\eta(\mathbf{r}')$$

then the ionization potential is introduced by letting η be equal to the ionization potential. The Fourier transform of this potential is

$$V(\mathbf{p}') = -\frac{\eta}{(2\pi\hbar)^{3}/_{2}} \tag{10}$$

and

$$\langle \mathbf{p}|V_{(\mathbf{p})}|\mathbf{p}'\rangle = -\frac{\eta}{(2\pi\hbar)^{3}/_{2}}\,\delta(\mathbf{p}-\mathbf{p}') \tag{11}$$

A way of further development is to use perturbation theory by modifying an equation by Harrison [10],

$$G(\mathbf{p},\mathbf{p}') = G^{o}(\mathbf{p})\delta(\mathbf{p}-\mathbf{p}')+G^{o}(\mathbf{p})\langle \mathbf{p}|V|\mathbf{p}'\rangle G^{o}(\mathbf{p}')$$
$$+\int G^{o}(\mathbf{p})\langle \mathbf{p}|V|\mathbf{p}_1\rangle dp\; G^{o}(\mathbf{p}_1)\langle \mathbf{p}_1|V|\mathbf{p}'\rangle G^{o}(\mathbf{p}')+\ldots \tag{12}$$

with the second wave number index being suppressed since the indices are always the same. The zero order Green's function, $G^{o}(\mathbf{p})$, is obtained from equation (5) by letting the potential energy of the Hamiltonian be equal to zero. If equation (11) is considered, evaluation of the second and third terms of equation (12) gives

$$-G^{o}(\mathbf{p})G^{o}(\mathbf{p}')\eta\delta(\mathbf{p}-\mathbf{p}') \tag{13}$$

and

$$G^{o}(\mathbf{p})G^{o}(\mathbf{p})G\,(\mathbf{p}')\eta^2\,\delta(\mathbf{p}-\mathbf{p}'). \tag{14}$$

It is obvious from the preceding that a series is being developed such that

$$G(\mathbf{p},\mathbf{p}') = \frac{(p^2/2m)\,\delta(\mathbf{p}-\mathbf{p}')}{1+\eta(p')^2/2m\,(2\pi\hbar)^{3}/_{2}} \tag{15}$$

With a suitable Green's function having been developed, then the individual wave functions in **p** space are obtained as

$$\chi(\mathbf{p}) = \int G(\mathbf{p},\mathbf{p}')\ \chi(\mathbf{p}')\ d\mathbf{p}' \tag{16}$$

The wave function in **r** space is obtained by an inverse Fourier transform of equation (16)

$$\Psi_{TB} = \frac{1}{(2\pi\hbar)^{3}/2}\int e^{i\mathbf{p}\cdot\mathbf{r}/\hbar}\chi_{TB}(\mathbf{p})d\mathbf{p} \tag{17}$$

with the subscript $_{TB}$ being used to indicate that the tight binding approximation has been used. The momentum wave functions that are used here are semi-empirical wave functions developed by Duncanson and Carlson [11, 12] based on initial work by Slater [13]. Slater developed some approximate analytical expressions which correct for the shielding effect of the core electrons of the nucleus.

III. THE FREE ELECTRON APPROXIMATION. If an electron is no longer primarily influenced by its parent ion, then its behavior is described by the quasi-free electron approximation. The influence of a periodic potential of a crystal causes the electron's motion to be no longer perfectly free. This potential energy affecting the electron can be considered as a combination of Coulombic and repulsive potential as a function of position; in order to simplify computations it will be considered as a constant. The effect of temperature could also be included at this time; again in order to simplify computations, this effect will be ignored.

These quasi-free electrons exist in a potential well of energy. The electrons fill up all the energy states in the well to a level above the bottom of the well with the potential depth being related to the cut off energy as

$$V_o = E_o + e\ \psi \tag{18}$$

The expression of e is the minimum energy to remove an electron from a metal and is usually considered as the work function. Consequently, if the work function is expressed in electron volts and the magnitude of the depth of the potential well for a particular valence electron is its ionization potential, then the cut off momentum value is

$$p_o = (2m(V_i - e\ \psi))^{\frac{1}{2}} \tag{19}$$

The expected value of the magnitude of the momentum is

$$|\bar{\mathbf{p}}_i| = \frac{\int_o^{p_o} p^2 dp}{\int_o^{p_o} p\,dp} = \frac{2}{3}\,(2m(V_i - e\ \psi))^{\frac{1}{2}} \tag{20}$$

such that a wave function related to an average value of the momentum is

$$\Psi_{QFE} = \frac{1}{(2\pi)^{3/2}} e^{i\bar{\mathbf{p}}_i \cdot \mathbf{r}/\hbar} \tag{21}$$

with the subscript $_{QFE}$ indicating that the quasi free approximation is being used.

The Block Wave Functions. Two wave functions have been developed, one suitable for the quasi-free approximation and the other for the tight binding approximation. These can be combined into a Bloch wave function as

$$= \sum_j \Psi_{QFE}(\mathbf{r})\, \Psi_{TB}(\mathbf{r} - \mathbf{r}_j) \tag{22}$$

with $\Psi_{TB}(\mathbf{r} - \mathbf{r}_j)$ being related to the periodic potential. The wave function for a sum of wave functions as would be reflected by the different ionization potentials is

$$\Psi = \frac{1}{\sqrt{j}} \sum \Psi_j \tag{23}$$

with the j being a subscript associated with the different ionization potentials. For example, for a divalent material with ionization potentials of V_1 and V_2 for the valence electrons,

$$= \frac{1}{\sqrt{2}} \sum_j e^{i(2m(V_i - e\psi))^{\frac{1}{2}} r} \int \frac{e^{i\mathbf{p}\cdot(\mathbf{r} - \mathbf{r}_j)} \chi_1(\mathbf{p}')(p^2/2m)\delta(\mathbf{p} - \mathbf{p}')d\mathbf{p}'}{1 + V_1((p')^2/2m)(2\pi\hbar)^{3/2}} \tag{24}$$

$$+ \frac{1}{\sqrt{2}} \sum_j e^{i(2m(V_2 - e\psi))^{\frac{1}{2}} r} \int \frac{e^{i\mathbf{p}\cdot(\mathbf{r} - \mathbf{r}_j)} \chi_2(\mathbf{p}')(p^2/2m)\delta(\mathbf{p} - \mathbf{p}')d\mathbf{p}'}{1 + V_2((p')^2/2m)(2\pi\hbar)^{3/2}}$$

IV. CONCLUSIONS. A prescription by which a Kroenig-Penney model of a crystal can be modified for specific materials has been obtained. This development can be modified for greater precision if the average potential of a crystal is given as a function of position or if the effect of temperature is included. The primary reason for this derivation is to facilitate calculations in which the change of internal energy of a crystal is a result of deformations.

