REPORT NO.
UCB/EERC-81/19
DECEMBER 1981

EARTHQUAKE ENGINEERING RESEARCH CENTER

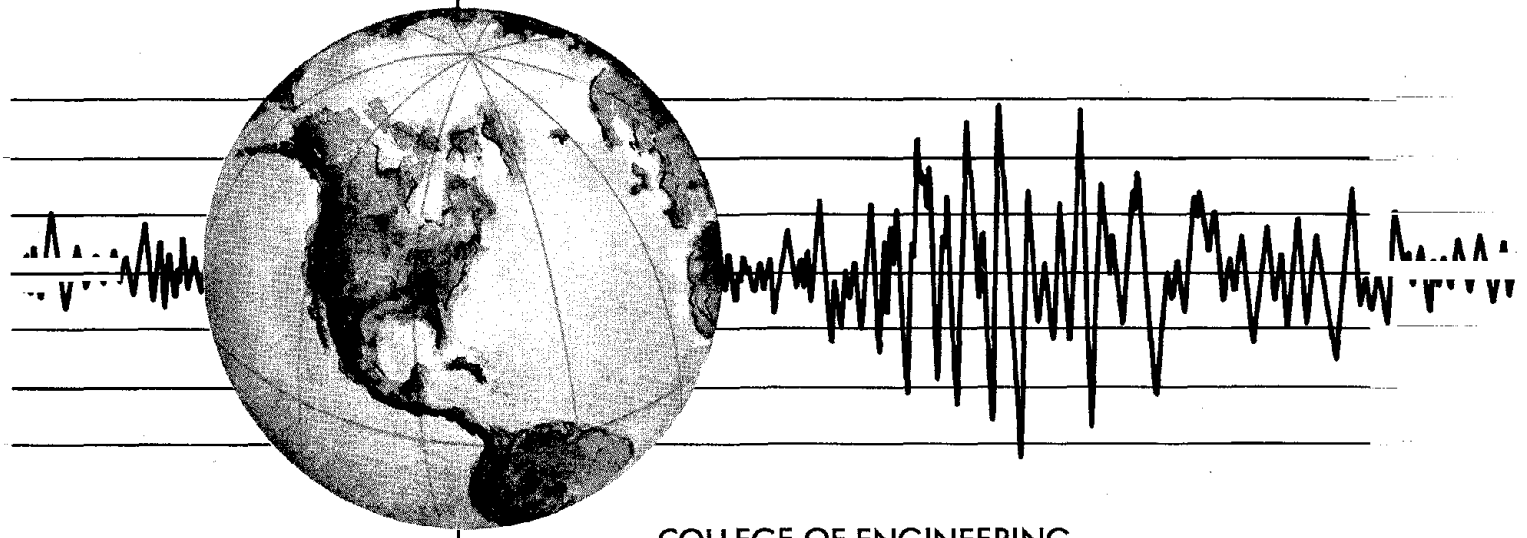
DELIGHT.STRUCT

A COMPUTER-AIDED DESIGN ENVIRONMENT FOR STRUCTURAL ENGINEERING

by

R. J. BALLING
K. S. PISTER
E. POLAK

Report to the National Science Foundation



COLLEGE OF ENGINEERING

UNIVERSITY OF CALIFORNIA · Berkeley, California

For sale by the National Technical Information Service, U.S. Department of Commerce, Springfield, Virginia 22161.

See back of report for up to date listing of EERC reports.

DISCLAIMER

50272-101

REPORT DOCUMENTATION PAGE	1. REPORT NO. NSF/CEE-81048	2.	3. Recipient's Accession No. PB82 218496
4. Title and Subtitle: DELIGHT . STRUCT A Computer-Aided Design Environment for Structural Engineering			5. Report Date: December 1981 6.
7. Author(s): R.J. Balling, K.S. Pister, E. Polak			8. Performing Organization Rept. No. UCB/EERC-81/19
9. Performing Organization Name and Address Earthquake Engineering Research Center University of California, Berkeley 47th Street & Hoffman Blvd. Richmond, California 94804			10. Project/Task/Work Unit No. 11. Contract(C) or Grant(G) No. (C) (G) PFR-7908261
12. Sponsoring Organization Name and Address National Science Foundation 1800 G Street, N.W. Washington, D.C. 20550			13. Type of Report & Period Covered 14.

15. Supplementary Notes

16. Abstract (Limit: 200 words)

This report describes an expandable software system for optimization-based, interactive computer-aided design of structures. This system can be used for the design of statically and/or dynamically loaded structures which exhibit linear or nonlinear response.

The software is the union of: (1) an interactive base code for the management of the computer-aided design process named DELIGHT, (2) a dynamic nonlinear general-purpose structural analysis package named ANSR, (3) a library of optimization algorithms specialized for the type of mathematical programming problems characteristic of structural design, and (4) specialized software for the design of seismic-resistant planar steel frames.

Flexibility has been emphasized in the development of this system so that a wide range of structural problems can be considered. The user describes his problem to the system by supplying a minimal amount of software or by selecting software from expandable libraries.

(a)

18. Availability Statement: Release Unlimited	19. Security Class (This Report)	21. No. of Pages 135
	20. Security Class (This Page)	22. Price

(See ANSI-Z39.18) See Instructions on Reverse

OPTIONAL FORM 272 (4-77)
(Formerly NTIS-35)
Department of Commerce

NOTICE

THIS DOCUMENT HAS BEEN REPRODUCED FROM THE BEST COPY FURNISHED US BY THE SPONSORING AGENCY. ALTHOUGH IT IS RECOGNIZED THAT CERTAIN PORTIONS ARE ILLEGIBLE, IT IS BEING RELEASED IN THE INTEREST OF MAKING AVAILABLE AS MUCH INFORMATION AS POSSIBLE.

DELIGHT.STRUCT: A Computer-Aided Design Environment for Structural Engineering

by

R. J. Balling

K. S. Pister

and

E. Polak

Prepared under the sponsorship of
the National Science Foundation
Grant PFR-7908261

Report No. UCB/EERC-81/19
Earthquake Engineering Research Center
College of Engineering
University of California
Berkeley, California

December 1981

ABSTRACT

This report describes an expandable software system for optimization-based, interactive computer-aided design of structures. This system can be used for the design of statically and/or dynamically loaded structures which exhibit linear or nonlinear response.

The software is the union of: (1) an interactive base code for the managment of the computer-aided design process named DELIGHT, (2) a dynamic nonlinear general-purpose structural analysis package named ANSR, (3) a library of optimization algorithms specialized for the type of mathematical programming problems characteristic of structural design, and (4) specialized software for the design of seismic-resistant planar steel frames.

Flexibility has been emphasized in the development of this system so that a wide range of structural problems can be considered. The user describes his problem to the system by supplying a minimal amount of software or by selecting software from expandable libraries.

ACKNOWLEDGEMENTS

This research was supported by the National Science Foundation under Grant PFR-7908261 with the University of California, Berkeley. Computing facilities were provided in part by equipment Grant ENG-7810442 from the National Science Foundation.

The authors also wish to acknowledge the developers of parts of software that were used in this work. Names of these researchers appear in references cited throughout the report. In particular, W. Nye, A. Tits, and A. Sangiovanni-Vincentelli are acknowledged as the developers of the DELIGHT software system used in this research.

TABLE OF CONTENTS

Page

1. INTRODUCTION

One of the most fundamental problems facing the structural engineer is that of design. This problem may be stated as, "Specify a good structure which will perform acceptably under possible loads". The way in which the engineer solves this problem is known as the design process. Typically the design process will have a "quantification" and a "computation" phase. In the quantification phase the engineer, after conversing with the owner, architect, contractor, etc., quantifies such questions as:

(1) What are the quantities which must be "specified"?

(2) What criteria are used to define "good"?

(3) What constitutes "acceptable performance"?

(4) What are the "possible loads"?

(5) What is a reasonable mathematical model?

(6) What is a good initial design?

It is important to note that because people have a tendency to change their minds, the answers to such questions are often revised throughout the entire design process. In the computation phase a trial-and-error scheme is employed in which the designer iterates by "analyzing" the design to obtain its response and "modifying" the design accordingly until he is satisfied. Any engineer who has designed a structure as simple as a reinforced concrete beam is familiar with this trial-and-error scheme.

The computer may be a very useful tool to the structural engineer. It can perform repetitious computations and logic very rapidly. Therefore, a good policy for the engineer to adopt is to let the computer perform those aspects of the design process which are repetitious and standard while the engineer carries out those aspects which require judgement and experience. Thus in the computer-aided-design (CAD) environment, engineer and computer are complementary as they work together to solve the design problem. This concept constitutes the premise of this report. It is foolish for a programmer to try to completely automate the design

process. It is also foolish for the engineer to waste time performing routine computations and logic by hand. The quantification phase of the design process is not easily generalized and requires much judgement on the part of the engineer. The computation phase has many repetitious aspects which can and should be automated.

One part of the computation phase which can be done by the computer is that of analysis. This may be defined as determining the response of a given structure under given loads. Most efforts to utilize the computer in structural engineering have been in the area of analysis, and much progress has been made. Indeed efficient general-purpose computer programs capable of modelling dynamic and nonlinear response have emerged in recent years [1]. The efficiency and generality of such programs have led to their popularity in industry. These programs typically require the user to supply data describing the particular analysis problem at hand. The programs work like a "black box" acting upon the data in a batch mode to compute the response.

Analysis is not the only part of the computation phase which may be automated. The way in which the structure is modified according to its response computed from analysis, thus, the trial-and-error process itself may be generalized. Optimization algorithms may be used to manage the trial-and-error process. These algorithms take advantage of the computer's ability to assimilate much information about the current design in order to determine the best set of modifications. Work in the area of optimization-based design has occurred and programs have emerged [2]. Typically they take the same format as the general-purpose analysis programs in requiring the user to supply data upon which the program acts in a batch mode. These programs have not enjoyed near the popularity as their analysis counterparts. In the author's opinion there are two main reasons for this lack of popularity. The first reason is that most of these programs lack sufficient flexibility. Normally they pre-specify the choice of cost and constraint functions. For example a common cost function is to minimize weight. However in most real design situations the choice of cost and constraints is one of the most judgement-based decisions in the design process. This decision is very problem-dependent and should not be

automated. The second reason for the lack of popularity of optimization-based design programs is that their convergence may be slow. The choice of algorithm and parameters for that algorithm which provides fastest convergence is not only problem-dependent but also iteration-dependent. Thus, by running in a batch mode a change of algorithm or algorithm parameters cannot be made.

The system OPTNSR was based on the philosophy that the user should supply programs to evaluate the cost and constraints and that he should be able to interact with algorithm parameters [3]. OPTNSR was the union of a general-purpose structural analysis program named ANSR and an interactive feasible directions optimization program named INTEROPTDYN. After gaining sufficient experience with OPTNSR, three shortcomings were recognized. First, it took too much time for the user to write the programs for cost and constraint evaluation because the format was complex and required a knowledge of the internal operation of ANSR. Second, once the user-supplied programs were specified, it was not allowed to later change them in the design process. Third, the possibility of changing algorithms interactively did not exist.

DELIGHT.STRUCT is a computer-aided structural design environment which has emerged after working with OPTNSR. Basically it is the union of ANSR, an engineering design system named DELIGHT, a library of optimum structural design software, and a library of software for specific classes of structural problems. DELIGHT.STRUCT is a system rather than a program. The philosophy adopted is that the user may interactively write software or choose software from existing expandable libraries. This allows the user to define his own cost and constraints, to utilize his own trial-and-error scheme, and to develop his own pre- and post-processors for communicating data in the way he thinks is best. If his software has some degree of generality, it may be added to expandable libraries to lessen future programming effort. Another aspect of the DELIGHT.STRUCT philosophy is that the user may interact with the execution in order to display and/or change the values of variables. Furthermore the user is allowed to interactively change or create programs during execution.

This report describes the DELIGHT.STRUCT system and how to use it. Section 2 provides background about the DELIGHT.STRUCT system. Organization and content of the different components of the system are described in this section. Section 3 describes the use of the DELIGHT.STRUCT system. Details on software and commands given by the user to solve specific structural design problems are explained in this section. Section 4 discusses future development of the DELIGHT.STRUCT system. Expansion of the libraries in the system as well as areas of improvement of the system itself are explored in this section.

2. BACKGROUND ON DELIGHT.STRUCT

DELIGHT.STRUCT is currently operational on a VAX 11/780 computer with a virtual memory version of the UNIX operating system. It is assumed, therefore, that the reader has some familiarity with UNIX [4]. This section of the report is intended to give the reader background on how the DELIGHT.STRUCT system is organized and how it operates. Details on how to use the system will be treated in the Section 3 of the report. As depicted in Figure 1, DELIGHT.STRUCT is the name of a directory which currently contains ten sub-directories. The directories *work.d, framework.d, save.d,* and *framesave.d* are the places where the user does the actual computing and storing of results. The directory *delight.d* contains files which interface the DELIGHT design system the, bulk of which resides in a central location external to DELIGHT.STRUCT. The main executable program, also named DELIGHT.STRUCT, is found in the directory *delight.d.* The DELIGHT design system will be briefly described in Subsection 2.1 which follows. Subsection 2.2 will describe the ANSR structural simulation package as it appears in the directories *ansr.d* and *element.d.* Subsection 2.3 will describe the software in the directory *structrattle.d* which is oriented toward optimum structural design. Subsection 2.4 will describe the software in the directories *framefort.d* and *framerattle.d* which relates to the seismic-resistant design of planar steel frames. Identification of each individual file in these directories is given in Appendix 1.

2.1. DELIGHT

DELIGHT, or "DEsign Laboratory with Interaction and Graphics for a Happier Tomorrow" is a system tailored for an engineering design environment. Although it is still in its developmental stages, it has been used successfully by research as in the design of electrical components and earthquake resistant structures. Thus, the bulk of the DELIGHT software resides in a central location while only those parts necessary to form the DELIGHT.STRUCT version reside in the directory *delight.d.* Three main goals were emphasized in the development of DELIGHT. First, the system was designed to simplify the programming process for the

user. Second, the system is based upon an interaction between the designer and the computer. Third, the system is organized to accomodate expandable libraries of software. Some of the features of the DELIGHT system which are directed toward these goals are depicted in Figure 2 and briefly described in the remainder of this subsection. For a more detailed description of the features of DELIGHT the reader is referred to the DELIGHT manual [5].

2.1.1. Fortran Interface

Engineering design software must necessarily include an analysis or simulation package. As mentioned in the Introduction, much research effort has been directed in recent decades toward the development of good engineering simulation programs. In order to expand DELIGHT to include such programs, DELIGHT allows easy interfacing with existing Fortran software. A Fortran subroutine may be interfaced to the DELIGHT program with virtually no modification so that it may later be called by a DELIGHT statement. Furthermore, variables within Fortran subroutines can be made "interactive" so that their values can be used or even changed with DELIGHT statements. This interfacing capability allows easy addition or substitution of superior simulation programs which are sure to appear in the future. A design group may form its own version of the DELIGHT program by interfacing only those programs which it needs. Thus, in the DELIGHT.STRUCT system the ANSR structural simulation program, as well as seismic resistant frame design software, are interfaced to the basic DELIGHT program. A description of how this was done follows.

To interface structural engineering subroutines the "built-in" part of DELIGHT was first modified. This part of DELIGHT contains information regarding the names and arguments of all Fortran subroutines which may be called from DELIGHT. This information for the structural subroutines was added by editing the files *builtnam, builtmid,* and *builttop.* The whole built-in part of DELIGHT was then recompiled. The compiled Fortran subroutines to be interfaced, the compiled subroutines which they call, the compiled built-in part of DELIGHT, and the compiled DELIGHT program itself were then loaded together to form the executable file

DELIGHT.STRUCT.

Certain variables and coefficients in the frame design subroutines were made interactive. This was done by writing the Fortran subroutine *fdeclr* (with no arguments) and interfacing it with the DELIGHT program in the manner described in the preceding paragraph. This subroutine contains the common blocks with all the variables which are interactive. Following these common blocks are calls to one of the DELIGHT subroutines *deci, decia1, decia2, decr, decra1,* or *decra2* for each such variable depending on its type. Execution of this subroutine is done once and causes all of these frame design variables to be made interactive.

2.1.2. Compilation / Execution

A useful engineering design system must be interactive. To help achieve this goal DELIGHT is based on the idea of interactive execution. When the designer types a command at the terminal, DELIGHT compiles and executes that command immediately. This means that DELIGHT performs line by line compilation and execution bypassing the usual load-linkage phase. This type of operation also allows the user to perform "scratchpad" calculations on the terminal in order to help him make decisions. In this respect the computer operates like a sophisticated hand-held calculator. In addition to this interactive execution capability DELIGHT also has interactive programming capabilities. The user is thus able to interactively program his own commands which will relieve him from having to re-type a series of statements over and over. The way these interactive execution and programming capabilities work will now be explained.

When the DELIGHT program is started, the prompt "1>" will appear on the terminal. At this point the user types a valid statement. DELIGHT "compiles" this statement into an intermediate form by translating it into a series of numeric codes using a modern compiler-compiler generated parser. Then the statement is "executed" by entering a large Fortran computed goto which acts on the numeric codes and sends execution to corresponding groups of Fortran statements. This results in interactive execution. There is no load-linkage phase.

Sometimes it is necessary to compile more than one statement before execution. For example a decision or a loop statement expects at least one following statement before it can be executed. DELIGHT automatically postpones execution for such statements. The user may also force postponement of execution by enclosing blocks of statements in curly brackets. In this case each statement is compiled as it is typed, but the statements are not executed until DELIGHT encounters the closing curly bracket.

Often a user would like to compile a group of statements once without execution and give them a name whereby they can be executed in the future without compilation. This is done by enclosing the statements in curly brackets and preceding them by a "procedure" statement. Such DELIGHT procedures are analogous to Fortran subroutines. The procedure is executed by calling the procedure name. Because there is no load-linkage phase in DELIGHT, a procedure called by another procedure must already exist in compiled form before the calling procedure can be compiled. The called procedure may be a dummy procedure which can later be replaced by the "real" procedure.

Two important features of DELIGHT are "defines" and "macros". The user can define any name to be a group of characters which is its definition. When the user types characters at the terminal they are broken up into names, numbers, etc. DELIGHT then checks the names to see if they are the define name from a previous define statement. If so, DELIGHT replaces them with the definition characters which are sent to be compiled and executed. Therefore, the user may actually type define names which may appear as invalid input but the DELIGHT compiler actually "sees" the definition which is valid input. Arguments, optional arguments with default values, and quoted strings may occur after the name and before the definition in define statements. Macros work similarly to defines except that they also allow operations and logic to be performed on the user's input and on the characters sent to the compiler.

Through the use of procedures, defines, and macros the user is able to interactively program. Several statements can be represented by a single statement. Procedures have the advantage that they are compiled only once. The statements replaced by defines and macros,

however, are re-compiled each time they are called. Defines and macros have the advantage that they do not use up a lot of memory by storing compiled code. Defines and macros also allow the user to create very readable command syntax. Thus, the user is able to expand DELIGHT according to his own needs or interests.

2.1.3. Included Files And Memfiles

Most of the software in DELIGHT is found among libraries of files. The user selects the software he wishes to use from such file libraries. This philosophy permits the user to have access to an extremely powerful program, but at the same time he is not bogged down with large amounts of software that he is not using. Furthermore, the user is also permitted to add to the file libraries, thus expanding the power of the system according to his needs. DELIGHT allows the user to access source files of DELIGHT statements which can then be compiled and executed. DELIGHT also allows the user to access binary files which are already in compiled form and may be executed directly.

Suppose the user creates a file in his working directory which contains delight statements after which he starts the DELIGHT program and awaits a prompt. DELIGHT is waiting for the user to input a statement from the terminal. With an "include" statement the user can direct that input be made from a file instead of from the terminal. Thus, DELIGHT will go to the file and compile and execute the statements therein just as if they had been typed at the terminal. The file may contain a procedure which the user has written. When the file is included, this procedure is compiled.

It would be ridiculous to force the user to have all the files in all the libraries which he may ever want to include in his working directory. Instead the user needs only a file *openhdtl,* which contains a list of the names of the directories containing the libraries of files. If the user wishes to include a particular file which is in one of these directories, he uses the include command but encloses the name of the file in triangular brackets. This causes DELIGHT to search through the directories listed in the file *openhdtl* for the file, which it then includes.

Often after the user has included many files and compiled many procedures, he would like to be able to store this compiled information in a binary file. Such binary files are known as "memfiles". Everything that has been compiled up to the current time can be stored in a memfile with the "store" command. This creates a binary memfile in the working directory. At a later date the user may start DELIGHT and "restore" from a memfile. Thus, he starts where he left off and does not have to re-compile all the previous software. The "openhdtl" philosophy applies to memfiles also. Thus, the user may restore from memfiles which are in the directories listed in the file *openhdtl.*

The bulk of the software in DELIGHT.STRUCT is in memfiles and libraries of files to be included. After the executable file DELIGHT.STRUCT was first loaded, it was forced to start and the files listed in the file *setup* were included. The files listed therein are those basic DELIGHT software files from the DELIGHT library thought to be useful for structural engineering. After these files were included, the compiled information was stored in the memfile *memfile.* Thus the memfile *memfile* contains basic DELIGHT software. Later DELIGHT.STRUCT was started, the software in the memfile *memfile* was restored, files containing software for optimum structural design were included, and the compiled information was stored in the memfile *memstruct.* Finally, DELIGHT.STRUCT was started, the software in the memfile *memstruct* was restored, files containing seimic-resistant frame design software were included, and the compiled information was stored in the memfile *memframe.*

2.1.4. Interrupts

DELIGHT possesses sophisticated interrupt capabilities. The user may interrupt execution and then examine the values of variables. From this information he can make changes, and he can decide whether to continue execution or start execution of something else. This interrupt capability is also useful for procedure debugging. Thus, the interrupt capabilities are essential to interactive programming and execution.

When the user writes a DELIGHT procedure, he may insert a "suspend" statement. When the procedure is later executed, it will interrupt at the suspend statement. Alternatively, the user may insert an "interaction" statement in a procedure. When executing the procedure if the user hits the "break" key on the terminal once, the execution will be interrupted when it reaches the next interaction statement. If the user does not hit the break key, execution will bypass the interaction statement. Finally, if the user hits the break key on the terminal twice during the execution of a procedure, execution will be interrupted no matter where it is in the procedure. These three methods of interrupting define "forced", "soft", and "hard" interrupts, respectively. One final type of interrupt is the "run-time error". For some execution errors such as "divide by zero" or "array subscript out of bounds" DELIGHT will not abort but instead will interrupt with a "run-time error" message.

Once execution has been interrupted, by whatever means, all DELIGHT variables, whether they are external or local to procedures, retain their most recent values from the execution. The user may then interact with these variables as he wishes. This is an important feature for debugging. If the execution interrupts due to a run-time error, the user can enter the procedure where the interrupt occurred and print out the current values of local variables to discover what went wrong.

After the user has finished interacting with DELIGHT variables following an interrupt, he has many options. He may "resume" execution of the interrupted procedure. He may begin execution of a second procedure, interrupt, begin execution of a third procedure, etc. up to five levels deep, and successively resume their executions from the innermost to the outermost. He may "reset", which causes him to leave the interrupted procedure to do something else. He may "quit", which exits from DELIGHT altogether. He may store into a memfile, which causes the current values of all DELIGHT variables to be stored in the memfile, which he can examine later.

2.1.5. Rattle Language

The programming language for DELIGHT is known as Rattle or "RATfor Terminal Language Environment". This language has been designed to simplify the programming process. Some of the tedious details of Fortran are eased in Rattle. Furthermore, there has been emphasis placed on readability in the development of Rattle syntax so that it is easy to become acquainted with an unfamiliar Rattle program.

Rattle is a structured programming language patterned after Ratfor or "RATional FORtran". Thus, it is very similar to Fortran, the main difference in Rattle being the absence of statement labels By convention the programmer indents all statements which are to be executed following a loop statement or as a result of a decision statement. The appearance of complicated branching statements can be simplified through successive indentation with a structured language. The user may also choose from several constructions such as "for", "while", and "repeat" for loops and "if", "else", and "case" for decisions. Finally, Rattle allows the user to "break" out of or skip to the "next" iteration of any level of nested looping. The result of this programming structure and style is enhanced program readability.

Some of the details typical to Fortran which are eliminated in Rattle include the following:

(1) All Rattle variables are initialized to zero automatically.

(2) If a loop end limit is less than its start limit, the loop is automatically bypassed.

(3) All local variables in a Rattle procedure retain their values from the last call of the procedure to the next call.

(4) Incomplete Rattle statements may be continued on the next line.

(5) All numbers in Rattle are double precision so there is no need to distinguish variable type.

(6) Anything following a "#" sign at any point on a line is taken as a comment.

(7) Certain variables may be designated as "global", meaning that they are automatically known to all procedures.

(8) Common constants such as "PI", "TWOPI", or "MAXREAL" are included among the global variables.

By convention the names of global variables consist of all capital letters; the names of local procedure variables consist of all small letters; and the names of regular Rattle variables begin with a capital letter followed by small letters.

Most of the nice features of Fortran are included in Rattle. Some examples are:

(1) Rattle includes the standard library of intrinsic mathematical functions.

(2) Input/ output to or from the terminal or files may be in free or specified format.

(3) Regular Rattle variables may be "imported" to any procedure analogous to a Fortran common statement.

2.1.6. Memory Manager

Another way in which DELIGHT simplifys the programming process is through its memory management system. Arrays with any number of subscripts may be variably dimensioned or redimensioned at any point within or outside a procedure. Dynamic array dimensioning may even occur as a result of a decision statement or within a loop statement. It is also permitted to dimension arrays to size zero. When an array is passed to a procedure through the argument list it is unnecessary to pass any of its dimensions. This completely dynamic memory management relieves many of the usual headaches of typical programming.

It is important to emphasize the efficiency of the memory manager in addition to the convenience it provides. When gaps are created in the large DELIGHT storage block due to the redimensioning of arrays and the recompilation of procedures, the user may force a "CRUNCH" to occur in which the allocated memory is basically compacted. An automatic CRUNCH will

occur as the storage block nears capacity in order to free up more memory space. If this fails to provide enough free space for the desired array, an automatic "FLOP" will occur, whereby array locations are swapped in order to free up space. Both the CRUNCH and the FLOP represent recent efficient techniques from the field of memory management in computer science.

2.1.7. Utility Commands

To give DELIGHT interactive power many utility commands have been devised. The commands "include", "system", "store", "restore", "suspend", "interaction", "resume", "reset", and "quit" have already been mentioned. Commands exist to help manipulate DELIGHT input and output. There are also commands to help the user keep track of the variables and procedures he has created. Finally, there are commands to help the user examine what has happened during execution. Some of these commands are described below. New commands are being made available which the user may or may not choose to include.

The user can cause DELIGHT input and/or output to be written on or appended to a file as well as output to the terminal with the "output_to", "echo_to", and "echo_io_to" commands. This allows the user to keep records of what he has been doing. When the user includes a file, switching DELIGHT input from the terminal to that file, he may monitor with the "echo" command. The "list" command allows the user to list on the terminal the contents of a file. The "history" command allows the user to list the most recent commands which have been given. With the "edit_history" command this history may be edited and written to a file.

All names of variables and procedures which have been created are stored in a symbol table. The command "whatis" permits the user to find out if a particular name corresponds to a variable, an array, a procedure, a define, etc. With the "whereis" command he can find the source file where the variable, array, procedure, define, etc. was created. An entry in the symbol table along with its memory allocation in the DELIGHT memory block can be removed with the "remove" command. Probably the most frequently used DELIGHT command is the "display" command. The names and values of some or all variables, global variables, local

variables, or system variables can be displayed with this command. Similarly the names of some or all of the arrays, defines, functions, macros, operators, or procedures in the symbol table can be displayed. The user can designate certain variables with which he often interacts as parameters with the "parameter" command. The parameters can later be displayed in a fancy way with their values, source file names, and descriptive strings.

When execution has been interrupted, the "trace" command will tell the user on which line of which procedure called from which line of which procedure etc. the execution was interrupted. The "display_time" command will display the cpu-time and number of calls that have been made to the most-used procedures. The "enter" command allows the user to enter a procedure where he may then display the values of local variables and arrays. The "!" command allows the user to re-execute any of the commands which appear in the history list.

2.1.8. High-Level Matrix Commands

In order to simplify the programming process for the user, DELIGHT has incorporated many high-level matrix commands. Algorithms from the LINPACK and Harwell libraries have been built into the system. Programs become much shorter and readable when a single statement can be used for a more complex matrix operation. Some of these powerful commands will now be described.

Vector and matrix functions include "det" for computing matrix determinants, "rcond" for computing matrix condition numbers, "|.|" for computing the euclidean norm of a vector, and "<<.,.>>" for computing the inner product of two vectors. Matrix algebra statements may follow the command "matop". These statements include adding, subtracting, multiplying, inverting, finding the eigenvalues, forming the singular value decomposition, and forming the Q-R decomposition of matrices. Arrays which are formed as a result of such operations are automatically dimensioned. The symbol "'" may be used throughout to denote the transpose of any matrix. Smaller arrays may be "clipped" out of larger arrays, and parts of larger arrays may be "filled" with smaller arrays. Algorithms for solving systems of linear equations, linear

programming problems, and quadratic programming problems may be accessed by a single statement. Array commands exist for finding the dimensions of an existing array, printing out or reading in values for array elements, and saving the values of an array in a file in the form of assignment statements so that the file can later be included.

2.1.9. High-Level Graphics Commands

High-level graphics commands which are necessary for interactive power also exist. The man-machine interface is an important consideration in the development of an engineering design system. Graphics aid the designer to rationally interpret large quantities of information quickly so that he can make decisions. Some of the DELIGHT graphics capabilities will now be descibed briefly.

The "terminal" and "grinit" commands allow the user to tell the DELIGHT system the type of terminal on which he is working and to initialize the terminal for graphics commands. The user may set up "viewports" and "windows" on the terminal screen representing graphics working areas, and he may endow them with their own "world" coordinate systems. Once the user has selected a viewport to work in, he may draw boxes, ovals, and vectors in whichever color he chooses. He may also display data in the form of "barplots" or continuous "curves" where only ordinates are given, or parametric "parcurves" where ordinates and abcissas are given. He can also position the graphics text cursor and display graphics text for labelling purposes. There is a sophisticated "plot" package which automatically scales, plots, and labels up to nine Rattle expressions vs a linear or logarithmic increment of any Rattle expression with specified limits. With this, a nice-looking plot of data can be made with one command. Finally, there are commands for erasing all or parts of the screen.

2.2. ANSR

The structural simulation program imbedded in DELIGHT.STRUCT is ANSR or "Analysis of Nonlinear Structural Response" [6]. This program has capabilities for analyzing static or dynamic, linear or nonlinear structural response. The main reason this program was chosen over other available structural simulation packages was because of its organization. It is quite modular and easy to follow. The organization of the ANSR program is depicted in Figure 3. The directory *ansr.d* contains the Fortran subroutines for central ANSR processing while the expandable directory *element.d* contains the Fortran subroutines for the various ANSR elements. Currently the directory *element.d* contains a three-dimensional elasto-plastic parallel-component truss element and the two-dimensional lumped-plasticity parallel-component beam-column element [7]. The subroutines in both the directories *ansr.d* and *element.d* have been loaded into the main executable program DELIGHT.STRUCT. The mini-ANSR program will be briefly described, and the optimization-simulation and simulation-optimization interfaces will then be explained. Finally the two subroutines in ANSR which may be called from DELIGHT.STRUCT will be described.

2.2.1. Mini-ANSR

Mini-ANSR is a version of ANSR adapted for mini virtual-memory computers such as the VAX. It differs from the original ANSR program in its memory management system, which was drastically simplified in order to take advantage of the existing virtual memory management. The mini-ANSR version also contains modifications at the element and global levels to enable the computation of the energy response of the structure.

To perform a mini-ANSR simulation, the subroutine *input* is first called. This subroutine reads input data from files both at the global level and at the element levels through the element subroutines *inel1, inel2,* etc. After reading in the raw data, the subroutine *input* processes it, compacts it, stores it into common blocks, and determines corresponding addresses. After the subroutine *input* is finished one of the computation management subroutines *static* or *dynmic*

is called depending on whether a static or dynamic simulation is to be performed. These subroutines loop through load or time steps linearizing to form the stiffness matrix in the subroutine *stiff,* solving the resulting set of linear equations for the incremental nodal displacements in the subroutine *optsol,* and finding the unbalanced nodal forces in the subroutine *respon.* When convergence is attained, the response envelopes may be output by the subroutine *envout.* The global subroutines *stiff, respon,* and *envout* call their element level counterpart subroutines *stif1, stif2,* etc., *resp1, resp2,* etc., and *out1, out2,* etc.

2.2.2. Optimization-Simulation Interface

One of the two main obstacles the software developer is faced with when interfacing a simulation program such as ANSR with an optimization-based CAD package such as DELIGHT.STRUCT is to devise a scheme whereby the simulation input can be modified according to the changing set of design variables. In other words as the structure in question is re-designed by the optimization process, data in the simulation program must be changed so that the "new" structure can be analyzed. Some limitations had to be made on the scope of structural design problems to be considered in order to generalize this process.

First, it was decided that permissible design variables may only affect element properties. Thus the distance between nodes or the magnitude of nodal loads must remain constant with changes in design variables. Second, as a new element is added to ANSR, the programmer must also add a subroutine named *mdfl* in which he pre-specifies the possible design properties for that element. For the truss element the possible design properties are section area, strain hardening ratio, tensile yield stress, and compressive yield or buckling stress. For the beam-column element the possible design properties are section area, section moment of inertia, strain hardening ratio, and plastic yield moment. The element subroutine *mdfl* has as arguments the values of these possible element design properties from which it enters element common blocks and makes the necessary changes in data affected by these properties.

