An Implementation of the Programming Structural Synthesis System (PROSSS)

James L. Rogers, Jr., Jaroslaw Sobieszczanski-Sobieski, and Rama B. Bhat

DECEMBER 1981
An Implementation of the Programming Structural Synthesis System (PROSSS)

James L. Rogers, Jr., and Jaroslaw Sobieszczanski-Sobieski
Langley Research Center
Hampton, Virginia

Rama B. Bhat
The George Washington University
Joint Institute for Advancement of Flight Sciences
Langley Research Center
Hampton, Virginia
CONTENTS

TABLES ........................................................................ vi
FIGURES ........................................................................ vi

1 INTRODUCTION ............................................................... 1

2 OVERVIEW OF THIS IMPLEMENTATION OF PROSSS ................. 1

3 STEPS FOR IMPLEMENTING PROSSS ........................................ 3

4 COMPONENTS OF THE SYSTEM ............................................. 7
  4.1 Option Files ................................................................. 7
  4.2 Procedure Files ............................................................. 7
    4.2.1 Nonrepeatable Model Definition: PRCNRPT ....................... 8
    4.2.2 Initialization: PRCINIT, PRCGETF ................................. 9
    4.2.3 Front Processing: PRCFPXX .......................................... 10
    4.2.4 Optimization: PRCOPTM ............................................. 10
    4.2.5 Analysis: PRCANAL .................................................... 10
    4.2.6 Analytical Gradient Calculations: PRCGRDS .................... 11
    4.2.7 End Processing: PRCEPXX .......................................... 11
    4.2.8 Printing Output: PRCEND ........................................... 11
  4.3 Program and Subroutine Files ............................................. 11
    4.3.1 Analysis Program: SPAR ............................................. 13
      4.3.1.1 Overall characteristics .......................................... 13
      4.3.1.2 Analytical calculation of gradients ............................. 15
      4.3.1.3 SPAR data storage and retrieval ............................... 15
    4.3.2 Optimization Program: CONMIN ..................................... 15
      4.3.2.1 Overall characteristics .......................................... 15
      4.3.2.2 Program: CONMS1 .............................................. 16
      4.3.2.3 Program: CONMS2 .............................................. 17
    4.3.3 Front Processor Program: FPROC ................................... 18
    4.3.4 End Processor Program: EPROC .................................... 19
    4.3.5 Control Programs ..................................................... 19
      4.3.5.1 Program to compute the finite difference gradients: EVALS .. 20
      4.3.5.2 Fully stressed design (FSD) program: FSDS .................. 20
      4.3.5.3 Program to select the active or violated constraints: SELECTS .. 21
      4.3.5.4 Program to rewrite the design variables to a different file: RERITES .. 21
    4.3.6 Gradient Programs ..................................................... 22
      4.3.6.1 Program to create input file for computing gradients in the analysis program in the nonrepeatable part: BLDELDS ........ 22
      4.3.6.2 Program to create input file for computing gradients in the analysis program in the repeatable part: GNGRDRS ........ 22
      4.3.6.3 Program to convert forces and moments and the derivatives of forces and moments to stresses and stress derivatives: DRVSTRES .... 23

iii
ENDIN ................................................................. 98
CONS ................................................................. 98
MODEL DATA FILES .................................................. 99
  NRRS (nonrepeatable SPAR runstream).......................... 99
  NGRS (repeatable SPAR runstream).............................. 101
TRANSFER DATA FILES ............................................... 102
  PCONRST .......................................................... 102
  PCNMNIO,TCNMNIO,CNMNIO(LINKE,LINKF) ....................... 110
  BLOCK,BLK ....................................................... 111
  SPFPOUT ................................................................ 119
  RSOUT (nonrepeatable part) ...................................... 120
  RSOUT (repeatable part) ......................................... 125
EDIT DATA FILES ..................................................... 127
  EDPASS1 ............................................................. 127
  EDPASS2 ............................................................. 127
  EDIT1 ................................................................. 127
  EDIT2 ................................................................. 127
  MERGFP .............................................................. 128
  EDGRDS .............................................................. 128

REFERENCES ................................................................ 129
TABLES

I OPTIONS FOR OPTIMIZATION .............................................................. 3
II PROCEDURE FILES IN PROSSS .......................................................... 8
III PROGRAM AND SUBROUTINE FILES IN PROSSS .................................. 12
IV SPAR PROCESSORS ........................................................................... 14
V NONREPEATABLE (N) AND REPEATABLE (R) PARTS IN FINITE
  ELEMENT ANALYSIS BASED ON DISPLACEMENT METHOD ..................... 15
VI DATA FILES IN PROSSS ................................................................. 25
VII COMPARISON OF RESULTS FROM DIFFERENT PROSSS OPTIONS .......... 30

FIGURES

1 Basic flow of PROSSS ........................................................................ 2
2 SPAR system organization ................................................................. 13
3 CONMIN program organization ......................................................... 16
4 Fuselage model used for testing ....................................................... 29
1 INTRODUCTION

Numerous approaches have been documented for combining optimization techniques with an analysis capability (e.g., refs. 1 to 3). The approach documented in this paper is a particular implementation of the method for combining analysis and optimization techniques with applications to structures (ref. 4). This method, called the programming structural synthesis system (PROSSS), combines a large, general purpose, finite element program for structural analysis, SPAR (ref. 5), with a large, general purpose, optimization program, CONMIN (ref. 6) and several, small, problem-dependent FORTRAN programs and subroutines which must be written by the user to interface the analysis and optimization programs. All of the programs are connected by a network of control cards in the standard, Control Data Corporation CYBER Control Language (CCL), documented in reference 7. Familiarity with the theory behind this method (ref. 4) and with the software (documented in refs. 5 to 7) is a prerequisite for understanding the remainder of this document.

This particular implementation of PROSSS is only the first step in a series of implementations. Other implementations are intended to give the user easier access to intermediate results and more control over the flow of the problem, as well as a capability for interactive modeling and data generation (ref. 8). Another implementation includes incorporating PROSSS entirely within the Engineering Analysis Language (EAL, ref. 9) computer program to simplify the maintenance, control, and data management aspects.

This paper describes a particular implementation of PROSSS. First, an overview is given which explains PROSSS in general with respect to this implementation. The second section describes how the input data are prepared. Next, each component of the system is explained in detail. These components include options, procedures, programs and subroutines, and data files used in this implementation. Finally, an example exercise for each option is given to allow the user to anticipate the type of results which might be expected. The appendixes contain annotated listings and flow-charts to clarify the descriptions of the components of the system presented within the body of this paper.

The purpose of this paper is to demonstrate one method for implementing a flexible software system combining large, complex programs with control language and small, user-supplied, problem-dependent programs. It is not intended to be a self-contained user's guide for PROSSS.

Identification of commercial products in this report is used to adequately describe the model. The identification of these commercial products does not constitute official endorsement, expressed or implied, of such products or manufacturers by the National Aeronautics and Space Administration.

2 OVERVIEW OF THIS IMPLEMENTATION OF PROSSS

This implementation of the programming structural synthesis system (PROSSS) combines a general purpose, finite element computer program for structural analysis (SPAR), a state-of-the-art optimization program (CONMIN), and several user-supplied,
problem-dependent computer programs. All of the programs are connected by the stand­
dard CCL. The results are flexibility of the optimization procedure organization and
versatility of the formulation of constraints and design variables.