REFERENCES

[1] Frohlich, H., "A Quantum Mechanical Discussion of the Cohesive Forces and Thermal Expansion Coefficients of the Alkali Metals", Proceedings of the Royal Society of London, A, Vol. 158 (1937), 97.

[2] Raimes, S., "The Cohesive Energy of Metallic Magnesium", Philosophical Magazine, Vol. 41 (1950), 568.

[3] Raimes, S., "A Calculation of the Cohesive Energies and Pressure/Volume Relations of the Divalent Metals", Philosophical Magazine, Vol. 43 (1952), 327.

[4] Sachs, Mendel, Solid State Theory, McGraw Hill, New York, N.Y., 1963, p. 297.

[5] Wigner, E. P. and Seitz, F., "On the Constitution of Metallic Sodium", Physical Review, Vol. 43 (1933), 804.

[6] Wigner, E. P. and Seitz, F., "On the Constitution of Metallic Sodium", Physical Review, Vol. 46 (1934), 509.

[7] Mott, M. F. and Jones, H., "The Theory of the Properties of Metals and Alloys", Chap. II, Oxford University Press, Fier Lawn, N.J., 1936.

[8] Bardeen, J., "Compressibilities of the Alkali Metals", Journal of Chemical Physics, Volume 6 (1938), 372.

[9] Nazieres, P. and Pines, D., "Correlation Energy of a Free Electron Gas", Physical Review, Vol. III, No. 2, (1958), 442.

[10] Harrison, Walter A., Solid State Theory, McGraw Hill, New York, N.Y., (1970), 220-222.

[11] Duncanson, W. E. and Carlson, C. A., "Atomic Wave Functions for Ground States of Elements Li to Ne", Proceedings of Royal Society of Edinburgh, 62A, 37 (1934).

[12] Duncanson, W. E. and Carlson, C. A., Proceedings, Physical Society (London), 57, 190 (1945).

[13] Slater, J. C. "Atomic Shielding Constants," Physical Review, Vol. 36, (1930), 57.

COMPUTATION OF MATRIX CHAIN PRODUCTS

T. C. Hu and M. T. Shing
University of California, San Diego
La Jolla, CA 92093

ABSTRACT. This paper considers the computation of matrix chain products of the form $M_1 \times M_2 \times \cdots \times M_{n-1}$. If the matrices are of different dimensions, the order in which the matrices are computed affects the number of operations. An optimum order is an order which minimizes the total number of operations. We present some theorems about an optimum order of computing the matrices. Based on these theorems, algorithms for finding an optimum order are developed.

1. INTRODUCTION. Consider the evaluation of the product of n-1 matrices

$$M = M_1 \times M_2 \times \cdots \times M_{n-1} \tag{1}$$

where M_i is a $w_i \times w_{i+1}$ matrix. Since matrix multiplication satisfies the associative law, the final result M in (1) is the same for all orders of multiplying the matrices. However, the order of multiplication greatly affects the total number of operations to evaluate M. The problem is to find an optimum order of multiplying the matrices such that the total number of operations is minimized. Here, we assume that the number of operations to multiply a $p \times q$ matrix by a $q \times r$ matrix is pqr.

In refs. 1 and 6, a dynamic programming algorithm is used to find an optimum order. The algorithm needs $O(n^3)$ time and $O(n^2)$ space. In ref. 2, Chandra proposed a heuristic algorithm to find an order of computation which requires no more than $2T_o$ operations where T_o is the total number of operations to evaluate (1) in an optimum order. This heuristic algorithm needs only O(n) time. Chin (ref. 3) proposed an improved heuristic algorithm to give an order of computation which requires no more than $1.25\,T_o$. This improved heuristic algorithm also needs only O(n) time.

In this paper we first transform the matrix chain product problem into a problem in graph theory - the problem of partitioning a convex polygon into non-intersecting triangles (see ref. 8), then we state several theorems about the optimum partitioning problem. Based on these theorems, algorithms for finding optimum partitions are developed.

2. PARTITIONING A CONVEX POLYGON. Given an n-sided convex polygon, such as the hexagon shown in Fig. 1, the number of ways to partition the polygon into (n-2) triangles by non-intersecting diagonals is the Catalan numbers (see, for example, ref. 7). Thus, there are two ways to partition a convex quadrilateral,

five ways to partition a convex pentagon, and fourteen ways to partition a convex hexagon.

Let every vertex V_i of the convex polygon have a weight w_i. We can define the cost of a given partition as follows: The cost of a triangle is the product of the weights of the three vertices, and the cost of partitioning a polygon is the sum of the costs of all its triangles. For example, the cost of the partition of the hexagon in Fig. 1 is

$$w_1w_2w_3 + w_1w_3w_6 + w_3w_4w_6 + w_4w_5w_6 \tag{2}$$

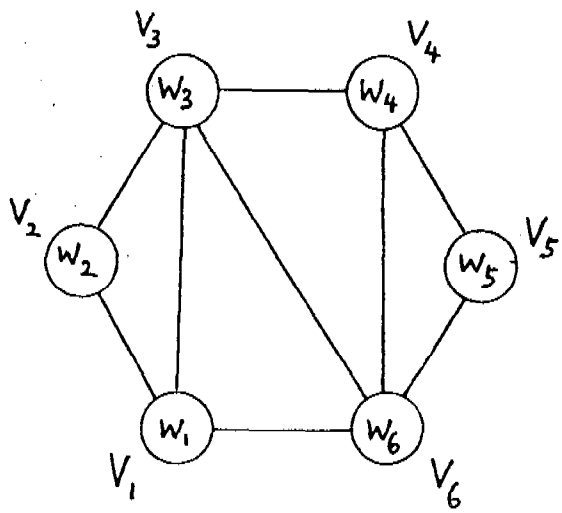


Fig. 1

If we erase the diagonal from V_3 to V_6 and replace it by the diagonal from V_1 to V_4, then the cost of the new partition will be