2.2.3. Simulation-Optimization Interface

The second obstacle in interfacing ANSR to DELIGHT.STRUCT involves storing in a general yet efficient manner the structural response from which the values of cost and constraints can be computed. The ANSR program normally prints the response information at each time step, and therefore it can use the same response arrays in the next time step by simply "writing over" the old information. However in the optimization setting the cost or constraints may be functions of the response at individual time steps meaning the entire response history should be stored for generality. It would obviously be inefficient if not impossible to store all possible response histories at every node and for every element. Therefore a general scheme must be devised whereby the user can communicate which response histories he wishes to store for his particular problem. The scheme should also include ordered storage of the specified histories so that the user knows where any particular history is located.

In order to avoid inefficient overdimensioning of several arrays, all response histories are stored in a large vector named *resp* which is the only member in the common block *bigres*. The first entries in *resp* are the values of the design variables from which the simulation generated the response to be stored. The user may have cost and constraint functions which correspond to different loadings, solution strategies, geometric modellings, etc. in his design problem. For example in the seismic resistant design problem the user may formulate one group of constraints corresponding to a static nonlinear model subjected to gravity loads, another group of constraints corresponding to a dynamic linear model subjected to moderate ground motion, and another group of constraints corresponding to a dynamic nonlinear model subjected to severe ground motion. To accomodate this situation the user would prepare three data files for ANSR named *ansrdata1, ansrdata2,* and *ansrdata3* which contain data for the three different analysis problems. The reponse histories corresponding to each ansrdata file are then stored in order in the vector *resp.* Therefore each ansrdata file must contain an integer which is the starting address in *resp* where its response histories are stored.

For each ansrdata file the possible global response histories which may be stored include seven different global energies from the energy balance equation, and relative displacements, relative velocities, or absolute accelerations in any of three orthogonal directions for any free node. Therefore each ansrdata file should include flags and lists expressing which histories for which nodes are to be stored. At each time step ANSR calls the subroutine *storsp* which takes care of the orderly storage of the response for that time step in *resp.* As an element is added to ANSR, the programmer must also add a subroutine named *stor* in which he pre-specifies which response quantities may be stored for that element. For the truss element it is possible to store axial displacement, axial force, or inelastic energy dissipation for each element. For the beam-column element it is possible to store axial force, plastic hinge rotation, plastic hinge energy dissipation, end moment, or total end rotation at both ends for each element. The subroutine *storsp* calls the element subroutines *stor1, stor2,* etc., which store element response histories according to element storage codes for each element. Each ansrdata file must specify such element storage codes.

2.2.4. Subroutines Analys And Resman

The subroutine which may be called from DELIGHT.STRUCT and which drives the ANSR simulation package is named *analys.* This subroutine performs the following four tasks:

(1) The storage sizes of ANSR common blocks are initialized and the arrays therein are set to zero.

(2) One of the six possible ansrdata files is opened and the subroutine *input* is called. Thus, the subroutine *analys* must be called for each ansrdata file in order to perform a complete simulation for a given set of design variables.

(3) The element subroutines *mdfl1, mdfl2,* etc. are called in order to modify the input data according to current values of the possible element design properties.

(4) The subroutines *static* and/or *dynmic* are called to carry out simulation.

Another subroutine which may be called directly from DELIGHT.STRUCT is *resman.* This subroutine manipulates the large vector *resp* in one of the following five ways, according to the value of its first argument flag:

(1) The contents of the vector *resp* may be written out to the binary file *response.*

(2) The vector *resp* may be restored from the binary file *response.* In this way the response from a previous simulation can be restored and used at a later date.

(3) Recall that the first things stored in *resp* are the values of the design variables which generated the stored response. Actually the responses from simulations corresponding to the two most recent design variable vectors may be stored in the vector *resp* one after another. The current set of design variables may be compared to the previous two sets, and if it is within a tolerance of either of them another flag is set and the corresponding response is copied into the front part of the vector *resp.* In this way one is able to obtain the response without performing another simulation in cases when the set of design variables is sufficiently close to either of the previous two sets.

(4) The two sets of design variables stored in the vector *resp* may be initialized.

(5) The vector *resp* may be made an interactive DELIGHT variable.

2.3. STRUCT

The directory *structrattle.d* contains software for general optimal structural design. The software is written in Rattle, and it has been compiled, combined with the basic DELIGHT software from the memfile *memfile,* and stored in the memfile *memstruct.* Structural design problems can be solved which may or may not employ ANSR simulation. At present this software has been utilized in seismic-resistant steel frame design which involved ANSR simulation as well as for the design of a large reflecting telescope support system which did not involve ANSR simulation. The software has been organized into three groups which will be

described in this subsection. These groups include software for the control of the computation phase of the design process, the optimization algorithm library, and the user software interface. The names of files of these groups begin with the capital letters "C", "A", and "S" respectively. The intricate relationships among the software in the directory *structrattle.d* are depicted in Figure 4.

2.3.1. Control Commands

There are control commands for the purpose of initialization according to the specific problem of the user. In particular the file *Cstartofffile* is used to initialize the computation of the cost and constraint functions. Specifically when this file is included the following five tasks are performed:

(1) The variables which describe the problem size are initialized by including a user-supplied file.

(2) The initial values of the design variables are set by including another user-supplied file.

(3) If ANSR simulation is involved, an ANSR simulation initialization file is included.

(4) Arrays for storing the values of constraints are dimensioned.

(5) A CRUNCH is forced.

The file *Coptimizefile* is used to initialize the optimization process. Specifically when this file is included the following five tasks are performed:

(1) The user is asked which optimization algorithm he wishes to employ and according to his answer the proper algorithm initialization file is included.

(2) The user is asked if he has supplied his own gradient computation scheme and if not the standard perturbation initialization file is included.

(3) The user is asked the maximum number of iterations he expects to complete and according to his answer, arrays are dimensioned for storing iteration histories.

(4) The cpu-time measurement features are initialized.

(5) An echo of DELIGHT input and output to the file *dialogue* is activated.

A number of commands have been created through DELIGHT defines for the control of the computation process. These commands are as follows:

(1) "startoff" causes the initialization file *Cstartofffile* to be included.

(2) "optimize" causes the initialization file *Coptimizefile* to be included.

(3) "simulate" causes the cost and constraint functions to be computed.

(4) "run n" causes the optimization algorithm to start running and suspend after it has completed n iterations.

(5) "cont n" causes the optimization algorithm to continue running and suspend after n additional iterations.

(6) "sleep n" causes the DELIGHT.STRUCT algorithm to sit idly for n seconds.

(7) "pajama" causes DELIGHT.STRUCT to include the file *pajama* the next time it suspends.

(8) "stop_opt" suspends the cpu-time measurement features and the echo to the file *dialogue.*

(9) "start_opt" resumes the cpu-time measurement features and the echo to the file *dialogue.*

(10) "saveall" saves the current ANSR simulation response in the binary file *response,* the current values of design variables in the file *xfile,* and the current state of DELIGHT execution in the memfile *memprob.*

(11) "onward" causes ANSR simulation response to be restored from the binary file *response.*

The way in which these commands are used is described in more detail in the Section 3 of this report.

All algorithms operate by successively "re-designing" the set of design variables. Each redesign constitutes an iteration. Some point in each iteration must be designated as the "main

breakpoint" of the algorithm. This is done by calling the procedure *interact* at this main breakpoint. This procedure does the following four things at each iteration:

(1) The iteration number, the value of the cost function, the value of the maximum of all the constraint functions, and the values of the design variables are printed.

(2) Three procedures which output information regarding the chosen optimization algorithm, information regarding ANSR simulation if pertinent, and any other information which the user wishes to output are called.

(3) The arrays for storing iteration histories are updated.

(4) Execution is suspended if the number of additional iterations has reached the value specified by the last "run" or "cont" command.

Post-processing control commands are available for doing the following:

(1) Iteration histories of the cost function, the maximum constraint value, and the cumulative cpu-time may be plotted on the terminal screen or routed to a hard-copy plotter.

(2) Iteration histories of the values of any of the design variables may be plotted on the terminal screen or routed to a hard-copy plotter.

(3) The values of the above iteration histories may be written to the file *history* in a format which can be later "included" by DELIGHT.STRUCT.

(4) The file *state* may be generated which contains the percentage violations of all the constraint functions for the current design.

Windows have been set up for plotting on the terminal screen, and they may be listed with the "window_list" command. The process of routing data to the hard-copy plotter involves the generation of data files to be used as input to available hard-copy plotter software external to the DELIGHT.STRUCT system.

2.3.2. Optimization Algorithm Library

One of the optimization algorithms available in the library is a feasible directions algorithm described in more detail in another reference [8]. Feasible directions algorithms are characterized by the property that when a design is "infeasible", or violates constraints, successive designs are monotonically decreasing in max constraint violation, and when a design is feasible successive designs are also feasible and monotonically decreasing in cost. Therefore, these algorithms are desirable for interactive CAD because the user is guaranteed a "better" design with each iteration. The available feasible directions algorithm is programmed as the procedure *feasdir*. This procedure calls two other procedures for each iteration. The first procedure is the procedure *actgrad,* which chooses a direction vector in the design variable space by the Gonzaga-Polak-Trahan [8] method. This direction is chosen so as to be opposite to the minimum vector in the convex hull of the cost and active constraint gradient vectors. The second procedure is the procedure *armijo,* which determines a step length in the chosen direction by the "armijo rule". The step length is chosen so that, if infeasible, a sufficient drop in max constraint violation is made, and if feasible, a sufficient drop in cost is made while remaining feasible. The algorithm automatically insures that all designs satisfy implied "side" constraints or constraints relating to the max and min values of design variables. The algorithm distinguishes between conventionally constraining a function to be less than a specified value, and functionally constraining the max over a given range of a parameter, such as time, of a function to be less than a specified value. This is done because the max function is nondifferentiable; therefore the nondifferentiable functional constraint functions must be handled differently than the differentiable conventional constraint functions.

Another algorithm available in the library is a conjugate gradient algorithm for unconstrained problems. This algorithm has a convergence rate which is quicker than the method of steepest descent but somewhat slower than Newton's method. However, unlike Newton's method, the hessian of the cost function is not required. This algorithm is programmed as the procedure *congrad.* Just as in the feasible directions algorithm, this algorithm calls a direction

finding procedure and a step length determination procedure at each iteration. The direction finding procedure is the procedure *polrib* in which a direction is selected by the Polak-Ribier method [9]. The direction selected is a linear combination of the opposite to the cost gradient vector and the direction from the previous iteration. The step length determination procedure is the procedure *linesearch,* which utilizes a line search algorithm suggested by Luenberger [10]. Basically, this algorithm iteratively approximates the value of the cost function along a line with a quadratic function, and chooses the step length so as to minimize the quadratic function.

Associated with the two algorithms are the procedures *feasdirout* and *congradout.* These procedures are called by the control procedure *interact* at each design iteration for the purpose of outputting information special to their respective algorithms. There are also the files *Afeasdirfile* and *Acongradfile* for the two algorithms. These files are included by the control initialization file *Coptimizefile* for the purpose of initializing their respective algorithm procedures and parameters.

2.3.3. User-Software Interface

All optimization algorithms require the evaluation of the cost and constraint functions. This is done by calling the procedure *state.* For a given set of design variables this procedure does the following five things:

(1) If ANSR simulation is involved, the procedure *ansrsim* is called.

(2) The user-supplied procedure *objective* is called to evaluate the cost function.

(3) The user-supplied procedure *conventional* is called to evaluate each conventional constraint.

(4) The user supplied procedure *functional* is called for each functional constraint and the max over time of each functional constraint is evaluated.

(5) The max value over all the constraints is found.

As mentioned above, if ANSR simulation is involved the procedure *state* calls the procedure *ansrsim*. This procedure manages the relevant ANSR simulation by performing the following four tasks:

(1) The user-supplied procedure *section* is called to evaluate the element design properties corresponding to the given set of design variables.

(2) The Fortran subroutine *resman* is called to check the given set of design variables against the sets of design variables used in the previous two ANSR simulations to see if it is possible to skip ANSR simulation.

(3) ANSR simulation is carried out for a specified number of time steps for each ansrdata file by calling the Fortran subroutine *analys*.

(4) Variables which record the number of ANSR simulations, the number of ANSR simulation time steps, the number of ANSR simulation stiffness matrix reformulations, and the maximum ANSR simulation error energy ratio are updated.

Some optimization algorithms require the computation of the gradients of cost and/or constraint functions with respect to the set of design variables. This is done by calling the procedure *sensitivity*. If the given set of design variables is the same as it was the last time the procedure *sensitivity* was called, then any cost and/or constraint gradients desired in the current call which were also desired in the previous call are simply copied from a stored array. Any remaining cost and/or constraint gradients which are desired are computed by calling the procedure *gradients*. If the user has not supplied the procedure *gradients*, then the procedure *gradients* simply calls the procedure *perturb* by default.

The procedure *perturb* manages the computation of the cost and/or constraint gradients by the method of perturbation or finite differences. For a given set of design variables the gradients of the desired cost and/or constraints are computed by performing the following three steps:

(1) The number of ANSR simulation time steps for each ansrdata file is determined. Thus, if the gradient of a functional constraint at time step i is desired, it is unnecessary to perform ANSR simulation up to the last time step, which may be much greater than i.

(2) The cost and/or constraint functions for which gradients are desired are evaluated for the given set of design variables by calling the procedure *ansrsim,* if pertinent, and by calling the relevant user-supplied procedures *objective, conventional,* or *functional.*

(3) Each component of the set of design variables is perturbed and the relevant cost and/or constraint functions are re-evaluated, differenced from their previous values, and divided by the amount of perturbation to obtain the gradients.

There is also the procedure *ansrsimout* which outputs information about the ANSR simulation, if pertinent, as called by the control procedure *interact* at each design iteration. If ANSR simulation is involved, the control initialization file *Cstartofffile* includes the ANSR simulation initialization file *Sansrsimfile.* This file initializes procedures and parameters used by ANSR simulation as well as restores the previously generated file *response* if available. If the user has not supplied the procedure *gradients,* the control initialization file *Coptimizefile* includes the perturbation initialization file *Sperturbfile.* This file initializes procedures and parameters for the computation of gradients by the perturbation method.

2.4. FRAME

The directories *framefort.d* and *framerattle.d* contain software for the seismic-resistant design of planar, rectangular, braced or unbraced, steel frames. This software is an example of a pre-formulated design problem for a general class of structures. It is hoped that similar software for other general classes of structures will be added to the DELIGHT.STRUCT system in the future. If the user employs this software, he no longer has to supply his own software defining the cost and constraint functions because such functions have been pre-defined. Thus this software allows the frame designer to reduce the amount of programming effort at the cost of surrendering problem flexibility. The actual definition of the cost and constraint functions

used in this software is described in a companion report [11]. It is necessary to interact with some parts of this frame design software and therefore these parts are written in Rattle and found in the directory *framerattle.d.* Other parts which are "non-interactive" have been written in Fortran, which executes faster, and are found in the directory *framefort.d.* The Fortran part has been loaded into the executable program DELIGHT.STRUCT, while the Rattle part has been compiled, combined with the optimum structural design software from the memfile *memstruct,* and stored in the memfile *memframe.* The pre-processing software which helps the user define his particular frame among this general class of frames, and the software interfacing this package to the optimum structural design package will be described. Finally, a desciption of the available software for post-processing the response computed by ANSR simulation will be given. The relationships among the frame software are depicted in Figure 5.

2.4.1. Pre-Processing

Information regarding frame geometry, element types, gravity loads, boundary conditions, element design status, and initial design are input through execution of the large Rattle procedure *finputter.* This procedure asks the user questions about the frame, which it displays graphically on the terminal screen in color. The user transforms the original grid into the proposed frame through his answers to the questions. Thus this information is conveyed to the computer in a straightforward manner.

After obtaining this raw data, the procedure *finputter* calls the subroutine *fprcss.* This subroutine processes the data by performing the following six tasks:

(1) The subroutine *fassum* is called, which reads assumed material constants from the file *assumptions* and reads the ground acceleration record from the file *record.*

(2) The main processing subroutine *fcrnch* is called.

(3) The subroutine *fandat* is called, which generates the relevant ansrdata files.

(4) The subroutine *fdescr* is called, which generates the file *description* which in turn describes the particular design problem for the benefit of the user.

(5) The subroutine *fpass* is called, which generates the binary file *passdata* containing information to be used later for the computation of the cost and constraints and for post-processing.

(6) The subroutine *fstart* is called, which generates the normally user-supplied files *sizefile* and *xfile* describing the problem size and the initial design, respectively.

When the execution of the procedure *finputter* has finished, the user may consult the file *description* for a list of the cost and constraint numbering in his problem. He may employ the procedure *fstructure* to plot on the terminal screen a picture of his frame with element numbering and nodal information. He may also use the procedure *frecord* to plot the ground acceleration record on the terminal screen or route this information to the hard-copy plotter.

2.4.2. Interface To STRUCT

This frame design software provides the procedures which the user normally supplies for the optimum structural design package. The procedures *objective, conventional,* and *functional* for evaluating the cost and constraint functions are set to call the Fortran subroutines *fobjec, fconve,* and *ffunct,* respectively. Since the design of frames involves ANSR simulation, the procedure *section* for computing element design properties from the design variables is set to call the Fortran subroutine *fsectn.*

Two of the commands for controlling the computation process have been modified in order to include the additional initialization needed for the frame design package. To initialize the computation process the command "fstartoff" is given instead of the usual command "startoff". This causes the file *Fstartofffile* to be included to perform the following four tasks:

(1) The Fortran subroutine *finitl* is called, which restores problem data from the binary file *passdata* created by the pre-processor.

(2) The user is asked to type in cost function coefficients from which the amount of simulation necessary to compute the cost is determined.

(3) Design parameters pertinent to the user's particular problem are initialized.

(4) The usual initialization command "startoff" is given.

The command "fonward" should be used instead of the usual command "onward". The only difference between these two commands is that the command "fonward" calls the Fortran subroutine *finitl* for restoring problem data from the binary file *passdata.*

2.4.3. Post-Processing

Post-processing commands are available for displaying the response of the frame under seismic loads as computed by ANSR simulation. The procedure *frameprint* may be used to output data regarding terms in the cost function and non-normalized values of the design variables to the file *frame.* It is also possible to display much of the response graphically. Specifically the user may plot the histories of terms in the global energy balance equation, story drifts, floor accelerations, total frame drifts, element energy dissipation, and element resultant forces. It is also possible to plot hysteresis loops for some elements. These plots may be made on the terminal screen or routed to the hard-copy plotter.

3. USE OF DELIGHT.STRUCT

The software and the commands which the user should give to solve structural engineering problems with the DELIGHT.STRUCT system are described in this section of the report. The description given here treats the DELIGHT.STRUCT system as a "black box", and explains only what the user sees and does during operation. For an explanation of what goes on behind the scenes regarding DELIGHT.STRUCT operation and organization the reader is referred to Section 2 of this report. Subsection 3.1 describes the software supplied by the user before the computation phase can begin. Subsection 3.2 describes the commands which the user may give to manage the computation phase. If the user opts to use the seismic-resistant steel frame design software, the instructions in Subsection 3.3 are applicable. Sample DELIGHT.STRUCT terminal, graphical, and file input and output for the example problem depicted in Figure 6 formulated with or without the frame design software are given in Appendices 4, 5, and 6, respectively.

3.1. USER-SUPPLIED SOFTWARE

The DELIGHT.STRUCT system has been designed so that it is necessary for the user to supply a minimal number of files and Rattle procedures. These files and procedures contain the problem-dependent information which results from the designer's answers to the questions comprising the quantification phase of the design process. This subsection provides explanations of the necessary files and procedures to be supplied by the user. Additional user-supplied software which is optional is also treated. Finally a brief explanation of how to write and debug software in the DELIGHT.STRUCT system is given. All files and procedures written by the user as described in this subsection are normally placed in the working directory *work.d*.

3.1.1. Necessary Files

It is necessary for the user to write a file named *sizefile.* In this file he specifies the values of certain global variables relating to the size of his problem and the amount of ANSR simulation needed. These variables are listed and identified in Appendix 2. The file *sizefile* is made up of Rattle assignment statements whereby each of these variables is given a numerical value.

It is necessary for the user to write a file named *xfile.* In this file he specifies the initial values of the design variables for his problem. Normally each design variable has "geometric" or min and max bounds. The user must normalize each of his variables by linearly mapping them from the interval [min,max] to the interval [-1,1]. This is done so that the optimization algorithms do not weight certain variables more than others. In cases where the user insists on using plus or minus infinity for his max or min value rather than just choosing a large or small number, he could employ an exponential map. The name of the global Rattle vector which contains the values of the design variables is *X.* The file *xfile* consists of Rattle assignment statements whereby each component of the vector *X* is given its initial numerical value between plus and minus one.

If the problem involves ANSR simulation for computation of cost and/or constraints, the user must supply "ansrdata" files. These files provide input for the ANSR program. If the user has constraints corresponding to different loadings, different modellings, etc. he may have more than one ansrdata file. Ansrdata files are named *ansrdata1, ansrdata2,* etc. up to a maximum of six files. The input syntax for an ansrdata file is given in Appendix 3. The element input in the ansrdata files that is dependent on the values of the design variables is arbitrary because this information will be changed automatically according to the values of the design variables.

The user may wish to create some Rattle variables. Such variables may contain information about the design problem which the user may wish to examine periodically during the optimization or simulation processes. These variables may also be problem-dependent coefficients which the user may want to interactively adjust during execution. Furthermore, these variables may contain information to be imported to problem-dependent pre- or post-

processing procedures. To create these variables the user writes a file named *startsetup.* This file contains Rattle "create" statements with syntax "create name" where name is the name of a scalar variable created. For coefficients which may be interactively adjusted, the "parameter" statement should be used instead of the create statement. To create array variables, an "array" statement which has exactly the same syntax as a Fortran dimension statement is used. Typically the dimensions of arrays are set to zero in the file *startsetup* and re-dimensioned later when they are used. All variables thus created in the file *startsetup* are Rattle variables and thus their names should begin with a capital letter followed by small letters.

3.1.2. Necessary Procedures

If the problem involves ANSR simulation, the user must write a procedure named *section.* This procedure has two arguments. The first is a design variable vector x. Sometimes this procedure is called with the vector x equal to the design variable vector X, and other times this procedure is called with the vector x equal to some perturbed or trial design variable vector *xpert* or *xnew.* The second argument in the procedure *section* is a matrix *secprop.* The number of rows in this matrix is equal to the number of elements in the structure to be simulated. The number of columns in the matrix *secprop* is equal to the maximum number of possible element design properties for the elements in the structure. The role of the procedure *section* is to compute the element section property matrix *secprop* corresponding to the given design variable vector x. The ith row contains the possible element design properties for the ith element as derived from the given design variable vector x. For beam-column elements the possible element design properties are cross-sectional area, moment of inertia, strain hardening ratio, and plastic yield moment. For truss elements the possible element design properties are cross-sectional area, strain hardening ratio, tensile yield stress, and compressive failure (yield or buckling) stress.

As an example suppose the ith design variable corresponds to the moment of inertia for elements (beams) m and n, and the jth design variable corresponds to the plastic yield moment

for elements m and n. In the procedure *section* the user would have to do the following three things:

(1) Because the ith and jth design variables are normalized to the interval [-1,1], they have to be scaled to obtain actual values of moment of inertia and plastic yield moment.

(2) The value of moment of inertia derived from the ith design variable is assigned to *secprop(m,2)* and *secprop(n,2)*, and the value of plastic yield moment derived from the jth design variable is assigned to *secprop(m,4)* and *secprop(n,4)*.

(3) The values of cross-sectional area and strain hardening ratio for elements m and n, though not design variables themselves, may be related by some approximate functions to the values of moment of inertia and/or plastic yield moment. The user computes the corresponding value for cross-sectional area and assigns it to *secprop(m,1)* and *secprop(n,1)*, and computes the corresponding value for strain hardening ratio and assigns it to *secprop(m,3)* and *secprop(n,3)*.

Some of the elements in the structure may be unaffected by the values of the design variable vector x. Nevertheless, their corresponding element design properties must be re-assigned by the procedure *section* to their constant values.

Whether or not the problem involves ANSR simulation, the user will have to supply a procedure named *objective*, which has two arguments. The first argument is a given design variable vector x from which this procedure computes the value of the cost function and returns it in the second argument, which is the scalar *cost*. If the user wishes to include conventional constraints in his problem, he must supply a procedure named *conventional*, with three arguments. The first argument is a given scalar i and the second argument is a given design variable vector x from which this procedure computes the value of the *ith* conventional constraint function and returns it in the third argument, the scalar *ineq*. If the user wishes to include functional constraints in his problem, he must supply a procedure named *functional*, with four arguments. The first argument is a given scalar i, the third argument is a given scalar *steps*, and the second argument is a given design variable vector x from which this procedure computes first

steps values of the *ith* functional constraint, and returns these values in the fourth argument, the vector *fineq*. To carry out these computations it may be useful for the user to write other procedures which are called by the procedures *objective, conventional,* or *functional.* It is suggested that the user normalize the constraint functions so that their range is in the interval [-1,0] when x is feasible, and the interval [0,+infinity] when x is infeasible. Furthermore, the user should normalize the cost function so that its range has the same order of magnitude as the constraint functions.

The cost and constraints are often functions of the response from an ANSR simulation made prior to calling the procedures *objective, conventional,* or *functional.* This response is stored in the large vector *Resp,* which must be imported to the procedures *objective, conventional,* or *functional.* The user must compute the values of cost or constraints from values in this vector *Resp.* The first values stored in the vector *Resp* are the values of the design variables used in the ANSR simulation generating the response. Following these, the response histories corresponding to the ANSR simulation for each ansrdata file appear. The order of storage of response histories for an ansrdata file is as follows:

(1) A zero history

(2) Global energy balance histories: quake input energy, work done by loads, elastic strain energy, inelastic energy dissipation, damped energy dissipation, kinetic energy, energy error

(3) Histories of relative displacements, relative velocities, and absolute accelerations in global x-direction for each node

(4) Histories of relative displacements, relative velocities, and absolute accelerations in global y-direction for each node

(5) Histories of relative displacements, relative velocities, and absolute accelerations in global z-direction for each node

(6) Element response histories

Not all of these histories are stored for every ansrdata file. In each ansrdata file the user specifies whether global energies are to be stored, which nodal motions for which nodes are to be stored, and which element responses for which elements are to be stored. Therefore, in the procedures *objective, conventional,* or *functional* the user must determine the address of the desired quantity in the vector *Resp* before it can be used for computation of the cost or of a constraint.

3.1.3. Optional Procedures

Most optimization algorithms require the evaluation of gradients of cost and constraint functions with respect to the design variables. The DELIGHT.STRUCT system currently computes such gradients by perturbation. However, the particular cost and constraint functions in the problem may be in a form in which the user may supply a more efficient scheme for computation of their gradients. If this is possible, the user should write a procedure named *gradients* with five arguments. The first argument is a given scalar *nact* which is the number of "active" constraints for which gradients are desired. The second argument is a given vector *list1* of dimension nact, the third argument is a given vector *list2* of dimension nact, and the fourth argument is a given design variable vector x. The last argument is the matrix *jacob* with dimensions equal to *nact* by the number of design variables. The procedure *gradients* computes the gradients of the active constraints specified in *list1* and *list2* with respect to the design variables evaluated at the design variable vector x and stores such gradients row-wise in the matrix *jacob.* If the ith element of *list1* equals zero and the ith element of *list2* equals zero, the gradient of the cost function should be computed and stored in the ith row of *jacob.* If the ith element of *list1* equals positive j and the ith element of *list2* equals zero, the gradient of the jth conventional constraint should be computed and stored in the ith row of *jacob.* If the ith element if *list1* equals positive j and the ith element of *list2* equals positive k, the gradient of the jth functional constraint at time step k should be computed and stored in the ith row of *jacob.*

If the user wishes to print out certain information particular to his problem at each design iteration, he must write a procedure named *uoutput* with no arguments. In this procedure he imports Rattle variables or accesses local procedure variables containing the information to be printed, and prints out their values in the format of his choice.

The user may wish to write procedures to post-process the results from his problem. Such procedures could display information in files, to the terminal screen, graphically as plots on the terminal screen, or in files compatible with hard-copy plotter software external to the DELIGHT.STRUCT system. If there is any degree of generality to the user's problem, he may wish to write procedures for pre-processing input information. Such procedures are useful if the problem is going to be done more than once with slightly different data. Pre-processing procedures read the values of variables which change from problem to problem from the terminal or from files. Then they generate some or all of the normally user-supplied files mentioned previously saving the user a certain amount of work. These pre-processors can be made quite friendly if they interactively ask the user questions and employ graphics.

The user may also wish to utilize the DELIGHT.STRUCT system in its sophisticated "hand-held calculator" mode to make "scratchpad" computations before, during, or after the optimization phase of his problem. Such computations are useful for obtaining a preliminary design, for determining appropriate adjustments for parameters, and for interpreting results. The user may construct variables and procedures which aid in these computations. Such scratchpad software is typically very problem-dependent.

3.1.4. Writing And Debugging Procedures

A good methodology to adopt for writing and debugging large (more than 20 lines) procedures is as follows:

(1) Leave the DELIGHT.STRUCT program with the "system" command, write the procedure in a file using the UNIX editor, and return to DELIGHT.STRUCT with the "exit" command.

(2) Cause DELIGHT.STRUCT to compile the procedure by "including" the new file. Any compiling errors will appear on the terminal screen with a diagnostic message.

(3) Leave DELIGHT.STRUCT, re-edit the file to correct the compiling errors, return to DELIGHT.STRUCT, and re-compile until a successful compilation has been made.

(4) Prepare test data and cause the procedure to execute by simply "calling" it. Most execution errors will cause the execution to interrupt with a "run-time error" message instead of aborting.

(5) Give the command "trace" to find out on which line execution was interrupted, "enter" the procedure, and display the values of local variables and arrays. From these values the source of the execution error can usually be determined after which the interrupt level should be "reset" to one.

(6) Leave DELIGHT.STRUCT, re-edit the file to correct the source of the execution error, return to DELIGHT.STRUCT, re-compile, and re-execute until a successful execution has been made.

An alternative methodology for writing and debugging small procedures using the DELIGHT "history" features is as follows:

(1) Prepare test data for the procedure.

(2) Begin typing the body of the procedure (without the "procedure", "inherited array", or "import" statements) directly from the terminal. The statements or statement blocks are compiled and executed as the user types them, and any compilation or execution errors will show up immediately.

(3) When a compilation error appears simply re-type the corrected statement.

(4) When a run-time execution error appears, discover its source by displaying the values of variables. Then reset the interrupt level to one.

(5) To correct the run-time execution error, obtain the history numbers of previous statements with the "history" command, re-execute those history numbers with the "!" command, and re-type the corrected statement.

(6) When the execution is error-free, "edit the history" taking out unnecessary statements and adding "procedure", "inherited array", and "import" statements at the top. Then "write out" the contents of the history editor to a file.

After the user has completed writing and debugging his software, it is useful to create the memfile *memstart* in which the DELIGHT.STRUCT software together with his compiled procedures is stored. The user can simply start from the memfile *memstart* when he begins the computation phase. This is especially useful if he is going to run more than one design problem because he won't have to re-compile all that software each time. The memfile *memstart* can be constructed by giving the UNIX command "make.memstart", which starts DELIGHT.STRUCT restoring from the memfile *memstruct,* takes input from the file *startsetup,* and writes out any errors or messages to the file *make.errors.* The user must append to the file *startsetup* DELIGHT "include" statements for each of the files containing the user-supplied procedures before the UNIX command "make.memstart" is given. Because there is no load-linkage in DELIGHT, all procedures called by a given procedure must already exist. Therefore the order in the file *startsetup* in which the procedure files are included is important. If the "include_and_print" statement is used instead of the ordinary include statement, a record of when each file was included will be recorded in the file *make.errors.* The final lines in the file *startsetup* are usually the DELIGHT statements, "clear_time", "store memstart 'starting memfile'", and "quit".

3.2. COMMANDS FOR COMPUTATION

In this subsection, the commands given by the user to carry out the computation phase of design are described. First, the business for getting started and initializing will be treated. Second, some of the possible DELIGHT commands for carrying out optimization will be discussed. Third, processing and storage of results will be described. Some of the commands for getting started and storing results are described for the UNIX operating system and therefore may be different for other operating systems.

3.2.1. 1. Getting Started

The user performs his work in the directory *work.d.* In fact if the DELIGHT.STRUCT system resides in a central location, users need only the directory *work.d.* This directory contains a copy of the UNIX file *make.memstart* as mentioned in Subsection 3.1. A copy of the file *openhdtl* which consists of a list of names of directories containing DELIGHT.STRUCT software is also found in the directory *work.d.* The user should put the directory *delight.d* at the end of his UNIX csh path. He should also create a UNIX alias equating "work" to changing to the directory *work.d,* and a UNIX alias equating "go" to DELIGHT.STRUCT.

To get started the user gives the UNIX command "work" which puts him in the directory *work.d.* Next he gives the UNIX command "go" which starts the DELIGHT.STRUCT program and restores software from the default memfile *memfile.* At this point the user-supplied software should be developed as described in the Subsection 3.1. After the files containing this software have been written and debugged, the user should stop the DELIGHT.STRUCT program with the DELIGHT command "quit". Then he should give the UNIX command "make.memstart" to create the memfile *memstart.* When this is done, the user is at the point where he may begin the computation phase of the design process. If he is going to run more than one design problem which uses the same software, he begins each time at this point.

The user starts the DELIGHT.STRUCT program restoring from the memfile *memstart* by giving the UNIX command "go memstart". At this point he may wish to execute scratchpad or

pre-processing software which he has written. Next he gives the initialization command "startoff". If ANSR simulation is involved he will be asked if the file *response* for the current design already exists, and he should give the proper answer. The user normally gives the command "simulate" at this point. This command causes the cost and constraints for the current design to be computed. At this point if the user wishes to proceed with optimization, he gives the initialization command "optimize". The user will be asked which optimization algorithm he wishes to use, whether he has supplied his own gradient computation scheme, and the maximum number of design iterations he expects to complete. Proper initializations are made depending on his answers.