A flowchart of the analysis-optimization process for this implementation is
shown in figure 1. The process results in a minimized objective function, typically

![Flowchart of PROSSS](image)

Figure 1.- Basic flow of PROSSS.

the mass defined in terms of a set of design variables, such as cross-sectional
dimension of a structural member. This member is subject to a set of constraints
such that stress must be less than some allowable value. Notice that the analysis
and optimization programs are executed repeatedly by looping through the system until
the process is stopped by a user-defined termination criterion. This part of the
system is referred to as the repeatable part of PROSSS. However, some of the analy­
sis, such as model definition, need only be done one time and the results saved for
future use. This analysis is performed outside of the loop and referred to as the
nonrepeatable part of PROSSS. The user must write some small FORTRAN programs (e.g.,
front processor and end processor) to interface between the analysis and optimization programs. The front processor converts the design variables output from the optimizer into a suitable format for input into the analyzer. The end processor retrieves the behavior variables (i.e., stresses or deflections due to loads) from the analysis program, evaluates the objective function and constraints, and optionally retrieves and evaluates their gradients. These quantities are output in a format suitable for input into the optimizer. These user-supplied programs are problem dependent, because they depend primarily upon the finite elements being used in the model.

Five options for organizing optimization procedures by combining nonlinear or piecewise linear programming methods with analytical or finite difference gradients are shown in table I. Each option is controlled by a complex CCL sequence of commands. These commands are modularized in the form of procedure files and perform the functions of executing programs in certain sequences, such as if-test branching, looping, and manipulating permanent and temporary files. Each procedure file is written using structured programming techniques to aid in readability.

Although presented in the context of structural analysis, the same system concept could be used for aerodynamic optimization of a wing, if SPAR were replaced by an analyzer with a capability for computational aerodynamics. The system is intended to be used in the following three basic ways:

(1) as a research tool for the development of optimization techniques that will interface with an efficient analysis program

(2) as a research tool for testing new analysis techniques that will interface with an efficient optimization program

(3) as an application tool that can be adapted to a wide range of problem types

### TABLE I.- OPTIONS FOR OPTIMIZATION

<table>
<thead>
<tr>
<th>Method</th>
<th>Gradients computed</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In CONMIN by</td>
<td>External to CONMIN by</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Finite difference</td>
<td>Finite difference</td>
<td>Analytical</td>
</tr>
<tr>
<td>Nonlinear programming</td>
<td>1.1</td>
<td>1.2</td>
<td>1.3</td>
</tr>
<tr>
<td>Piecewise linear programming</td>
<td>N/A</td>
<td>2.2</td>
<td>2.3</td>
</tr>
</tbody>
</table>

3 STEPS FOR IMPLEMENTING PROSSS

The user has access to all of the files containing the components of PROSSS (except SPAR and CONMIN), so that he can modify its programs, procedure files, and flow organization. This capability for component modification makes PROSSS suitable
for use as a research test bed. The following step-by-step description explains how to start a new implementation using SPAR as the analyzer and CONMIN as the optimizer. (Details of the files discussed are given in sec 4.)

1. Write a SPAR runstream (see sec 4.4.2) for the entire problem, execute it, and verify the results. Save the SPARLA (see sec 4.4.3) as a permanent file.

2. Divide the SPAR runstream into the nonrepeatable and repeatable parts. The nonrepeatable part contains the TAB (ref. 5) and ELD processors while the repeatable part contains the rest. Execute the nonrepeatable part and save its SPARLA as a permanent file with a different name than the one used in step 1.

3. Write a front processor and its input file CNMNIO. The input file contains the design variables yielding the same structural parameters (behavior variables) as those used as SPAR input in step 1. The front processor reads these design variables and outputs them in a form suitable for input into the SPAR TAB processor. Execute the front processor by using the procedure file PRCFPXX and verify the output file SPFPOUT.

4. Execute the procedure files PRCFPXX and PRCANAL in this order. SPAR output should agree with that obtained in step 1. Save the SPARLA from this execution as a permanent file with name different from those used in steps 1 and 2.

5. Write an end processor and its input data file ENDN. The end processor converts data output by SPAR on SPARLA in step 4 into objective function and constraint data for input into CONMIN. Execute the end processor by using the procedure file PRCEPXX and the SPARLA from step 4 as input. Verify the results on output file CNMNIO.

6. Execute the procedure files PRCFPXX, PRCANAL, PRCEPXX in this order. Verify that the CNMNIO file is the same as that obtained in step 5.

7. If the gradients are to be computed analytically, perform the following additional steps:

   (a) Write input files INPT and, if needed, CONS required by the procedure file PRCGRDS, and rerun step 2 by using these files.

   (b) Execute the procedure file PRCGRDS using the SPARLA saved in step 4, as input. Save the SPARLD as a permanent file. Verify the SPAR output containing the desired gradient values.

   (c) Expand the end processor written in step 5 to handle the gradients available in the SPARLD saved in step 7(b). Execute the end processor and verify its output file CNMNIO to see that its contents are augmented by the gradient values.

   (d) Execute the procedure files PRCFPXX, PRCANAL, PRCGRDS, and PRCEPXX in this order and verify that file CNMNIO is the same as that obtained in step 7(c).

8. Write an input file PCNPR containing the CONMIN control parameters. Set ITMAX = 1 (Maximum iterations = 1).
9. Write a file PSTRT containing the starting values of the design variables (same as those used in step 3). Save this file as a permanent file.

10. Select an execution option.

11. In program CONMS1 or CONMS2 (EVALS and FSDS, if applicable), change the dimension statements as required and recompile the program.

12. Write an input file for PROSCR to convert the general names in the option files (app. A) to specific, problem-dependent names.

13. Execute PROSCR and verify that the output file CNMNIO is the same as that obtained in step 6 or 7(d).

14. Relax the ITMAX = 1 restriction in the CONMIN control parameter file PCNPR. Proceed with optimization. If option 2.2 or 2.3 (table I) is chosen, allow completion of one linear stage and verify its results before continuing.

15. When the optimization is underway, periodically inspect the CONOUT file for acceptability of the direction the process is taking.

After step 15 has been completed, only the input data files should be changed for a different application to a problem of the same class. In this way, PROSSS is executed like a "black box" because PROSSS is set up and debugged for a specific type of application. All of its components, except the problem-dependent input data files, are protected from unauthorized access. The user thus retains the freedom to change input data as he studies a class of problems, but he cannot change the inner workings of the system. Such restricted use is desirable in production applications, because it facilitates the separation of responsibilities of a support staff specialist from those of the engineer user.

Using PROSSS as a black box is very simple. The user must create an input file PROSSIN for a small program called PROSCR. (See step 12.) This input file contains the specific names and numbers to replace the general names in the option files (app. A) to solve a particular problem (see sec 4.1). The first column of names, each beginning in column 1, are the general names and the second column of names and numbers, each beginning in column 11, are the specific names and numbers supplied by the user. The first name in the input file must be the option number, thereafter the order is not important. The following are the general names that are to be replaced:

- **POPT**: option number (11, 12, 13, 22, or 23)
- **NONREPT**: 1, if nonrepeatable part to be executed; 0, otherwise
- **NRRS**: name of runstream input to SPAR for nonrepeatable part, can be omitted if NONREPT is 0
- **FUSD**: 1, if fully stressed design is to be used for optimization program; 0, otherwise
- **FSDSUB**: file containing fully stressed design subroutines to be appended to main program, can be omitted if FUSD is 0
- **CONMIN**: file containing main program for optimization
ENDP1  file containing end processor program
FRNT  file containing front processor program
ENDN  file containing input to end processor
PCNPR  file containing input to optimization program
PSTRT  file containing starting values for design variables
PCONRST  file containing restart values for optimization program, a transfer file
PCNMNIO  file containing data transferred to/from the optimization program
SAVCOUT  file containing cumulative output listing from the optimization program
NSPARLA  SPAR library saved from nonrepeatable part
NGRS  
   or  
   RGS  
   spar runstream for repeatable part
FLENGTH  field length (octal) required for SPAR execution
BLK  file containing objective function and constraint data, a transfer file
SAVSPLD  SPAR library saved from repeatable part
CNT  file containing testing information for termination criterion
CONS  input file of constants such as the cross-sectional areas of beams
   for user-supplied subroutines to analytically calculate the
   gradients and the front processor