$$w_1w_2w_3 + w_1w_3w_4 + w_1w_4w_6 + w_4w_5w_6 \ . \tag{3}$$

We will prove that an order of multiplying (n-1) matrices corresponds to a partition of a convex polygon with n sides. The cost of the partition is the total number of operations needed in multiplying the matrices. For brevity, we shall use n-gon to mean a convex polygon with n sides, and the partition of n-gon to mean the partitioning of an n-gon into (n-2) non-intersecting triangles.

For any n-gon, one side of the n-gon will be considered to be its base, and will usually be drawn horizontally at the bottom such as the side V_1-V_6 in Fig. 1.

This side will be called the base, all other sides are considered in a clockwise way. Thus, V_1-V_2 is the first side, V_2-V_3 the second side,..., and V_5-V_6 the fifth side.

The first side represents the first matrix in the matrix chain and the base represents the final result M in (1). The dimensions of a matrix are the two weights associated with the two end vertices of the side. Since the adjacent matrices are compatible, the dimensions $w_1 \times w_2$, $w_2 \times w_3, \ldots, w_{n-1} \times w_n$ can be written inside the vertices as $w_1, w_2, \ldots, w_n$. The diagonals are the partial products. A partition of an n-gon corresponds to an alphabetic tree of n-1 leaves or the parenthesis problem of n-1 symbols (see, for example, ref. 5). It is easy to see the one-to-one correspondence between the multiplication of n-1 matrices to either the alphabetic tree or the parenthesis problem of n-1 symbols. We state this fact as Lemma 1.

Lemma 1. Any order of multiplying n-1 matrices corresponds to a partition of an n-gon. ■

We can also establish the correspondence between the matrix-chain products and the partitions of a convex polygon directly. See ref. 8 for more details.

Lemma 2. The minimum number of operations to evaluate the following matrix chain products are identical.

$$M_1 \times M_2 \times \cdots \times M_{n-2} \times M_{n-1}$$

$$M_n \times M_1 \times \cdots \times M_{n-3} \times M_{n-2}$$

$$\vdots$$

$$M_2 \times M_3 \times \cdots \times M_{n-1} \times M_n$$

where M_i has dimension $w_i \times w_{i+1}$ and $w_{n+1} \equiv w_1$. Note that in the first matrix chain, the resulting matrix is of dimension w_1 by w_n. In the last matrix chain, the resulting matrix is of dimension w_2 by w_1. But in all cases, the total number of operations in the optimum orders of multiplication is the same.

Proof. The cyclic permutations of the n-1 matrices all correspond to the same n-gon and thus have the same optimum partitions. ■
(This Lemma was obtained independently in ref. 4 with a long proof.)

From now on, we shall concentrate only on the partitioning problem.

The diagonals inside the polygon are called arcs. Thus, every partition consists of n-2 triangles formed by n-3 arcs and n sides.

In a partition of an n-gon, the degree of a vertex is the number of arcs incident on the vertex plus two (since there are two sides incident on every vertex).

Lemma 3. In any partition of an n-gon, $n \geq 4$, there are at least two triangles, each has a vertex of degree two. (For example, in Fig. 1, the triangle $V_1V_2V_3$ has vertex V_2 with degree 2 and the triangle $V_4V_5V_6$ has vertex V_5 with degree 2.) ■

Lemma 4. Let P and P′ both be n-gons where the corresponding weights of the vertices satisfy $w_i \leq w_i'$, then the cost of an optimum partition of P is less than or equal to the cost of an optimum partition of P′. ■

If we use $C(w_1, w_2, w_3, \ldots, w_k)$ to mean the minimum cost of partitioning the k-gon with weights w_i optimally, Lemma 4 can be stated as

$$C(w_1, w_2, \ldots, w_k) \leq C(w_1', w_2', \ldots, w_k') \text{ if } w_i \leq w_i'$$

We say that two vertices are connected in an optimum partition if the two vertices are connected by an arc or if the two vertices are adjacent to the same side.

In the rest of the paper, we shall use $V_1, V_2, \ldots, V_n$ to denote vertices which are ordered according to their weights, i.e., $w_1 \leq w_2 \leq \cdots \leq w_n$. To facilitate the presentation, we introduce a tie-breaking rule for vertices of equal weights.

If there are two or more vertices with weights equal to the smallest weight w_1, we can arbitrarily choose one of these vertices to be the vertex V_1. Once the vertex V_1 is chosen, further ties in equal weights are resolved by regarding the vertex which is closer to V_1 in the clockwise direction to be of less weight. With this tie-breaking rule, we can unambiguously label the vertices $V_1, V_2, \ldots, V_n$ for each choice of V_1.

We shall use $V_a, V_b, \ldots$ to denote vertices which are unordered in weights, and T_{ijk} to denote the product of the weights of any three vertices V_i, V_j and V_k.

3. SOME CHARACTERISTICS OF THE OPTIMUM PARTITIONS. First, let us consider the polygons where there are two or more vertices with equal weights w_1.

Lemma 5. For every choice of $V_1, V_2, \ldots$ (as prescribed), if the weights of the vertices satisfy the condition

$$w_1 = w_2 < w_3 \leq \cdots \leq w_n ,$$

then V_1-V_2 exists in every optimum partition of the n-gon.

Proof. The lemma is true if V_1-V_2 is a side of the n-gon. Hence, we can assume that V_1, V_2 are not adjacent to the same side of the n-gon.

The proof is by induction on the size of the n-gon. The lemma is true for a triangle and a quadrilateral. Assume that the lemma is true for all k-gons $(3 \leq k \leq n-1)$ and consider the optimum partitions of an n-gon.

By Lemma 3, we know that there are at least two vertices with degree two in each optimum partition of the n-gon. We have the following two cases.