3.2.2. Optimization Process

To start the optimization algorithm the command "run n" is given where the optional argument n is the number of design iterations to be performed. After the n iterations are completed, execution will be suspended at the main breakpoint in the algorithm. At this point the user may interact with the results. When he is ready to continue execution of the algorithm, he gives the command "cont m" which causes m additional design iterations to be performed. There are other breakpoints in the algorithms besides the main breakpoint. If the user hits the "break" key once during execution of the design iterations, the execution will suspend when it reaches the next breakpoint, and a message will appear on the terminal screen telling the user at which breakpoint he is. When the user is ready to resume execution from this breakpoint, he gives the command "resume". It is also possible to interrupt execution between breakpoints. If the user hits the "break" key twice during execution, the execution is suspended no matter where it is. The user should then give the command "trace" to find out where he is in the execution. When he is ready to resume execution again, he gives the command "resume".

When the user has suspended execution, he may interact with the results. He may display the current values of variables with the "display" command. He may display the current values of variables designated as parameters with the "dp" command. He may "enter" a

procedure, "display" the current values of local variables, and "leave" the procedure. He may also print the current values of the elements of arrays with the "printv" command. He may display the amount of cpu-time used in the various procedures with the "dt" command. He may even display results graphically with the "plot" command. The user may wish to change the values of some variables, particularly those which have been designated as parameters. This may be done with Rattle assignment statements. Some of the DELIGHT.STRUCT variables and parameters with which he may want to interact are described in Appendix 2. In order to aid the designer in making decisions regarding which variables to modify, how many more iterations to carry out, etc. he may perform scratchpad computations with DELIGHT in its sophisticated "hand-held calculator" mode.

The commands given by the user while execution of the algorithm is suspended are echoed to the file *dialogue.* Anything following a "#" symbol is taken as a comment, which provides a means whereby the user may record an explanation in the file *dialogue* of what he is doing during the design process. The cumulative cpu-time for each design iteration is also measured and stored. If the user decides he doesn't want a group of commands to be recorded in the file *dialogue* nor to be taken into account in the running total cpu-time, he precedes them with the command "stop_opt" and follows them with the command "start_opt". If the user decides he is finished with the optimization process, he gives the commands "stop_opt" and "reset".

When execution has been suspended at the main breakpoint in the algorithm, the user may wish to stop and continue execution at a later date. This will be possible if he gives the commands "stop_opt", "simulate", "saveall", and "quit". The command "saveall" causes the values of DELIGHT variables to be stored into the memfile *memprob,* the values of the ANSR simulation response vector *Resp* to be stored in the file *response,* and the values of the current design variable vector X to be stored in the file *xfile.* When he is ready to begin again at a later date, he should make a backup copy of the memfile *memprob,* start DELIGHT.STRUCT with the UNIX command "go memprob", give the command "onward" which restores from the

stored files, and finally give the command "start_opt". After doing this, he may continue where he previously left off.

For larger problems the execution of each design iteration may take a long time, and the user may wish to operate in a semi-interactive mode which has come to be known as the "pajama" mode. In this mode the user initializes his problem with the "startoff" and "optimize" commands just as in the usual fully-interactive mode. Then he gives the command "pajama". This will cause the DELIGHT.STRUCT program to take commands from the file also named *pajama* the next time execution suspends. The user then gives the command "sleep n ; run m". This will cause the DELIGHT.STRUCT program to remain idle for n seconds after which m design iterations will be executed. After giving these commands the user may put the DELIGHT.STRUCT program in the background, which is done with the UNIX commands "cntl-z" and "bg". He may then log out and go home. Thus the user may give these commands during the day causing the program to "sleep" for n seconds idly until say midnight when the computing rates go down. At midnight the program automatically begins execution of the design iterations after which the execution suspends and receives commands from the file *pajama.* Typically the user places the commands "stop_opt", "simulate", "saveall", and "quit" in his *pajama* file. When the user returns in the morning he finds his job completed. He may start the execution where it left off with the commands "go memprob", "onward", and "start_opt" and interact to find out what happened over night. Then he may set up further pajama mode iterations for the following night with the commands "pajama", "sleep n ; cont m", "cntl-z", and "bg". The name "pajama mode" obviously originated from the fact that this method was used for overnight computation. With this method the user can still interact without having to sit and wait for a long time at a terminal while simulation or design iterations are being executed.

3.2.3. Storage Of Results

When the user thinks he has completed enough design iterations, he should save and store results. Histories of cost, maximum constraint value, cumulative cpu-time, and design variables vs iteration have been stored throughout the optimization process. These histories may be plotted graphically on the terminal screen. The terminal must be initialized for graphics with the "terminal" and "grinit" commands. The available windows may be listed with the "window_list" command. The user must select one of these windows with the "window" command. The command "graph_cost from n to m" will create a plot in the selected window of the cost history from iteration n to iteration m. The arguments n and m are optional with default values being zero and the current iteration number, respectively. Similarly the commands "graph_psi", "graph_time", and "graph_x" can be used to obtain plots of maximum constraint value, cumulative cpu-time, and design variables vs iteration. By replacing the characters "graph" with the characters "plot" in these commands, files are created which are compatible with available hard-copy plotter software. The command "savehist" causes these histories to be saved in the file *history* in a format which is "includable" by DELIGHT.STRUCT. The command "printstate" causes the value of the cost and the percentage violation of each of the constraints to be printed out to the file *state.* In addition to these commands, the user may employ his own post-processing software commands.

The commands "saveall", "savehist", "printstate", etc. generate files in the working directory *work.d* containing the results of the design process. It is suggested that the user create a directory named *save.d* in which he can store such results so that the working directory may be cleared and made ready for another design problem. The user may wish to save copies of any or all of the files *dialogue, history, xfile, state, response, memprob, sizefile,* the ansrdata files, and the hard-copy plotter files in the directory *save.d.* He may also wish to save copies of the user-supplied procedures for his problem.

3.3. SEISMIC-RESISTANT DESIGN OF FRAMES

If the user elects to utilize the existing software for seismic-resistant design of planar rectangular braced or unbraced steel frames, he does not have to supply much of the software mentioned earlier. The procedures *section, objective, conventional,* and *functional* as well as pre- and post-processing procedures have already been written in a general fashion. Of course the user may add further scratchpad procedures if he desires. The files *sizefile* and *xfile* and the ansrdata files are generated by the pre-processing software from a minimal amount of information given by the user in a painless manner which describes his specific frame from among a large number of possible frames. The software has been set up to divide the design process into two distinct phases. These are the problem definition phase and the computation phase. These phases are described in this subsection.

3.3.1. Problem Definition Phase

It is suggested that the frame designer create another working directory named *framework.d.* It is convenient to create a UNIX alias so that the UNIX command "fwork" will put the user in this directory. If DELIGHT.STRUCT resides in a central location, the directory *framework.d* is the only directory the frame designer needs. This directory should contain a copy of the file *openhdtl* consisting of names of software directories. A copy of the file *assumptions* should also reside in this directory. This file lists assumed parameter values for quantities such as elastic modulus, damping ratio, and coefficients for relationships among element section properties. If the user wishes to replace these values with others, he can do so by simply editing the file. A copy of the file *record* should also exist in the directory *framework.d.* This file contains the digitized ground acceleration record to be used for the design. The user is free to replace this file with another ground acceleration record file with similar format. To start the DELIGHT.STRUCT program, the user should give the UNIX command "go '<memframe>'", which restores software from the memfile *memframe.*

After starting the DELIGHT.STRUCT program, the user is ready to execute the pre-processor. He should be on a color graphics terminal, and he should initialize the terminal for graphics with the "terminal" command. Next he gives the command "finput". He is asked how many stories and bays in the frame and the values of the respective story heights and bay widths. The frame grid is then displayed on the terminal screen with nodal and element numbering. He can place cross-bracing in panels of his choice in the grid. Such bracing is graphically displayed and numbered. He can "erase" elements and nodes of his choice. He can designate girders of his choice as "shear link elements" for eccentrically braced frames or "dissipator elements" for base isolation systems, and he can designate columns of his choice as "rubber bearing elements" for base isolation systems. All of this element information is displayed graphically. In this way the geometry of the particular frame is easily conveyed to the computer. The user may then place uniform gravity loads on girders of his choice and downward point gravity loads on nodes of his choice. He may also give nodal constraint codes for nodes of his choice. Again this nodal information is displayed graphically. The user may specify certain elements as elements whose size is not to change during the design process. He may specify certain elements as elements which have no constraints on their behavior (for example the frame may have sufficient symmetry so that constraints on some elements may be redundant). He may also specify groups of elements to have the same size throughout the design process. This information is also displayed graphically. Finally the user specifies the initial sizes of the elements in the frame, and the corresponding elements are erased as he does so. This interactive graphical pre-processor has sufficient intelligence to detect many illogical responses from the user, and in such cases the user is asked to try the particular response again. There is also an "undo" capability with which the user may undo previous accidental responses in each part.

The interactive graphical pre-processor sends all this information to a processor which generates the files *sizefile, xfile, ansrdata1, ansrdata2,* and *ansrdata3.* The processor also generates the file *description* which identifies and numbers the constraints in the particular problem

for the benefit of the user. A final file generated by the processor is the binary file *passdata,* which contains all other information about the user's frame needed later for computation of cost and constraints.

The user may display the frame as defined by the pre-processor on the terminal screen for inspection with the command "fstruct". He may also select a window with the "window" command and display for inspection a plot of the ground motion accelerogram with the command "graph_frecord". The record may also be written to files compatible with the hard-copy plotter software with the command "plot_frecord". If the user is satisfied with the frame as described by this problem definition phase, he may stop the DELIGHT.STRUCT program with the "quit" command and start the computation phase at a later time.

3.3.2. Computation Phase

To begin the computation phase the user again goes to the directory *framework.d* with the UNIX command "fwork" and starts the DELIGHT.STRUCT program with the UNIX command "go '<memframe>'". The initialization command "fstartoff" is then given instead of the usual command "startoff". The user will be asked questions about his choice of cost function coefficients. Afterwards the user may wish to perform a simulation by giving the usual command "simulate", and/or he may wish to initialize for optimization by giving the usual command "optimize".

To carry out the optimization process the commands are the same as the usual commands for other structural problems with the single exception that the command "onward" should be replaced by the command "fonward". The files *dialogue, memprob, response,* and *xfile* are likewise created and used in the same way. Additional frame design variables and parameters with which the user may wish to interact are described in Appendix 2.

There are frame post-processing commands available in addition to the usual post-processing commands for structural problems. The command "fprint" causes information regarding the terms in the cost function and the values of design variables in terms of member

sizes to be printed out to the file *frame*. Plots on the terminal screen of moderate quake story drifts, moderate quake floor accelerations, and severe quake total structure sway vs time can be made for the current design with the commands "graph_fdrift", "graph_faccel", and "graph_fsway", respectively. A plot of terms in the global energy balance equation vs time can be made with the command "graph_fenergy". Plots of severe quake element energy dissipation and moderate quake element resultant forces vs time can be made with the commands "graph_felenergy" and "graph_felforce", respectively. Plots of hysteresis loops for braces, shear link elements, or base-isolation dissipator elements can be made with the command "graph_felhyst". In all the above mentioned "graph" commands, one can instead generate plot files compatible with the hard-copy plotter software by replacing the characters "graph" with the characters "plot".

It is suggested that the user create the directory *framesave.d* analogous to the usual directory *save.d* where results from seismic-resistant frame design problems can be stored. In addition to the usual structural design files the user may wish to save copies of any or all of the files *frame, description, assumptions, record,* and *passdata.*

4. FUTURE DEVELOPMENT

The DELIGHT.STRUCT system as it now stands is a research tool. Before it could be used in an industrial environment, future development must occur in two main areas. First, the libraries in the system must be expanded, thus increasing the power of the system for solving a wider range of problems. An explanation of how to expand the libraries will be given in in Subsection 4.1 which follows. Second, there are many specific improvements which should be made in the operation and organization of the system itself. However, the incorporation of these improvements should not be attempted until sufficient experience has been gained with the present system in order to pinpoint the exact sources of inefficiency. A discussion of areas in which improvements could be made is found in Subsection 4.2.

4.1. LIBRARY EXPANSION

The organization of DELIGHT.STRUCT is such that users may expand and adapt the libraries of software according to their needs and interests. Specifically, the libraries which may be expanded include the library of basic DELIGHT commands, the library of ANSR elements, the library of optimization algorithms, and the library of general classes of structural design problems. Some of these libraries consist of Fortran subroutines which are loaded directly into the executable file DELIGHT.STRUCT. Other libraries consist of Rattle files which are compiled into memfiles. Still other libraries consist of Rattle files which can be included. Each of these libraries currently contains software which the programmer can use as an example when expanding them.

4.1.1. DELIGHT Commands

The library of DELIGHT commands exists in the central directory containing DELIGHT software. This software is in the form of files written in Rattle source code. Each file provides the software for a basic DELIGHT feature. Files and thus features are being added to this directory continually. The latest features which have been added to this directory are described

in the file *features* also in this directory. Some of the files in this directory were selected to be included in the compiled software in the DELIGHT.STRUCT memfiles *memfile, memstruct,* and *memframe.* If the user wishes to use one of the features found in a file that was not included in these memfiles, he must simply "include" the file after starting the DELIGHT.STRUCT program.

4.1.2. ANSR Elements

Currently the library of ANSR elements contains a three-dimensional elasto-plastic parallel-component truss element and a two-dimensional lumped-plasticity parallel-component beam-column element. Future additional elements may include a plane-stress, plane-strain element, a plate element, a shell element, a solid element, etc. Specialized elements such as a shear link element or a hysteretic dissipator element may also be added. Addition of an element involves adding the six subroutines *inel, stif, resp, out, stor,* and *mdfl* for that element to the directory *element.d,* and removing the corresponding six dummy subroutines from the file *eldum.f* also in the directory *element.d.* Then one must go to the directory *delight.d* and reload the executable file DELIGHT.STRUCT by giving the UNIX command "load.delight". The memfiles must be re-made by giving the UNIX command "make.memfile" in the directory *delight.d,* followed by the UNIX command "make.memstruct" in the directory *structrattle.d,* followed by the UNIX command "make.memframe" in the directory *framerattle.d.*

Four of the six element subroutines, namely *inel, stif, resp,* and *out,* are written and added in exactly the same way as elements are added to the original ANSR-I program, with two exceptions. These exceptions result from the fact that the original ANSR-I program was modified to form the mini-ANSR version, which is capable of computing energy response and is compatible with virtual memory computing systems. The first exception is that the element information must be separated into element "integer" information and element "real" information. This means the common block *infel* becomes the common blocks *infeli* and *infelr,* the argument vector *coms* becomes the argument vectors *icoms* and *coms,* the argument scalar *ninfc*

becomes the argument scalars *ninfci* and *ninfcr,* and the local vector *com* becomes the local vectors *icom* and *com.* The second exception is that the first four members of the element real information common block must be the following quantities, which must be computed in the subroutine *resp:*

(1) *eelas* = elastic strain energy in the element

(2) *einel* = cumulative inelastic energy dissipated in the element

(3) *fee* = vector of components of equivalent non-geometric element resistence forces in local coordinates for the element

(4) *fdd* = vector of components of equivalent damping nodal forces in local coordinates for the element.

The user must also add the element subroutine *mdfl.* The arguments for this subroutine are:

(1) *icoms* = element integer information vector

(2) *coms* = element real information vector

(3) *ninfci* = element integer information size

(4) *ninfcr* = element real information size

(5) *secpro* = matrix of element design properties

(6) *nels*

= number of elements in the structural model

The transfer of element information is made in the same way as in the other ANSR element subroutines. The task of this subroutine is to modify the members of the element real common block *infelr* according to the values of the element design properties found in the matrix *secpro.*

The user must also add the element subroutine *stor.* The arguments of this subroutine are *icoms, coms, ninfci,* and *ninfcr* which have the same meanings as defined above. In addition to the usual element common blocks *infeli* and *infelr,* the common blocks *stores* and *bigres* must

also be included in this subroutine. The transfer of element information is made in the same way as in the other ANSR element subroutines. The task of this subroutine is store values of certain members of the element common block *infelr* into appropriate locations of the vector *resp* found in the common block *bigres.*

4.1.3. Optimization Algorithms

The library of optimization algorithms currently contains a feasible-directions algorithm for constrained optimization with possible functional constraints and a conjugate-gradient algorithm for unconstrained minimization. A new algorithm for this library must be written as a Rattle procedure with no arguments. The algorithm may be modularized to call other Rattle procedures. All files associated with the new algorithm should have names which begin with the capital letter A and should be placed in the directory *structrattle.d.* Typically optimization algorithms iterate over successively improved designs. A main breakpoint in the design iteration loop should be chosen and established by calling the procedure *interact* which has no arguments.

Optimization algorithms must somewhere require the evaluation of cost and constraint functions. This may be done by calling the user-supplied procedures *objective, conventional,* or *functional* directly. If the values of these functions depend on the response determined from ANSR simulation, the procedure *ansrsim* should be called first. The procedure *ansrsim* has two arguments: a design variable vector x and the scalar *steps,* which is the number of time steps for which simulation is desired. As an alternative to calling the procedure *ansrsim* and then the procedures *objective, conventional,* or *functional* directly, an algorithm may call the procedure *state.* The first argument of this procedure is a given design variable vector x from which it returns the value of the cost function in the second argument *cost* and returns the value of the maximum of all the constraint functions in the third argument *psi.* Many optimization algorithms also require evaluation of the gradients of the cost and constraint functions with respect to the design variables. This may be done by calling the procedure *sensitivity,* which has the

same arguments as the optional user-supplied procedure *gradients* described earlier.

After supplying files containing Rattle source code for the new algorithm in the directory *structrattle.d,* the following details must be carried out to complete expansion of the algorithm library:

(1) An output procedure with no arguments to be called at each design iteration should be written. This procedure prints out useful information about the performance of the algorithm.

(2) Any interactive parameters used by the algorithm should be created in the file *structsetup* and imported to the algorithm procedure. The file *structsetup* should also be modified to include the files containing the algorithm procedure, all procedures called by the algorithm procedure, and the algorithm output procedure.

(3) An initialization file for the new algorithm should be written which sets the procedure *algo* to call the algorithm procedure, sets the procedure *aoutput* to call the algorithm output procedure, and initializes the algorithm parameters with "parameter" statements.

(4) The file *Coptimizefile* should be modified to include the new algorithm among the choice of algorithms, and the initialization file for the new algorithm should be included if it is chosen.

(5) The UNIX command "make.memstruct" should be given in the directory *structrattle.d* and the UNIX command "make.memframe" should be given in the directory *framerattle.d* causing the memfiles *memstruct* and *memframe* to be re-compiled.

4.1.4. Classes Of Structural Problems

The library of software for classes of structural design problems currently contains software for the seismic-resistant design of steel frames. Examples of other structural design problem classes may include design of steel bridge decks, design of concrete dams, etc. The seismic-resistant frame design software should be used as an example in developing software

for other classes. Such software pre-specifies the normally user-supplied software in a somewhat general way. Thus the software should supply the procedures *section, objective, conventional,* and *functional.* The software should include a pre-processor which takes input from the user which distinguishes his specific problem from other possible problems in the class and generates the files *sizefile, xfile,* and the ansrdata files. Post-processing software should also be included which allows the user to interpret results quickly and efficiently.

The software may consist of a directory containing Fortran code as well as a directory containing Rattle code. The following work must be done to interface the Fortran code to the DELIGHT.STRUCT system:

(1) The built-in part of the DELIGHT.STRUCT system must be modified. This is done by editing the files *builttop, builtmid,* and *builtnam* in the directory *delight.d* to include the name and argument information of the new subroutines. Then the file *builtn.r* in the directory *delight.d* should be re-compiled with the rat4 compiler.

(2) The DELIGHT.STRUCT program should be re-loaded by adding the names of the files containing the compiled Fortran code to the UNIX file *load.delight* in the directory *delight.d* and giving the UNIX command "load.delight".

(3) The memfiles *memfile* and *memstruct* should be re-made by giving the UNIX commands "make.memfile" in the directory *delight.d* and "make.memstruct" in the directory *structrattle.d.*

To interface the Rattle code with the DELIGHT.STRUCT system a setup *file* and a UNIX file should be constructed for making a memfile which contains the new Rattle code and the software from the memfile *memstruct.* After interfacing the new Fortran and Rattle code, directories for working with the new software and for saving the corresponding results should be formed. The *openhdtl* file in the working directory should contain the name of the directory with the new Rattle code.

4.2. IMPROVEMENT

After working with the DELIGHT.STRUCT system for a period of time the author has been able to recognize some of the deficiencies of the system. Alternatives for correcting such deficiencies have also been considered. In this subsection possible improvements on the DELIGHT.STRUCT system are treated briefly. First the problem of DELIGHT.STRUCT speed and size will be considered. Then, a closer look at how gradients are computed for the sensitivity analysis will be made. Suggestions for improving the simulation package will be given. A brief discussion regarding optimization algorithms will follow. Then, the optimization-simulation interface will be examined. Finally some mention is made of enhancing the pre- and post-processing power of the system.

4.2.1. DELIGHT.STRUCT Speed And Size

The primary goals in the development of the DELIGHT system did not include optimal execution speed and minimal program size. It has been discovered that the cpu-time required for executing many loops of compiled DELIGHT statements is more than an order of magnitude larger than the cpu-time required for executing the same number of loops of compiled Fortran statements. This is a serious drawback. The current size of the DELIGHT.STRUCT program is a little over four mega-bytes. This means that its use practically requires a virtual-memory system. There are two large data spaces in the system which are the main culprits for its size. One of the spaces is the simulation response storage vector *Resp* which currently has a dimension of 300000 double-precision numbers accounting for over two mega-bytes of space. This array was nearly filled when designing a seismic-resistant steel frame with 20 degrees of freedom using 1100 analysis time steps. The other large data space is the DELIGHT data space, which contains compiled procedure code and values of DELIGHT variables. This is the data which can be written out into binary memfiles. It can have a size up to a little under one mega-byte, and well over half of this maximum size has been used in design problems encountered thus far.

One alternative to correct the slow execution speed of the DELIGHT program is to make a systematic overhaul of the program with efficiency as the primary goal. Such an overhaul is planned by its creator in the near future. A systematic examination of the data spaces would reveal that much of the stored information is not necessary to the design process, and schemes could be incorporated which cut down the large size of the system.

Another alternative solution to the slow execution speed and the large size of the system is based on a completely different organization. The current organization of the system is such that the ANSR simulation package is loaded together with optimization algorithms, the processors, the interfaces, and the management package DELIGHT. The alternative is to organize so that each of these modules is a stand-alone executable program. In such an organization the management system would not be DELIGHT, but rather the operating system of the computer. Much of the interactive power and friendliness of the DELIGHT system would be lost in this organization. A data-base management system would be necessary to handle the data input/output needed to pass from one stand-alone module to another. The space needed to operate would have to be only as large as the largest module in the package. Furthermore, modules could be written in Fortran for faster execution. Finally, modules which involve large amounts of number-crunching could be executed on array-processors while other modules could be executed on "slower" hardware.

4.2.2. Gradient Computation

In order to compute the sensitivity of the cost and constraint functions with respect to changes in the design variables, the author has resorted to a "brute force" finite-difference scheme up to this point. This is extremely inefficient because it ignores much of the useful information generated during simulation. However, the finite-difference scheme can be programmed external to the simulation package whereas more efficient schemes involve extensive modification of the simulation package itself. One efficient scheme which has been used by researchers is based on an adjoint method [12].

Another alternative gradient computation scheme is based on implicit differentiation of the structural equation of motion with respect to a design variable. This results in an equation for the derivatives of the displacements which has the same form as the original equation of motion except that there is a new right-hand-side load vector. This means that at each time step in the simulation the gradients of the displacements may be obtained by simply resolving the already triangularized tangent dynamic stiffness matrix for each design variable. The reduction in the number of multiplications in this scheme from the finite-difference scheme is by a factor approximately equal to the number of degrees of freedom in the system. However, the storage requirements may increase in order to store these gradient histories along with the usual simulation response histories.

4.2.3. Simulation Package

Much of the computational effort spent in the optimization process occurs during simulation. Simulation approximations may be incorporated into the DELIGHT.STRUCT system which lessen these lengthy computations. One obvious suggestion takes advantage of the fact that after gradients have been computed for a given design, the simulation response for other designs may be approximated by the first two terms in the Taylor's series expansion. Another suggestion is to incorporate iterative rather than direct equation solvers in the simulation package because the response from previous designs provides a good starting point for the iterative solvers and because the tolerance for simulation accuracy depends on how close one is to the optimum.

One drawback to the ANSR simulation package is its lack of flexibility. A more ideal package would allow the user to interactively construct "runstreams" which dictate the solution strategy. These runstreams consist of combinations of the modularized operations normally found in structural simulation packages. One operation might be to formulate the tangent stiffness matrix while another might be to solve the eigenproblem for the natural frequencies and mode shapes. This runstream philosophy is more consistent with the general philosophy of

modularity in the DELIGHT.STRUCT system.

4.2.4. Optimization Algorithms

An extensive library of optimization algorithms has been developed for the DELIGHT system. This library was not incorporated into the DELIGHT.STRUCT system because the format of the algorithms was inefficient for structural engineering design problems. Nevertheless the DELIGHT.STRUCT system and the library should be modified so that the two could be brought together. In such a case the library of optimization algorithms would be moved from the directory *structrattle.d* to a central directory used by other groups of engineers in addition to structural engineers.

Many algorithms utilize some sort of line-search which usually accounts for a large percentage of the simulation time. The line-search could be made more efficient by incorporating information from available gradients as well as from previous simulations along the line. Furthermore a hierarchy of analyses could be established such that is a point in the line-search failed from the results of simpler analyses, more complex analyses at that point could be bypassed.

4.2.5. Optimization-Simulation Interface

The interface from optimization to simulation centers on the task of setting up a simulation for the new set of design variables computed by the optimization algorithm. Currently this interface has been constructed so as to allow certain element section properties to be designated as design variables. The ANSR simulation program reads input data, processes it, stores it into common blocks, then calls this interface to modify the data in the common blocks according to the design variables before proceding with simulation. An alternative scheme would be to read input data, modify it according to the design variables, then process it, store it into common blocks and proceed with simulation. Such a scheme would allow any piece of input information to be a design variable. By considering nodal coordinates as design variables, some interesting

shape optimization problems could be treated. This scheme tends to bring the interface up to a level of generality which exploits the full capabilities of the ANSR simulation package. This scheme would also be less complicated since knowledge of the common blocks in ANSR is not needed.

The interface from simulation to optimization centers on the task of extracting the data from the simulation necessary for the computation of the cost and constraint functions. The current interface requires the user to do alot of bookkeeping to keep track of the addresses of various simulation data stored in the vector *Resp.* Use of a friendly data manager would be preferable.

4.2.6. Processing Packages

The modularity of the DELIGHT.STRUCT system allows the possibility of interfacing some of the sophisticated pre- and post-processing software packages currently available. Such software enhances communication between man and machine. In particular graphics software coupled with modern hardware is sure to aid in achieving the real goal of design efficiency if utilized properly.

REFERENCES

1. K.J. Bathe, E.L. Wilson, and F.E. Peterson, "SAP IV - A Structural Analysis Program for Static and Dynamic Response of Linear Systems", Report No. EERC 73-11, Earthquake Engineering Research Center, Univ. of Ca., Berkeley, Ca., June, 1973

2. C. Fleury, R.K. Ramanathan, M. Salama, and L.A. Schmit Jr., "ACCESS Computer Program for the Synthesis of Large Structural Systems", Proceedings of the International Symposium on Optimum Structural Design, Univ. of Ariz., Tucson, Az., October 19-22, 1981

3. M.A. Bhatti, V. Ciampi, K.S.Pister, and E. Polak, "OPTNSR An Iteractive Software System for Optimal Design of Statically and Dynamically Loaded Structures with Nonlinear Response", Report No. EERC 81-02, Earthquake Engineering Research Center, Univ. of Ca., Berkeley, Ca., January, 1981

4. "UNIX Programmer's Manual" Seventh Edition, VAX-11 Version, Bell Telephone Laboratories, Inc., Holmdel, N.J., December, 1978

5. W. Nye, E. Polak, and A. Sangiovanni-Vincentilli, "DELIGHT DEsign Laboratory with Interaction and Graphics for a Happier Tomorrow", Dept. of Electrical Engineering and Computer Sciences, Univ. of Ca., Berkeley, Ca., April, 1981

6. D.P. Mondkar and G.H. Powell, "ANSR-I General Purpose Program for Analysis of Nonlinear Structural Response", Report No. EERC 75-37, Earthquake Engineering Research Center, Univ. of Ca., Berkeley, Ca., December, 1975

7. A. Riahi, D.G. Row, and G.H. Powell, "Three Dimensional Inelastic Frame Elements for the ANSR-I Program", Report No. EERC 78-06, Earthquake Engineering Research Center, Univ. of Ca., Berkeley, Ca., August, 1978

8. G. Gonzaga, E. Polak, and R. Trahan, "An Improved Algorithm for Optimization Problems with Functional Inequality Constraints", Memorandum No. UCB/ERL/ M78/56, Electronics Research Laboratory, Univ. of Ca., Berkeley, Ca., September, 1977

9. E. Polak, Computational Methods in Optimization, Academic Press, New York, N.Y., 1971

10. D.G. Luenberger, Introduction to Linear and Nonlinear Programming, Addison-Wesley, Reading, Mass., 1973

11. R.J. Balling, V.Ciampi, K.S. Pister, and E. Polak, "Optimal Design of Seismic-Resistant Planar Steel Frames", Report No. EERC 81-20, Earthquake Engineering Research Center, Univ. of Ca., Berkeley, Ca., December, 1981

12. R.K. Brayton and S.W. Director, "Computation of Delay Time Sensitivities for Use in Time Domain Optimization", IEEE Transactions on Circuits and Systems, vol. cas-22, no. 12, December, 1975