   The following names can be omitted if options 1.3 or 2.3 (use of analytical
   gradients) are not chosen:

   SUBS  n, the number of subroutines supplied by the user to calculate
   analytical gradients; one subroutine is needed for each element type
   containing more than one design variable

   BINDEPB  file containing 2n subroutines supplied by the user for calculating
   analytical gradients; can be omitted if n = 0

   INPT  input file of control parameters for calculating analytical gradients

Once the input file, PROSSIN, has been created, it is input into the PROSCRS
program. PROSCRS creates three output files: a file of control cards; a file of
Text Editor commands (ref. 10) to replace the general names with the specific names;
and a file of Text Editor commands (ref. 10) to remove unwanted blanks in the preceding file. The control file is rewound and executed after PROSCRS has completed. The control file performs the following four functions: gets the PROSOPT file (see sec 4.1) and copies the correct option onto file OPTION; edits the file containing edit commands to remove the blanks; edits the OPTION file to change general names to specific names; and starts the option executing.
The program header card for PROSCRS is as follows:

```
PROGRAM PROSCRS (TAPE8, TAPE9, TAPE10, TAPE11)
```

where

- **TAPE8** is the input file of general and specific names
- **TAPE9** is the output file of control cards
- **TAPE10** is the output file of edit commands to change from general names to specific names
- **TAPE11** is the output file of edit commands to remove blanks

Listings of PROSCRS, the input files, the control card file, and the edit file also are presented in appendix A.

4 COMPONENTS OF THE SYSTEM

PROSSS is composed of a system of files consisting of four primary components: option files, procedure files, program files, and data files. Each file is explained in detail with annotated examples listed in the appendixes. For those files which are problem dependent and supplied by the user, the descriptions are given with emphasis on the way the files interface with each other.

4.1 Option Files

There are five option files in PROSSS (table I): options 1.1, 1.2, 1.3, 2.2, and 2.3. Options 1.1, 1.2, and 1.3 use nonlinear mathematical programming, while options 2.2 and 2.3 use piecewise linear programming. Option 1.1 uses gradients calculated inside CONMIN. Options 1.2 and 2.2 use gradients calculated by the finite difference method, while options 1.3 and 2.3 use analytically computed gradients. Five procedure files, one for each of the five options, exist on file PROSOPT. These procedure files consist of a sequence of control cards in CCL (ref. 7). Listings and flowcharts of each option are shown in appendix B. The BEGIN cards in these procedure files control the sequence of execution of the procedure files described in section 4.2. No option procedures have any key words associated with them. Each procedure does, however, have many general names within it. These general names are replaced by specific names (using the Text Editor (ref. 10)) before the option begins executing. These names are described in detail in sections 4.2 to 4.4, and the replacement process was previously explained in section 3. This process was chosen over key word substitution because one BEGIN card (ref. 7) would not hold all the necessary key words and no continuation card is allowed.

4.2 Procedure Files

PROSSS is controlled by nine procedure files, all of which are located on a file named PROSPRC. The specific functions performed by the procedures are (1) nonrepeatable model definition (PRCNRPT), (2) and (3) initialization (PRCINIT and PRCGETF), (4) front processing (PRCFPXX), (5) optimization (PRCOPTM), (6) analysis
(PRCANAL), (7) analytical gradient calculation (PRCGRDS), (8) end processing (PRCEPXX), and (9) printing output (PRCEND). Each procedure consists of a sequence of control cards in CCL (ref. 7). A procedure is called by a BEGIN statement. Commented listings of each of the nine procedure files are shown in appendix C. Each procedure begins with a header card containing a list of key words, if needed. Key word substitution allows the user to substitute key words in the procedure body with parameters specified on the BEGIN statement. Table II provides a quick reference for each procedure file by giving the file name, the purpose, and the calling sequence.

**TABLE II.- PROCEDURE FILES IN PROSSS**

<table>
<thead>
<tr>
<th>File name</th>
<th>Purpose</th>
<th>File call command</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRCNRPT</td>
<td>Executes nonrepeatable analysis program</td>
<td>.PROC,PRCNRPT,NROPT,NRRS,FLX,I,NRLA.</td>
</tr>
<tr>
<td>PRCINIT</td>
<td>Assembles user supplied programs and subroutines</td>
<td>.PROC,PRCINIT,OP,A,B,NSUB,C,FSD,FSUB.</td>
</tr>
<tr>
<td>PROGETF</td>
<td>Retrieves remainder of files needed for execution</td>
<td>.PROC,PROGETF,OP,F,E,CN,S,I,C,CT,RS,RGS.</td>
</tr>
<tr>
<td>PRCPFXX</td>
<td>Executes the front processor</td>
<td>.PROC,PRCPFXX.</td>
</tr>
<tr>
<td>PRCOPTM</td>
<td>Executes the optimization program</td>
<td>.PROC,PRCLOPTM,C,D,F.</td>
</tr>
<tr>
<td>PRCANAL</td>
<td>Executes the repeatable analysis program</td>
<td>.PROC,PRCANAL,NRLA,FLX,SAUELD.</td>
</tr>
<tr>
<td>PRCGRDS</td>
<td>Creates a runstream for input to the analysis program, then executes the analysis program and, optionally, a post processor to find analytical gradients</td>
<td>.PROC,PRCGRDS,NSUB,SAUELD.</td>
</tr>
<tr>
<td>PRCEPXX</td>
<td>Executes the end processor</td>
<td>.PROC,PRCEPXX,BLK.</td>
</tr>
<tr>
<td>PRCEND</td>
<td>Outputs important files</td>
<td>.PROC,PRCEND.</td>
</tr>
</tbody>
</table>

4.2.1 Nonrepeatable Model Definition: PRCNRPT

This procedure PRCNRPT creates a SPAR library of the joint and element information for the finite element model (app. C). If analytical gradients are required, the derivatives of the stiffness and mass matrices with respect to the design variables are also computed and stored on the library. The procedure header card is as follows:

```
.PROC,PRCNRPT,NROPT,NRRS,FLX,I,NRLA.
```

where PRCNRPT is the name of procedure file, with key words:

- **NROPT**: option number
- **NRRS**: nonrepeatable SPAR runstream
- **FLX**: field length (octal)
input file for analytical gradient calculation (see INPT)

NRLA SPAR library

4.2.2 Initialization: PRCINIT,PRCGETF

Two procedure files, PRCINIT and PRCGETF, are used in the initialization process. The first, PRCINIT, gets the programs required for a particular option (app. C). Some programs must be assembled using various user-supplied main programs and/or subroutines, the names of which are passed through the header card. The procedure header card is as follows:

```
.PROC,PRCINIT,OP,A,B,BB,NSUB,C,FSD,FSUB.
```

where PRCINIT is the name of procedure file, with key words:

- **OP**: option number
- **A**: main program for optimization
- **B**: main program for end processor with no gradients
- **BB**: main program for end processor with gradients
- **NSUB**: number of user-supplied subroutines for analytical gradient calculation
- **C**: file containing all user-supplied subroutines for analytical gradient calculation. The first NSUB subroutines are combined with program GNGRDRS. The second NSUB subroutines are combined with program DRVSTRS. Each program is a physical record (not needed if NSUB is zero)
- **FSD**: 1, implies fully stressed design is required; 0, implies no fully stressed design is required
- **FSUB**: file containing fully stressed design subroutines (not needed if FSD is zero)

The second procedure file PRCGETF, used in the initialization process retrieves the remainder of the files required for executing a particular option (app. C). The procedure header card is as follows:

```
.PROC,PRCGETF,OP,F,E,CN,S,I,C,CT,RS,RGS.
```

where PRCGETF is the name of procedure file, with key words:

- **OP**: option number
- **F**: front processor program (see FPROC)
- **E**: input file to end processor (see ENDN)
- **CN**: input file to optimization program (see PCNPR,CONPAR)
S input files to optimization program (see PSTRT,STARTX)
I input file for analytical gradient calculation (see INPT)
C input file of constants for analytical gradient calculation (see CONS)
CT input file of constants for optimization program (see CNT)
RS SPAR runstream (no gradients, see NGRS)
RGS SPAR runstream (gradients, see RGS)

4.2.3 Front Processing: PRCFPXX

The procedure file PRCFPXX executes the front processor (app. C). The procedure header card is as follows:

.PROC,PRCFPXX.

where PRCFPXX is the name of procedure file. There are no key words.

4.2.4 Optimization: PROCOPTM

The procedure file PROCOPTM executes the optimization program (app. C). The procedure header card is as follows:

.PROC,PROCOPTM,C,D,F.

where PROCOPTM is the name of procedure file, with key words:

C restart data for optimization program (see PCONRST)
D transfer data to/from optimization program (see PCNMNIO)
F cumulative output from optimization program (see SAVCOUT)

4.2.5 Analysis: PRCANAL

The procedure file PRCANAL merges the output file (SPFPOUT) from the front processor into the SPAR runstream file (SPARRS) and executes the SPAR analysis program. (See app. C.) The initial SPAR input and output are saved for later listing. The procedure header card is as follows:

.PROC,PRCANAL,NRLA,FLX,SAVELD.

where PRCANAL is the name of procedure file, with key words:

NRLA SPAR library from nonrepeatable part
FLX field length (octal)
SAVELD SPAR library for use by end processor
4.2.6 Analytical Gradient Calculations: PRCGRDS

This procedure file, PRCGRDS, creates a SPAR runstream and then uses SPAR and a postprocessor (see sec 4.3.6.3) to SPAR to calculate stress derivatives when forces and moments and derivatives of forces and moments must be converted to stresses and stress derivatives. (See app. C.) The procedure header card is as follows:

PROC,PRCGRDS,NSUB,SAVELD.

where PRCGRDS is the name of procedure file, with key words:

NSUB number of user-supplied subroutines for analytical gradient calculations

SAVELD SPAR library for use by end processor

4.2.7 End Processing: PRCEPXX

This procedure file, PRCEPXX, executes the end processor. (See app. C.) The procedure header card is as follows:

PROC,PRCEPXX,BLK.

where PRCEPXX is the name of procedure file, with key word:

BLK transfer data file (see BLOCK)

4.2.8 Printing Output: PRCEND

This procedure file, PRCEND, prints CONMIN and SPAR output files and SPAR runstreams. (See app. C.) The procedure header card is as follows:

PROC,PRCEND.

where PRCEND is the name of procedure file. There are no key words.

4.3 Program and Subroutine Files

PROSSS uses the following six types of programs during the analysis-optimization process: analysis (SPAR), optimization (CONMIN), front processor (FPROC), end processor (EPROC), control programs, and analytical gradient programs. The analysis program SPAR and the CONMIN subroutine library are not given in the appendixes because they are standard for the system and are documented in references 5 and 6. The user does not have to create either SPAR or the CONMIN subroutine library. Sample listings for all other programs, including the main driver program for the CONMIN subroutine library, are shown in appendix D. Table III provides a quick reference for each of the programs by giving the options and procedures in which the program is used, whether or not the program must be created by the user, and a brief comment about the function of the program.
<table>
<thead>
<tr>
<th>Program (P) or subroutine (S) name</th>
<th>User created</th>
<th>Option(s)</th>
<th>Procedure(s)</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPAR (P)</td>
<td>No</td>
<td>All</td>
<td>PRCRNPT,PRCNAL, PROCSRS</td>
<td>Finite element structural analysis program and its data management system.</td>
</tr>
<tr>
<td>CONMIN (S) subroutines</td>
<td>No</td>
<td>All</td>
<td>PROCPYM</td>
<td>Problem independent set of optimization subroutines.</td>
</tr>
<tr>
<td>CONMS1 (P)</td>
<td>Yes</td>
<td>1.1, 1.2, 1.3</td>
<td>PROCPYM</td>
<td>Driver program for CONMIN optimization. Has problem dependent dimensioned variables in blank common. (Nonlinear)</td>
</tr>
<tr>
<td>CONMS2 (P)</td>
<td>Yes</td>
<td>2.2, 2.3</td>
<td>PROCPYM</td>
<td>Driver program for CONMIN with linear extrapolation in lieu of full analysis. Has problem dependent dimensioned variables in blank common. (Piecewise linear)</td>
</tr>
<tr>
<td>FPROC (P)</td>
<td>Yes</td>
<td>All</td>
<td>PROCPXK</td>
<td>Creates a file of design variables in a format meaningful for input into SPAR.</td>
</tr>
<tr>
<td>EPROC (P)</td>
<td>Yes</td>
<td>All</td>
<td>PRCEPXK</td>
<td>Creates a file of objective function, constraints, and optionally, their gradients in a format meaningful for input into CONMIN.</td>
</tr>
<tr>
<td>EVALS (P)</td>
<td>Yes</td>
<td>1.2, 2.2, 2.3</td>
<td>None, called from option file</td>
<td>Computes finite difference gradients. Dimensioned variables are problem dependent.</td>
</tr>
<tr>
<td>FSDS (P)</td>
<td>Yes</td>
<td>All</td>
<td>PROCPYM</td>
<td>Replaces CONMIN. Modifies values of initial design variables by means of fully stressed design techniques. Dimensioned variables are problem dependent. Can be used as an inexpensive means for finding starting design variables.</td>
</tr>
<tr>
<td>FSDSUBS (S)</td>
<td>Yes</td>
<td>All</td>
<td>PROCPYM</td>
<td>Subroutines used in conjunction with FSDS.</td>
</tr>
<tr>
<td>SELECTS (P)</td>
<td>Yes</td>
<td>1.2, 1.3</td>
<td>None, called from option file</td>
<td>Determines if constraints are active or violated.</td>
</tr>
<tr>
<td>RERITES (P)</td>
<td>No</td>
<td>1.2</td>
<td>None, called from option file</td>
<td>Copies design variables from one file to another.</td>
</tr>
<tr>
<td>BLDELDS (F)</td>
<td>No</td>
<td>1.3, 2.3</td>
<td>PRCRNPT</td>
<td>Creates an input file for computing gradients in the analysis program in the nonrepeatable part.</td>
</tr>
<tr>
<td>GNGRDRS (P)</td>
<td>No</td>
<td>1.3, 2.3</td>
<td>PROCRDS</td>
<td>Creates an input file for computing gradients in the analysis program in the repeatable part.</td>
</tr>
<tr>
<td>DKDVE21 (S)</td>
<td>Yes</td>
<td>1.3, 2.3</td>
<td>PROCRDS</td>
<td>Creates part of the input to compute derivatives of the mass and stiffness matrices with respect to the design variables. These subroutines are used with GNGRDRS.</td>
</tr>
<tr>
<td>DKDVE22</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DKDVE23</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DKDVE43</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRVSTRS (P)</td>
<td>No</td>
<td>1.3, 2.3</td>
<td>PROCRDS</td>
<td>Converts forces and moments and derivatives of forces and moments to stresses and stress derivatives.</td>
</tr>
<tr>
<td>BMGSTRS (S)</td>
<td>Yes</td>
<td>1.3, 2.3</td>
<td>PROCRDS</td>
<td>Computes stresses and stress derivatives for beam and plate elements. (Subroutines)</td>
</tr>
</tbody>
</table>
4.3.1 Analysis Program: SPAR