(i) In an optimum partition of an n-gon, one of the vertices with degree two, say V_i, has weights larger than w_1. In this case, we can form an (n-1)-gon by removing V_i with its two sides. By induction assumption, V_1-V_2 is present in every optimum partition of the (n-1)-gon.

(ii) Consider the complementary case of (i), i.e. all vertices with degree two have weights equal to w_1 in an optimum partition of the n-gon. In other words, V_1 and V_2 are the only two vertices with degree two in that optimum partition, as shown symbolically in Fig. 2a. Note that every arc in the optimum partition must dissect the n-gon into two subpolygons in such a way that V_1, V_2 can never appear in any subpolygon together, else there will be more than two vertices with degree two in the optimum partition. In Fig. 2b, we show a partition of the n-gon in which V_1 and V_2 are connected. Let us denote the n-2 triangles in Fig. 2a by $P_1, P_2, \ldots, P_{n-2}$. Except P_1 and P_{n-2}, all the other n-4 triangles are made up of one side and two arcs each. For each of these n-4 triangles, we can find a unique triangle in Fig. 2b such that they both consist of the same side. We use P_i' to denote the image of P_i in Fig. 2b. The only two triangles left unmatched in Fig. 2b are $V_1V_aV_2$ and $V_1V_2V_i$ and they are the images of P_1 and P_{n-2}, respectively. Let the cost of P_i be C_i and the cost of P_i' be C_i'. Since $C_i' \leq C_i$ for $1 \leq i \leq n-2$, the partition in Fig. 2b is cheaper than that in Fig. 2a and we have a contradiction. ■

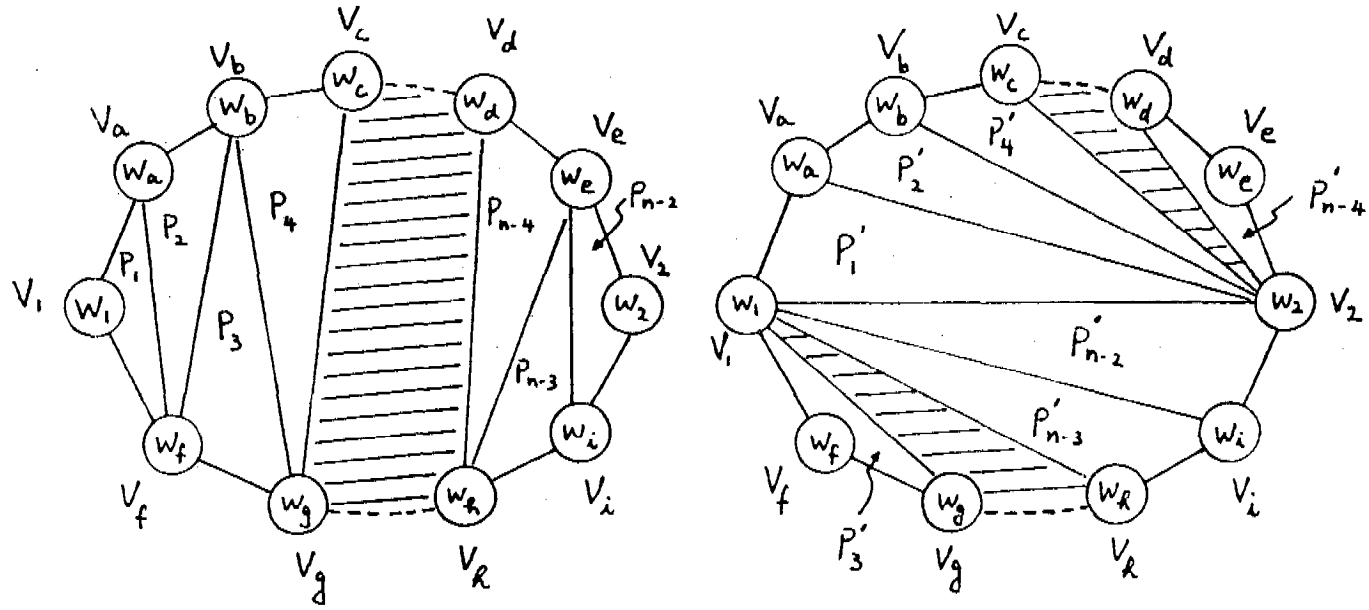


Fig. 2

Theorem 1. For every choice of $V_1, V_2, \ldots$ (as prescribed), if the weights of the vertices satisfy the condition

$$w_1 = w_2 < w_3 \leq w_4 \leq \cdots \leq w_n ,$$

then every optimum partition of the n-gon must contain a triangle $V_1V_2V_p$ for some vertex V_p with weight equal to w_3. Note that if $w_1 = w_2 < w_3 < w_4 \leq \cdots \leq w_n$, then every optimum partition must contain the triangle $V_1V_2V_3$ since there is a unique choice of V_3.

Proof. Similar to Lemma 5, we can prove this theorem by induction on the size of the n-gon. The theorem is true for any triangle or quadrilateral satisfying the above condition. Assume the theorem is true for all k-gons ($3 \leq k \leq n-1$) and consider the optimum partitions of an n-gon.

From Lemma 5, we know that V_1, V_2 are always connected in every optimum partition. Hence, without loss of generality, we can assume V_1, V_2 to be adjacent to the same side of the n-gon. Again, we have the following two cases.

(i) In an optimum partition, one of the vertices with degree two, say V_i, has weight larger than w_3. In this case, we can remove V_i with its sides and form an (n-1)-gon. By induction assumption, every optimum partition of the (n-1)-gon contains a triangle $V_1V_2V_p$ where $w_p = w_3$.

(ii) Consider the complementary case if (i), in an optimum partition of the n-gon, all vertices with degree two have weights less than or equal to w_3. Since V_1-V_2 is a side of the n-gon, for $n \geq 4$, either V_1 or V_2 (but not both) can have degree two. We have the following two subcases:

(a) If there are more than one vertex whose weight equals w_3, we can form an (n-1)-gon by removing one of those degree two vertices whose weight equals w_3. By induction assumption, every optimum partition of the (n-1)-gon contains a triangle $V_1V_2V_p$ for some V_p with $w_p = w_3$.