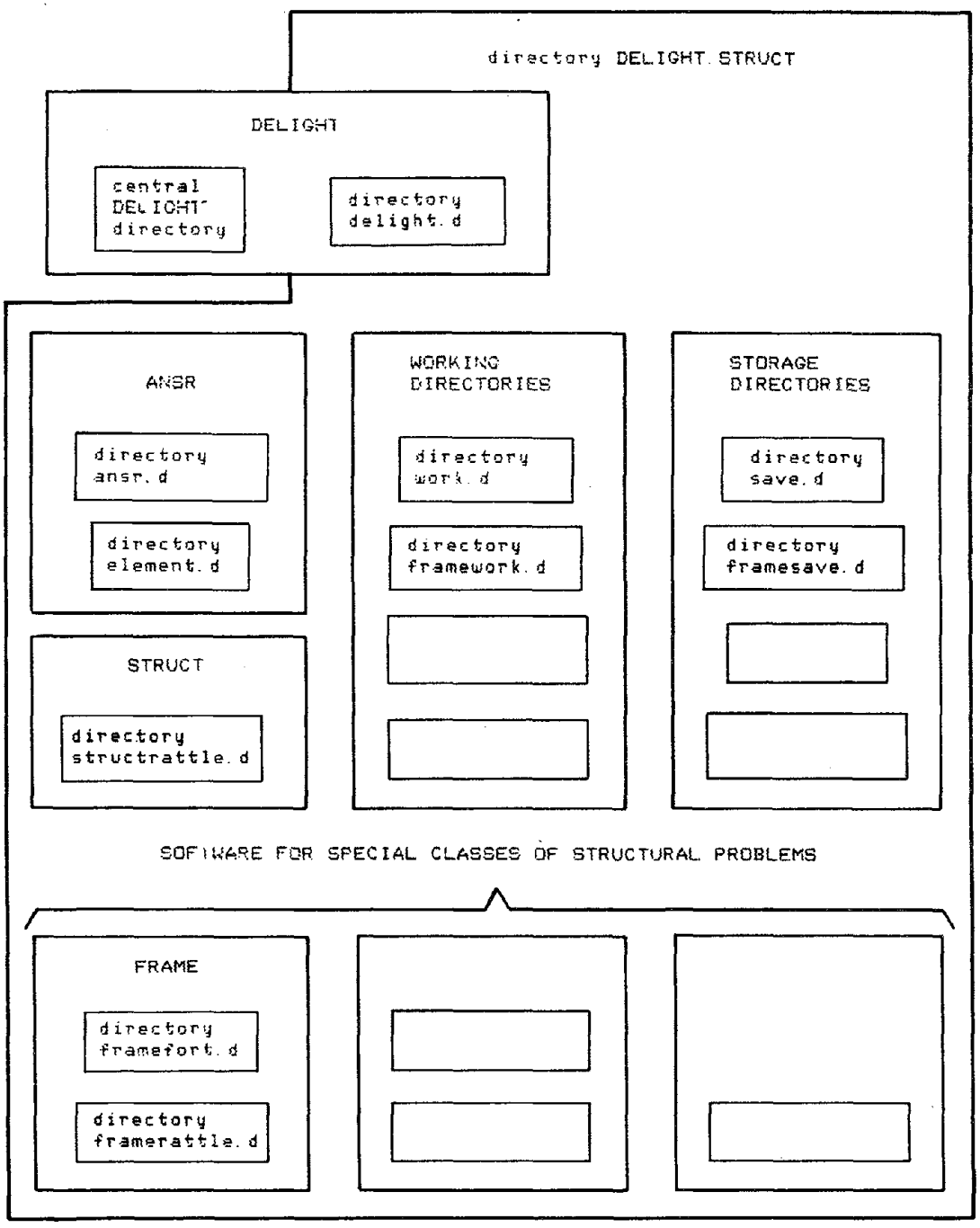


FIGURE 1 : ORGANIZATION OF DELIGHT.STRUCT

GOAL #1 : SIMPLIFY PROGRAMMING PROCESS

* FRIENDLY RATTLE LANGUAGE
* SOPHISTICATED DYNAMIC MEMORY MANAGER
* HIGH-LEVEL MATRIX OPERATION STATEMENTS

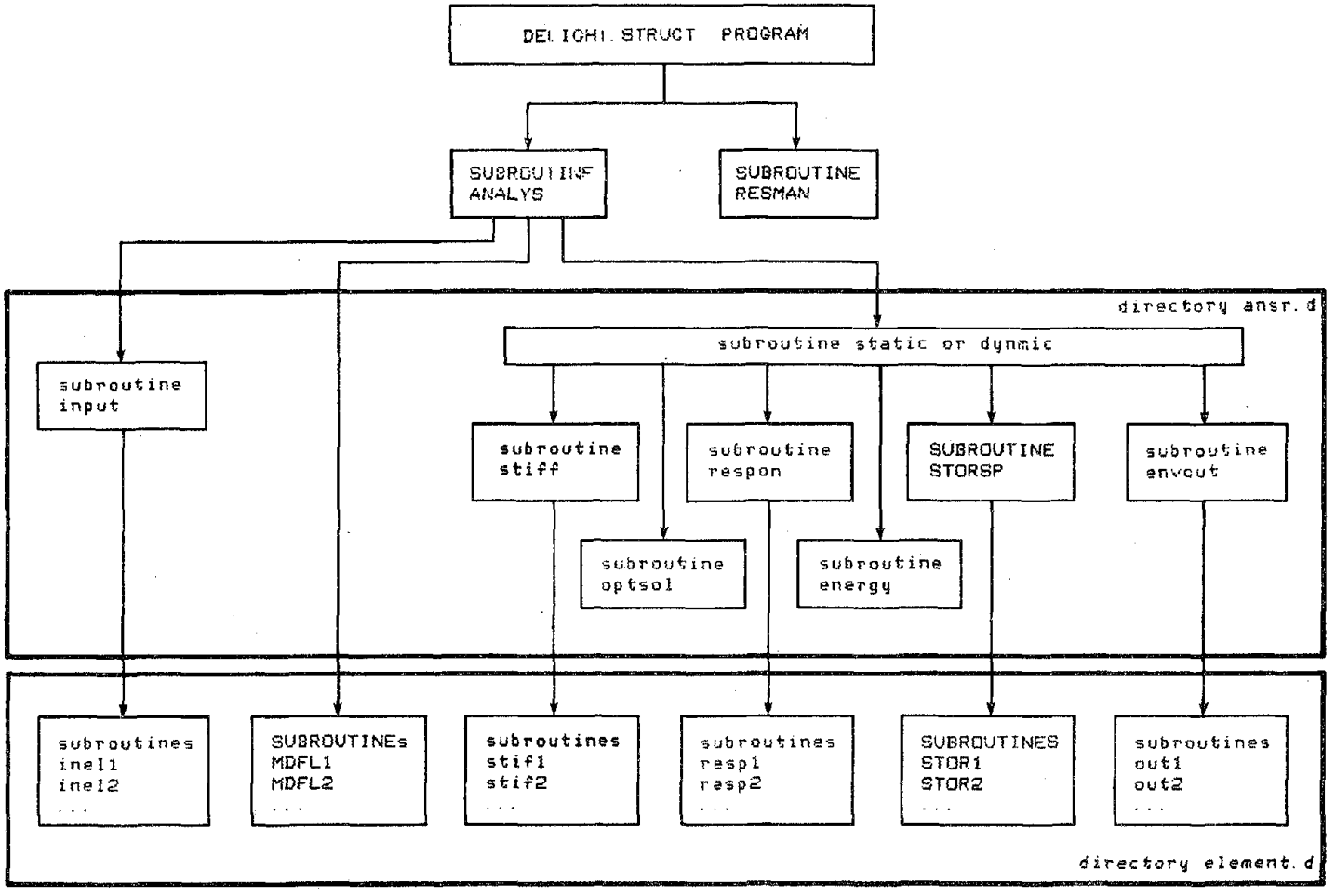
GOAL #2 : ALLOW INTERACTION

* NO LOAD-LINKAGE IN COMPILATION / EXECUTION
* EXECUTION INTERRUPTS
* UTILITY COMMANDS
* HIGH-LEVEL GRAPHICS COMMANDS

GOAL #3 : ACCOMODATE EXPANDABLE LIBRARIES

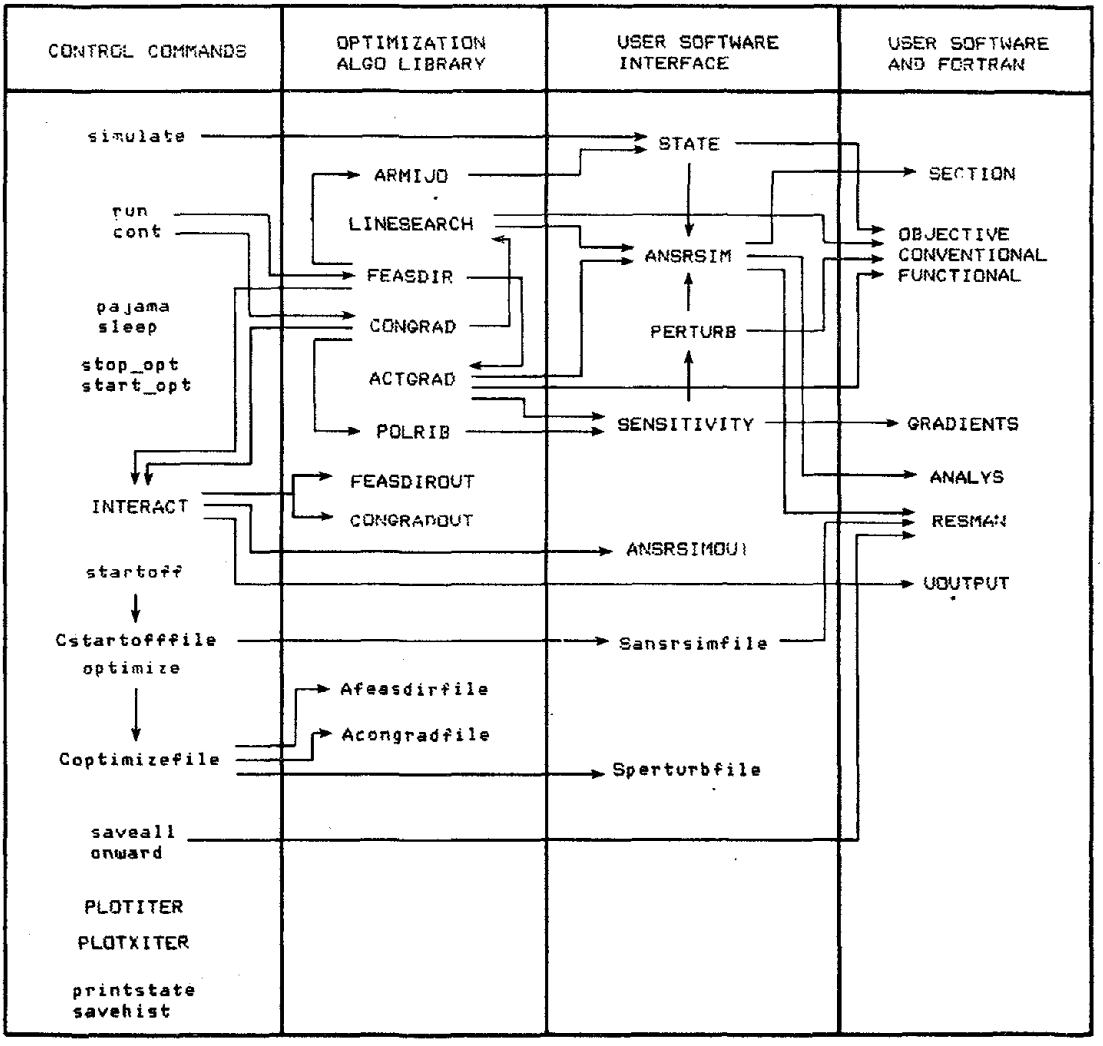
* INTERFACE EXISTING FORTRAN CODE
* INCLUDE SOURCE FILES AND ACCESS COMPILED "MEMFILES"

FIGURE 2 : GOALS AND FEATURES OF DELIGHT



original mini-ANSR in small letters
MODIFICATIONS IN CAPITAL LETTERS

FIGURE 3 : ORGANIZATION OF ANSR



PROCEDURES AND SUBROUTINES ARE IN CAPITAL LETTERS
defines and files are in small letters

FIGURE 4 : ORGANIZATION OF STRUCT

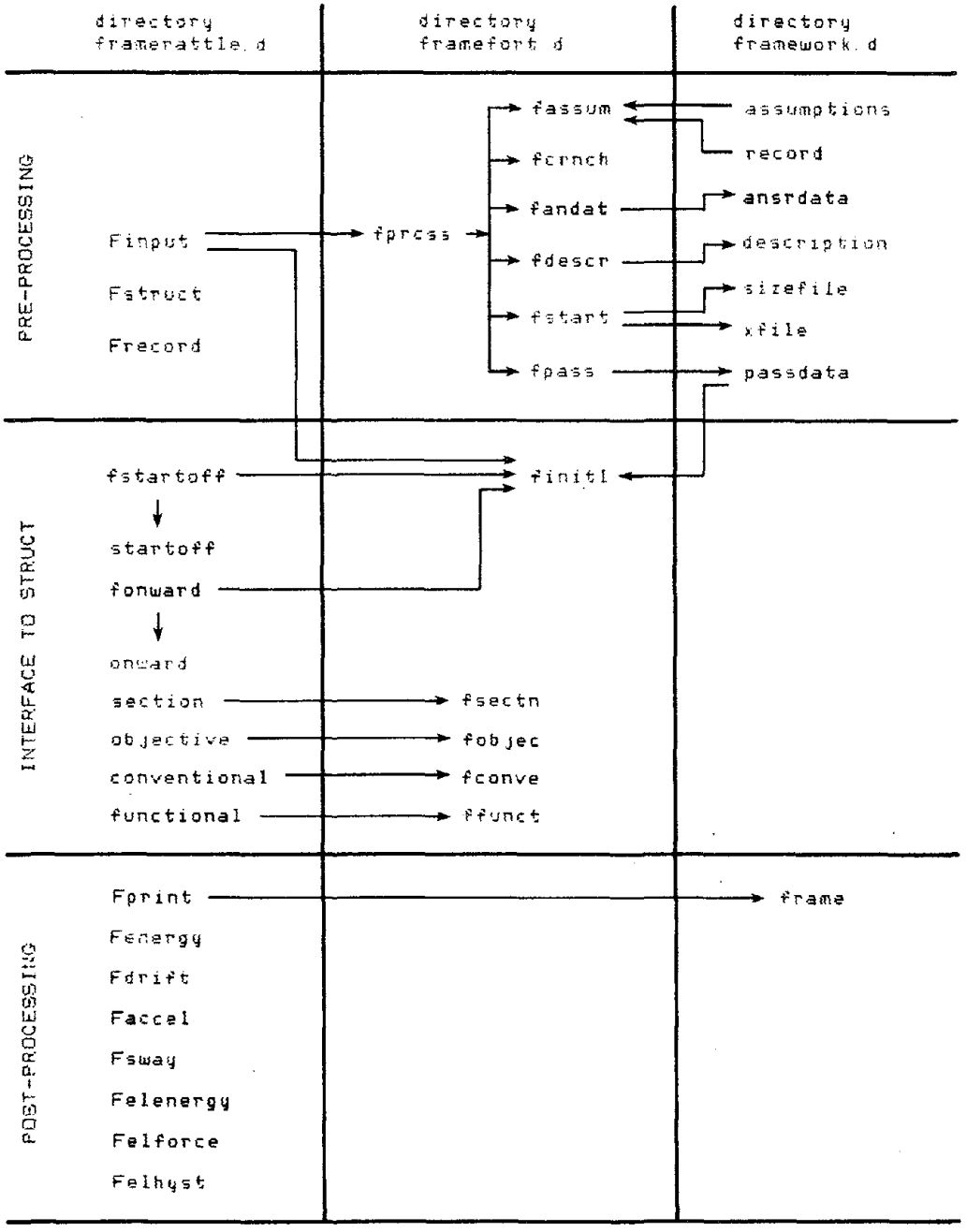
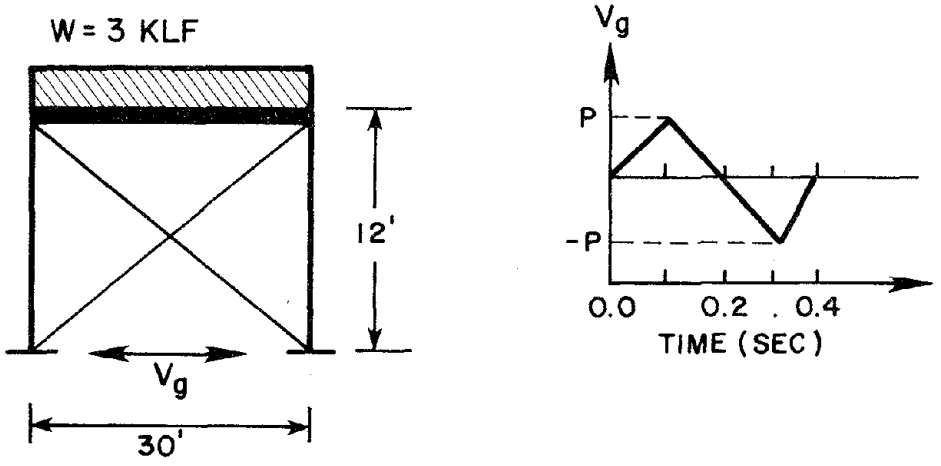


FIGURE 5 : ORGANIZATION OF FRAME



DESIGN VARIABLES

#1 = column moment of inertia
#2 = brace cross-sectional area
#3 = dummy story drift variable

COST FUNCTION

minimize linear combination of square of max of moderate quake story drift and severe quake inelastic energy

CONVENTIONAL CONSTRAINTS

1	:	volume	< C1
2	:	!column axial gravity force!	< C2
3,4	:	!column gravity end moments!	< C3
5,6	:	!brace gravity axial force!	< C4
7,8	:	severe quake column energy dissipation	< C5
9,10	:	severe quake brace energy dissipation	< C6

FUNCTIONAL CONSTRAINTS

1,2	:	!moderate quake column end moments!	< C7
3,4	:	!moderate quake brace axial force!	< C8
5	:	!moderate quake story drift!	< C9
6	:	!moderate quake floor acceleration!	< C10
7	:	square moderate quake story drift	< des var #3
8	:	!severe quake structure sway!	< C11

FIGURE 6 : EXAMPLE PROBLEM

APPENDIX 1 : IDENTIFICATION OF FILES IN DELIGHT.STRUCT

Files in the directory delight.d:

builtmid, builttop, builtnam
: Ratfor code for interfacing Fortran to DELIGHT

builtn.r, cexeval, chrdefs, ciochans, cralloc, ctk7data, exdefs, interupt, iodefs, machdep, maxdefs, style
: linked files representing the built-in part of DELIGHT

rat4
: linked Ratfor compiler

fort.7
: Fortran code translated from built-in part of DELIGHT

builtn.o
: compiled built-in part of DELIGHT

meMfio.f
: Fortran code for storing Fortran variables in memfiles

meMfio.o
: compiled meMfio.f

testrr.o
: linked compiled main DELIGHT code

load.delight
: UNIX code for loading the DELIGHT.STRUCT program

DELIGHT.STRUCT
: executable main program in the DELIGHT.STRUCT system

openhdtl
: formatted basic DELIGHT software directory list

setup
: Rattle code selecting software for the memfile memfile

make.memfile
: UNIX code for making the memfile memfile

memfile
: binary compiled basic DELIGHT software

make.errors
: formatted output while making the memfile memfile

Files in the directory ansr.d

analys.f
: Fortran code for driving ANSR simulation

resman.f
: Fortran code for manipulating response storage vector

storsp.f
: Fortran code for storing response at each time step

remaining ".f"
: Fortran code representing mini-ANSR simulation package

".o"
: compiled versions of source counterparts

Files in the directory element.d

elem3.f, elem4.f, elem5.f, elem6.f, elem7.f, elem8.f, elem9,f, elem10.f

Fortran code for dummy ANSR elements
inel1.f, stif1.f, resp1.f, out1.f
Fortran code for 3-D inelastic truss element
inel2.f, stif2.f, resp2.f, out2.f
Fortran code for 2-D lumped-plasticity beam element
stor1.f, stor2.f
Fortran code for storing element response at each step
mdfl1.f, mdfl2.f
Fortran code modifying element data according to design
".o"
compiled versions of source counterparts

Files in the directory.structrattle.d

Afeasdir
Rattle code for feasible directions algorithm
Aactgrad
Rattle code for Gonzaga-Polak-Trahan direction finder
Aarmijo
Rattle code for Armijo step length determination
Afeasout
Rattle code for feasible directions output
Afeasdirfile
Rattle code for feasible directions initialization
Acongrad
Rattle code for conjugate gradient algorithm
Apolrib
Rattle code for Polak-Ribier direction finder
Alinesearch
Rattle code for Luenberger line search
Acongradout
Rattle code for conjugate gradient output
Acongradfile
Rattle code for conjugate gradient initialization
Coperate
Rattle code containing defines for control commands
Cinteract
Rattle code governing interaction at each iteration
Cstartofffile
Rattle code for initialization of computation process
Coptimizefile
Rattle code for initialization of optimization process
Cwindow
Rattle code setting up windows for plotting
Cplotiter
Rattle code for plotting COST, PSI, cpu-time histories
Cplotxiter
Rattle code for plotting design variable history
Csavehist
Rattle code for saving histories on file
Cprintstate
Rattle code printing constraint percentages on file
Sstate

Rattle code for computing cost and constraints

Sansrsim
Rattle code for performing relevant ANSR simulations

Sansrsimout
Rattle code for ANSR simulation output

Sansrsimfile
Rattle code for ANSR simulation initialization

Ssensitivity
Rattle code for performing sensitivity analysis

Sperturb
Rattle code for computing gradients by perturbation

Sperturbfile
Rattle code for perturbation initialization

openhdtl
formatted structural software directory list

structsetup
Rattle code selecting software for memfile memstruct

make.memstruct
UNIX code for making the memfile memstruct

memstruct
binary compiled structural software

make.errors
formatted output while making the memfile memstruct

Files in the directory framefort.d

fdeclr.f
Fortran code declaring frame variables as interactive

fprcss.f
Fortran code managing the pre-processing of input data

fassum.f
Fortran code reading record and assumed material values

fcrnch.f
Fortran code digesting the input data

fandat.f
Fortran code creating ansrdata files

fdescr.f
Fortran code printing a description of the problem

fpass.f
Fortran code creating binary data file passdata

fstart.f
Fortran code creating files sizefile and xfile

finitl.f
Fortran code reading from binary data file passdata

fsectn.f
Fortran code computing element section properties

fobjec.f
Fortran code computing the cost function

fconve.f
Fortran code computing the conventional constraints

ffunct.f
Fortran code computing the functional constraints

".o"

compiled versions of source counterparts

Files in the directory framerattle.d

Finterface
: Rattle code for interfacing frame software to system

Fstartofffile
: Rattle code for initializing frame software

Finput
: Rattle code for the frame pre-processor

Fstruct
: Rattle code plotting frame geometry and loads

Frecord
: Rattle code plotting ground acceleration record

Fprint
: Rattle code giving printout of frame information

Fenergy
: Rattle code plotting terms in global energy balance

Fdrift
: Rattle code plotting moderate quake story drifts

Faccel
: Rattle code plotting moderate quake floor acceleration

Fsway
: Rattle code plotting severe quake structure drift

Felenergy
: Rattle code plotting severe quake element energy

Felforce
: Rattle code plotting moderate quake element forces

Felhyst
: Rattle code plotting severe quake element hysteresis

openhdtl
: formatted frame software directory list

framesetup
: Rattle code selecting software for memfile memframe

make.memframe
: UNIX code for making the memfile memframe

memframe
: binary compiled frame software

make.errors
: formatted output while making the memfile memframe

Files in the directory work.d

objective
: Rattle code for user-supplied cost function

conventional
: Rattle code for user-supplied conventional constraints

functional
: Rattle code for user-supplied functional constraints

section
: Rattle code for user-supplied procedure section

uoutput

Rattle code for user-supplied output at each iteration

preprocessor

Rattle code for user-supplied pre-processing software

postprocessor

Rattle code for user-supplied post-processing software

scratchpad

Rattle code for user-supplied scratchpad software

sizefile

Rattle code initializing problem size variables

xfile

Rattle code initializing design variable set

ansrdata

formatted ANSR simulation input data

startsetup

Rattle code compiling problem software

openhdtl

linked structural software directory list

make.memstart

UNIX code for making the memfile memstart

memstart

binary compiled problem software

make.errors

formatted output while making the memfile memstart

dialogue

formatted record of optimization process

memprob

binary temporary information state in optimization

backup

binary spare copy of previous memprob

response

binary most recent ANSR simulation response data

pajama

Rattle code giving commands when execution suspends

ansroutput

formatted output from ANSR simulation

history

Rattle code for restoring iteration history arrays

state

formatted current cost and constraint percentages

plot

formatted data for hard-copy plotter software

plotdesc

formatted description of data in plot

Files in the directory framework.d

assumptions

formatted assumptions used in ANSR simulation

record

formatted ground acceleration record

description

formatted description of optimization problem

passdata

binary frame information generated by pre-processor
openhdtl
linked frame software directory list
frame
formatted values of useful frame quantities
remaining files as in the directory work.d

Files in the directory save.d

files saved by user from directory work.d (not listed here)

Files in the directory framesave.d

files saved by user from directory framework.d (not listed here)

APPENDIX 2 : INTERACTIVE VARIABLES IN DELIGHT.STRUCT

Global Variables To Be Initialized By The User In The File Sizefile

NPARAM	total number of design variables
NSIMPAR	number of design variables which affect ANSR simulation
NLOAD	number of ansrdata files for ANSR simulation
NINEQ	number of conventional constraints
NFINEQ	number of functional constraints
NPROP	maximum number of element design properties for ANSR simulation
NELS	number of structural elements for ANSR simulation
NRESP	amount of response storage space needed for ANSR simulation
COSTL	ANSR simulation ansrdata files needed to compute the cost (if simulation with ansrdata1 file is needed COSTL = 1, if both ansrdata1 and ansrdata2 are needed COSTL = 21, etc.)
Q(NLOAD)	number of ANSR simulation time steps for each ansrdata file
INEQL(NLOAD)	beginning conventional constraint number for each ansrdata file
FINEQL(NLOAD)	beginning functional constraint number for each ansrdata file

if ANSR simulation is not involved, NSIMPAR=NLOAD=NPROP=NELS=NRESP=COSTL=0

Other Global Structural Variables

ITER	optimization design iteration number
X(NPARAM)	vector of current design variables
COST	current value of the cost function
INEQ(NINEQ)	current values of the conventional constraints
NQ	maximum number of time steps in functional constraints
FINEQ(NQ)	values of a functional constraint over time
MFINEQ(NFINEQ,2)	max time step and max value of each functional constraint
PSI	value of max constraint
IPSI	max constraint number
JPSI	max constraint time step (zero if conventional constraint)

Possible Structural Parameters

Tolx	tolerance for ANSR resimulation
Numsim	number of ANSR simulations
Numsteps	number of ANSR simulation time steps
Numref	number ANSR stiffness reformulations
Maxerr	maximum ANSR ratio of error energy to input energy
Print	ANSR simulation printout code (Print=1 for no printout)
Eps	active constraint band-width
Dist	step length in optimal direction
Delta	eps reduction control
Epstol	convergence tolerance on eps
Psitol	convergence tolerance on psi
Geomslope	geometric constraint slope
Alpha	cost or psi reduction angle
Beta	step length line-search factor
Distmax	maximum step length

Scale	scale for gradients of active constraints
Gamma	cost penalty when infeasible
Deltax	gradient perturbation
Maxiter	maximum number of iterations in step length determination
Tolf	tolerance for line search
Tolgrad	tolerance on norm of cost gradient

Other Structural Variables

Startime	starting cpu-clock time
Iterstop	design iteration number when next interaction will occur
Costsave()	iteration history of cost
Psisave()	iteration history of max constraint value
Timesave()	iteration history of cumulative cpu time
Xsave(NPARAM,)	iteration history of design variables
Resp(300000)	ANSR simulation response storage vector

Algorithm Local Variables Output At Each Iteration

direct(NPARAM)	(procedure feasdir) direction vector
theta	(procedure feasdir) value of the optimality function
nact	(procedure actgrad) number of active constraints
list1(nact)	(procedure actgrad) constraint numbers of active constraints
list2(nact)	(procedure actgrad) time step numbers of active functional constraints
ngeom	(procedure actgrad) number of active geometric constraints
glist(ngeom)	(procedure actgrad) active geometric constraint numbers
jacobian(nact,NPARAM)	(procedure actgrad) gradients of active constraints
norms(nact)	(procedure actgrad) norms of active constraint gradients
angles(nact)	(procedure actgrad) angles from direction to active constraints
narm	(procedure armijo) number of step length iterations
distart	(procedure armijo) starting step length
stepviol(narm)	(procedure armijo) history of step length iterations
direct(NPARAM)	(procedure congrad) direction vector
gradnorm	(procedure congrad) norm of current cost gradient
angles	(procedure polrib) angle from direction vector to cost gradient
ncost	(procedure linesearch) number of cost evaluations in line search

Possible Frame Design Parameters (Interactive Fortran Variables)

Cosvol	volume cost coefficient
Cosdri	moderate quake drifts cost coefficient
Cosinp	severe quake input energy cost coefficient
Cosdis	severe quake inelastic energy cost coefficient
Cosfus	severe quake fuse inelastic energy cost coefficient
Coscol	severe quake column inelastic energy cost coefficient
Volmax	max structural volume
Drift	max moderate quake story drift
Accel	max moderate quake floor accel in g's
Sway	max severe quake structure sway
Colax	gravity load column axial force factor
Colgra	gravity load column end moment yield factor
Girgra	gravity load girder end moment yield factor

Girdef	max gravity load girder midspan deflection
Bragra	gravity load brace yield or buckling factor
Colyld	moderate quake column yield factor
Colduc	severe quake column ductility
Berten	severe quake rubber bearing tensile stress factor
Giryld	moderate quake girder yield factor
Girduc	severe quake girder ductility
Shryld	moderate quake shear element yield factor
Shrduc	severe quake shear element ductility
Disyld	moderate quake dissipator element yield factor
Disduc	severe quake dissipator element ductility
Brayld	moderate quake brace yield or buckling factor
Braduc	severe quake brace ductility

Other Frame Design Interactive Fortran Variables

Nc(6), Nstnd, Naccel, Ntpnd, Ksever, Kmoder, Nodsev, Ldrift, Nelm, Kinde(60), Xplus(6), Xminus(6), Kaccel(26), Kdrift(26,2), Hstnd(26), Nec, Mapct(60), Kelem(60,3), Ksway(26,2), Htpnd(26), Xy(26,2), Totht, Totwd, Kind(60), Nodes, Konn(60,2), Force(26,2), Ndcon, Konst(26,2), Ns, Sh(12), Nb, Bw(12), Nrecd, Tend, Record(1100)

APPENDIX 3 : ANSRDATA FILE SYNTAX

Control And Nodal Information

one line (a5,18a4)
  type the word START
  problem title

one line (i10)
  starting address for storing response in vector resp

one line (16i5)
  total number of nodes
  number of control nodes = NCNOD
  number of coordinate generation commands = NODGC
  number of zero displacement commands = NDCON
  number of equal displacement commands = NIDDOF
  number of nodal mass commands = NMSGC
  number of element groups = NELGR
  execution code
    (=0 full execution; =1 data checking only)

NCNOD lines (i5,3f10.0)
  node number
  x-coordinate
  y-coordinate
  z-coordinate

NODGC lines (4i5,f10.0,10i5)
  node number at beginning of generation line
  node number at end of generation line
  number of nodes to generated along the line
  node number difference between successive generated nodes
  node spacing
    (=0 uniform spacing; <1 spacing is this proportion
    of total line length; >1 spacing is this distance)
  list of nodes to be generated

NDCON lines (16i5)
  node number of first node
  constraint code for x-displacement
    (=0 not constrained to be zero;
    =1 constrained to be zero)
  constraint code for y-displacement
  constraint code for z-displacement
  constraint code for x-rotation
  constraint code for y-rotation
  constraint code for z-rotation
  node number of last node
  node number difference between successive nodes
  number of nodes in following list
  list of nodes with this constraint code

NIDDOF lines (16i5)

equal displacement code for x-displacement
(=0 displacement not equal; =1 displacement equal)
equal displacement code for y-displacement
equal displacement code for z-displacement
equal displacement code for x-rotation
equal displacement code for y-rotation
equal displacement code for z-rotation
number of nodes in following list
list of nodes for which displacement code applies

NMSGC lines (i5,6f10.0,2i5)
node number of first node
x-displacement mass
y-displacement mass
z-displacement mass
x-rotation mass
y-rotation mass
z-rotation mass
node number of last node
node number difference between successive nodes

Load Information

one line (8i5,3f10.0)
code for static and/or dynamic analysis = KSTAT
(=0 dynamic analysis only;
=1 static and dynamic analysis;
=-1 static analysis only)
number of static force patterns = NSPAT
number of static force load increments = NSLGC
code for ground motion records = IGR
(=0 no ground motion records;
=1 ground motion records exist)
number of dynamic force records = NDLR
largest number of points on any dynamic force record
number of dynamic force application commands = NDLGC
number of integration time steps
integration time step
Newmark integration parameter delta
Newmark integration parameter beta

NSPAT sets of lines

one line (i5,18a4)
number of nodal load commands for this pattern = NSLC
load pattern title

NSLC lines (i5,6f10.0,2i5)
node number of first node
load in x-direction
load in y-direction
load in z-direction
moment about x-axis
moment about y-axis

moment about z-axis
node number of last node
node number difference between successive nodes

one line if IGR = 1 (4i5,6f10.0)
number of points in x-direction record = NIPX
number of points in y-direction record = NIPY
number of points in z-direction record = NIPZ
print code
(=0 no printout; =1 records printed as input and scaled;
=-1 records printed as input, scaled, and interpolated)
time interval for x-direction record
time interval for y-direction record
time interval for z-direction record
scale factor for x-direction record
scale factor for y-direction record
scale factor for z-direction record

one line if NIPX > 0 (15a4,5a4)
x-direction record title
input format for the NIPX points

lines for NIPX points (format as specified on preceding line)
values of ground accelerations in x-direction

one line if NIPY > 0 (15a4,5a4)
y-direction record title
input format for the NIPY points

lines for NIPY points (format as specified on preceding line)
values of ground accelerations in y-direction

one line if NIPZ > 0 (15a4,5a4)
z-direction record title
input format for the NIPZ points

lines for NIPZ points (format as specified on preceding line)
values of ground accelerations in z-direction

NDLR sets of lines

one line (2i5,2f10.0,8a4,4a4)
number of points defining record = NIPT
print code
(=0 no printout; =1 record printed as input and scaled;
=-1 record printed as input, scaled, and interpolated)
record time interval
record scale factor
record title
input format for record

lines for NIPT points (format as specified on preceding line)
values of forces

NDLGC lines (16i5)
  dynamic force record number
  direction code
    (=1 x-translation; =2 y-translation; =3 z-translation;
    =4 x-rotation; =5 y-rotation; =6 z-rotation)
  list of nodes for which record is to be applied

one line if KSTAT > -1 (3f10.0)
  mass proportional damping factor
  tangent stiffness proportional damping factor
  initial stiffness proportional damping factor

Analysis And Output Information

NSLGC sets of lines

  one line if KSTAT < 1 (8i5,4f10.0)
    number of static load steps in this load increment
    static iteration type
      (=0 Newton-Raphson iteration; =n constant stiffness
      iteration reinitializing stiffness every n iterations)
    state determination type
      (=0 path independent; =1 path dependent)
    stiffness reformation code
      (=0 stiffness not reformed;
      =n stiffness reformed every n load steps)
    termination code
      (=0 continue regardless of convergence;
      =1 terminate if there is no convergence)
    print code
      (=-1 no printout;
      =0 results printed at end of increment;
      =1 results printed at each load step;
      =2 results printed at each iteration)
    maximum number of cycles of iteration in any load step
    maximum number of iterations within any cycle
    nodal force convergence tolerance for last load step
    nodal force convergence tolerance for all load steps
    nodal force tolerance for changing stiffness
    maximum nodal displacement

  as may lines as needed for NSPAT values (8f10.0)
    scale factors for the static force patterns

one line if KSTAT > -1 (7i5,4f10.0,i5)
  dynamic iteration type
    (=0 Newton-Raphson iteration; =n constant stiffness
    iteration reinitializing stiffness every n iterations)
  state determination type
    (=0 path independent; = 1 path dependent)
  stiffness reformation code
    (=0 stiffness not reformed;
    =n stiffness reformed every n load steps)
  termination code

(=0 continue regardless of convergence;
=1 terminate if there is no convergence)
maximum number of cycles of iteration within any time step
maximum number of iterations within any cycle
number of time steps for fine convergence tolerance
fine nodal force convergence tolerance
coarse nodal force convergence tolerance
nodal force tolerance for changing stiffness
maximum nodal displacement
number of initial condition commands = NICGC

NICGC lines (i5,2f10.0,11i5)
direction code
(=1 x-translation; =2 y-translation; =3 z-translation;
=4 x-rotation; =5 y-rotation; =6 z-rotation)
initial velocity
initial acceleration
list of nodes having these initial conditions

one line (10i5)
time step interval for printout of nodal displacements
time step interval for printout of element response
time step interval for printout of envelopes
number of nodes for x-direction output and storage = NODSX
number of nodes for y-direction output and storage = NODSY
number of nodes for z-direction output and storage = NODSZ
code for storage of nodal displacements
(=0 no storage; =1 storage)
code for storage of nodal velocities
(=0 no storage; =1 storage)
code for storage of nodal accelerations
(=0 no storage; =1 storage)
code for storage of global energies
(=0 no storage; =1 storage)

as many lines as needed for the NODSX nodes (16i5)
list of nodes for x-direction output and storage

as many lines as needed for the NODSY nodes (16i5)
list of nodes for y-direction output and storage

as many lines as needed for the NODSZ nodes (16i5)
list of nodes for z-direction output and storage