4.3.1.1 Overall characteristics.- SPAR is a system of processors which perform linear, finite element, structural analysis (ref. 5). It can compute static deflections, stresses, vibration frequencies and modes, dynamic responses, buckling loads, and mode shapes. Shown in figure 2 is the organization of the SPAR processors and data flow. The individual processors communicate with a central body of information known as the data complex. The data complex consists of one or more libraries, which contain the data sets output from the different processors. Each data set has a specific identifying name by which any processor can access it whenever it is required as input for particular computations. A list of the SPAR processors and their functions is given in table IV. The numeral next to the processor names refers to the rows of table V. In table V certain operations performed in SPAR are defined and the operation is broken down according to the type of variable and whether or not the operation occurs in the repeatable or nonrepeatable part of PROSSS. Taken together, tables IV and V show a division of the SPAR processors between nonrepeatable and repeatable parts. Processors without numerals are utilities for functions such as plotting, printing, eigenvalue extraction, etc.

SPAR executes on a processor-by-processor basis. Each processor is executed by a separate explicit command. A runstream consisting of a string of such commands interlaced with the numerical input data is written by the user for a specific problem. Runstreams are described in section 4.4.2. This modularity of SPAR organization and execution makes it well suited for optimization applications.
TABLE IV.- SPAR PROCESSORS

<table>
<thead>
<tr>
<th>Operation number in table V</th>
<th>Name</th>
<th>Processor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Function</td>
</tr>
<tr>
<td>1, 2, 3, 6</td>
<td>*TAB</td>
<td>Creates data sets containing tables of joint locations, section properties, material constants, etc.</td>
</tr>
<tr>
<td>5, 7</td>
<td>ELD</td>
<td>Defines finite elements making up model</td>
</tr>
<tr>
<td>5, 6, 7</td>
<td>E</td>
<td>Generates sets of information for each element, including connected joint numbers, geometrical data, material and section property data</td>
</tr>
<tr>
<td>8, 9</td>
<td>EKS</td>
<td>Adds stiffness and stress matrices for each element to set of information produced by E processor</td>
</tr>
<tr>
<td>10</td>
<td>TOPO</td>
<td>Analyzes element interconnection topology and creates data sets used to assemble and factor system mass and stiffness matrices</td>
</tr>
<tr>
<td>10</td>
<td>K</td>
<td>Assembles unconstrained system stiffness matrix in sparse format</td>
</tr>
<tr>
<td>11</td>
<td>M</td>
<td>Assembles unconstrained system mass matrix in sparse format</td>
</tr>
<tr>
<td>15</td>
<td>KG</td>
<td>Assembles unconstrained system initial-stress (geometric) stiffness matrix in sparse format</td>
</tr>
<tr>
<td>12</td>
<td>INV</td>
<td>Factors assembled system matrices</td>
</tr>
<tr>
<td>4</td>
<td>EQNF</td>
<td>Computes equivalent joint loading associated with thermal, dislocational, and pressure loading</td>
</tr>
<tr>
<td>13</td>
<td>SSOL</td>
<td>Computes displacements and reactions due to loading applied at joints</td>
</tr>
<tr>
<td>14, 16</td>
<td>GSF</td>
<td>Generates element stresses and internal loads</td>
</tr>
<tr>
<td></td>
<td>PSF</td>
<td>Prints information generated by GSF processor</td>
</tr>
<tr>
<td>17</td>
<td>EIG</td>
<td>Solves linear vibration and bifurcation buckling eigenproblems</td>
</tr>
<tr>
<td></td>
<td>DR</td>
<td>Performs dynamic response analysis</td>
</tr>
<tr>
<td></td>
<td>SYN</td>
<td>Produces mass and stiffness matrices for systems comprised of interconnected substructures</td>
</tr>
<tr>
<td></td>
<td>STRP</td>
<td>Computes eigenvalues and eigenvectors of substructured systems</td>
</tr>
<tr>
<td>4</td>
<td>AUS</td>
<td>Performs array of matrix arithmetic functions and is used in construction, editing, and modification of data sets</td>
</tr>
<tr>
<td></td>
<td>DCU</td>
<td>Performs array of data management functions including display of table of contents, data transfer between libraries, changing data set names, printing data sets, and transferring data between libraries and sequential files</td>
</tr>
<tr>
<td></td>
<td>VPRT</td>
<td>Performs editing and printing of data sets which are in form of vectors on data libraries</td>
</tr>
<tr>
<td></td>
<td>PLTA</td>
<td>Produces data sets containing plot specifications</td>
</tr>
<tr>
<td></td>
<td>PLTB</td>
<td>Generates graphical displays which are specified by PLTA processor</td>
</tr>
</tbody>
</table>

*This processor can operate in an update mode in the repeatable part. (See sec 3.1 in ref. 5.)
### TABLE V. NONREPEATABLE (N) AND REPEATABLE (R) PARTS IN FINITE ELEMENT ANALYSIS BASED ON DISPLACEMENT METHOD

<table>
<thead>
<tr>
<th>Number</th>
<th>Operation</th>
<th>Cross-sectional dimensions</th>
<th>Nodal coordinates</th>
<th>Connectivity</th>
<th>Element type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Define material properties</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>2</td>
<td>Define coordinates of nodes</td>
<td>N</td>
<td>R</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>3</td>
<td>Define each node's degrees of freedom</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>4</td>
<td>Define loads</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>5</td>
<td>Define types of elements</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>R</td>
</tr>
<tr>
<td>6</td>
<td>Define cross-sectional dimensions</td>
<td>R</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>7</td>
<td>Define element-node connectivity</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>R</td>
</tr>
<tr>
<td>8</td>
<td>Compute elemental stiffness matrices</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>9</td>
<td>Compute elemental mass matrices</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>10</td>
<td>Assemble structure stiffness matrix</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>11</td>
<td>Assemble structure mass matrix</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>12</td>
<td>Decompose stiffness matrix</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>13</td>
<td>Compute displacements</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>14</td>
<td>Compute loads on elements</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>15</td>
<td>Assemble structure geometrical stiffness matrix</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>16</td>
<td>Compute stresses</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>17</td>
<td>Compute eigenvalues and eigenmodes for vibration and/or buckling</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
</tbody>
</table>

#### 4.3.1.2 Analytical calculation of gradients.
Gradients can be calculated in SPAR by using runstreams established specifically for this purpose. The general analysis capability of SPAR is augmented by this runstream to calculate structural response derivatives for static displacements and stresses. The runstream is dependent upon the types of elements used to model the structure. This method of calculating gradients for static analysis is represented in PROSSS by procedure files discussed in section 4.2.6.