(b) There exists only one vertex of weight w_3. In this case, there must be only two vertices with degree two in the optimum partition of the n-gon. These two vertices are V_3 and either V_1 or V_2. Without loss of generality, we can assume V_1 has degree 2. The situation is shown symbolically in Fig. 3a. Again, every arc in the optimum partition must dissect the n-gon in such a way that V_1 and V_3 can never appear in any subpolygon together. In Fig. 3b, we show a partition containing the triangle $V_1V_2V_3$. Using arguments similar to those in the proof of Lemma 5, we can show that the partition in Fig. 3b is cheaper and we obtain a contradiction. ■

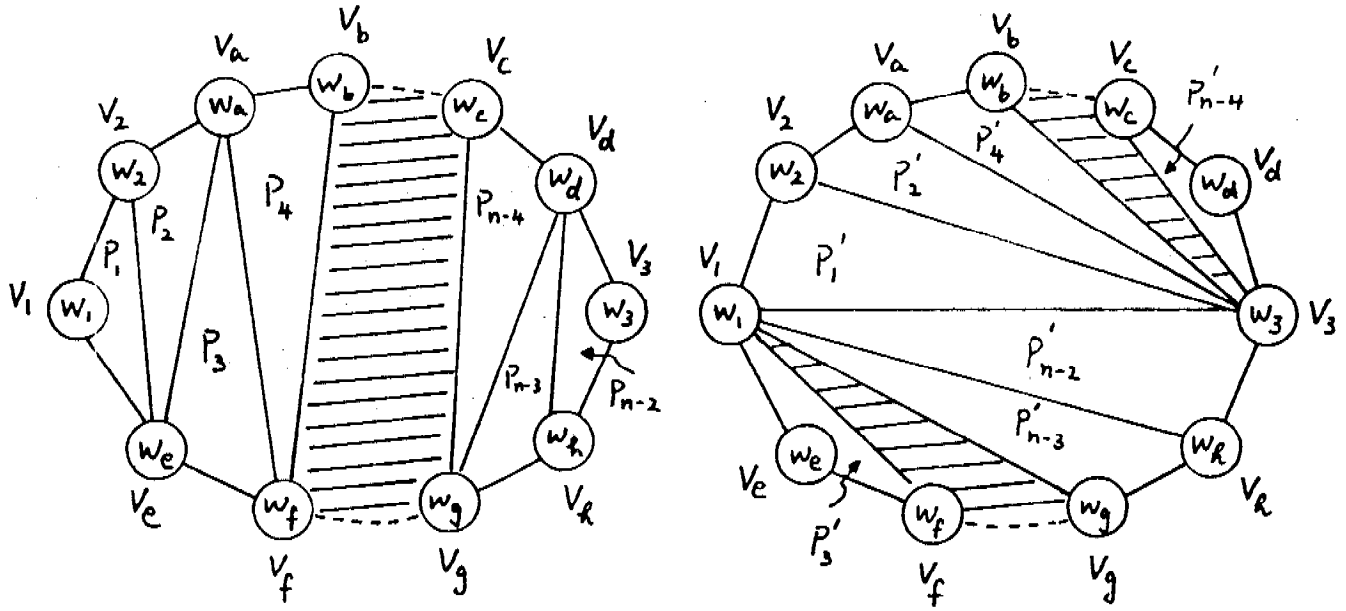


Fig. 3

Theorem 2. For every choice of $V_1, V_2, \ldots$ (as prescribed), if the weights of the vertices of the n-gon satisfy the following condition,

$$w_1 = w_2 = \cdots = w_k < w_{k+1} \leq \cdots \leq w_n$$

for some k, $3 \leq k \leq n$, then every optimum partition of the n-gon contains the k-gon $V_1\text{-}V_2\text{-}\cdots\text{-}V_k$.

Proof. The proof is by induction on the size of the n-gon. The theorem is true for any triangle and quadrilateral. Suppose the theorem is true for all polygons with (n-1) sides or less and consider the optimum partitions of an n-gon.

From Lemma 3, there exist at least two vertices having degree two in every optimum partition. We have the following two cases.

(i) In an optimum partition of the n-gon, one of the vertices with degree two, say V_i, has weight larger than w_1. In this case, we can remove the vertex V_i with its two sides and obtain an (n-1)-gon. By induction assumption, every optimum partition of the (n-1)-gon contains the k-gon $V_1\text{-}V_2\text{-}\cdots\text{-}V_k$.

(ii) Consider the complementary case of (i), i.e., all the vertices with degree two have weights equal to w_1 in an optimum partition. Let two of these vertices be V_i, V_j. We have the following two subcases:

(a) $k > 3$. We first form an (n-1)-gon by removing V_i and its two sides. There are (k-1) vertices with weights equal to w_1 in the (n-1)-gon. By induction assumption, every optimum partition of the (n-1)-gon contains the (k-1)-gon which includes V_j as one of its vertices. Since V_j has degree two in the optimum partition, its two neighboring vertices, say V_x and V_y, must also have weights equal to w_1 and the arc V_x-V_y exists in the optimum partition (Fig. 4). Similarly, we can remove the vertex V_j with its two sides V_j-V_x and V_j-V_y and form an (n-1)-gon. By induction assumption, every optimum partition of the (n-1)-gon contains the (k-1)-gon formed by the (k-1) vertices with weights equal to w_1 in the (n-1)-gon and V_i is one of the vertices in the (k-1)-gon. Now, by pasting the triangle $V_xV_jV_y$ and the (k-1)-gon together, we form a k-gon which includes all the vertices with weight equal to w_1 in the n-gon and this k-gon is contained in the optimum partition of the n-gon.