Lumped-Plasticity Parallel-Component 2-D Beam-Column Element

NELGR sets of lines

one line (10i5,6f5.0)
element group indicator
(=2 for beam-column element)
number of elements in this group = NMEM
element number of the first element in this group
number of element stiffness types = NMBT

number of end eccentricity types = NECC
number of yield interaction surface types = NSURF
number of initial force patterns = NINT
blank
blank
blank
initial stiffness element damping factor
tangent stiffness element damping factor

NMBT lines (i5,4f10.0,3f5.0,f10.0,2f5.0)
stiffness type number
modulus of elasticity
strain hardening ratio
average cross-sectional area
reference moment of inertia
flexural stiffness factor kii
(=4 for prismatic)
flexural stiffness factor kjj
(=4 for prismatic)
flexural stiffness factor kij
(=2 for prismatic)
effective shear area
poisson's ratio
ratio of minor to major axis bending stiffness

NECC lines (i5,6f10.0)
end eccentricity type number
x-eccentricity at end i
x-eccentricity at end j
y-eccentricity at end i
y-eccentricity at end j
z-eccentricity at end i
z-eccentricity at end j

NSURF lines (2i5,5f10.0,4f5.0)
yield surface number
yield surface shape code
(=1 beam type; =2 steel I-beam type;
=3 reinforced concrete column type)
scale factor
positive yield moment
negative yield moment
compressive axial yield force
tensile axial yield force
moment fraction of positive moment corner
axial force fraction of positive moment corner
moment fraction of negative moment corner
axial force fraction of negative moment corner

NINT lines (i5,6f10.0)
initial force pattern number
axial force at end i
shear force at end i
moment at end i

axial force at end j
shear force at end j
moment at end j

lines for the NMEM elements (10i4,i3,i10,i5,f5.0)
element number
node number at end i
node number at end j
node number increment for element generation
number of node to which i end is slaved
number of node to which j end is slaved
stiffness type number
end eccentricity type number
yield surface type number for end i
yield surface type number for end j
code for geometric stiffness
(=0 neglect geometric stiffness;
=1 consider geometric stiffness)
code for response storage and output
(up to a 5 digit number with codes as follows:
=0 no storage or output for this element;
=1 store and output end axial forces;
=2 store and output end plastic rotations;
=3 store and output end inelastic energies;
=4 store and output end moments;
=5 store and output end rotations)
initial force pattern number
scale factor for initial force pattern

Nonlinear Parallel-Component 3-D Truss Element

one line (10i5,6f5.0)
element group indicator
(=1 for truss element)
number of elements in this group = NMEM
element number of the first element in this group
number of element material types = NMAT
blank
blank
blank
blank
blank
blank
initial stiffness element damping factor
tangent stiffness element damping factor

NMAT lines (i5,5f10.0)
material type number
modulus of elasticity
strain hardening ratio
yield stress in tension
yield or buckling stress in compression
scale factor for yield or buckling stress

lines for the NMEM elements (4i5,2f10.0,4i5)
  element number
  node number at end i
  node number at end j
  material type number
  cross-sectional area
  initial axial force on element
  node number increment for element generation
  code for large displacements
    (=0 neglect large displacements;
    =1 consider large displacements)
  code for response storage and output
    (up to a 3 digit number with codes as follows:
    =0 no storage or output for this element;
    =1 store and output axial force;
    =2 store and output axial displacement;
    =3 store and output inelastic energy)
  code for buckling
    (=0 yields in compression;
    =1 buckles in compression)

APPENDIX 4 : SAMPLE DELIGHT.STRUCT TERMINAL INPUT AND OUTPUT

Example of using DELIGHT.STRUCT in a hand-calculator mode to obtain an initial design for the example problem

```
% go
Restoring from <memfile> ...
Identifier: basic memfile
----- LONG LIVE RATTLE/DELIGHT !!! -----
List <features> to see the new features.

1> # Design columns to take all of the gravity axial force.
1> axfor = 360*.25/2
1> icol1 = (axfor*2/36/.8)**2
1> icol2 = (144/PI)**2*axfor*2/29000
1> icolmin = 50
1> icol = max (icol1,icol2,icolmin)
1> printf 'column moment of inertia = %r/n' icol
column moment of inertia =  5.000e+1

1> # Design brace area to supply necessary stiffness to keep
1> # severe quake sway below limit.
1> # Approximate as undamped SDOF system under sinusoidal pulse.
1> # Use iterative procedure to find necessary stiffness.
1> k1 = 100 ; k2 = 200
1> t = .4 ; w = 2*PI/t ; g = 386.088 ; mass = 2*axfor*.8/g
1> d1 = g*w/(k1/m-w**2)*sqrt(2*m/k1*(1-cos(t*sqrt(k1/m))))
RUN-TIME ERROR: overflow or other floating point exception.
Interrupt...
2> # Oops, lets try that one again.
2> reset
1> d1 = g*w/(k1/mass-w**2)*sqrt(2*mass/k1*(1-cos(t*sqrt(k1/mass))))
1> repeat {
1> d2 = g*w/(k2/mass-w**2)*sqrt(2*mass/k2*(1-cos(t*sqrt(k2/mass))))
1> dk = (k2-k1)*(144*.01-d2)/(d2-d1)
1> kk = k2+dk
1> if (abs(dk/k2) < .0001) break
1> k1 = k2 ; d1 = d2 ; k2 = kk
1> }
1> forever
1> abra = (kk-24*29000*icol/144**3)*(144**2+360**2)/2/360/29000
1> abramin = 1
1> abra = max (abra,abramin)
1> printf 'brace area = %r/n' abra
brace area =  1.000

1> # The initial design is quite uninteresting since it is just
1> # the minimum section values, but this was intended to be an
1> # illustration of the DELIGHT.STRUCT scratchpad mode.
1> quit
Goodbye Sir
```

Example of initialization for the optimization process for the example problem using the existing seismic-resistant frame design software

```
% go '<memframe>'
Restoring from <memframe> ...
Identifier: frame memfile
----- LONG LIVE RATTLE/DELIGHT !!! -----
List <features> to see the new features.

1> fstartoff
Input cost function coefficients for:
  volume of designed elements: 0
  sum of squares of moderate quake story drifts: 1
  input energy from severe quake: 0
  inelastically dissipated energy from severe quake: 1
  fuse dissipated energy from severe quake: 0
  column dissipated energy from severe quake: 0
PARAMETER: Cosvol = Cosvol : volume cost coefficient
PARAMETER: Cosdri = Cosdri : drift cost coefficient
PARAMETER: Cosinp = Cosinp : input energy cost coefficient
PARAMETER: Cosdis = Cosdis : plastic energy cost coefficient
PARAMETER: Cosfus = Cosfus : fuse energy cost coefficient
PARAMETER: Coscol = Coscol : column energy cost coefficient
PARAMETER: Volmax = MAXREAL : max structural volume
PARAMETER: Drift = .005 : max moderate story drift
PARAMETER: Accel = .5 : max moderate floor accel in gs
PARAMETER: Sway = .01 : max severe structure sway
PARAMETER: Colax = .5 : gravity column axial force factor
PARAMETER: Colgra = .6 : gravity column yield factor
PARAMETER: Colyld = 1 : moderate column yield factor
PARAMETER: Colduc = 3 : severe column ductility
PARAMETER: Bragra = .6 : gravity brace yield factor
PARAMETER: Brayld = 1 : moderate brace yield factor
PARAMETER: Braduc = 2 : severe brace ductility
PARAMETER: Tolx = 0.0001 : tolerance for resimulating
PARAMETER: Numsim = 0 : number of simulations
PARAMETER: Numsteps = 0 : number of simulation time steps
PARAMETER: Numref = 0 : number stiffness reformulations
PARAMETER: Maxerr = 0.0 : maximum energy error ratio
PARAMETER: Print = 1 : simulation printout code
Type "1" if a response file exists, otherwise type "0": 0
---------- Next CRUNCH is forced. ----------
CRUNCH ...

1> Volmax = 10000
1> simulate
begin simulation...
end simulation...
begin simulation...
end simulation...
begin simulation...
end simulation...
1> print PSI
5.882e+1
```

```
1> # This design is very infeasible.
1> print IPSI
4.000
1> # The brace constraint is the culprit.
1> # I will increase the brace size a little.
1> print X(2)
-1.000
1> X(2) = -.8
1> simulate
begin simulation...
end simulation...
begin simulation...
end simulation...
begin simulation...
end simulation...
1> print PSI
6.162
1> # This looks like a reasonable starting design.
1> fprint
Created file "frame"
1> list frame
-------------------- frame --------------------
Values Of Different Terms In The Cost Function:
   Volume of designed elements    =  3.108e+3
   Sum of squares of story drifts =  6.922e-7
   Severe quake input energy      =  9.548e+1
   Severe quake dissipated energy =  7.568e+1
   Fuse dissipated energy         =  0.000
   Column dissipated energy       =  1.150
Element Section Design Variable Values:
  X1 = moment of inertia    = 50
  X2 = brace section area   =  1.900
Dummy Design Variable Values:
  X3 =  1.000,  should be = -.9446
-------------------- frame --------------------
1> # From the results listed above in the file frame,
1> # I will make the following adjustments:
1> Volmax = 6000
1> Cosdri = 10000000
1> Cosdis = .1
1> X(3) = -.94
1> simulate

1> optimize
Which optimization algo do you wish to use?
  0 = user supplied
  1 = feasible directions
  2 = conjugate gradient
  Type the corresponding number: 1
PARAMETER:  Eps = 0.2 : active constraint width
PARAMETER:  Dist = 0.2 : step length
PARAMETER:  Delta = 1.0 : eps reduction control
PARAMETER:  Epstol = 0.00001 : convergence tolerance on eps
PARAMETER:  Psitol = 0.00001 : convergence tolerance on psi
```

```
PARAMETER:  Geomslope = 100.0 : geometric constraint slope
PARAMETER:  Alpha = 0.2 : cost or psi reduction angle
PARAMETER:  Beta = 0.5 : step length search factor
PARAMETER:  Distmax = 1.0 : maximum step length
PARAMETER:  Scale = 1.0 : scale for active constraints
PARAMETER:  Gamma = 1.0 : infeasible cost penalty
PARAMETER:  Maxiter = 50 : maximum number of armijo loops
Type "1" if you have supplied a gradient scheme, otherwise type "0": 0
PARAMETER:  Deltax = 0.0001 : gradient perturbation
What is the maximum number of iterations you expect to complete? 30
Created file "dialogue"
```

Example of compiling the user-supplied software and initialization for the example problem without using the existing seismic-resistant frame design software

```
% make.memstart &
Beginning make.memstart
Restoring from <memstruct> ...
Identifier: structural memfile
----- LONG LIVE RATTLE/DELIGHT !!! -----
List <features> to see the new features.
1> PARAMETER:  Cosdri = Cosdri : drift cost coefficient
1> PARAMETER:  Cosdis = Cosdis : plastic energy cost coefficient
1> PARAMETER:  Volmax = 10000 : max structural volume
1> PARAMETER:  Drift = .005 : max moderate story drift
1> PARAMETER:  Accel = .5 : max moderate floor accel in gs
1> PARAMETER:  Sway = .01 : max severe structure sway
1> PARAMETER:  Colax = .5 : gravity column axial force factor
1> PARAMETER:  Colgra = .6 : gravity column yield factor
1> PARAMETER:  Colyld = 1 : moderate column yield factor
1> PARAMETER:  Colduc = 3 : severe column ductility
1> PARAMETER:  Bragra = .6 : gravity brace yield factor
1> PARAMETER:  Brayld = 1 : moderate brace yield factor
1> PARAMETER:  Braduc = 2 : severe brace ductility
1> including section        (9sec)
1> including objective      (15sec)
1> including conventional   (17sec)
1> including functional     (24sec)
1> Storing into memstart ...
CRUNCH ...
1> Goodbye Sir
End make.memstart

% go memstart
Restoring from memstart ...
Identifier: starting memfile
----- LONG LIVE RATTLE/DELIGHT !!! -----
List <features> to see the new features.
1> startoff
PARAMETER:  Tolx = 0.0001 : tolerance for resimulating
PARAMETER:  Numsim = 0 : number of simulations
PARAMETER:  Numsteps = 0 : number of simulation time steps
PARAMETER:  Numref = 0 : number stiffness reformulations
PARAMETER:  Maxerr = 0.0 : maximum energy error ratio
```

```
PARAMETER:  Print = 1 : simulation printout code
Type "1" if a response file exists, otherwise type "0": 0
---------- Next CRUNCH is forced. ----------
CRUNCH ...

1> # From here on the computation phase proceeds almost exactly
1> # the same as it does with the existing frame design software.
```

Example of the dialogue file generated for a couple of iterations of optimization for the example problem

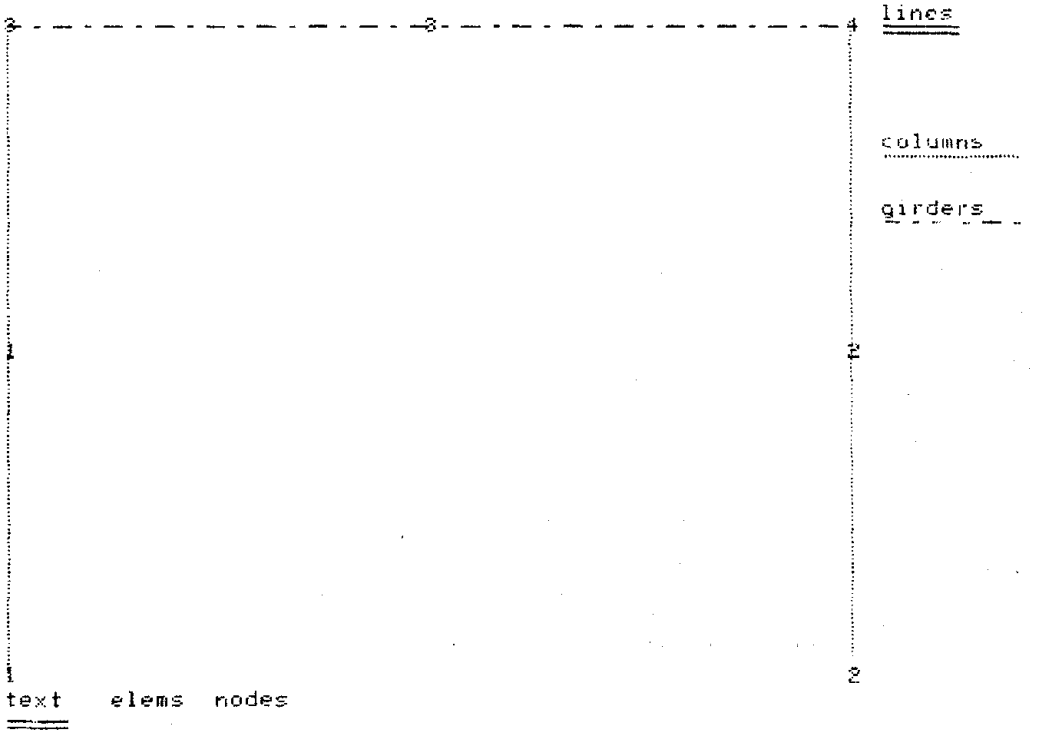
```
>> Gamma = 100
>> run 1

ITER = 0, COST =  1.507e+1, PSI =  6.162
Column X(3):
    -1.000
    -.8000
    -.9400
--------simulation output-------
number of simulations = 3
number of time steps = 283
number of stiffness reformulations = 173
maximum energy error ratio =  5.340e-5

ITER = 1, COST =  .7876, PSI =  3.807
Column X(3):
    -.9059
    -.6319
    -.9937
--------algorithm output--------
Eps =  .2000, theta = -1.106e+3, Dist =  .2000
direction finding phase:
Column direct(3):
     .4704
     .8406
    -.2685
  COST  norm =  1.250e+2  angle =  1.056e+2
  FINEQ(3,14)  norm =  5.316e+1  angle =  1.522e+2
  -X(1)  norm =  1.000e+2  angle =  1.181e+2
Matrix jacobian(3,3):
     0.000      0.000   1.250e+2
    -5.524  -5.287e+1      0.000
 -1.000e+2      0.000      0.000
steplength determination phase:
  first try in armijo did it
  nothing was violated
  geometric constraints violated
--------simulation output-------
number of simulations = 8
number of time steps = 848
number of stiffness reformulations = 356
maximum energy error ratio =  5.340e-5
```

```
Interrupt...
>> # Lets try the pajama mode.
>> pajama
>> sleep 10 ; cont 1

ITER = 2, COST =  3.833e+1, PSI =  .3886
Column X(3):
    -.6988
  8.008e-2
    -.6933
--------algorithm output--------
Eps =  .2000, theta = -3.268e+2, Dist =  .8000
direction finding phase:
Column direct(3):
     .2589
     .8900
     .3754
  COST  norm =  1.250e+2  angle =  6.795e+1
  FINEQ(7,34)  norm =  7.510e+2  angle =  1.126e+2
Matrix jacobian(2,3):
     0.000     0.000   1.250e+2
    -2.051    -7.049  -7.510e+2
steplength determination phase:
  armijo loop was increasing
  nothing was violated
  nothing was violated
  nothing was violated
  step size exceeded max
--------simulation output-------
number of simulations = 19
number of time steps = 1979
number of stiffness reformulations = 866
maximum energy error ratio =  5.340e-5

Interrupt...
>> simulate
>> dt 8
 TOTAL    DIRECT   NUMBER
 SECONDS  SECONDS  OF CALLS  PROCEDURE/MACRO NAME
 2.46e+2    ----     ---     Cpu-time since last "clear_time"
 2.05e+2    .367      17     ansrsim
 2.03e+2  2.03e+2     16     analys
 1.56e+2  1.08e+1      5     state
 1.55e+2    .683       2     armijo
 7.71e+1  1.27e+1      2     actgrad
 6.17e+1    .633       2     sensitivity
 6.08e+1    .417       2     perturb
 6.08e+1    0.00       2     gradients
>> stop_opt
>> # At this point I would probably increase Volmax and
>> # continue on with pajama mode for 2 or 3 more iterations.
```

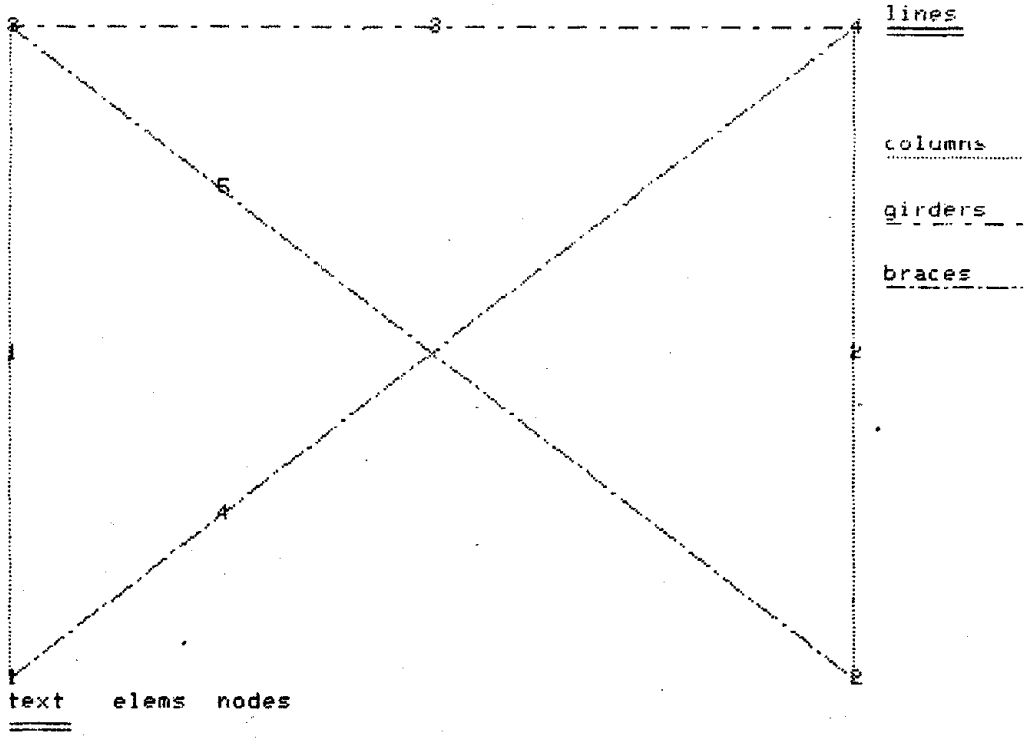
APPENDIX 5 : SAMPLE DELIGHT.STRUCT GRAPHICAL INPUT AND OUTPUT

If the seismic-resistant frame design software is used an interactive graphical pre-processor defines the frame at hand as shown. Also shown is a graphical display of the frame and the ground acceleration record after pre-processing.

```
% go '(memframe)'
Restoring from (memframe) ...
Identifier: frame memfile

------ LONG LIVE RATTLE/DELIGHT !!! ------

List (features) to see the new features.

1) terminal hp
1) finput
INTERACTIVE GRAPHICAL INPUT OF GEOMETRY AND LOADS
   (you should be on a color terminal, Sir)
   (units are inches, kips, and seconds throughout)

Type the number of stories: 1
Type the number of bays: 1
Type story heights from bottom to top (one per line):
144
Type bay widths from left to right (one per line):
360
-------------------------------------------------------------------------
From this point on the user inputs lists of various data.
Each line of the list may contain one or more elements.
The list is terminated by a line consisting of a single zero.
While making a list, errors may be undone.  This is done by
   typing the negative of the first element of the bad line.
-------------------------------------------------------------------------
Type a list of story and bay numbers of braced panels:
```

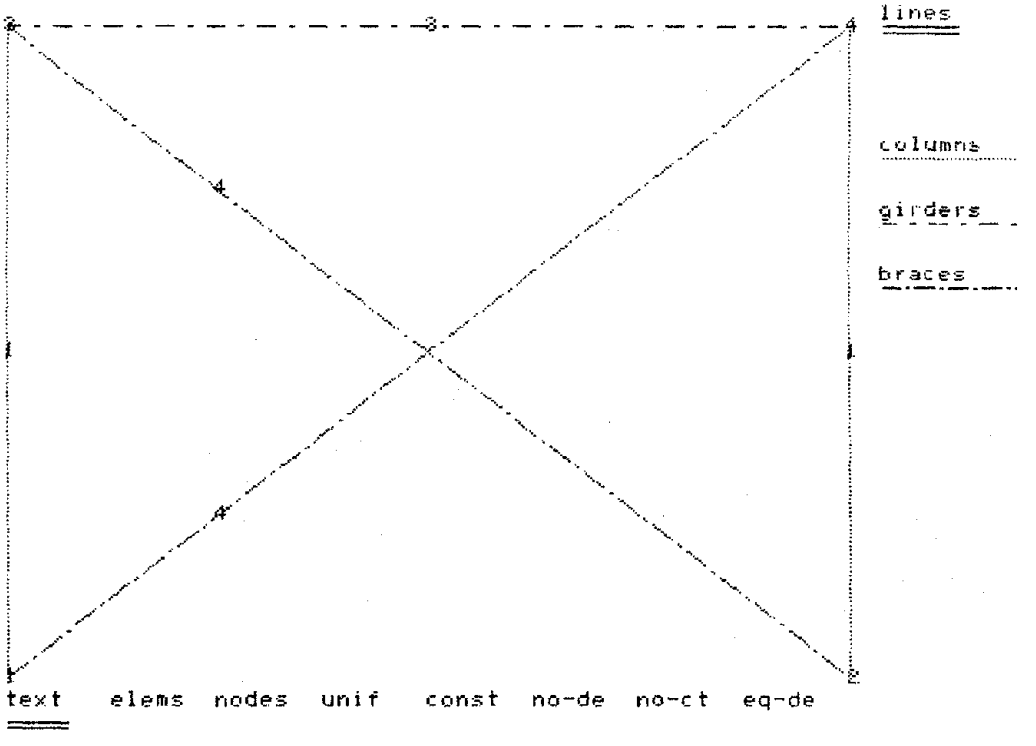


```
Type a list of story and bay numbers of braced panels:
1 1
0
Type a list of numbers of elements to be erased:
0
Type a list of numbers of nodes to be erased:
0
Type a list of shear girder element numbers:
0
Type a list of dissipator element numbers:
0
Type a list of rubber bearing element numbers:
0
Type a list of loaded girder numbers with their uniform load:
```



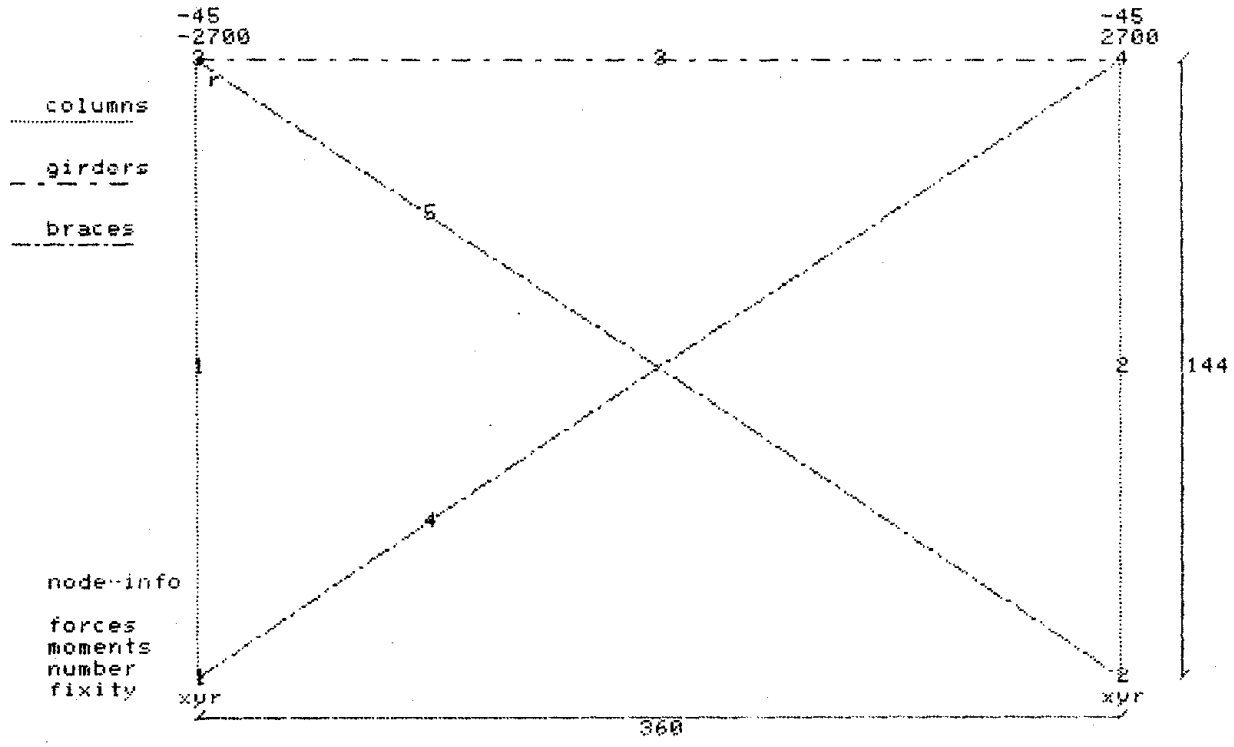

```
Type a list of loaded girder numbers with their uniform load:
4 .25
   Specified element is not a girder, try again Sir:
3 .25
0
Type a list of loaded node numbers with their point loads:
0
Type a list of constrained node numbers with their code:
   (code is 3-digit integer where digits represent hori-disp,
    vert-disp, rotation with 0=free, 1=constrained)
   (default code is 111 for the base nodes):
3 001
4 001
0
Type a list of numbers of elements not subject to design:
```

```
Type a list of numbers of elements not subject to design:
3
2
-2
0
Type a list of numbers of elements not subject to constraint:
2
3
0
Type a list of groups of elements constrained equal during design
   (end each line with a zero):
1 2 0
4 5 0
0
Type a list of element numbers with their initial design values
   (moments of inertia for columns and girders, areas for braces
    thickness for truss girders, and edge lengths for bearings)
   (elements are systematically erased as values are specified)
   (default values are the max values for the elements):
```

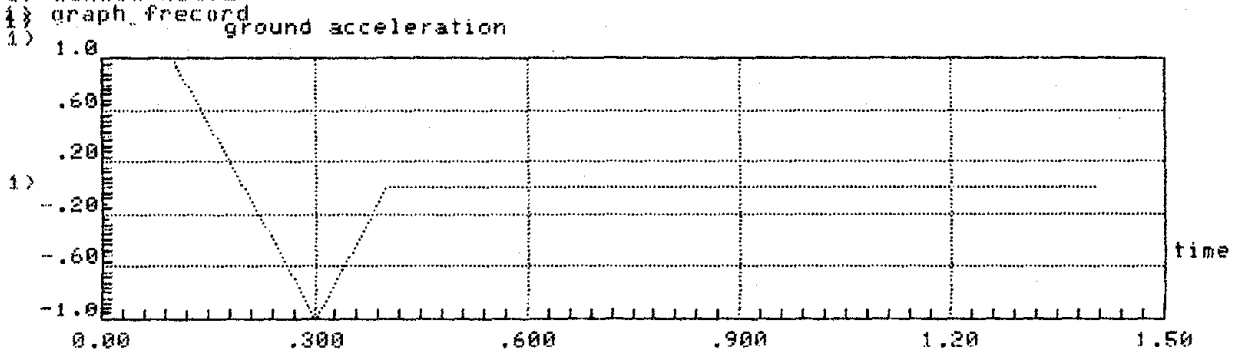



```
Type a list of element numbers with their initial design values
   (moments of inertia for columns and girders, areas for braces
    thickness for truss girders, and edge lengths for bearings)
   (elements are systematically erased as values are specified)
   (default values are the max values for the elements):
1 50
4 1
0
```

```
1> fstructy
1>
```




```
1> erase
1> window_list
window name ------ viewport coordinates ----- ====== world coordinates ======
     screen      0.0      0.0      1.0      1.0      0.0      0.0      1.0      1.0
      half1      0.0      0.0      1.0      .50      0.0      0.0      1.0      1.0
      half2      0.0      .50      1.0      1.0      0.0      0.0      1.0      1.0
     third1      0.0      0.0      1.0      .34      0.0      0.0      1.0      1.0
     third2      0.0      .34      1.0      .66      0.0      0.0      1.0      1.0
     third3      0.0      .66      1.0      1.0      0.0      0.0      1.0      1.0
     quart1      0.0      0.0      1.0      .25      0.0      0.0      1.0      1.0
     quart2      0.0      .25      1.0      .50      0.0      0.0      1.0      1.0
     quart3      0.0      .50      1.0      .75      0.0      0.0      1.0      1.0
     quart4      0.0      .75      1.0      1.0      0.0      0.0      1.0      1.0
    fourth1      0.0      0.0      .50      .50      0.0      0.0      1.0      1.0
    fourth2      .50      0.0      1.0      .50      0.0      0.0      1.0      1.0
    fourth3      0.0      .50      .50      1.0      0.0      0.0      1.0      1.0
    fourth4      .50      .50      1.0      1.0      0.0      0.0      1.0      1.0
1> window half2
1> graph frecord
1>
```

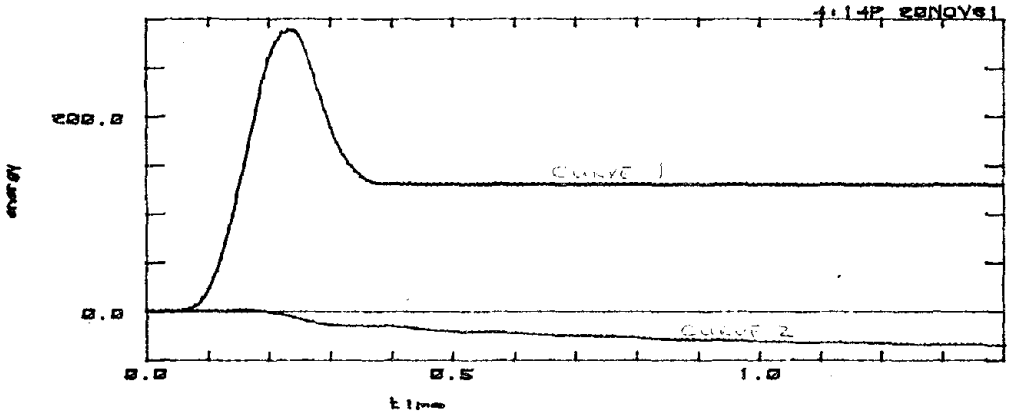


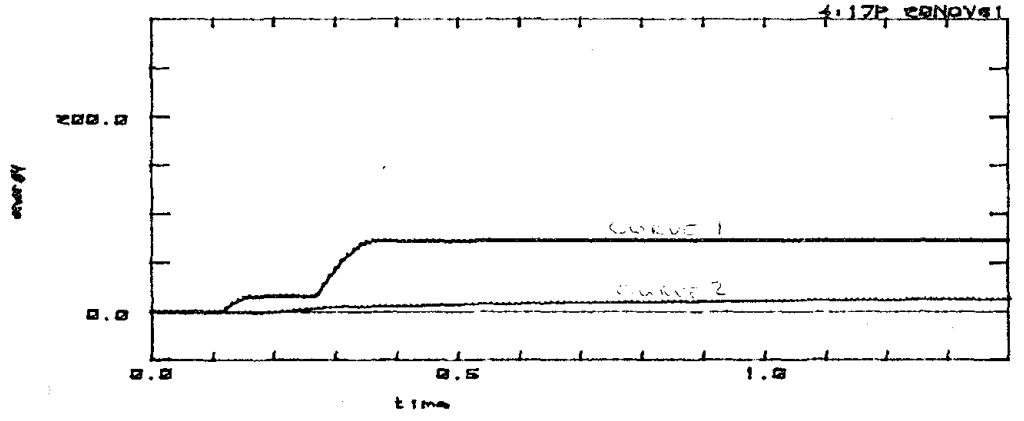

```
1>
```

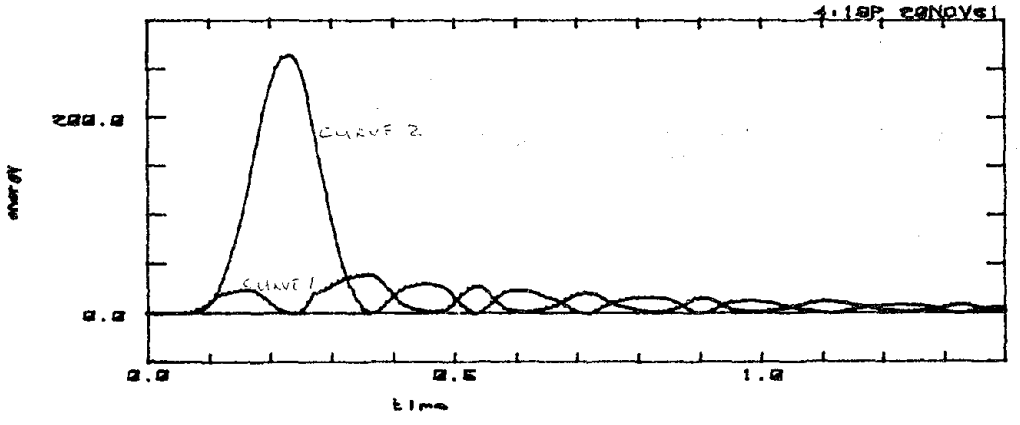
The seismic-resistant frame design post-processing software allows plotting of various response quantities following simulation as shown.