#### 4.3.1.3 SPAR data storage and retrieval.
A unique feature of SPAR relevant to PROSSS organization is its set of data libraries. A group of data sets can be assembled to form a named library. Subroutines documented in reference 11 are available to store and retrieve the SPAR library data sets. These subroutines can be executed by FORTRAN CALL statements and, hence, can be used to make the SPAR data storage accessible to non-SPAR FORTRAN programs.

#### 4.3.2 Optimization Program: CONMIN

#### 4.3.2.1 Overall characteristics.
CONMIN is a general purpose, optimization subroutine library capable of solving linear or nonlinear constrained optimization problems. The basic optimization algorithm is the method of feasible directions. A user's manual describing the program and its execution options (ref. 6) explains all the control parameters used by CONMIN and the CONMIN execution modes.

In CONMIN, the objective function and the constraint functions must be continuous functions of the design variables. The design variables must also be continuous. Therefore, only these two types of optimization problems can be handled by CONMIN and, consequently, by PROSSS.
The CONMIN program organization, shown in figure 3, consists of a main program and the CONMIN subroutine library. The main program reads the initial values of design variables and CONMIN control parameters. The computation of the constraints, the objective function, and the gradients of both can be carried out in the main program if the problem is computationally small. In PROSSS, this mode of operation is used for options 2.2 and 2.3 (table I) by taking advantage of the simplicity and speed of the linear extrapolation procedure. If the problem is large, the computation of objective function, constraints, and gradients is executed externally by stopping the main program, performing the external analysis, and restarting the main program. This mode of operation is used in all other options.

Actual optimization is carried out by the CONMIN subroutine library and is controlled by a set of parameters input through the main program. The CONMIN subroutine library also contains a set of the termination criteria. The user can select a criterion from that set, and establish the numerical values associated with that criterion by means of the input control parameters.

In PROSSS, the optimizer CONMIN appears in the form of

1. subroutine CONMIN and a set of associated subroutines
2. two versions of CONMIN main programs
   (a) CONMS1, to be used with options 1.1, 1.2, 1.3
   (b) CONMS2, to be used with options 2.2, 2.3

4.3.2.2 Program: CONMS1.- The program is problem independent except for the blank common statement. (See app. D.) For a new application, the arrays in this statement must be inspected and adjusted according to reference 6, if necessary, to accommodate the problem size. The source CONMS1 must then be recompiled by the user, and the binary code is used in execution.

The program functions are

1. read CONMIN control parameters from PCNPR the first time CONMS1 is executed

Figure 3.- CONMIN program organization.
2. read analyzer output in the second and subsequent executions for file CNMNIO

3. call subroutine CONMIN

4. output new vector of design variables for the optimizer on file CNMNIO

5. stop itself to permit the external analysis

6. write and save all data needed for subsequent restarts on file PCONRST

7. generate a message on file GONOGO to indicate that the nonlinear programming (NLP) be stopped when subroutine CONMIN detects satisfaction of its termination criteria

The program card is as follows:

```plaintext
PROGRAM CONMS1 (INPUT, OUTPUT, TAPE8, TAPE7, TAPE9, TAPE11, TAPE10, TAPE5 = INPUT,
TAPE6 = OUTPUT)
```

where

<table>
<thead>
<tr>
<th>INPUT</th>
<th>PCNPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPUT</td>
<td>SAVCOUT</td>
</tr>
<tr>
<td>TAPE7</td>
<td>PCONRST</td>
</tr>
<tr>
<td>TAPE8</td>
<td>PSTRT</td>
</tr>
<tr>
<td>TAPE9</td>
<td>CNMNIO</td>
</tr>
<tr>
<td>TAPE10</td>
<td>PASS</td>
</tr>
<tr>
<td>TAPE11</td>
<td>GONOGO</td>
</tr>
</tbody>
</table>

4.3.2.3 Program: CONMS2.- The foregoing description of the program CONMS1 applies to CONMS2 with the following differences. (See app. D.)

The program functions are

1. read CONMIN control parameters from file PCNPR

2. read data for linear extrapolation analysis from file BLOCK

3. call subroutine CONMIN

4. execute the linear extrapolation analysis

5. repeat functions 3 and 4 until two changes in the objective function, obtained by comparing results of three consecutive linear stages, are smaller than a prescribed limit and generate a message to CCL that the piecewise linear programming (PLP) outer loop should be stopped
The program card is as follows:

PROGRAM CONMS2 (INPUT,OUTPUT,TAPE7,TAPE8,TAPE9,TAPE10,TAPE11,TAPE5 = INPUT, TAPE6 = OUTPUT)

where

<table>
<thead>
<tr>
<th>INPUT</th>
<th>PCNPR</th>
<th>Same as CONMS1</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPUT</td>
<td>SAVCOUT</td>
<td></td>
</tr>
<tr>
<td>TAPE7</td>
<td>PSTRT</td>
<td></td>
</tr>
<tr>
<td>TAPE8</td>
<td>BLOCK</td>
<td></td>
</tr>
<tr>
<td>TAPE9</td>
<td>CNT</td>
<td></td>
</tr>
<tr>
<td>TAPE10</td>
<td>PASS</td>
<td>Same as CONMS1</td>
</tr>
<tr>
<td>TAPE11</td>
<td>GONOGO</td>
<td></td>
</tr>
</tbody>
</table>

4.3.3 Front Processor Program: FPROC

This program does not exist in the PROSSS until it is coded by the user for a specific problem. The front processor program must be compiled by the user, and the binary code must be stored to be used in the PROSSS execution. The name of the binary file must be supplied in the input file to PROSCRS. (See app. D for an example of a front processor program.)

The program functions are

1. read the design variables output by CONMIN from file CNMNIO
2. calculate behavior variables using the design variables on file CNMNIO
3. output these behavior variables in a format meaningful for SPAR. (See SPFPOUT in sec 4.4.3.)

The program card is as follows:

PROGRAM FPROC (INPUT,OUTPUT,TAPE7,TAPE5 = INPUT,TAPE6 = OUTPUT)

where

<table>
<thead>
<tr>
<th>FPROC</th>
<th>program name</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPUT</td>
<td>CNMNIO</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>SPFPOUT</td>
</tr>
<tr>
<td>TAPE7</td>
<td>CONS</td>
</tr>
</tbody>
</table>
4.3.4 End Processor Program: EPROC

This program must also be coded by the user. (See app. D.) The end processor program must be compiled by the user, and the binary code must be stored to be used in the PROSSS execution. The name of the binary file must be supplied in the input file to PROSCRS.

The program functions are

1. read user-supplied constants defining limits imposed on the behavior variables from file ENDN; these limits are to be used for the constraint function evaluations

2. retrieve the SPAR output data sets (behavior variables and, optionally, their gradients) from the SPAR libraries and store them in arrays

3. extract from these arrays the design variables to be used for the constraint function evaluations and for computing the objective function

4. evaluate the constraints and the objective function and, optionally, their gradients

5. output these quantities in NAMELIST form on file CNMNIO to be read by CONMS1 or CONMS2 main programs (see the LINKE NAMELIST in app. D)

The SPAR libraries are in a special binary format accessible only by two FORTRAN callable subroutines, DAL and FIN, which are used as standard parts of the end processor code to carry out program function 2. These subroutines are documented in reference 11. Both routines are used in SPAR and must be retrieved from the SPAR relocatable subroutines and loaded with the end processor.