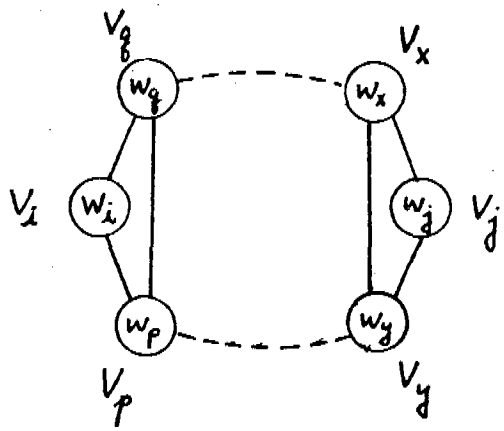


Fig. 4

(b) $k = 3$. In this case, we have $w_1 = w_2 = w_3 < w_4 \leq \cdots \leq w_n$. Without loss of generality, we can assume V_1 and V_2 both have degree two in an optimum partition. Again, we can form an (n-1)-gon by removing V_1 and its two sides. By Lemma 5, V_2 and V_3 are connected in every optimum partition of the (n-1)-gon. Since V_2 has degree two, V_2-V_3 must be a side of the n-gon. Next, we can remove V_2 with its two sides and form an (n-1)-gon. By Lemma 5, V_1, V_3 are connected by a side of the n-gon. The situation is shown in Fig. 5a. Then, the partition in Fig. 5b is cheaper because

$$T_{123} + T_{12y} \leq T_{13x} + T_{23y} \ .$$

and $$C(w_1, w_x, \ldots, w_y) \leq C(w_3, w_x, \ldots, w_y) \ . \ \blacksquare$$

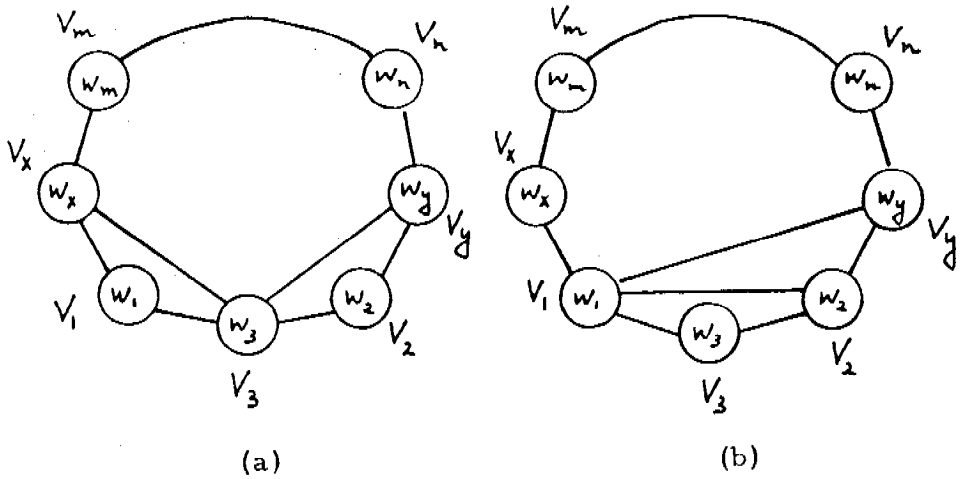


Fig. 5

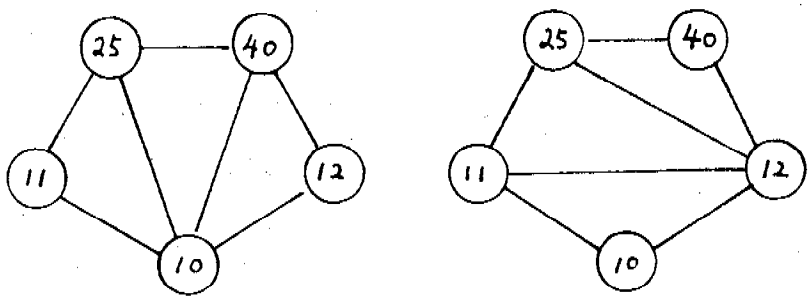
Now, whenever we have three or more vertices with weights equal to w_1 in the n-gon, we can decompose the n-gon into subpolygons by forming the k-gon in Theorem 2. The partition of the k-gon can be arbitrary, since all vertices of the k-gon are of equal weight. For any subpolygon with two vertices of weights equal to w_1, we can always apply Theorem 1 and decompose the subpolygon into smaller subpolygons. Hence, we have only to consider the polygons with a unique choice of V_1, i.e., each polygon has only one vertex with weight equal to w_1.

Theorem 3. For every choice of $V_1, V_2, \ldots$ (as prescribed), if the weights of the vertices satisfy the condition

$$w_1 < w_2 \leq w_3 \leq \cdots \leq w_n \ ,$$

then V_1-V_2 and V_1-V_3 exist in every optimum partition of the n-gon.

Proof. We can again use Lemma 3 and prove Theorem 3 by the induction on the size of the n-gon. ■



(a) A stable partition (b) An optimum partition

Fig. 6

In any partition of an n-gon, every arc dissects a unique quadrilateral. Let V_x, V_y, V_z, V_w be the four vertices of an inscribed quadrilateral and V_x-V_z be the arc which dissects the quadrilateral. We define V_x-V_z to be a vertical arc if (6) or (7) is satisifed.

$$\min(w_x, w_z) < \min(w_y, w_w) \tag{6}$$

$$\left.\begin{aligned} \min(w_x, w_z) &= \min(w_y, w_w) \\ \max(w_x, w_z) &\le \max(w_y, w_w) \end{aligned}\right\} \tag{7}$$

We define V_x-V_z to be a horizontal arc if (8) is satisfied

$$\left.\begin{aligned} \min(w_x, w_z) &> \min(w_y, w_w) \\ \max(w_x, w_z) &< \max(w_y, w_w) \end{aligned}\right\} \tag{8}$$

For brevity, we shall use h-arcs and v-arcs to denote horizontal arcs and vertical arcs from now on.