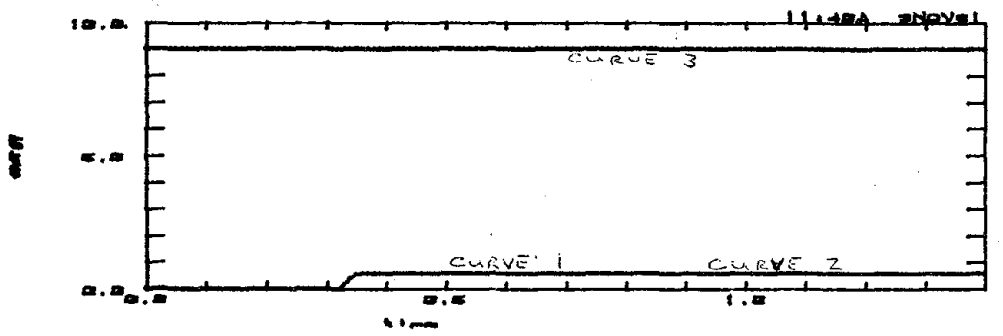
plot 1 curve 1	input energy for starting design under severe quake
curve 2	work done by the gravity loads for the starting design under severe quake
plot 2 curve 1	plastic dissipated energy for the starting design under severe quake
curve 2	damped energy for the starting design under severe quake
plot 3 curve 1	elastic strain energy for the starting design under severe quake
curve 2	kinetic energy for the starting design under severe quake
plot 4 curve 1	column bottom node energy dissipation for starting design under severe quake
curve 2	column top node energy dissipation for starting design under severe quake
curve 3	allowable level
plot 5 curve 1	first brace energy dissipation for starting design under severe quake
curve 2	second brace energy dissipation for starting design under severe quake
curve 3	allowable level
plot 6 curve 1	first brace force-displacement hysteresis for starting design under severe quake
curve 2	second brace force-displacement hysteresis for starting design under severe quake
plot 7 curve 1	starting design story drift under moderate quake
curve 2	final design story drift under moderate quake
curves 3	allowable levels
plot 8 curve 1	starting design floor acceleration under moderate quake
curve 2	final design floor acceleration under moderate quake
curves 3	allowable levels
plot 9 curve 1	starting design structure sway under severe quake
curve 2	final design structure sway under severe quake
curves 3	allowable levels
plot 10 curve 1	column bottom node end moment for starting design under moderate quake
curve 2	column bottom node end moment for starting design under moderate quake
curves 3	allowable levels
plot 11 curve 1	column bottom node end moment for final design under moderate quake
curve 2	column bottom node end moment for final design under moderate quake
curves 3	allowable levels
plot 12 curve 1	first brace axial force for starting design under moderate quake
curve 2	second brace axial force for starting design under moderate quake
curves 3	allowable levels

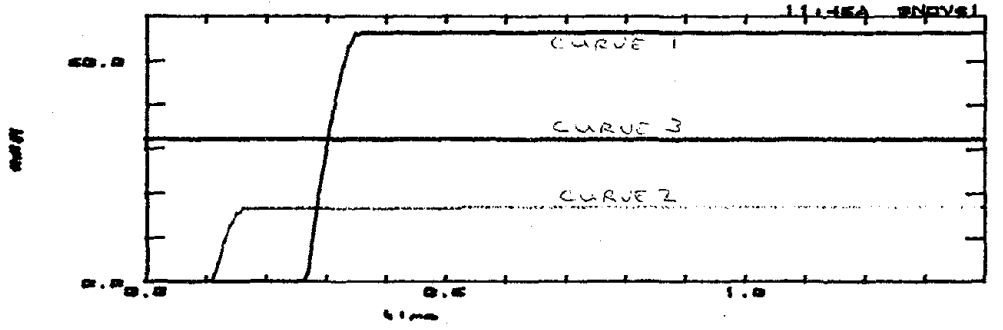
plot 13 curve 1	first brace axial force for final design under moderate quake
curve 2	second brace axial force for final design under moderate quake
curves 3	allowable levels

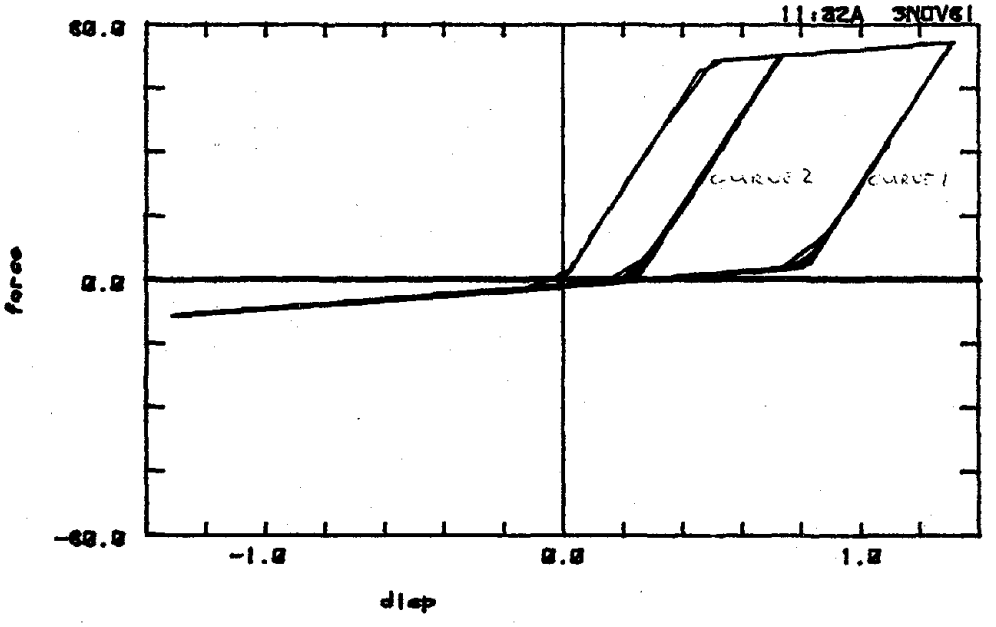


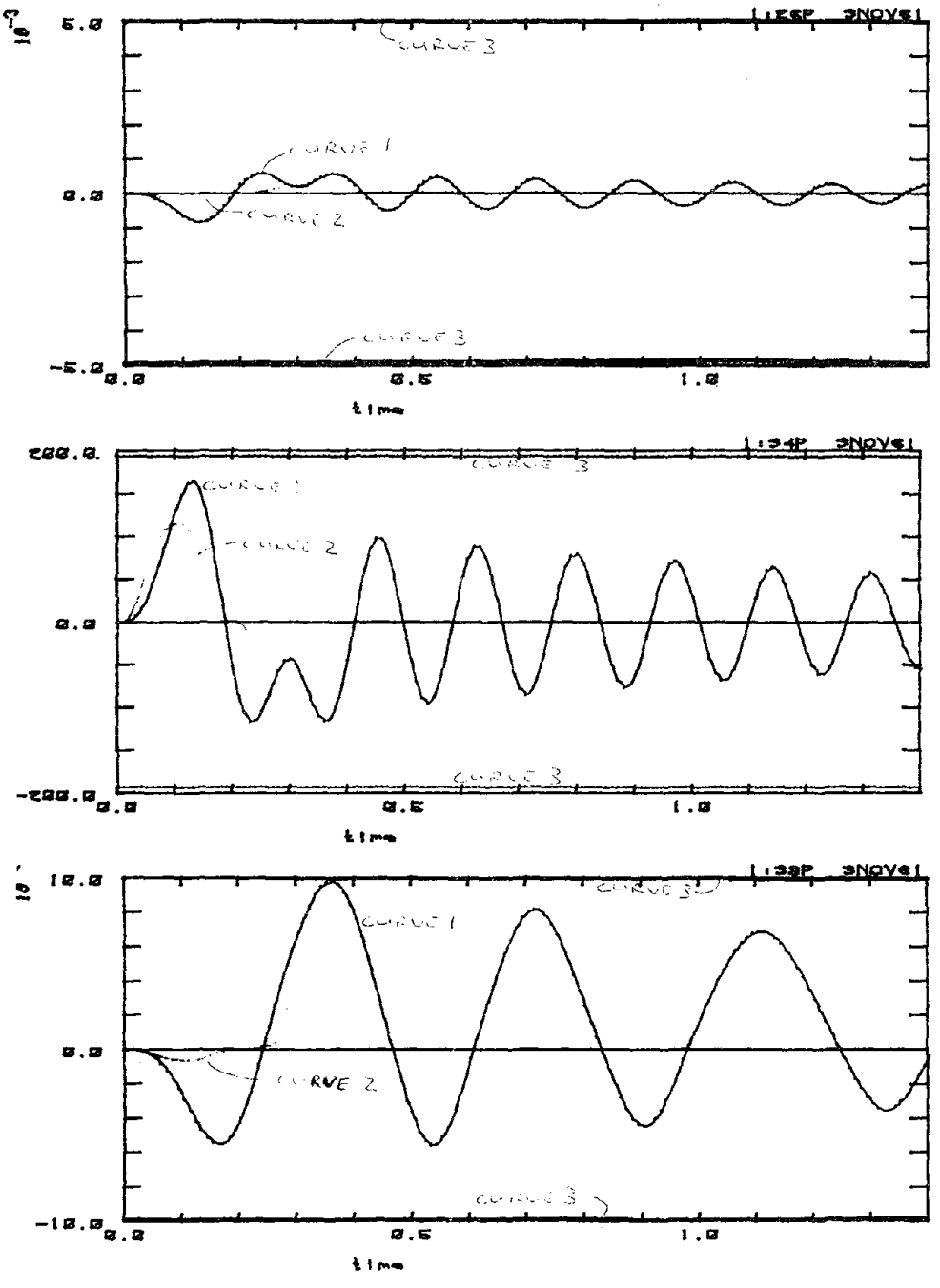


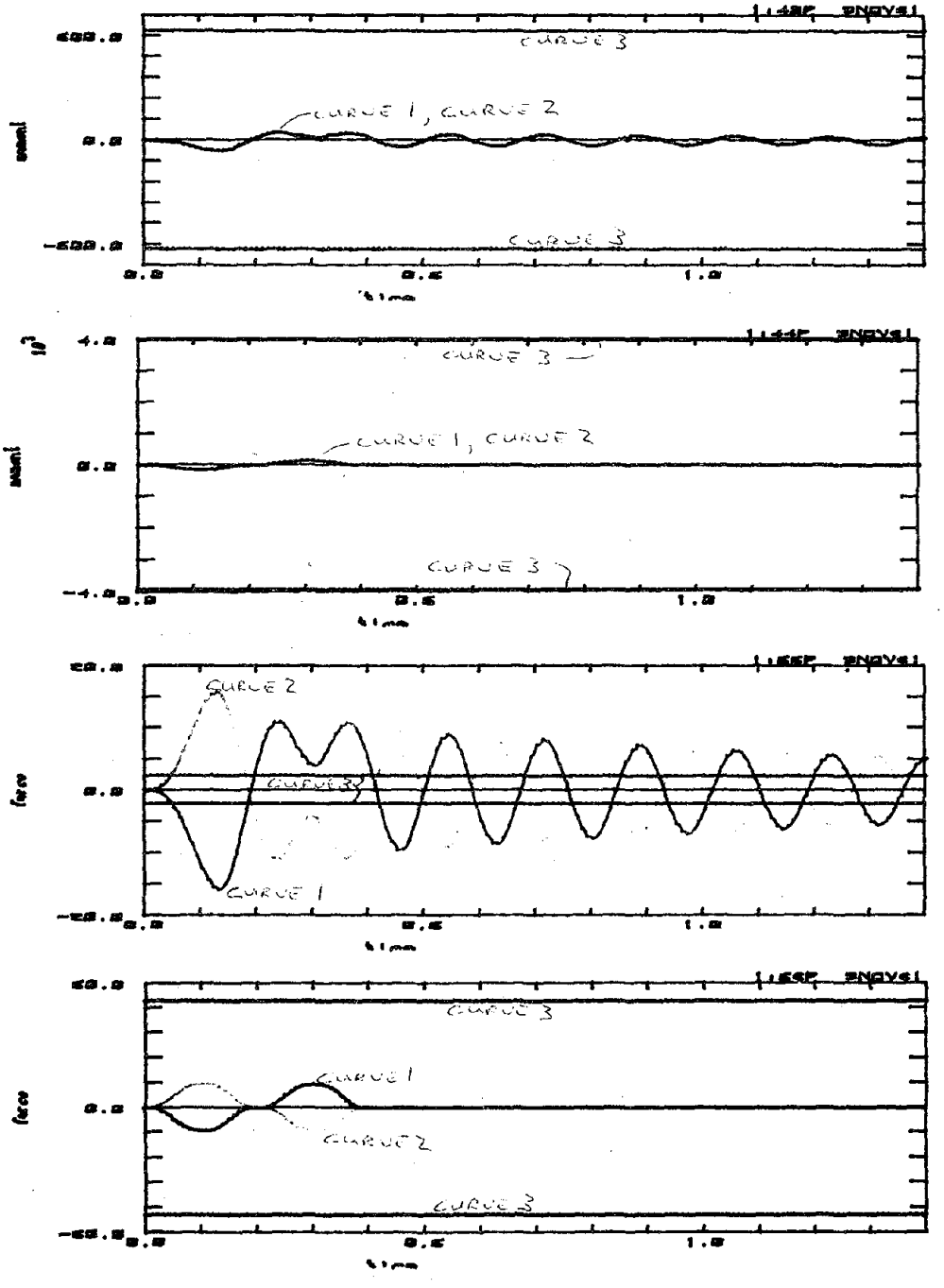



CURVE 3
CURVE 1
CURVE 2


CURVE 1
CURVE 3
CURVE 2

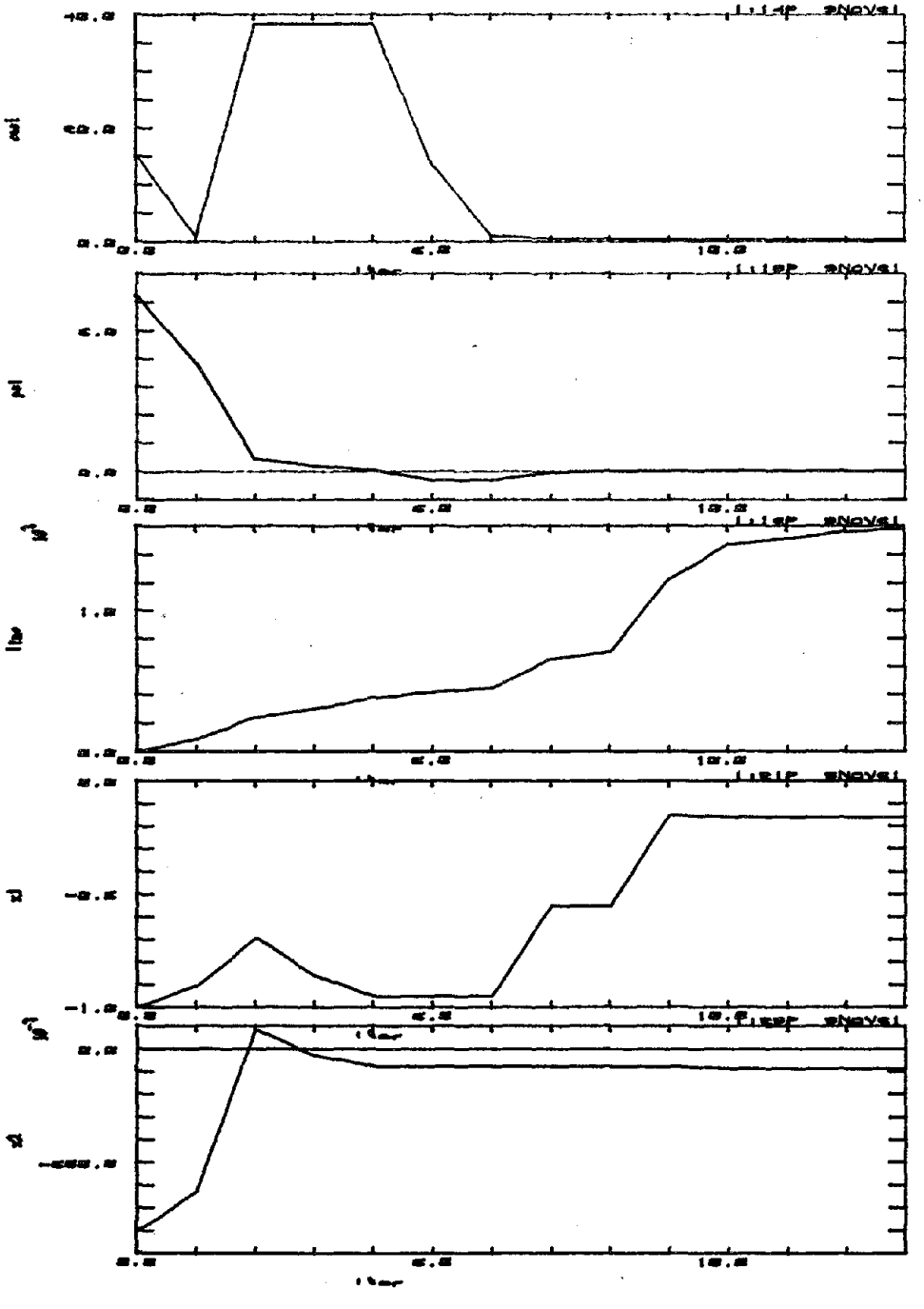

11:32A 3NOV81
60.0
0.0
-60.0
force
-1.0
0.0
1.0
disp
CURVE 2
CURVE 1




CURVE 3
CURVE 1, CURVE 2
CURVE 3
CURVE 3
CURVE 1, CURVE 2
CURVE 3
CURVE 2
CURVE 3
CURVE 1
CURVE 3
CURVE 1
CURVE 2
CURVE 3

Software is available for plotting optimization iteration histories for various quantities as shown.

plot 1	cost function vs. iteration
plot 2	maximum constraint value vs. iteration
plot 3	cumulative cpu-time vs. iteration
plot 4	design variable number one (column moment of inertia) vs. iteration
plot 5	design variable number two (brace cross-sectional area) vs. iteration



APPENDIX 6 : SAMPLE DELIGHT.STRUCT FILE INPUT AND OUTPUT

If the existing seismic-resistant frame design software is not used, the user must supply a file named startsetup such as the following

```
COSTL = 2

parameter Cosdri=Cosdri 'drift cost coefficient'
parameter Cosdis=Cosdis 'plastic energy cost coefficient'
parameter Volmax=10000 'max structural volume'
parameter Drift=.005 'max moderate story drift'
parameter Accel=.5 'max moderate floor accel in gs'
parameter Sway=.01 'max severe structure sway'
parameter Colax=.5 'gravity column axial force factor'
parameter Colgra=.6 'gravity column yield factor'
parameter Colyld=1 'moderate column yield factor'
parameter Colduc=3 'severe column ductility'
parameter Bragra=.6 'gravity brace yield factor'
parameter Brayld=1 'moderate brace yield factor'
parameter Braduc=2 'severe brace ductility'

include_and_print section
include_and_print objective
include_and_print conventional
include_and_print functional

clear_time
store memstart 'starting memfile'
quit
```

If the existing seismic-resistant frame design software is not used, the user must supply a file named section such as the following

```
# procedure for determining element section properties
procedure section (x,secprop) {
  array x(), secprop(,)

  # determine column and brace section properties
  icol = (1550+1450*x(1))/2
  acol = .8*sqrt(icol)
  dcol = 2*icol**.25/.78
  mpcol = 36*(acol*dcol/8+1.5*icol/dcol)
  abra = (11+9*x(2))/2
  ibra = .169*abra**3
  tbra = 36
  cbra = (PI/387.732)**2*29000*ibra/abra

  # fill in matrix secprop
  for i = 1 to 2 {
    secprop(i,1) = acol  ; secprop(i,2) = icol
    secprop(i,3) = .05   ; secprop(i,4) = mpcol
    secprop(i+3,1) = abra ; secprop(i+3,2) = .05
    secprop(i+3,3) = tbra ; secprop(i+3,4) = cbra
    }
```

```
  # re-assign properties for girder
  secprop(3,1) = 24.53 ; secprop(3,2) = 2500
  secprop(3,3) = .05   ; secprop(3,4) = 8147
  }
```

If the existing seismic-resistant frame design software is not used, the user must supply a file named objective such as the following

```
# procedure for computing the cost function
procedure objective (x,cost) {
  array x()
  import Cosdri, Cosdis, Resp, Drift

  # add contribution from moderate quake drift bounds
  cost = Cosdri*Drift**2*(x(3)+1)/2

  # add contribution from severe quake dissipated energy
  cost = cost+Cosdis*Resp(9+12*141)
  }
```

If the existing seismic-resistant frame design software is not used, the user must supply a file named conventional such as the following

```
# procedure for computing conventional constraint functions
procedure conventional (i,x,ineq) {
  array x()
  import Resp, Volmax, Colax, Colgra, Bragra, Colduc, Braduc

  # evaluate volume constraint
  if (i == 1)
    ineq = 2*(144*SECPROP(1,1)+387.732*SECPROP(4,1))/Volmax-1

  # evaluate column gravity load axial constraint
  if (i == 2) {
    if (Resp(5) >= 0) ineq = Resp(5)/36/SECPROP(1,1)/Colax-1
    else {
      allow=min(36*SECPROP(1,1),(PI/144)**2*29000*SECPROP(1,2))
      ineq = -Resp(5)/allow/Colax-1
      }
    }

  # evaluate column gravity load moment constraints
  if (i == 3 | i == 4) {
    ax = abs(Resp(5))/.15/36/SECPROP(1,1)
    if (ax > 1) factor = (1-.15*ax)/.85
    else factor = 1
    ineq = abs(Resp(i+3))/SECPROP(1,4)/factor/Colgra-1
    }

  # evaluate brace gravity load force constraints
  if (i == 5 | i == 6) {
    if (Resp(i+3) >= 0)
      ineq = Resp(i+3)/SECPROP(4,1)/SECPROP(4,3)/Bragra-1
    else ineq = -Resp(i+3)/SECPROP(4,1)/SECPROP(4,4)/Bragra-1
```

```
    }

  # evaluate column severe quake energy constraints
  if (i == 7 | i == 8) {
    ax = 45/.15/36/SECPROP(1,1)
    if (ax > 1) factor = (1-.15*ax)/.85
    else factor = 1
    eyield = (SECPROP(1,4)*factor)**2*144/12/29000/SECPROP(1,2)
    duct = (Colduc-1)*.95*(2+.05*(Colduc-1))
    ineq = Resp(9+(10+i)*141)/eyield/duct-1
    }

  # evaluate brace severe quake energy constraints
  if (i == 9 | i == 10) {
    eyield = 36**2*SECPROP(4,1)*387.732/29000/2
    duct = (Braduc-1)*.95*(2+.05*(Braduc-1))
    ineq = Resp(9+(3*i-8)*141)/eyield/duct-1
    }
  }
```

If the existing seismic-resistant frame design software is not used, the user must supply a file named functional such as the following

```
# procedure for computing functional constraint functions
procedure functional (i,x,steps,fineq) {
  array x(), fineq()
  import Resp, Colyld, Brayld, Drift, Accel, Sway

  # evaluate column moderate quake moment constraints
  if (i == 1 | i == 2) {
    ax = 45/.15/36/SECPROP(1,1)
    if (ax > 1) myield = SECPROP(1,4)*Colyld*(1-.15*ax)/.85
    else myield = SECPROP(1,4)*Colyld
    for j = 1 to steps
      fineq(j) = abs(Resp(9+(2+i)*141+j))/myield-1
    }

  # evaluate brace moderate quake force constraints
  if (i == 3 | i == 4) {
    fyield1 = SECPROP(4,1)*SECPROP(4,3)*Brayld
    fyield2 = SECPROP(4,1)*SECPROP(4,4)*Brayld
    for j = 1 to steps {
      resp = Resp(9+(2+i)*141+j)
      if (resp >= 0) fineq(j) = resp/fyield1-1
      else fineq(j) = -resp/fyield2-1
      }
    }

  # evaluate moderate quake story drift constraint
  if (i == 5)
    for j = 1 to steps
      fineq(j) = abs(Resp(9+141+j))/144/Drift-1

  # evaluate moderate quake floor acceleration constraint
```

```
if (i == 6)
  for j = 1 to steps
    fineq(j) = abs(Resp(9+2*141+j))/386.088/Accel-1

# evaluate moderate quake dummy variable drift constraint
if (i == 7)
  for j = 1 to steps
    fineq(j) = 2*(Resp(9+141+j)/Drift/144)**2/(x(3)+1)-1

# evaluate severe quake structure sway constraint
if (i == 8)
  for j = 1 to steps
    fineq(j) = abs(Resp(9+15*141+j))/144/Sway-1
}
```

If the seismic-resistant frame design software is used, the user should check and possibly modify the assumed values of constants in the file assumptions which follows

*** ASSUMED PARAMETER VALUES ***

```
Ratio of live uniform load to total uniform load = 0.2
Global damping ratio = 0.02
Ratio of number of stories to fundamental period = 4.52
Ratio of fundamental period to second period = 3.
Young's modulus for steel = 29000.
Yield stress for steel = 36.
Strain hardening ratio for steel = 0.05
For columns:
  50.      < inertia     < 1500.
  moment yield coordinate fraction = 1.
  axial yield coordinate fraction = 0.15
  radius of gyration = 0.436     * depth   ** 1.
  for inertia <= 429.
  depth            = 2.56      * inertia ** 0.25
  otherwise
  depth            = 2.56      * inertia ** 0.25
For rubber bearings:
  5.       < edge length < 100.
  shear modulus of rubber = 0.0984
  bearing height = 25.
For girders:
  125.     < inertia     < 2500.
  steel poisson ratio = 0.3
  radius of gyration = 0.52      * depth   ** 0.92
  depth            = 2.66      * inertia ** 0.287
For dissipators:
  0.25     < thickness   < 10.
  height of dissipator = 75.
  width of dissipator = 36.
For braces:
  1.       < area        < 10.
  inertia          = 0.169     * area    ** 3.
```

If the seismic-resistant frame design software is used, the user should supply the file record which provides the ground motion record such as the following

*** GROUND ACCELERATION RECORD ***

```
  the maximum allowable number of data points in a record is 999
  units for the accelerations are inches per seconds squared
  EXAMPLE PULSE
  number of data points = 4
  time increment in seconds = 0.1
  peak acceleration for this record in inches per seconds squared = 1.
  severe quake acceleration in g's = 1.
  moderate quake acceleration in g's = 0.3
  format for the following accelerations is (4f4.0)
 1. 0. -1. 0.
```

If the seismic-resistant frame design software is used, the ansrdata files are automatically generated otherwise these files must be written by the user. For the example problem, the files ansrdata1, ansrdata2, and ansrdata3 are used. The file ansrdata3 for the severe quake loading is listed here, while the files ansrdata1 for gravity loading and ansrdata2 for the moderate quake loading are similar.

```
start  SEISMIC ANALYSIS OF RECTANGULAR BRACED PLANAR FRAMES
     996
   4   4   0   5   3   2   3
   10.   d+000.   d+00
   20.3600d+030.   d+00
   30.   d+000.1440d+03
   40.3600d+030.1440d+03
   1   0   0   1   1   1   0   4   1
   1   1   1   1   1   1   1
   2   1   1   1   1   1   1
   3   0   0   1   1   1   1
   4   0   0   1   1   1   1
   0   1   1   1   1   0   2   1   3
   0   1   1   1   1   0   2   2   4
   1   0   1   1   1   0   2   3   4
   30.9324d-010.9324d-01
   40.9324d-010.9324d-01
   1   1   1   1   0   0   0  1400.1000e-01
   2  GRAVITY LOADS
   30.   e+00-.4500d+020.   e+000.   e+000.   e+00-.2700d+04
   40.   e+00-.4500d+020.   e+000.   e+000.   e+000.2700d+04
   4   0   0   10.1000d+000.   e+000.   e+000.3861d+03
  EXAMPLE PULSE                                    (4f4.0)
 1. 0. -1. 0.
0.8520d+000.   e+000.3521d-03
   5   0   1   1   0   1   1  200.1000e-010.1000e-010.1000e-01
0.1000e+01
   0   1   1   0   1  20   10.1000e-010.1000e-010.1000e-01
  10  50   0   1   0   0   1   0   0   1
   3
   2   2   1   2   0   2   2
   10.2900d+050.5000d-010.5676d+010.5000d+02 4.00 4.00 2.00
```

```
 20.2900d+050.5000d-010.5676d+010.5000d+02 4.00 4.00 2.00
 1   20.1000e+010.5705d+030.5705d+030.2043d+030.2043d+03 1.00 0.15 1.00 0.15
 2   20.1000e+010.5705d+030.5705d+030.2043d+030.2043d+03 1.00 0.15 1.00 0.15
 10.4500d+020.   e+000.   e+00-.4500d+02
 20.4500d+020.   e+000.   e+00-.4500d+02
1  1  3  0  0  0  1  0  1  1  1       3   1 1.00
2  2  4  0  0  0  2  0  2  2  1       0   2 1.00
 2   1   3   1   0   1   1
 10.2900d+050.5000d-010.2453d+020.2500d+04 4.00 4.00 2.00
 1   10.1000e+010.8147d+040.8147d+04
 10.   e+000.4500d+020.2700d+040.   e+000.4500d+02-.2700d+04
3  3  4  0  0  0  1  0  1  1  0       0   1 1.00
 1   2   4   2
 10.2900d+050.5000d-010.3600d+020.3218d+000.1000e+01
 20.2900d+050.5000d-010.3600d+020.3218d+000.1000e+01
 4   1   4   10.1000d+010.   e+00   0   0 213   1
 5   3   2   20.1000d+010.   e+00   0   0 213   1
```

If the seismic-resistant frame design software is used the following file description is generated which identifies and numbers the constraints for the benefit of the user

*** DESCRIPTION OF THE OPTIMIZATION PROBLEM ***

The Problem:
  Design variable list: Xi (i = 1 to 3)
  Minimize the cost function: F(X)
  Conventional constraints: Gj(X) < 0 (j = 1 to 10)
  Functional constraints: max over t (Hk(X,t)) < 0 (k = 1 to 8)

Design Variables:
  X 1 = column moment of inertia for element(s):
      1 2
  X 2 = brace area for element(s):
      4 5
  X 3 = dummy variable for story drift

Cost Function:
  F = Cosvol*structural volume of designed elements
    + Cosdri*sum of squares of moderate quake story drifts
    + Cosinp*input energy from severe quake
    + Cosdis*severe quake inelastic energy
    + Cosfus*fuse severe quake inelastic energy
    + Coscol*column severe quake inelastic energy

Volume Conventional Constraint:
  volume of designed elements < Volmax
    G 1

Gravity Load Conventional Constraints:
  |column axial force| < column yield or buckling force*Colax
    G 2: element 1
  |column end moment| < column yield moment*Colgra
    G 3: element 1 node 1
    G 4: element 1 node 3

```
  |brace axial force| < brace yield or buckling force*Bragra
    G  5:   element  4
    G  6:   element  5

Severe Quake Conventional Constraints:
  column end energy < column failure energy (Colduc)
    G  7:   element  1  node  1
    G  8:   element  1  node  3
  brace energy < brace failure energy (Braduc)
    G  9:   element  4
    G 10:   element  5

Moderate Quake Functional Constraints:
  |column end moment| < column yield moment*Colyld
    H  1:   element  1  node  1
    H  2:   element  1  node  3
  |brace force| < brace yield or buckling force*Brayld
    H  3:   element  4
    H  4:   element  5
  |story drift| < Drift
    H  5:   top node  3  bottom node  0
  |floor acceleraction| < Accel*g
    H  6:   floor node  3
  (story drift)**2 < dummy design variable
    H  7:   drift  1  dummy variable  3

Severe Quake Functional Constraints:
  |structure sway| < Sway
    H  8:   top node  3  bottom node  0
```