The end processor program card is as follows:

PROGRAM EPROC (INPUT,TAPE6,TAPE8,TAPE5 = INPUT)

where

EPROC     program name
TAPE5     ENDN
TAPE6     CNMNIO
TAPE8     BLK

4.3.5 Control Programs

There are four problem independent control codes: EVALS, evaluates the finite difference gradients; FSDS, incorporates the fully stressed design technique; SELECTS, selects active or violated constraints; and RERITES, rewrites the design variables to a new file.
4.3.5.1 Program to compute the finite difference gradients: EVALS.—Its functions are

1. read CONMIN control parameters supplied by user from file PCNPR
2. read the design variables from file PSTRT
3. stop itself to allow execution of the front processor/SPAR/end processor sequence
4. restart - read (from file CNMNIO) and store the objective function and constraint values
5. increment one design variable
6. restart - reset the design variable incremented in step 5 to its original value
7. using the stored objective function and constraint values, compute the finite difference gradients of these functions for the design variable
8. repeat steps 4 to 7 until all variables have been incremented

The program card is as follows:

```
PROGRAM EVALS (INPUT, OUTPUT, TAPE5 = INPUT, TAPE7, TAPE8, TAPE9, TAPE10, TAPE11)
```

where

- TAPE5 PCNPR
- TAPE7 PASS
- TAPE8 PSTRT
- TAPE9 CNMNIO
- TAPE10 BLOCK
- TAPE11 CHECK

See appendix D for a sample listing.

4.3.5.2 Fully stressed design (FSD) program: FSDS.—The program uses a fully stressed design technique to modify the initial design variables. The program functions are

1. read control parameters supplied by user from file PCNPR
2. read the design variables from file PSTRT
3. stop itself to allow execution of the front processor/SPAR/end processor sequence
4. restart - read values of the stress constraints from file BLOCK
5. change the design variables by FSD technique using the constraint values
6. repeat from 3

The program card is as follows:

PROGRAM FSDS (INPUT,OUTPUT,TAPE8,TAPE7,TAPE9,TAPE11,TAPE10,TAPE5 = INPUT, TAPE6 = OUTPUT)

All files are defined exactly the same as in the program CONMS1. (See app. D for a sample listing.) The program calls a problem independent subroutine FSDSUBS (also in app. D) that carries out FSDS function 5.

4.3.5.3 Program to select the active or violated constraints: SELECTS.- The program functions are

1. read the constraints from file BLOCK
2. determine if the constraint is less than the constraint thickness parameter for linear and side constraints and, thus, active or violated
3. if the constraint is active or violated, the analytical gradient of that constraint is stored and a pointer is set in an array to denote whether or not the constraint is active or violated

The program card is as follows:

PROGRAM SELECTS (INPUT,OUTPUT,TAPE5 = INPUT,TAPE6 = OUTPUT)

where

TAPE5 BLOCK
TAPE6 CONREST

See appendix D for a sample listing.

4.3.5.4 Program to rewrite the design variables to a different file: RERITES.- The program RERITES copies the design variables from CNMNIO to SO.

The program card is as follows:

PROGRAM RERITES (INPUT,TAPE5 = INPUT,TAPE6)

where

TAPE5 CNMNIO
TAPE6 SO

See appendix D for a sample listing.
4.3.6 Gradient Programs

There are three problem independent programs, BLDELDS, GNGRDRS, and DRVSTRS, that aid in calculating analytical gradients. The first, BLDELDS, builds a file for input into the analysis program to calculate the derivatives for the mass and stiffness matrices with respect to the design variables in the nonrepeatable part of PROSSS. The second, GNGRDRS, builds a file in the repeatable part for input into the analysis program to compute the derivatives of the stresses and of the forces and moments with respect to the design variables. The third, DRVSTRS, converts the forces and moments and the derivatives of the forces and moments into stresses and derivatives of stresses.

4.3.6.1 Program to create input file for computing gradients in the analysis program in the nonrepeatable part: BLDELDS.- BLDELDS performs the following functions:

1. Read control parameters supplied by the user from file INPT.
2. Read the input model data file for the analysis program from file NRRS.
3. Change all design variables in the file NRRS to unity and place on file RSOUT.
4. Create remainder of file RSOUT to calculate the derivative of the stiffness and mass matrices with respect to the different design variables.

The program card is as follows:

PROGRAM BLDELDS (TAPES,TAPE23,TAPE20,TAPE21,TAPE22,OUTPUT)

where

TAPE5 INPT
TAPE23 NRRS (after editing)
TAPE20 RSOUT
TAPE21 Scratch file
TAPE22 Scratch file
OUTPUT Output file for error messages

See appendix D for a sample listing.

4.3.6.2 Program to create input file for computing gradients in the analysis program in the repeatable part: GNGRDRS.- GNGRDRS performs the following functions:

1. Reads control parameters supplied by user from file INPT.
2. Creates the runstream to calculate the derivatives of the stiffness and the mass matrices with respect to the design variables when an element (such as a beam or plate) has more than one contributing factor. The program calls user-supplied subroutine(s) with any of the following names DKDVE21, DKDVE22, DKDVE33, and/or
DKDVE43. These subroutines compute the derivatives of the stiffness matrix with respect to a design variable for a particular element type in SPAR (e.g., E21 or E43). The name(s) used depends upon the elements used in the finite element model. If a subroutine is not used, it remains unsatisfied. Two integer parameters used in naming the created data sets are passed to each subroutine. The first is the counter for the design variable and the second is the number of unconstrained degrees of freedom (from 1 to 6). A listing of a sample DKDVE21 subroutine is shown in appendix D.

The subroutine card for DKDVE21 is as follows:

SUBROUTINE DKDVE21(NDVJIM,NDF)

where

NDVJIM
the number of the design variables (first, second, third, etc.) for which the derivative of the stiffness and mass matrices with respect to the design variable is being calculated

NDF
number of degrees of freedom per joint squared

3. Create the remainder of the runstream and place on file RSOUT.

The program card is as follows:

PROGRAM GNGRDRS (INPUT,TAPE30,TAPE31,TAPE10,TAPE5 = INPUT,OUTPUT)

where

TAPE5
INPT

TAPE30
CONS

TAPE31
CNMNIO

TAPE10
RSOUT

OUTPUT
Output file for error messages

See appendix D for listing of GNGRDRS.

4.3.6.3 Program to convert forces and moments and the derivatives of forces and moments to stresses and stress derivatives: DRVSTRS.- DRVSTRS performs the following functions:

1. Reads control parameters supplied by user from file INPT.

2. Reads forces and moments and the derivatives of forces and moments from file SPARLD, a library created by SPAR.

3. Computes stresses and stress derivatives using a user-supplied subroutine named BMSTRS (for beams) or PLTSTRS (for plates). As before, if a name is not used, it remains unsatisfied. Four integer parameters are passed to BMSTRS. They are (1) a switch and (2) a counter, so certain computations can be skipped if they are not needed; (3) a block counter for accessing an array; and (4) another switch to
determine if the beam is the contributing factor to the stress derivative. The first three parameters are also passed to PLTSTRS. A listing of a sample BMSTRS subroutine is provided in appendix D.

The subroutine card for BMSTRS is as follows:

```
SUBROUTINE BMSTRS (ISW,KCNT,JCNT,IBEAM)
```

where

- **ISW, KCNT** a switch and a counter used to store certain beam data (e.g., moments of inertia) and make certain computations only on the first time BMSTRS is called from DRVSTRS
- **JCNT** a counter to find the location in blank common for various data items, depends on the number of beam elements
- **IBEAM** 1, if beam is a contributing factor to stress derivatives; 0, otherwise

4. The stress and stress derivatives are written on SPARLD for processing by the end processor EPROC (sec 4.3.4).

The program card is as follows:

```
PROGRAM DRVSTRS (INPUT,TAPE30,TAPE31,TAPE5 = INPUT,TAPE15,TAPE16,OUTPUT, TAPE6 = OUTPUT)
```

where

- **TAPE5** INPT
- **TAPE30** CONS
- **TAPE31** CNMNIO Used in user-supplied subroutine
- **TAPE15** Scratch file
- **TAPE16** Scratch file
- **OUTPUT** Output file for error messages

See appendix D for a sample listing of DRVSTRS.