Corollary 2. All arcs in an optimum partition must be either vertical arcs or horizontal arcs. ■

Theorem 5. Let V_x and V_z be two arbitrary vertices which are not adjacent in a polygon, and V_w be the smallest vertex from V_x to V_z in the clockwise manner ($V_w \neq V_x$, $V_w \neq V_z$), and V_y be the smallest vertex from V_z to V_x in the clockwise manner ($V_y \neq V_x$, $V_y \neq V_z$). This is shown in Fig. 7 where we assume that

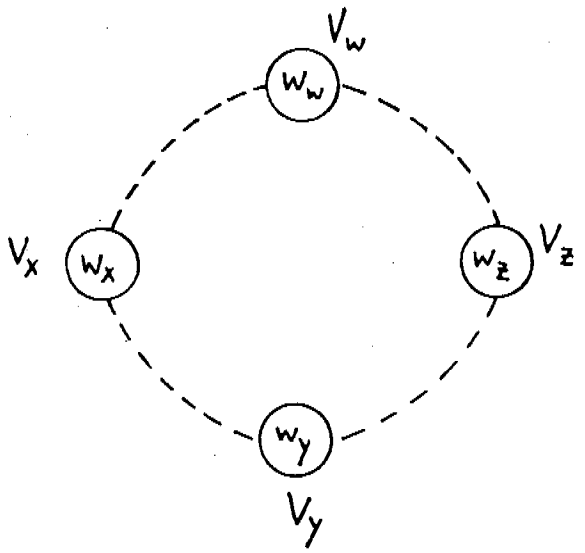


Fig. 7

$w_x \leq w_z$ and $w_y \leq w_w$. The necessary condition for V_x-V_z to exist as an h-arc in any optimum partition is

$$w_y < w_x \leq w_z < w_w \ . \ \blacksquare$$

We call any arc which satisfied this necessary condition a potential h-arc. Let P be the set of potential h-arcs in the n-gon and H be the set of h-arcs in the optimum partitions; we have $P \supseteq H$ where the inclusion could be proper.

Corollary 3. Let V_w be the largest vertex in the polygon and V_x and V_z be its two neighboring vertices. If there exists a vertex V_y such that $w_y < w_x$ and $w_y < w_z$. then V_x-V_z is a potential h-arc. ■

Two arcs are called compatible if both arcs can exist simultaneously in a partition. Assume that all weights of vertices are distinct, then there are (n-1)! distinct permutations of the weights around an n-gon. For example, the weights 10, 11, 25, 40, 12 in Fig. 6(a) correspond to the permutations w_1, w_2, w_4, w_5, w_3 (where $w_1 < w_2 < w_3 < w_4 < w_5$). There are infinitely many values of the weights which correspond to the same permutation. For example, 1, 16, 34, 77, 29 also correspond to w_1, w_2, w_4, w_5, w_3 but its optimum partition is different from that of 10, 11, 25, 40, 12. However, all the potential h-arcs in all the n-gons with the same permutation of weights are compatible. We state this remarkable fact as Theorem 6.

Theorem 6. All potential h-arcs are compatible. ■

Note that any potential h-arc V_x-V_z, like the one in Fig. 7, always dissects the n-gon into two subpolygons and one of these subpolygons has the property that all its vertices except V_x and V_z have weights larger than $\max(w_x, w_z)$. We shall call this subpolygon the upper subpolygon of V_x-V_z. For example, the subpolygon V_x - . . . - V_w - . . . - V_z in Fig. 7 is the upper subpolygon of V_x-V_z.

Using Corollary 3 and Theorem 6, we can generate all the potential h-arcs of a polygon.

Let V_x-V_z be the arc defined in Corollary 3. The arc V_x-V_z is a potential h-arc compatible to all other potential h-arcs in the n-gon. Furthermore, there is no other potential h-arc in its upper subpolygon. Now consider the (n-1)-gon obtained by cutting out V_w. In this (n-1)-gon, let $V_{w'}$ be the largest vertex and $V_{x'}$ and $V_{z'}$ be the two neighbors of $V_{w'}$. Then $V_{x'}$ - $V_{z'}$ is again a potential h-arc compatible to all other potential h-arcs in the n-gon and there is no other potential h-arc in its upper subpolygon which has not been generated. This is true even if V_w is in the upper subpolygon of $V_{x'}$ - $V_{z'}$. If we repeat the process of cutting out the largest vertex, we get n-3 arcs, all arcs satisfy Theorem 3.

The set of h-arcs of the optimum partitions must be a subset of these n-3 arcs.

The process of cutting out the largest vertex can be made into an algorithm which is O(n). We shall call this algorithm the one-sweep algorithm. The output of the one-sweep algorithm is a set S of n-3 arcs. S is empty initially.

The one-sweep algorithm:

Starting from the smallest vertex, say V_1, we travel clockwisely around the polygon and push the weights of the vertices successively onto the stack as follows (w_1 will be at the bottom of the stack).

(a) Let V_t be the top element on the stack, V_{t-1} be the element immediately below V_t, and V_c be the element to be pushed onto the stack. If there are two or more vertices on the stack and $w_t > w_c$, add V_{t-1}-V_c to S, pop V_t off the stack; if there is only one vertex on the stack or $w_t \leq w_c$, push w_c onto the stack. Repeat this step until the nth vertex has been pushed onto the stack.

(b) If there are more than three vertices on the stack, add V_{t-1}-V_c to S, pop V_t off the stack and repeat this step, else stop.

Since we do not check for the existence of a smallest vertex whose weight is strictly less than those of the two neighbors of the largest vertex, i.e. the existence of the vertex V_y in Theorem 3, not all the n-3 arcs generated by the algorithm are potential h-arcs. However, the one-sweep algorithm always generates a set S of n-3 arcs which contains the set P of all potential h-arcs which contains the set H of all h-arcs in the optimum partitions of the n-gon, i.e.,

$S \supseteq P \supseteq H$

where each inclusion could be proper. For example, if the weights of the vertices around the n-gon in the clockwise direction are $w_1, w_2, \ldots, w_n$ where $w_1 \leq w_2 \leq \ldots \leq w_n$, none of the arcs in the n-gon can satisfy Theorem 3 and hence there is no potential h-arcs in the n-gon. The one-sweep algorithm would still generate n-3 arcs for the n-gon but none of the arcs generated are potential h-arcs.

4. CONCLUSION. In this paper, we have shown the one-to-one correspondence between the orders of multiplying a chain of matrices and the partitions of an n-sided convex polygon. Then some theorems on the properties of the optimum partitions are presented. We have skipped some of the proofs and interested readers should refer to ref. 8 for details. Based on these theorems, an O(n) algorithm for finding a near-optimum partition can be developed (ref. 9). The cost of the partition produced by the heuristic algorithm never exceeds 1.155 Copt, where Copt is the optimum cost of partitioning the polygon. An O(n log n) algorithm for finding an optimum partition is also presented in ref. 8.