If the seismic-resistant frame design software is used the following file sizefile is automatically generated, otherwise this file must be supplied by the user

```
array Q(3),INEQL(3),FINEQL(3)
NPARAM =  3
NSIMPAR =  2
NLOAD = 3
NINEQ =  10
NFINEQ =  8
NPROP = 4
NELS =  5
NRESP =      6786
Q(1) = 1
Q(2) =  141
Q(3) =  141
INEQL(1) = 2
INEQL(2) =  7
INEQL(3) =  7
FINEQL(1) = 1
FINEQL(2) = 1
FINEQL(3) =  8
```

If the seismic-resistant frame design software is used the following file xfile is automatically generated, otherwise this file must be supplied by the user

X(1)=-.1617
X(2)=-8.704e-2
X(3)=-.9973

The following file state provides information regarding the amount of constraint violation following a simulation

*** COST = .3346
PSI = -6.422e-4 IPSI = 7 JPSI = 12

INEQ(1): percent = 99
INEQ(2): percent = 12
INEQ(3): percent = 0
INEQ(4): percent = 0
INEQ(5): percent = 5
INEQ(6): percent = 5
INEQ(7): percent = 0
INEQ(8): percent = 0
INEQ(9): percent = 0
INEQ(10): percent = 0

FINEQ(1,12): percent = 4
FINEQ(2,12): percent = 4
FINEQ(3,12): percent = 22
FINEQ(4,32): percent = 22
FINEQ(5,12): percent = 4
FINEQ(6,12): percent = 59
FINEQ(7,12): percent = 100
FINEQ(8,12): percent = 6

If the seismic-resistant frame design software is used the following file frame may be generated may be generated following a simulation to provide more information

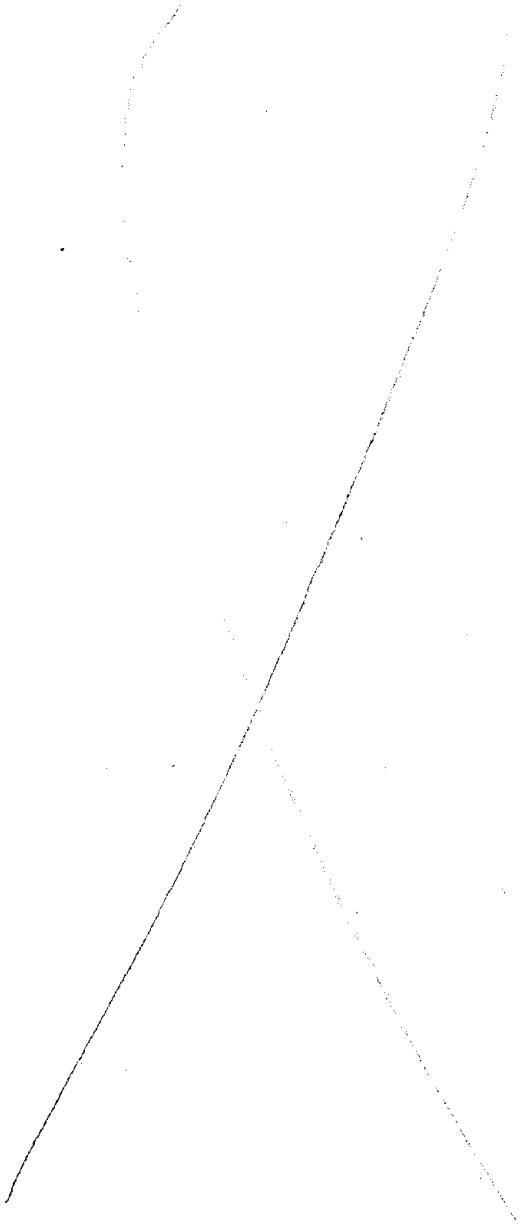
Values Of Different Terms In The Cost Function:
Volume of designed elements = 9.890e+3
Sum of squares of story drifts = 3.344e-8
Severe quake input energy = 2.367
Severe quake dissipated energy = 0.000
Fuse dissipated energy = 0.000
Column dissipated energy = 0.000

Element Section Design Variable Values:
X1 = moment of inertia = 658
X2 = brace section area = 5.108

Dummy Design Variable Values:
X3 = -.9973, should be = -.9973

The iteration histories of the cost, max constraint value, cumulative cpu time, and design variables may be saved in the following file history following optimization

ITER = 4 ; NPARAM = 3
array Costsave(ITER+1),Psisave(ITER+1),Timesave(ITER+1)
array Xsave(NPARAM,ITER+1)
Costsave(1)= 1.507e+1
Costsave(2)= .7876
Costsave(3)= 3.833e+1
Costsave(4)= 3.833e+1
Costsave(5)= 3.833e+1
Psisave(1)= 6.162
Psisave(2)= 3.807
Psisave(3)= .3886
Psisave(4)= .1615
Psisave(5)= 1.853e-2
Timesave(1)= 5.400
Timesave(2)= 9.027e+1
Timesave(3)= 2.437e+2
Timesave(4)= 3.038e+2
Timesave(5)= 3.789e+2
Xsave(1,1)=-1.000
Xsave(2,1)=-.8000
Xsave(3,1)=-.9400
Xsave(1,2)=-.9059
Xsave(2,2)=-.6319
Xsave(3,2)=-.9937
Xsave(1,3)=-.6988
Xsave(2,3)= 8.008e-2
Xsave(3,3)=-.6933
Xsave(1,4)=-.8640
Xsave(2,4)=-3.263e-2
Xsave(3,4)=-.6933
Xsave(1,5)=-.9532
Xsave(2,5)=-7.789e-2
Xsave(3,5)=-.6933



EARTHQUAKE ENGINEERING RESEARCH CENTER REPORTS

NOTE: Numbers in parenthesis are Accession Numbers assigned by the National Technical Information Service; these are followed by a price code. Copies of the reports may be ordered from the National Technical Information Service, 5285 Port Royal Road, Springfield, Virginia, 22161. Accession Numbers should be quoted on orders for reports (PB --- ---) and remittance must accompany each order. Reports without this information were not available at time of printing. Upon request, EERC will mail inquirers this information when it becomes available.

EERC 67-1 "Feasibility Study Large-Scale Earthquake Simulator Facility," by J. Penzien, J.G. Bouwkamp, R.W. Clough and D. Rea - 1967 (PB 187 905)A07

EERC 68-1 Unassigned

EERC 68-2 "Inelastic Behavior of Beam-to-Column Subassemblages Under Repeated Loading," by V.V. Bertero - 1968 (PB 184 888)A05

EERC 68-3 "A Graphical Method for Solving the Wave Reflection-Refraction Problem," by H.D. McNiven and Y. Mengi - 1968 (PB 187 943)A03

EERC 68-4 "Dynamic Properties of McKinley School Buildings," by D. Rea, J.G. Bouwkamp and R.W. Clough - 1968 (PB 187 902)A07

EERC 68-5 "Characteristics of Rock Motions During Earthquakes," by H.B. Seed, I.M. Idriss and F.W. Kiefer - 1968 (PB 188 338)A03

EERC 69-1 "Earthquake Engineering Research at Berkeley," - 1969 (PB 187 906)A11

EERC 69-2 "Nonlinear Seismic Response of Earth Structures," by M. Dibaj and J. Penzien - 1969 (PB 187 904)A08

EERC 69-3 "Probabilistic Study of the Behavior of Structures During Earthquakes," by R. Ruiz and J. Penzien - 1969 (PB 187 886)A06

EERC 69-4 "Numerical Solution of Boundary Value Problems in Structural Mechanics by Reduction to an Initial Value Formulation," by N. Distefano and J. Schujman - 1969 (PB 187 942)A02

EERC 69-5 "Dynamic Programming and the Solution of the Biharmonic Equation," by N. Distefano - 1969 (PB 187 941)A03

EERC 69-6 "Stochastic Analysis of Offshore Tower Structures," by A.K. Malhotra and J. Penzien - 1969 (PB 187 903)A09

EERC 69-7 "Rock Motion Accelerograms for High Magnitude Earthquakes," by H.B. Seed and I.M. Idriss - 1969 (PB 187 940)A02

EERC 69-8 "Structural Dynamics Testing Facilities at the University of California, Berkeley," by R.M. Stephen, J.G. Bouwkamp, R.W. Clough and J. Penzien - 1969 (PB 189 111)A04

EERC 69-9 "Seismic Response of Soil Deposits Underlain by Sloping Rock Boundaries," by H. Dezfulian and H.B. Seed 1969 (PB 189 114)A03

EERC 69-10 "Dynamic Stress Analysis of Axisymmetric Structures Under Arbitrary Loading," by S. Ghosh and E.L. Wilson 1969 (PB 189 026)A10

EERC 69-11 "Seismic Behavior of Multistory Frames Designed by Different Philosophies," by J.C. Anderson and V. V. Bertero - 1969 (PB 190 662)A10

EERC 69-12 "Stiffness Degradation of Reinforcing Concrete Members Subjected to Cyclic Flexural Moments," by V.V. Bertero, B. Bresler and H. Ming Liao - 1969 (PB 202 942)A07

EERC 69-13 "Response of Non-Uniform Soil Deposits to Travelling Seismic Waves," by H. Dezfulian and H.B. Seed - 1969 (PB 191 023)A03

EERC 69-14 "Damping Capacity of a Model Steel Structure," by D. Rea, R.W. Clough and J.G. Bouwkamp - 1969 (PB 190 663)A06

EERC 69-15 "Influence of Local Soil Conditions on Building Damage Potential during Earthquakes," by H.B. Seed and I.M. Idriss - 1969 (PB 191 036)A03

EERC 69-16 "The Behavior of Sands Under Seismic Loading Conditions," by M.L. Silver and H.B. Seed - 1969 (AD 714 982)A07

EERC 70-1 "Earthquake Response of Gravity Dams," by A.K. Chopra - 1970 (AD 709 640)A03

EERC 70-2 "Relationships between Soil Conditions and Building Damage in the Caracas Earthquake of July 29, 1967," by H.B. Seed, I.M. Idriss and H. Dezfulian - 1970 (PB 195 762)A05

EERC 70-3 "Cyclic Loading of Full Size Steel Connections," by E.P. Popov and R.M. Stephen - 1970 (PB 213 545)A04

EERC 70-4 "Seismic Analysis of the Charaima Building, Caraballeda, Venezuela," by Subcommittee of the SEAONC Research Committee: V.V. Bertero, P.F. Fratessa, S.A. Mahin, J.H. Sexton, A.C. Scordelis, E.L. Wilson, L.A. Wyllie, H.B. Seed and J. Penzien, Chairman - 1970 (PB 201 455)A06

EERC 70-5 "A Computer Program for Earthquake Analysis of Dams," by A.K. Chopra and P. Chakrabarti - 1970 (AD 723 994)A05

EERC 70-6 "The Propagation of Love Waves Across Non-Horizontally Layered Structures," by J. Lysmer and L.A. Drake 1970 (PB 197 896)A03

EERC 70-7 "Influence of Base Rock Characteristics on Ground Response," by J. Lysmer, H.B. Seed and P.B. Schnabel 1970 (PB 197 897)A03

EERC 70-8 "Applicability of Laboratory Test Procedures for Measuring Soil Liquefaction Characteristics under Cyclic Loading," by H.B. Seed and W.H. Peacock - 1970 (PB 198 016)A03

EERC 70-9 "A Simplified Procedure for Evaluating Soil Liquefaction Potential," by H.B. Seed and I.M. Idriss - 1970 (PB 198 009)A03

EERC 70-10 "Soil Moduli and Damping Factors for Dynamic Response Analysis," by H.B. Seed and I.M. Idriss - 1970 (PB 197 869)A03

EERC 71-1 "Koyna Earthquake of December 11, 1967 and the Performance of Koyna Dam," by A.K. Chopra and P. Chakrabarti 1971 (AD 731 496)A06

EERC 71-2 "Preliminary In-Situ Measurements of Anelastic Absorption in Soils Using a Prototype Earthquake Simulator," by R.D. Borcherdt and P.W. Rodgers - 1971 (PB 201 454)A03

EERC 71-3 "Static and Dynamic Analysis of Inelastic Frame Structures," by F.L. Porter and G.H. Powell - 1971 (PB 210 135)A06

EERC 71-4 "Research Needs in Limit Design of Reinforced Concrete Structures," by V.V. Bertero - 1971 (PB 202 943)A04

EERC 71-5 "Dynamic Behavior of a High-Rise Diagonally Braced Steel Building," by D. Rea, A.A. Shah and J.G. Bouwkamp 1971 (PB 203 584)A06

EERC 71-6 "Dynamic Stress Analysis of Porous Elastic Solids Saturated with Compressible Fluids," by J. Ghaboussi and E. L. Wilson - 1971 (PB 211 396)A06

EERC 71-7 "Inelastic Behavior of Steel Beam-to-Column Subassemblages," by H. Krawinkler, V.V. Bertero and E.P. Popov 1971 (PB 211 335)A14

EERC 71-8 "Modification of Seismograph Records for Effects of Local Soil Conditions," by P. Schnabel, H.B. Seed and J. Lysmer - 1971 (PB 214 450)A03

EERC 72-1 "Static and Earthquake Analysis of Three Dimensional Frame and Shear Wall Buildings," by E.L. Wilson and H.H. Dovey - 1972 (PB 212 904)A05

EERC 72-2 "Accelerations in Rock for Earthquakes in the Western United States," by P.B. Schnabel and H.B. Seed - 1972 (PB 213 100)A03

EERC 72-3 "Elastic-Plastic Earthquake Response of Soil-Building Systems," by T. Minami - 1972 (PB 214 868)A08

EERC 72-4 "Stochastic Inelastic Response of Offshore Towers to Strong Motion Earthquakes," by M.K. Kaul - 1972 (PB 215 713)A05

EERC 72-5 "Cyclic Behavior of Three Reinforced Concrete Flexural Members with High Shear," by E.P. Popov, V.V. Bertero and H. Krawinkler - 1972 (PB 214 555)A05

EERC 72-6 "Earthquake Response of Gravity Dams Including Reservoir Interaction Effects," by P. Chakrabarti and A.K. Chopra - 1972 (AD 762 330)A08

EERC 72-7 "Dynamic Properties of Pine Flat Dam," by D. Rea, C.Y. Liaw and A.K. Chopra - 1972 (AD 763 928)A05

EERC 72-8 "Three Dimensional Analysis of Building Systems," by E.L. Wilson and H.H. Dovey - 1972 (PB 222 438)A06

EERC 72-9 "Rate of Loading Effects on Uncracked and Repaired Reinforced Concrete Members," by S. Mahin, V.V. Bertero, D. Rea and M. Atalay - 1972 (PB 224 520)A08

EERC 72-10 "Computer Program for Static and Dynamic Analysis of Linear Structural Systems," by E.L. Wilson, K.-J. Bathe, J.E. Peterson and H.H.Dovey - 1972 (PB 220 437)A04

EERC 72-11 "Literature Survey - Seismic Effects on Highway Bridges," by T. Iwasaki, J. Penzien and R.W. Clough - 1972 (PB 215 613)A19

EERC 72-12 "SHAKE-A Computer Program for Earthquake Response Analysis of Horizontally Layered Sites," by P.B. Schnabel and J. Lysmer - 1972 (PB 220 207)A06

EERC 73-1 "Optimal Seismic Design of Multistory Frames," by V.V. Bertero and H. Kamil - 1973

EERC 73-2 "Analysis of the Slides in the San Fernando Dams During the Earthquake of February 9, 1971," by H.B. Seed, K.L. Lee, I.M. Idriss and F. Makdisi - 1973 (PB 223 402)A14

EERC 73-3 "Computer Aided Ultimate Load Design of Unbraced Multistory Steel Frames," by M.B. El-Hafez and G.H. Powell 1973 (PB 248 315)A09

EERC 73-4 "Experimental Investigation into the Seismic Behavior of Critical Regions of Reinforced Concrete Components as Influenced by Moment and Shear," by M. Celebi and J. Penzien - 1973 (PB 215 884)A09

EERC 73-5 "Hysteretic Behavior of Epoxy-Repaired Reinforced Concrete Beams," by M. Celebi and J. Penzien - 1973 (PB 239 568)A03

EERC 73-6 "General Purpose Computer Program for Inelastic Dynamic Response of Plane Structures," by A. Kanaan and G.H. Powell - 1973 (PB 221 260)A08

EERC 73-7 "A Computer Program for Earthquake Analysis of Gravity Dams Including Reservoir Interaction," by P. Chakrabarti and A.K. Chopra - 1973 (AD 766 271)A04

EERC 73-8 "Behavior of Reinforced Concrete Deep Beam-Column Subassemblages Under Cyclic Loads," by O. Küstü and J.G. Bouwkamp - 1973 (PB 246 117)A12

EERC 73-9 "Earthquake Analysis of Structure-Foundation Systems," by A.K. Vaish and A.K. Chopra - 1973 (AD 766 272)A07

EERC 73-10 "Deconvolution of Seismic Response for Linear Systems," by R.B. Reimer - 1973 (PB 227 179)A08

EERC 73-11 "SAP IV: A Structural Analysis Program for Static and Dynamic Response of Linear Systems," by K.-J. Bathe, E.L. Wilson and F.E. Peterson - 1973 (PB 221 967)A09

EERC 73-12 "Analytical Investigations of the Seismic Response of Long, Multiple Span Highway Bridges," by W.S. Tseng and J. Penzien - 1973 (PB 227 816)A10

EERC 73-13 "Earthquake Analysis of Multi-Story Buildings Including Foundation Interaction," by A.K. Chopra and J.A. Gutierrez - 1973 (PB 222 970)A03

EERC 73-14 "ADAP: A Computer Program for Static and Dynamic Analysis of Arch Dams," by R.W. Clough, J.M. Raphael and S. Mojtahedi - 1973 (PB 223 763)A09

EERC 73-15 "Cyclic Plastic Analysis of Structural Steel Joints," by R.B. Pinkney and R.W. Clough - 1973 (PB 226 843)A08

EERC 73-16 "QUAD-4: A Computer Program for Evaluating the Seismic Response of Soil Structures by Variable Damping Finite Element Procedures," by I.M. Idriss, J. Lysmer, R. Hwang and H.B. Seed - 1973 (PB 229 424)A05

EERC 73-17 "Dynamic Behavior of a Multi-Story Pyramid Shaped Building," by R.M. Stephen, J.P. Hollings and J.G. Bouwkamp - 1973 (PB 240 718)A06

EERC 73-18 "Effect of Different Types of Reinforcing on Seismic Behavior of Short Concrete Columns," by V.V. Bertero, J. Hollings, O. Küstü, R.M. Stephen and J.G. Bouwkamp - 1973

EERC 73-19 "Olive View Medical Center Materials Studies, Phase I," by B. Bresler and V.V. Bertero - 1973 (PB 235 986)A06

EERC 73-20 "Linear and Nonlinear Seismic Analysis Computer Programs for Long Multiple-Span Highway Bridges," by W.S. Tseng and J. Penzien - 1973

EERC 73-21 "Constitutive Models for Cyclic Plastic Deformation of Engineering Materials," by J.M. Kelly and P.P. Gillis 1973 (PB 226 024)A03

EERC 73-22 "DRAIN - 2D User's Guide," by G.H. Powell - 1973 (PB 227 016)A05

EERC 73-23 "Earthquake Engineering at Berkeley - 1973," (PB 226 033)A11

EERC 73-24 Unassigned

EERC 73-25 "Earthquake Response of Axisymmetric Tower Structures Surrounded by Water," by C.Y. Liaw and A.K. Chopra 1973 (AD 773 052)A09

EERC 73-26 "Investigation of the Failures of the Olive View Stairtowers During the San Fernando Earthquake and Their Implications on Seismic Design," by V.V. Bertero and R.G. Collins - 1973 (PB 235 106)A13

EERC 73-27 "Further Studies on Seismic Behavior of Steel Beam-Column Subassemblages," by V.V. Bertero, H. Krawinkler and E.P. Popov - 1973 (PB 234 172)A06

EERC 74-1 "Seismic Risk Analysis," by C.S. Oliveira - 1974 (PB 235 920)A06

EERC 74-2 "Settlement and Liquefaction of Sands Under Multi-Directional Shaking," by R. Pyke, C.K. Chan and H.B. Seed 1974

EERC 74-3 "Optimum Design of Earthquake Resistant Shear Buildings," by D. Ray, K.S. Pister and A.K. Chopra - 1974 (PB 231 172)A06

EERC 74-4 "LUSH - A Computer Program for Complex Response Analysis of Soil-Structure Systems," by J. Lysmer, T. Udaka, H.B. Seed and R. Hwang - 1974 (PB 236 796)A05

EERC 74-5 "Sensitivity Analysis for Hysteretic Dynamic Systems: Applications to Earthquake Engineering," by D. Ray 1974 (PB 233 213)A06

EERC 74-6 "Soil Structure Interaction Analyses for Evaluating Seismic Response," by H.B. Seed, J. Lysmer and R. Hwang 1974 (PB 236 519)A04

EERC 74-7 Unassigned

EERC 74-8 "Shaking Table Tests of a Steel Frame - A Progress Report," by R.W. Clough and D. Tang - 1974 (PB 240 869)A03

EERC 74-9 "Hysteretic Behavior of Reinforced Concrete Flexural Members with Special Web Reinforcement," by V.V. Bertero, E.P. Popov and T.Y. Wang - 1974 (PB 236 797)A07

EERC 74-10 "Applications of Reliability-Based, Global Cost Optimization to Design of Earthquake Resistant Structures," by E. Vitiello and K.S. Pister - 1974 (PB 237 231)A06

EERC 74-11 "Liquefaction of Gravelly Soils Under Cyclic Loading Conditions," by R.T. Wong, H.B. Seed and C.K. Chan 1974 (PB 242 042)A03

EERC 74-12 "Site-Dependent Spectra for Earthquake-Resistant Design," by H.B. Seed, C. Ugas and J. Lysmer - 1974 (PB 240 953)A03

EERC 74-13 "Earthquake Simulator Study of a Reinforced Concrete Frame," by P. Hidalgo and R.W. Clough - 1974 (PB 241 944)A13

EERC 74-14 "Nonlinear Earthquake Response of Concrete Gravity Dams," by N. Pal - 1974 (AD/A 006 583)A06

EERC 74-15 "Modeling and Identification in Nonlinear Structural Dynamics - I. One Degree of Freedom Models," by N. Distefano and A. Rath - 1974 (PB 241 548)A06

EERC 75-1 "Determination of Seismic Design Criteria for the Dumbarton Bridge Replacement Structure, Vol. I: Description, Theory and Analytical Modeling of Bridge and Parameters," by F. Baron and S.-H. Pang - 1975 (PB 259 407)A15

EERC 75-2 "Determination of Seismic Design Criteria for the Dumbarton Bridge Replacement Structure, Vol. II: Numerical Studies and Establishment of Seismic Design Criteria," by F. Baron and S.-H. Pang - 1975 (PB 259 408)A11 (For set of EERC 75-1 and 75-2 (PB 259 406))

EERC 75-3 "Seismic Risk Analysis for a Site and a Metropolitan Area," by C.S. Oliveira - 1975 (PB 248 134)A09

EERC 75-4 "Analytical Investigations of Seismic Response of Short, Single or Multiple-Span Highway Bridges," by M.-C. Chen and J. Penzien - 1975 (PB 241 454)A09

EERC 75-5 "An Evaluation of Some Methods for Predicting Seismic Behavior of Reinforced Concrete Buildings," by S.A. Mahin and V.V. Bertero - 1975 (PB 246 306)A16

EERC 75-6 "Earthquake Simulator Study of a Steel Frame Structure, Vol. I: Experimental Results," by R.W. Clough and D.T. Tang - 1975 (PB 243 981)A13

EERC 75-7 "Dynamic Properties of San Bernardino Intake Tower," by D. Rea, C.-Y. Liaw and A.K. Chopra - 1975 (AD/A008 406) A05

EERC 75-8 "Seismic Studies of the Articulation for the Dumbarton Bridge Replacement Structure, Vol. I: Description, Theory and Analytical Modeling of Bridge Components," by F. Baron and R.E. Hamati - 1975 (PB 251 539)A07

EERC 75-9 "Seismic Studies of the Articulation for the Dumbarton Bridge Replacement Structure, Vol. 2: Numerical Studies of Steel and Concrete Girder Alternates," by F. Baron and R.E. Hamati - 1975 (PB 251 540)A10

EERC 75-10 "Static and Dynamic Analysis of Nonlinear Structures," by D.P. Mondkar and G.H. Powell - 1975 (PB 242 434)A08

EERC 75-11 "Hysteretic Behavior of Steel Columns," by E.P. Popov, V.V. Bertero and S. Chandramouli - 1975 (PB 252 365)A11

EERC 75-12 "Earthquake Engineering Research Center Library Printed Catalog," - 1975 (PB 243 711)A26

EERC 75-13 "Three Dimensional Analysis of Building Systems (Extended Version)," by E.L. Wilson, J.P. Hollings and H.H. Dovey - 1975 (PB 243 989)A07

EERC 75-14 "Determination of Soil Liquefaction Characteristics by Large-Scale Laboratory Tests," by P. De Alba, C.K. Chan and H.B. Seed - 1975 (NUREG 0027)A08

EERC 75-15 "A Literature Survey - Compressive, Tensile, Bond and Shear Strength of Masonry," by R.L. Mayes and R.W. Clough - 1975 (PB 246 292)A10

EERC 75-16 "Hysteretic Behavior of Ductile Moment Resisting Reinforced Concrete Frame Components," by V.V. Bertero and E.P. Popov - 1975 (PB 246 388)A05

EERC 75-17 "Relationships Between Maximum Acceleration, Maximum Velocity, Distance from Source, Local Site Conditions for Moderately Strong Earthquakes," by H.B. Seed, R. Murarka, J. Lysmer and I.M. Idriss - 1975 (PB 248 172)A03

EERC 75-18 "The Effects of Method of Sample Preparation on the Cyclic Stress-Strain Behavior of Sands," by J. Mulilis, C.K. Chan and H.B. Seed - 1975 (Summarized in EERC 75-28)

EERC 75-19 "The Seismic Behavior of Critical Regions of Reinforced Concrete Components as Influenced by Moment, Shear and Axial Force," by M.B. Atalay and J. Penzien - 1975 (PB 258 842)A11

EERC 75-20 "Dynamic Properties of an Eleven Story Masonry Building," by R.M. Stephen, J.P. Hollings, J.G. Bouwkamp and D. Jurukovski - 1975 (PB 246 945)A04

EERC 75-21 "State-of-the-Art in Seismic Strength of Masonry - An Evaluation and Review," by R.L. Mayes and R.W. Clough 1975 (PB 249 040)A07

EERC 75-22 "Frequency Dependent Stiffness Matrices for Viscoelastic Half-Plane Foundations," by A.K. Chopra, P. Chakrabarti and G. Dasgupta - 1975 (PB 248 121)A07

EERC 75-23 "Hysteretic Behavior of Reinforced Concrete Framed Walls," by T.Y. Wong, V.V. Bertero and E.P. Popov - 1975

EERC 75-24 "Testing Facility for Subassemblages of Frame-Wall Structural Systems," by V.V. Bertero, E.P. Popov and T. Endo - 1975

EERC 75-25 "Influence of Seismic History on the Liquefaction Characteristics of Sands," by H.B. Seed, K. Mori and C.K. Chan - 1975 (Summarized in EERC 75-28)

EERC 75-26 "The Generation and Dissipation of Pore Water Pressures during Soil Liquefaction," by H.B. Seed, P.P. Martin and J. Lysmer - 1975 (PB 252 648)A03

EERC 75-27 "Identification of Research Needs for Improving Aseismic Design of Building Structures," by V.V. Bertero 1975 (PB 248 136)A05

EERC 75-28 "Evaluation of Soil Liquefaction Potential during Earthquakes," by H.B. Seed, I. Arango and C.K. Chan - 1975 (NUREG 0026)A13

EERC 75-29 "Representation of Irregular Stress Time Histories by Equivalent Uniform Stress Series in Liquefaction Analyses," by H.B. Seed, I.M. Idriss, F. Makdisi and N. Banerjee - 1975 (PB 252 635)A03

EERC 75-30 "FLUSH - A Computer Program for Approximate 3-D Analysis of Soil-Structure Interaction Problems," by J. Lysmer, T. Udaka, C.-F. Tsai and H.B. Seed - 1975 (PB 259 332)A07

EERC 75-31 "ALUSH - A Computer Program for Seismic Response Analysis of Axisymmetric Soil-Structure Systems," by E. Berger, J. Lysmer and H.B. Seed - 1975

EERC 75-32 "TRIP and TRAVEL - Computer Programs for Soil-Structure Interaction Analysis with Horizontally Travelling Waves," by T. Udaka, J. Lysmer and H.B. Seed - 1975

EERC 75-33 "Predicting the Performance of Structures in Regions of High Seismicity," by J. Penzien - 1975 (PB 248 130)A03

EERC 75-34 "Efficient Finite Element Analysis of Seismic Structure - Soil - Direction," by J. Lysmer, H.B. Seed, T. Udaka, R.N. Hwang and C.-F. Tsai - 1975 (PB 253 570)A03

EERC 75-35 "The Dynamic Behavior of a First Story Girder of a Three-Story Steel Frame Subjected to Earthquake Loading," by R.W. Clough and L.-Y. Li - 1975 (PB 248 841)A05

EERC 75-36 "Earthquake Simulator Study of a Steel Frame Structure, Volume II - Analytical Results," by D.T. Tang - 1975 (PB 252 926)A10

EERC 75-37 "ANSR-I General Purpose Computer Program for Analysis of Non-Linear Structural Response," by D.P. Mondkar and G.H. Powell - 1975 (PB 252 386)A08

EERC 75-38 "Nonlinear Response Spectra for Probabilistic Seismic Design and Damage Assessment of Reinforced Concrete Structures," by M. Murakami and J. Penzien - 1975 (PB 259 530)A05

EERC 75-39 "Study of a Method of Feasible Directions for Optimal Elastic Design of Frame Structures Subjected to Earthquake Loading," by N.D. Walker and K.S. Pister - 1975 (PB 257 781)A06

EERC 75-40 "An Alternative Representation of the Elastic-Viscoelastic Analogy," by G. Dasgupta and J.L. Sackman - 1975 (PB 252 173)A03

EERC 75-41 "Effect of Multi-Directional Shaking on Liquefaction of Sands," by H.B. Seed, R. Pyke and G.R. Martin - 1975 (PB 258 781)A03

EERC 76-1 "Strength and Ductility Evaluation of Existing Low-Rise Reinforced Concrete Buildings - Screening Method," by T. Okada and B. Bresler - 1976 (PB 257 906)A11

EERC 76-2 "Experimental and Analytical Studies on the Hysteretic Behavior of Reinforced Concrete Rectangular and T-Beams," by S.-Y.M. Ma, E.P. Popov and V.V. Bertero - 1976 (PB 260 843)A12

EERC 76-3 "Dynamic Behavior of a Multistory Triangular-Shaped Building," by J. Petrovski, R.M. Stephen, E. Gartenbaum and J.G. Bouwkamp - 1976 (PB 273 279)A07

EERC 76-4 "Earthquake Induced Deformations of Earth Dams," by N. Serff, H.B. Seed, F.I. Makdisi & C.-Y. Chang - 1976 (PB 292 065)A08

EERC 76-5 "Analysis and Design of Tube-Type Tall Building Structures," by H. de Clercq and G.H. Powell - 1976 (PB 252 220) A10

EERC 76-6 "Time and Frequency Domain Analysis of Three-Dimensional Ground Motions, San Fernando Earthquake," by T. Kubo and J. Penzien (PB 260 556)A11

EERC 76-7 "Expected Performance of Uniform Building Code Design Masonry Structures," by R.L. Mayes, Y. Omote, S.W. Chen and R.W. Clough - 1976 (PB 270 098)A05

EERC 76-8 "Cyclic Shear Tests of Masonry Piers, Volume 1 - Test Results," by R.L. Mayes, Y. Omote, R.W. Clough - 1976 (PB 264 424)A06

EERC 76-9 "A Substructure Method for Earthquake Analysis of Structure - Soil Interaction," by J.A. Gutierrez and A.K. Chopra - 1976 (PB 257 783)A08

EERC 76-10 "Stabilization of Potentially Liquefiable Sand Deposits using Gravel Drain Systems," by H.B. Seed and J.R. Booker - 1976 (PB 258 820)A04

EERC 76-11 "Influence of Design and Analysis Assumptions on Computed Inelastic Response of Moderately Tall Frames," by G.H. Powell and D.G. Row - 1976 (PB 271 409)A06

EERC 76-12 "Sensitivity Analysis for Hysteretic Dynamic Systems: Theory and Applications," by D. Ray, K.S. Pister and E. Polak - 1976 (PB 262 859)A04

EERC 76-13 "Coupled Lateral Torsional Response of Buildings to Ground Shaking," by C.L. Kan and A.K. Chopra - 1976 (PB 257 907)A09

EERC 76-14 "Seismic Analyses of the Banco de America," by V.V. Bertero, S.A. Mahin and J.A. Hollings - 1976

EERC 76-15 "Reinforced Concrete Frame 2: Seismic Testing and Analytical Correlation," by R.W. Clough and J. Gidwani - 1976 (PB 261 323)A08

EERC 76-16 "Cyclic Shear Tests of Masonry Piers, Volume 2 - Analysis of Test Results," by R.L. Mayes, Y. Omote and R.W. Clough - 1976

EERC 76-17 "Structural Steel Bracing Systems: Behavior Under Cyclic Loading," by E.P. Popov, K. Takanashi and C.W. Roeder - 1976 (PB 260 715)A05

EERC 76-18 "Experimental Model Studies on Seismic Response of High Curved Overcrossings," by D. Williams and W.G. Godden - 1976 (PB 269 548)A08