5. ACKNOWLEDGMENT. The authors would like to thank the U.S. Army Research Office for their continuing support during the previous years.

REFERENCES

1. A. V. Aho, J. E. Hopcroft and J. D. Ullman, "The Design and Analysis of Computer Algorithms," Addison-Wesley, 1974.
2. A. K. Chandra, "Computing Matrix Chain Product in Near Optimum Time," IBM Res. Rept. RC5626 (#24393), IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 1975.
3. F. Y. Chin, "An O(n) Algorithm for Determining a Near Optimal Computation Order of Matrix Chain Product," Communication of ACM, Vol. 21, No. 7, July 1978, pp. 544-549.
4. L. E. Deimel, Jr. and T. A. Lampe, "An Invariance Theorem Concerning Optimal Computation of Matrix Chain Products," North Carolina State Univ. Report TR79-14.
5. M. Gardner, "Catalan Numbers," Scientific American, June 1976, pp. 120-124.
6. S. S. Godbole, "An Efficient Computation of Matrix Chain Products," IEEE Trans. Computers C-22, 9 Sept. 1973, pp. 864-866.
7. H. W. Gould, "Bell and Catalan Numbers," Combinatorial Research Institute, Morgantown, W. Va., June 1977.
8. T. C. Hu and M. T. Shing, "Computation of Matrix Chain Product," to appear in SIAM J. on Computing, 1981.
9. T. C. Hu and M. T. Shing, "An O(n) Algorithm to Find a Near-Optimum Partition of a Convex Polygon," to appear in the Journal of Algorithms, 1981.

ATTENDANCE LIST
1981 Army Numerical Analysis & Computers Conference

NAME	ORGANIZATION
Asquith, C. Frank	MICOM
Blanco, Abel J.	Army Electronics R&D Command
Boggs, Paul T.	ARO
Bryan, Ferrell	MICOM
Carroll, Edward J.	ARRADCOM
Caviness, B. F.	GE R&D
Chandra, Jagdish	Army Research Office
Chen, Peter C. T.	Benet Weapons Laboratory
Cody, William J.	Argonne National Laboratory
Coleman, Norman P.	ARRADCOM
Conlon, John C.	AMSAA
Dickson, Richard E.	MICOM
Faber, Conrad	Army Aviation R&D Command
Garcia, Charles R.	WSMR
Gazaway, M.	MICOM
Gunzberger, Max	University of Tennessee
Glimm, Jim	Rockefeller University
Goldstein, Marvin J.	Naval Underwater Systems Center
Hackett, Robert M.	MICOM
Hafen, John A.	WSMR
Hausner, Arthur	Harry Diamond Laboratories
Henriksen, Bruce B.	BRL
Herman, Glenn	WSMR
Hirschberg, Morton A.	BRL
Jenkins, Billy Z.	MICOM
John, Billy H.	Army Corps of Engineers
Johnson, Billy H.	Waterways Experiment Station
Johnson, James H.	Constr Engr Res Lab
Komoriya, H.	Argonne National Laboratory
Kring, Jonathan F.	TACOM
Kurtz, Keith	Army Engineer Topographic Laboratories
Ladouceur, Pierre,	Department of National Defence, Ottawa
Launer, Robert L.	ARO
Lenoe, E. M.	AMMRC
Martin, Donald L., Jr.	MICOM
McConnell, Peter J.	Army Mobility Equipment R&D Command
Morris, Alfred H., Jr.	Naval Surface Weapons Center
Nohel, John A.	University of Wisconsin
Opalka, Klaus O.	ARRADCOM/BRL
Osterweil, Leon J.	University of Colorado

Attendance List
1981 Army Numerical Analysis and Computers Conference

Parker, A. P.	AMMRC
Plemmons, Robert J.	University of Tennessee
Rice, Bill	WSMR
Riesenfeld, Richard F.	University of Utah
Roache, Patrick J.	Ecodynamics Research Associates, Inc.
Selig, John M.	MICOM
Serbin, Steven M.	University of Tennessee
Shen, C. N.	Waterliet Arsenal
Shen, Wen-Wu	MERADCOM
Stenger, Frank	University of Utah
Thornton, Thomas L.	SAI
Wagner, Clifford C., Jr.	USAMIA
Walbert, James N.	Army Ballistic Research Laboratory
Wang, Chia Ping	Army Natick R&D Laboratories
Ward, Robert C.	Union Carbide Corp
Wu, Julian J.	Benet Weapons Laboratory
Zabusky, N.	University of Pittsburgh

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: ARO Report 81-3

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle): Proceedings of the 1981 Army Numerical Analysis and Computers Conference

5. TYPE OF REPORT & PERIOD COVERED: Interim Technical Report

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS: Army Mathematics Steering Committee on behalf of the Chief of Research, Development and Acquisition

12. REPORT DATE: August 1981

13. NUMBER OF PAGES: 630

14. MONITORING AGENCY NAME & ADDRESS (If different from Controlling Office): U. S. Army Research Office, ATTN: DRXRO-MA, P. O. Box 12211, Research Triangle Park, NC 27709

15. SECURITY CLASS. (of this report)

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited. The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

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

18. SUPPLEMENTARY NOTES

This is a technical report resulting from the 1981 Army Numerical Analysis and Computers Conference. It contains papers on computer aided designs and engineering as well as papers on numerical analysis.

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

mathematical software
beta and triangular distributions
multinomial variables
military construction
boundary layer equations
blast loads
hyperbolic conservation laws
hydrodynamic models
the Gem code
variational methods
time-stepping methods
hyperbolic systems
meteorological data
parabolic equation
quadratic λ-matrices
ohmic losses
algebraic computation
gun tube problems
compressible material
land mines
elastic-plastic structures
control algorithms
minicomputers
beam vibrations
analytical solutions
dynamic testing
Kroenig-Penney model
matrix chain products

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)