EERC 76-19 "Effects of Non-Uniform Seismic Disturbances on the Dumbarton Bridge Replacement Structure," by F. Baron and R.E. Hamati - 1976 (PB 282 981)A16

EERC 76-20 "Investigation of the Inelastic Characteristics of a Single Story Steel Structure Using System Identification and Shaking Table Experiments," by V.C. Matzen and H.D. McNiven - 1976 (PB 258 453)A07

EERC 76-21 "Capacity of Columns with Splice Imperfections," by E.P. Popov, R.M. Stephen and R. Philbrick - 1976 (PB 260 378)A04

EERC 76-22 "Response of the Olive View Hospital Main Building during the San Fernando Earthquake," by S. A. Mahin, V.V. Bertero, A.K. Chopra and R. Collins - 1976 (PB 271 425)A14

EERC 76-23 "A Study on the Major Factors Influencing the Strength of Masonry Prisms," by N.M. Mostaghel, R.L. Mayes, R. W. Clough and S.W. Chen - 1976 (Not published)

EERC 76-24 "GADFLEA - A Computer Program for the Analysis of Pore Pressure Generation and Dissipation during Cyclic or Earthquake Loading," by J.R. Booker, M.S. Rahman and H.B. Seed - 1976 (PB 263 947)A04

EERC 76-25 "Seismic Safety Evaluation of a R/C School Building," by B. Bresler and J. Axley - 1976

EERC 76-26 "Correlative Investigations on Theoretical and Experimental Dynamic Behavior of a Model Bridge Structure," by K. Kawashima and J. Penzien - 1976 (PB 263 388)A11

EERC 76-27 "Earthquake Response of Coupled Shear Wall Buildings," by T. Srichatrapimuk - 1976 (PB 265 157)A07

EERC 76-28 "Tensile Capacity of Partial Penetration Welds," by E.P. Popov and R.M. Stephen - 1976 (PB 262 899)A03

EERC 76-29 "Analysis and Design of Numerical Integration Methods in Structural Dynamics," by H.M. Hilber - 1976 (PB 264 410)A06

EERC 76-30 "Contribution of a Floor System to the Dynamic Characteristics of Reinforced Concrete Buildings," by L.E. Malik and V.V. Bertero - 1976 (PB 272 247)A13

EERC 76-31 "The Effects of Seismic Disturbances on the Golden Gate Bridge," by F. Baron, M. Arikan and R.E. Hamati - 1976 (PB 272 279)A09

EERC 76-32 "Infilled Frames in Earthquake Resistant Construction," by R.E. Klingner and V.V. Bertero - 1976 (PB 265 892)A13

UCB/EERC-77/01 "PLUSH - A Computer Program for Probabilistic Finite Element Analysis of Seismic Soil-Structure Interaction," by M.P. Romo Organista, J. Lysmer and H.B. Seed - 1977

UCB/EERC-77/02 "Soil-Structure Interaction Effects at the Humboldt Bay Power Plant in the Ferndale Earthquake of June 7, 1975," by J.E. Valera, H.B. Seed, C.F. Tsai and J. Lysmer - 1977 (PB 265 795)A04

UCB/EERC-77/03 "Influence of Sample Disturbance on Sand Response to Cyclic Loading," by K. Mori, H.B. Seed and C.K. Chan - 1977 (PB 267 352)A04

UCB/EERC-77/04 "Seismological Studies of Strong Motion Records," by J. Shoja-Taheri - 1977 (PB 269 655)A10

UCB/EERC-77/05 "Testing Facility for Coupled-Shear Walls," by L. Li-Hyung, V.V. Bertero and E.P. Popov - 1977

UCB/EERC-77/06 "Developing Methodologies for Evaluating the Earthquake Safety of Existing Buildings," by No. 1 - B. Bresler; No. 2 - B. Bresler, T. Okada and D. Zisling; No. 3 - T. Okada and B. Bresler; No. 4 - V.V. Bertero and B. Bresler - 1977 (PB 267 354)A08

UCB/EERC-77/07 "A Literature Survey - Transverse Strength of Masonry Walls," by Y. Omote, R.L. Mayes, S.W. Chen and R.W. Clough - 1977 (PB 277 933)A07

UCB/EERC-77/08 "DRAIN-TABS: A Computer Program for Inelastic Earthquake Response of Three Dimensional Buildings," by R. Guendelman-Israel and G.H. Powell - 1977 (PB 270 693)A07

UCB/EERC-77/09 "SUBWALL: A Special Purpose Finite Element Computer Program for Practical Elastic Analysis and Design of Structural Walls with Substructure Option," by D.Q. Le, H. Peterson and E.P. Popov - 1977 (PB 270 567)A05

UCB/EERC-77/10 "Experimental Evaluation of Seismic Design Methods for Broad Cylindrical Tanks," by D.P. Clough (PB 272 280)A13

UCB/EERC-77/11 "Earthquake Engineering Research at Berkeley - 1976," - 1977 (PB 273 507)A09

UCB/EERC-77/12 "Automated Design of Earthquake Resistant Multistory Steel Building Frames," by N.D. Walker, Jr. - 1977 (PB 276 526)A09

UCB/EERC-77/13 "Concrete Confined by Rectangular Hoops Subjected to Axial Loads," by J. Vallenas, V.V. Bertero and E.P. Popov - 1977 (PB 275 165)A06

UCB/EERC-77/14 "Seismic Strain Induced in the Ground During Earthquakes," by Y. Sugimura - 1977 (PB 284 201)A04

UCB/EERC-77/15 "Bond Deterioration under Generalized Loading," by V.V. Bertero, E.P. Popov and S. Viwathanatepa - 1977

UCB/EERC-77/16 "Computer Aided Optimum Design of Ductile Reinforced Concrete Moment Resisting Frames," by S.W. Zagajeski and V.V. Bertero - 1977 (PB 280 137)A07

UCB/EERC-77/17 "Earthquake Simulation Testing of a Stepping Frame with Energy-Absorbing Devices," by J.M. Kelly and D.F. Tsztoo - 1977 (PB 273 506)A04

UCB/EERC-77/18 "Inelastic Behavior of Eccentrically Braced Steel Frames under Cyclic Loadings," by C.W. Roeder and E.P. Popov - 1977 (PB 275 526)A15

UCB/EERC-77/19 "A Simplified Procedure for Estimating Earthquake-Induced Deformations in Dams and Embankments," by F.I. Makdisi and H.B. Seed - 1977 (PB 276 820)A04

UCB/EERC-77/20 "The Performance of Earth Dams during Earthquakes," by H.B. Seed, F.I. Makdisi and P. de Alba - 1977 (PB 276 821)A04

UCB/EERC-77/21 "Dynamic Plastic Analysis Using Stress Resultant Finite Element Formulation," by P. Lukkunapvasit and J.M. Kelly - 1977 (PB 275 453)A04

UCB/EERC-77/22 "Preliminary Experimental Study of Seismic Uplift of a Steel Frame," by R.W. Clough and A.A. Huckelbridge 1977 (PB 278 769)A08

UCB/EERC-77/23 "Earthquake Simulator Tests of a Nine-Story Steel Frame with Columns Allowed to Uplift," by A.A. Huckelbridge - 1977 (PB 277 944)A09

UCB/EERC-77/24 "Nonlinear Soil-Structure Interaction of Skew Highway Bridges," by M.-C. Chen and J. Penzien - 1977 (PB 276 176)A07

UCB/EERC-77/25 "Seismic Analysis of an Offshore Structure Supported on Pile Foundations," by D.D.-N. Liou and J. Penzien 1977 (PB 283 180)A06

UCB/EERC-77/26 "Dynamic Stiffness Matrices for Homogeneous Viscoelastic Half-Planes," by G. Dasgupta and A.K. Chopra - 1977 (PB 279 654)A06

UCB/EERC-77/27 "A Practical Soft Story Earthquake Isolation System," by J.M. Kelly, J.M. Eidinger and C.J. Derham - 1977 (PB 276 814)A07

UCB/EERC-77/28 "Seismic Safety of Existing Buildings and Incentives for Hazard Mitigation in San Francisco: An Exploratory Study," by A.J. Meltsner - 1977 (PB 281 970)A05

UCB/EERC-77/29 "Dynamic Analysis of Electrohydraulic Shaking Tables," by D. Rea, S. Abedi-Hayati and Y. Takahashi 1977 (PB 282 569)A04

UCB/EERC-77/30 "An Approach for Improving Seismic - Resistant Behavior of Reinforced Concrete Interior Joints," by B. Galunic, V.V. Bertero and E.P. Popov - 1977 (PB 290 870)A06

UCB/EERC-78/01 "The Development of Energy-Absorbing Devices for Aseismic Base Isolation Systems," by J.M. Kelly and D.F. Tsztoo - 1978 (PB 284 978)A04

UCB/EERC-78/02 "Effect of Tensile Prestrain on the Cyclic Response of Structural Steel Connections, by J.G. Bouwkamp and A. Mukhopadhyay - 1978

UCB/EERC-78/03 "Experimental Results of an Earthquake Isolation System using Natural Rubber Bearings," by J.M. Eidinger and J.M. Kelly - 1978 (PB 281 686)A04

UCB/EERC-78/04 "Seismic Behavior of Tall Liquid Storage Tanks," by A. Niwa - 1978 (PB 284 017)A14

UCB/EERC-78/05 "Hysteretic Behavior of Reinforced Concrete Columns Subjected to High Axial and Cyclic Shear Forces," by S.W. Zagajeski, V.V. Bertero and J.G. Bouwkamp - 1978 (PB 283 858)A13

UCB/EERC-78/06 "Inelastic Beam-Column Elements for the ANSR-1 Program," by A. Riahi, D.G. Row and G.H. Powell - 1978

UCB/EERC-78/07 "Studies of Structural Response to Earthquake Ground Motion," by O.A. Lopez and A.K. Chopra - 1978 (PB 282 790)A05

UCB/EERC-78/08 "A Laboratory Study of the Fluid-Structure Interaction of Submerged Tanks and Caissons in Earthquakes," by R.C. Byrd - 1978 (PB 284 957)A08

UCB/EERC-78/09 "Model for Evaluating Damageability of Structures," by I. Sakamoto and B. Bresler - 1978

UCB/EERC-78/10 "Seismic Performance of Nonstructural and Secondary Structural Elements," by I. Sakamoto - 1978

UCB/EERC-78/11 "Mathematical Modelling of Hysteresis Loops for Reinforced Concrete Columns," by S. Nakata, T. Sproul and J. Penzien - 1978

UCB/EERC-78/12 "Damageability in Existing Buildings," by T. Blejwas and B. Bresler - 1978

UCB/EERC-78/13 "Dynamic Behavior of a Pedestal Base Multistory Building," by R.M. Stephen, E.L. Wilson, J.G. Bouwkamp and M. Button - 1978 (PB 286 650)A08

UCB/EERC-78/14 "Seismic Response of Bridges - Case Studies," by R.A. Imbsen, V. Nutt and J. Penzien - 1978 (PB 286 503)A10

UCB/EERC-78/15 "A Substructure Technique for Nonlinear Static and Dynamic Analysis," by D.G. Row and G.H. Powell - 1978 (PB 288 077)A10

UCB/EERC-78/16 "Seismic Risk Studies for San Francisco and for the Greater San Francisco Bay Area," by C.S. Oliveira - 1978

UCB/EERC-78/17 "Strength of Timber Roof Connections Subjected to Cyclic Loads," by P. Gülkan, R.L. Mayes and R.W. Clough - 1978

UCB/EERC-78/18 "Response of K-Braced Steel Frame Models to Lateral Loads," by J.G. Bouwkamp, R.M. Stephen and E.P. Popov - 1978

UCB/EERC-78/19 "Rational Design Methods for Light Equipment in Structures Subjected to Ground Motion," by J.L. Sackman and J.M. Kelly - 1978 (PB 292 357)A04

UCB/EERC-78/20 "Testing of a Wind Restraint for Aseismic Base Isolation," by J.M. Kelly and D.E. Chitty - 1978 (PB 292 833)A03

UCB/EERC-78/21 "APOLLO - A Computer Program for the Analysis of Pore Pressure Generation and Dissipation in Horizontal Sand Layers During Cyclic or Earthquake Loading," by P.P. Martin and H.B. Seed - 1978 (PB 292 835)A04

UCB/EERC-78/22 "Optimal Design of an Earthquake Isolation System," by M.A. Bhatti, K.S. Pister and E. Polak - 1978 (PB 294 735)A06

UCB/EERC-78/23 "MASH - A Computer Program for the Non-Linear Analysis of Vertically Propagating Shear Waves in Horizontally Layered Deposits," by P.P. Martin and H.B. Seed - 1978 (PB 293 101)A05

UCB/EERC-78/24 "Investigation of the Elastic Characteristics of a Three Story Steel Frame Using System Identification," by I. Kaya and H.D. McNiven - 1978

UCB/EERC-78/25 "Investigation of the Nonlinear Characteristics of a Three-Story Steel Frame Using System Identification," by I. Kaya and H.D. McNiven - 1978

UCB/EERC-78/26 "Studies of Strong Ground Motion in Taiwan," by Y.M. Hsiung, B.A. Bolt and J. Penzien - 1978

UCB/EERC-78/27 "Cyclic Loading Tests of Masonry Single Piers: Volume 1 - Height to Width Ratio of 2," by P.A. Hidalgo, R.L. Mayes, H.D. McNiven and R.W. Clough - 1978

UCB/EERC-78/28 "Cyclic Loading Tests of Masonry Single Piers: Volume 2 - Height to Width Ratio of 1," by S.-W.J. Chen, P.A. Hidalgo, R.L. Mayes, R.W. Clough and H.D. McNiven - 1978

UCB/EERC-78/29 "Analytical Procedures in Soil Dynamics," by J. Lysmer - 1978

UCB/EERC-79/01 "Hysteretic Behavior of Lightweight Reinforced Concrete Beam-Column Subassemblages," by B. Forzani, E.P. Popov and V.V. Bertero - April 1979(PB 298 267)A06

UCB/EERC-79/02 "The Development of a Mathematical Model to Predict the Flexural Response of Reinforced Concrete Beams to Cyclic Loads, Using System Identification," by J. Stanton & H. McNiven - Jan. 1979(PB 295 875)A10

UCB/EERC-79/03 "Linear and Nonlinear Earthquake Response of Simple Torsionally Coupled Systems," by C.L. Kan and A.K. Chopra - Feb. 1979(PB 298 262)A06

UCB/EERC-79/04 "A Mathematical Model of Masonry for Predicting its Linear Seismic Response Characteristics," by Y. Mengi and H.D. McNiven - Feb. 1979(PB 298 266)A06

UCB/EERC-79/05 "Mechanical Behavior of Lightweight Concrete Confined by Different Types of Lateral Reinforcement," by M.A. Manrique, V.V. Bertero and E.P. Popov - May 1979(PB 301 114)A06

UCB/EERC-79/06 "Static Tilt Tests of a Tall Cylindrical Liquid Storage Tank," by R.W. Clough and A. Niwa - Feb. 1979 (PB 301 167)A06

UCB/EERC-79/07 "The Design of Steel Energy Absorbing Restrainers and Their Incorporation into Nuclear Power Plants for Enhanced Safety: Volume 1 - Summary Report," by P.N. Spencer, V.F. Zackay, and E.R. Parker - Feb. 1979(UCB/EERC-79/07)A09

UCB/EERC-79/08 "The Design of Steel Energy Absorbing Restrainers and Their Incorporation into Nuclear Power Plants for Enhanced Safety: Volume 2 - The Development of Analyses for Reactor System Piping, "Simple Systems" by M.C. Lee, J. Penzien, A.K. Chopra and K. Suzuki "Complex Systems" by G.H. Powell, E.L. Wilson, R.W. Clough and D.G. Row - Feb. 1979(UCB/EERC-79/08)A10

UCB/EERC-79/09 "The Design of Steel Energy Absorbing Restrainers and Their Incorporation into Nuclear Power Plants for Enhanced Safety: Volume 3 - Evaluation of Commercial Steels," by W.S. Owen, R.M.N. Pelloux, R.O. Ritchie, M. Faral, T. Ohhashi, J. Toplosky, S.J. Hartman, V.F. Zackay and E.R. Parker - Feb. 1979(UCB/EERC-79/09)A04

UCB/EERC-79/10 "The Design of Steel Energy Absorbing Restrainers and Their Incorporation into Nuclear Power Plants for Enhanced Safety: Volume 4 - A Review of Energy-Absorbing Devices," by J.M. Kelly and M.S. Skinner - Feb. 1979(UCB/EERC-79/10)A04

UCB/EERC-79/11 "Conservatism In Summation Rules for Closely Spaced Modes," by J.M. Kelly and J.L. Sackman - May 1979(PB 301 328)A03

UCB/EERC-79/12 "Cyclic Loading Tests of Masonry Single Piers; Volume 3 - Height to Width Ratio of 0.5," by P.A. Hidalgo, R.L. Mayes, H.D. McNiven and R.W. Clough - May 1979(PB 301 321)A08

UCB/EERC-79/13 "Cyclic Behavior of Dense Course-Grained Materials in Relation to the Seismic Stability of Dams," by N.G. Banerjee, H.B. Seed and C.K. Chan - June 1979(PB 301 373)A13

UCB/EERC-79/14 "Seismic Behavior of Reinforced Concrete Interior Beam-Column Subassemblages," by S. Viwathanatepa, E.P. Popov and V.V. Bertero - June 1979(PB 301 326)A10

UCB/EERC-79/15 "Optimal Design of Localized Nonlinear Systems with Dual Performance Criteria Under Earthquake Excitations," by M.A. Bhatti - July 1979(PB 80 167 109)A06

UCB/EERC-79/16 "OPTDYN - A General Purpose Optimization Program for Problems with or without Dynamic Constraints," by M.A. Bhatti, E. Polak and K.S. Pister - July 1979(PB 80 167 091)A05

UCB/EERC-79/17 "ANSR-II, Analysis of Nonlinear Structural Response, Users Manual," by D.P. Mondkar and G.H. Powell - July 1979(PB 80 113 301)A05

UCB/EERC-79/18 "Soil Structure Interaction in Different Seismic Environments," A. Gomez-Masso, J. Lysmer, J.-C. Chen and H.B. Seed - August 1979(PB 80 101 520)A04

UCB/EERC-79/19 "ARMA Models for Earthquake Ground Motions," by M.K. Chang, J.W. Kwiatkowski, R.F. Nau, R.M. Oliver and K.S. Pister - July 1979(PB 301 166)A05

UCB/EERC-79/20 "Hysteretic Behavior of Reinforced Concrete Structural Walls," by J.M. Vallenas, V.V. Bertero and E.P. Popov - August 1979(PB 80 165 905)A12

UCB/EERC-79/21 "Studies on High-Frequency Vibrations of Buildings - 1: The Column Effect," by J. Lubliner - August 1979 (PB 80 158 553)A03

UCB/EERC-79/22 "Effects of Generalized Loadings on Bond Reinforcing Bars Embedded in Confined Concrete Blocks," by S. Viwathanatepa, E.P. Popov and V.V. Bertero - August 1979

UCB/EERC-79/23 "Shaking Table Study of Single-Story Masonry Houses, Volume 1: Test Structures 1 and 2," by P. Gülkan, R.L. Mayes and R.W. Clough - Sept. 1979

UCB/EERC-79/24 "Shaking Table Study of Single-Story Masonry Houses, Volume 2: Test Structures 3 and 4," by P. Gülkan, R.L. Mayes and R.W. Clough - Sept. 1979

UCB/EERC-79/25 "Shaking Table Study of Single-Story Masonry Houses, Volume 3: Summary, Conclusions and Recommendations," by R.W. Clough, R.L. Mayes and P. Gülkan - Sept. 1979

UCB/EERC-79/26 "Recommendations for a U.S.-Japan Cooperative Research Program Utilizing Large-Scale Testing Facilities," by U.S.-Japan Planning Group - Sept. 1979(PB 301 407)A06

UCB/EERC-79/27 "Earthquake-Induced Liquefaction Near Lake Amatitlan, Guatemala," by H.B. Seed, I. Arango, C.K. Chan, A. Gomez-Masso and R. Grant de Ascoli - Sept. 1979(NUREG-CR1341)A03

UCB/EERC-79/28 "Infill Panels: Their Influence on Seismic Response of Buildings," by J.W. Axley and V.V. Bertero - Sept. 1979(PB 80 163 371)A10

UCB/EERC-79/29 "3D Truss Bar Element (Type 1) for the ANSR-II Program," by D.P. Mondkar and G.H. Powell - Nov. 1979 (PB 80 169 709)A02

UCB/EERC-79/30 "2D Beam-Column Element (Type 5 - Parallel Element Theory) for the ANSR-II Program," by D.G. Row, G.H. Powell and D.P. Mondkar - Dec. 1979(PB 80 167 224)A03

UCB/EERC-79/31 "3D Beam-Column Element (Type 2 - Parallel Element Theory) for the ANSR-II Program," by A. Riahi, G.H. Powell and D.P. Mondkar - Dec. 1979(PB 80 167 216)A03

UCB/EERC-79/32 "On Response of Structures to Stationary Excitation," by A. Der Kiureghian - Dec. 1979(PB 80166 929)A03

UCB/EERC-79/33 "Undisturbed Sampling and Cyclic Load Testing of Sands," by S. Singh, H.B. Seed and C.K. Chan - Dec. 1979(

UCB/EERC-79/34 "Interaction Effects of Simultaneous Torsional and Compressional Cyclic Loading of Sand," by P.M. Griffin and W.N. Houston - Dec. 1979

UCB/EERC-80/01 "Earthquake Response of Concrete Gravity Dams Including Hydrodynamic and Foundation Interaction Effects," by A.K. Chopra, P. Chakrabarti and S. Gupta - Jan. 1980(AD-A087297)A10

UCB/EERC-80/02 "Rocking Response of Rigid Blocks to Earthquakes," by C.S. Yim, A.K. Chopra and J. Penzien - Jan. 1980 (PB80 166 002)A04

UCB/EERC-80/03 "Optimum Inelastic Design of Seismic-Resistant Reinforced Concrete Frame Structures," by S.W. Zagajeski and V.V. Bertero - Jan. 1980(PB80 164 635)A06

UCB/EERC-80/04 "Effects of Amount and Arrangement of Wall-Panel Reinforcement on Hysteretic Behavior of Reinforced Concrete Walls," by R. Iliya and V.V. Bertero - Feb. 1980(PB81 122 525)A09

UCB/EERC-80/05 "Shaking Table Research on Concrete Dam Models," by A. Niwa and R.W. Clough - Sept. 1980(PB81 122 368)A06

UCB/EERC-80/06 "The Design of Steel Energy-Absorbing Restrainers and their Incorporation into Nuclear Power Plants for Enhanced Safety (Vol 1A): Piping with Energy Absorbing Restrainers: Parameter Study on Small Systems," by G.H. Powell, C. Oughourlian and J. Simons - June 1980

UCB/EERC-80/07 "Inelastic Torsional Response of Structures Subjected to Earthquake Ground Motions," by Y. Yamazaki April 1980(PB81 122 327)A08

UCB/EERC-80/08 "Study of X-Braced Steel Frame Structures Under Earthquake Simulation," by Y. Ghanaat - April 1980 (PB81 122 335)A11

UCB/EERC-80/09 "Hybrid Modelling of Soil-Structure Interaction," by S. Gupta, T.W. Lin, J. Penzien and C.S. Yeh May 1980(PB81 122 319)A07

UCB/EERC-80/10 "General Applicability of a Nonlinear Model of a One Story Steel Frame," by B.I. Sveinsson and H.D. McNiven - May 1980(PB81 124 877)A06

UCB/EERC-80/11 "A Green-Function Method for Wave Interaction with a Submerged Body," by W. Kioka - April 1980 (PB81 122 269)A07

UCB/EERC-80/12 "Hydrodynamic Pressure and Added Mass for Axisymmetric Bodies," by F. Nilrat - May 1980(PB81 122 343)A08

UCB/EERC-80/13 "Treatment of Non-Linear Drag Forces Acting on Offshore Platforms," by B.V. Dao and J. Penzien May 1980(PB81 153 413)A07

UCB/EERC-80/14 "2D Plane/Axisymmetric Solid Element (Type 3 - Elastic or Elastic-Perfectly Plastic) for the ANSR-II Program," by D.P. Mondkar and G.H. Powell - July 1980(PB81 122 350)A03

UCB/EERC-80/15 "A Response Spectrum Method for Random Vibrations," by A. Der Kiureghian - June 1980(PB81 122 301)A03

UCB/EERC-80/16 "Cyclic Inelastic Buckling of Tubular Steel Braces," by V.A. Zayas, E.P. Popov and S.A. Mahin June 1980(PB81 124 885)A10

UCB/EERC-80/17 "Dynamic Response of Simple Arch Dams Including Hydrodynamic Interaction," by C.S. Porter and A.K. Chopra - July 1980(PB81 124 000)A13

UCB/EERC-80/18 "Experimental Testing of a Friction Damped Aseismic Base Isolation System with Fail-Safe Characteristics," by J.M. Kelly, K.E. Beucke and M.S. Skinner - July 1980(PB81 148 595)A04

UCB/EERC-80/19 "The Design of Steel Energy-Absorbing Restrainers and their Incorporation into Nuclear Power Plants for Enhanced Safety (Vol 1B): Stochastic Seismic Analyses of Nuclear Power Plant Structures and Piping Systems Subjected to Multiple Support Excitations," by M.C. Lee and J. Penzien - June 1980

UCB/EERC-80/20 "The Design of Steel Energy-Absorbing Restrainers and their Incorporation into Nuclear Power Plants for Enhanced Safety (Vol 1C): Numerical Method for Dynamic Substructure Analysis," by J.M. Dickens and E.L. Wilson - June 1980

UCB/EERC-80/21 "The Design of Steel Energy-Absorbing Restrainers and their Incorporation into Nuclear Power Plants for Enhanced Safety (Vol 2): Development and Testing of Restraints for Nuclear Piping Systems," by J.M. Kelly and M.S. Skinner - June 1980

UCB/EERC-80/22 "3D Solid Element (Type 4-Elastic or Elastic-Perfectly-Plastic) for the ANSR-II Program," by D.P. Mondkar and G.H. Powell - July 1980(PB81 123 242)A03

UCB/EERC-80/23 "Gap-Friction Element (Type 5) for the ANSR-II Program," by D.P. Mondkar and G.H. Powell - July 1980 (PB81 122 285)A03

UCB/EERC-80/24 "U-Bar Restraint Element (Type 11) for the ANSR-II Program," by C. Oughourlian and G.H. Powell July 1980(PB81 122 293)A03

UCB/EERC-80/25 "Testing of a Natural Rubber Base Isolation System by an Explosively Simulated Earthquake," by J.M. Kelly - August 1980

UCB/EERC-80/26 "Input Identification from Structural Vibrational Response," by Y. Hu - August 1980(PB81 152 308)A05

UCB/EERC-80/27 "Cyclic Inelastic Behavior of Steel Offshore Structures," by V.A. Zayas, S.A. Mahin and E.P. Popov August 1980

UCB/EERC-80/28 "Shaking Table Testing of a Reinforced Concrete Frame with Biaxial Response," by M.G. Oliva October 1980(PB81 154 304)A10

UCB/EERC-80/29 "Dynamic Properties of a Twelve-Story Prefabricated Panel Building," by J.G. Bouwkamp, J.P. Kollegger and R.M. Stephen - October 1980

UCB/EERC-80/30 "Dynamic Properties of an Eight-Story Prefabricated Panel Building," by J.G. Bouwkamp, J.P. Kollegger and R.M. Stephen - October 1980

UCB/EERC-80/31 "Predictive Dynamic Response of Panel Type Structures Under Earthquakes," by J.P. Kollegger and J.G. Bouwkamp - October 1980(PB81 152 316)A04

UCB/EERC-80/32 "The Design of Steel Energy-Absorbing Restrainers and their Incorporation into Nuclear Power Plants for Enhanced Safety (Vol 3): Testing of Commercial Steels in Low-Cycle Torsional Fatigue," by P. Spencer, E.R. Parker, E. Jongewaard and M. Drory

UCB/EERC-80/33 "The Design of Steel Energy-Absorbing Restrainers and their Incorporation into Nuclear Power Plants for Enhanced Safety (Vol 4): Shaking Table Tests of Piping Systems with Energy-Absorbing Restrainers," by S.F. Stiemer and W.G. Godden - Sept. 1980

UCB/EERC-80/34 "The Design of Steel Energy-Absorbing Restrainers and their Incorporation into Nuclear Power Plants for Enhanced Safety (Vol 5): Summary Report," by P. Spencer

UCB/EERC-80/35 "Experimental Testing of an Energy-Absorbing Base Isolation System," by J.M. Kelly, M.S. Skinner and K.E. Beucke - October 1980(PB81 154 072)A04

UCB/EERC-80/36 "Simulating and Analyzing Artificial Non-Stationary Earthquake Ground Motions," by R.F. Nau, R.M. Oliver and K.S. Pister - October 1980(PB81 153 397)A04

UCB/EERC-80/37 "Earthquake Engineering at Berkeley - 1980," - Sept. 1980

UCB/EERC-80/38 "Inelastic Seismic Analysis of Large Panel Buildings," by V. Schricker and G.H. Powell - Sept. 1980 (PB81 154 338)A13

UCB/EERC-80/39 "Dynamic Response of Embankment, Concrete-Gravity and Arch Dams Including Hydrodynamic Interaction," by J.F. Hall and A.K. Chopra - October 1980(PB81 152 324)A11

UCB/EERC-80/40 "Inelastic Buckling of Steel Struts Under Cyclic Load Reversal," by R.G. Black, W.A. Wenger and E.P. Popov - October 1980(PB81 154 312)A08

UCB/EERC-80/41 "Influence of Site Characteristics on Building Damage During the October 3, 1974 Lima Earthquake," by P. Repetto, I. Arango and H.B. Seed - Sept. 1980(PB81 161 739)A05

UCB/EERC-80/42 "Evaluation of a Shaking Table Test Program on Response Behavior of a Two Story Reinforced Concrete Frame," by J.M. Blondet, R.W. Clough and S.A. Mahin

UCB/EERC-80/43 "Modelling of Soil-Structure Interaction by Finite and Infinite Elements," by F. Medina

UCB/EERC-81/01 "Control of Seismic Response of Piping Systems and Other Structures by Base Isolation," edited by J.M. Kelly - January 1981 (PB81 200 735)A05

UCB/EERC-81/02 "OPTNSR - An Interactive Software System for Optimal Design of Statically and Dynamically Loaded Structures with Nonlinear Response," by M.A. Bhatti, V. Ciampi and K.S. Pister - January 1981 (PB81 218 851)A09

UCB/EERC-81/03 "Analysis of Local Variations in Free Field Seismic Ground Motion," by J.-C. Chen, J. Lysmer and H.B. Seed - January 1981 (AD-A099508)A13

UCB/EERC-81/04 "Inelastic Structural Modeling of Braced Offshore Platforms for Seismic Loading," by V.A. Zayas, P.-S. B. Shing, S.A. Mahin and E.P. Popov - January 1981 (PB

UCB/EERC-81/05 "Dynamic Response of Light Equipment in Structures," by A. Der Kiureghian, J.L. Sackman and B. Nour-Omid - April 1981 (PB81 218 497)A04

UCB/EERC-81/06 "Preliminary Experimental Investigation of a Broad Base Liquid Storage Tank," by J.G. Bouwkamp, J.P. Kollegger and R.M. Stephen - May 1981

UCB/EERC-81/07 "The Seismic Resistant Design of Reinforced Concrete Coupled Structural Walls," by A.E. Aktan and V.V. Bertero - June 1981 (PB82 113 358)A11

UCB/EERC-81/08 "The Undrained Shearing Resistance of Cohesive Soils at Large Deformation," by M.R. Pyles and H.B. Seed - August 1981

UCB/EERC-81/09 "Experimental Behavior of a Spatial Piping System with Steel Energy Absorbers Subjected to a Simulated Differential Seismic Input," by S.F. Stiemer, W.G. Godden and J.M. Kelly - July 1981

UCB/EERC-81/10 "Evaluation of Seismic Design Provisions for Masonry in the United States," by B.I. Sveinsson, R.L. Mayes and H.D. McNiven - August 1981

UCB/EERC-81/11 "Two-Dimensional Hybrid Modelling of Soil-Structure Interaction," by T.-J Tzong, Sunil Gupta and J. Penzien - August 1981

UCB/EERC-81/12 "Studies on Effects of Infills in Seismic Resistant R/C Construction," by S. Brokken and V.V. Bertero - September 1981

UCB/EERC-81/13 "Linear Models to Predict the Nonlinear Seismic Behavior of a One-Story Steel Frame," by H. Valdimarsson, A.H. Shah and H.D. McNiven - September 1981

UCB/EERC-81/14 "TLUSH: A Computer Program for the Three-Dimensional Dynamic Analysis of Earth Dams," by T. Kagawa, L.H. Mejia, H.B. Seed and J. Lysmer - September 1981

UCB/EERC-81/15 "Three Dimensional Dynamic Response Analysis of Earth Dams," by L.H. Mejia and H.B. Seed - September 1981

UCB/EERC-81/16 "Experimental Study of Lead and Elastomeric Dampers for Base Isolation Systems," by J.M. Kelly and S.B. Hodder - October 1981

UCB/EERC-81/17 "The Influence of Base Isolation on the Seismic Response of Light Secondary Equipment," by J.M. Kelly - April 1981

UCB/EERC-81/18 "Studies on Evaluation of Shaking Table Response Analysis Procedures," by J. Marcial Blondet - November 1981

UCB/EERC-81/19 "DELIGHT.STRUCT: A Computer-Aided Design Environment for Structural Engineering," by R.J. Balling, K.S. Pister and E. Polak - December 1981

UCB/EERC-81/20 "Optimal Design of Seismic-Resistant Planar Steel Frames," by R.J. Balling, V. Ciampi, K.S. Pister and E. Polak - December 1981