4.4 Data Files

There are five types of data files in PROSSS: input, model, transfer, edit, and save. Sample annotated listings of each file are shown in appendix E. Table VI
<table>
<thead>
<tr>
<th>File name</th>
<th>Type</th>
<th>Created by</th>
<th>Storage</th>
<th>Problem</th>
<th>Program(s) using file</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCONPAR, CONPAR</td>
<td>Input</td>
<td>User</td>
<td>Permanent</td>
<td>Dependent</td>
<td>CONMS1,CONMS2, EVALS</td>
<td>Initializes values for optimization program</td>
</tr>
<tr>
<td>PSTART STARTX</td>
<td>Input</td>
<td>User</td>
<td>Permanent</td>
<td>Dependent</td>
<td>CONMS1,CONMS2</td>
<td>Initializes design variables for optimization program</td>
</tr>
<tr>
<td>PASS</td>
<td>Input</td>
<td>EDPASS1, EDPASS2</td>
<td>Local</td>
<td>Independent</td>
<td>CONMS1,EVALS</td>
<td>Test variable for first pass through system. Created by EDPASS1 file.</td>
</tr>
<tr>
<td>INPT</td>
<td>Input</td>
<td>User</td>
<td>Permanent</td>
<td>Dependent</td>
<td>BLDELDS,GNGRDRS, DRVSTRS</td>
<td>Initializes values for gradient calculation</td>
</tr>
<tr>
<td>CNT</td>
<td>Input</td>
<td>User</td>
<td>Permanent</td>
<td>Dependent</td>
<td>CONMS1,CONMS2</td>
<td>Objective functions and tolerance for termination test</td>
</tr>
<tr>
<td>ENDS</td>
<td>Input</td>
<td>User</td>
<td>Permanent</td>
<td>Dependent</td>
<td>EPROC</td>
<td>Defines limits imposed on design variables in end processor</td>
</tr>
<tr>
<td>CONS</td>
<td>Input</td>
<td>User</td>
<td>Permanent</td>
<td>Dependent</td>
<td>GNGRDRS,DRVSTRS</td>
<td>Defines certain constants used in analytical gradient calculation</td>
</tr>
<tr>
<td>NRRS</td>
<td>Model</td>
<td>User</td>
<td>Permanent</td>
<td>Dependent</td>
<td>SPAR</td>
<td>Nonrepeatable input runstream to analysis program</td>
</tr>
<tr>
<td>SPARRS</td>
<td>Model</td>
<td>User</td>
<td>Permanent</td>
<td>Dependent</td>
<td>SPAR</td>
<td>Repeatable input runstream to analysis program</td>
</tr>
<tr>
<td>PCONRST</td>
<td>Transfer Programs</td>
<td>Local</td>
<td>Dependent</td>
<td>CONMS1,CONMS2, SELECTS</td>
<td>Data saved for subsequent passes through optimization program. Created in optimization program</td>
<td></td>
</tr>
<tr>
<td>BLOCK, BLK</td>
<td>Transfer Programs</td>
<td>Local</td>
<td>Dependent</td>
<td>CONMS1,CONMS2, EVALS, EPROC, SELECTS</td>
<td>Contains objective function and constraint data</td>
<td></td>
</tr>
<tr>
<td>PCMN0IO, CMMNIO</td>
<td>Transfer Programs</td>
<td>Local</td>
<td>Dependent</td>
<td>FFPROC,CONMS1, CONMS2, GNGRDRS, DRVSTRS, EPROC, RERITES, EVALS</td>
<td>Transfer design variable, objective function and constraint data between optimization program and front and end processors</td>
<td></td>
</tr>
<tr>
<td>CHECK</td>
<td>Transfer Programs</td>
<td>Local</td>
<td>Dependent</td>
<td>EVALS</td>
<td>Created when analysis has been performed for each design variable combination</td>
<td></td>
</tr>
<tr>
<td>GGN0GO</td>
<td>Transfer Programs</td>
<td>Local</td>
<td>Dependent</td>
<td>CONMS1,CONMS2</td>
<td>Created to terminate PROSSS</td>
<td></td>
</tr>
<tr>
<td>SPFPOUT</td>
<td>Transfer Program</td>
<td>Local</td>
<td>Dependent</td>
<td>FFPROC,SPAR</td>
<td>Contains updated design variables for input into analysis program</td>
<td></td>
</tr>
<tr>
<td>ROUT</td>
<td>Transfer Program</td>
<td>Local</td>
<td>Dependent</td>
<td>BLDELDS,GNGRDRS, SPAR</td>
<td>Input runstream to analysis program</td>
<td></td>
</tr>
<tr>
<td>SC</td>
<td>Transfer Program</td>
<td>Local</td>
<td>Dependent</td>
<td>EVALS, RERITES</td>
<td>Design variable information</td>
<td></td>
</tr>
<tr>
<td>SPARLA, SPARLB, SPARLC, SPARLD</td>
<td>Transfer Program</td>
<td>Local</td>
<td>Dependent</td>
<td>SPAR, DRVSTRS, EPROC</td>
<td>Data library created by analysis program</td>
<td></td>
</tr>
<tr>
<td>EDPASS1</td>
<td>Edit</td>
<td>External</td>
<td>Permanent</td>
<td>Independent</td>
<td>None</td>
<td>Initializes PASS variable to 1</td>
</tr>
<tr>
<td>EDPASS2</td>
<td>Edit</td>
<td>External</td>
<td>Permanent</td>
<td>Independent</td>
<td>None</td>
<td>Reinitializes PASS variable to 1</td>
</tr>
<tr>
<td>EDIT1</td>
<td>Edit</td>
<td>External</td>
<td>Permanent</td>
<td>Independent</td>
<td>None</td>
<td>Removes all but element connection data from NRRS file</td>
</tr>
<tr>
<td>EDIT2</td>
<td>Edit</td>
<td>External</td>
<td>Permanent</td>
<td>Independent</td>
<td>None</td>
<td>Prepares nonrepeatable RROUT file for input into analysis program</td>
</tr>
<tr>
<td>MERGFP</td>
<td>Edit</td>
<td>External</td>
<td>Permanent</td>
<td>Independent</td>
<td>None</td>
<td>Merges SPFPOUT file into SPARRS file</td>
</tr>
<tr>
<td>EDGDRUS</td>
<td>Edit</td>
<td>External</td>
<td>Permanent</td>
<td>Independent</td>
<td>None</td>
<td>Prepares repeatable RROUT file for input into analysis program</td>
</tr>
<tr>
<td>SAVCOUT</td>
<td>Save</td>
<td>Program</td>
<td>Permanent</td>
<td>Dependent</td>
<td>CONMS1,CONMS2</td>
<td>Cumulative list of data output from optimization program</td>
</tr>
<tr>
<td>NRLA</td>
<td>Save</td>
<td>Program</td>
<td>Permanent</td>
<td>Dependent</td>
<td>SPAR</td>
<td>Data library output from nonrepeatable analysis program</td>
</tr>
<tr>
<td>SAVSPLD</td>
<td>Save</td>
<td>Program</td>
<td>Permanent</td>
<td>Dependent</td>
<td>SPAR, DRVSTRS</td>
<td>Data library output from repeatable analysis program</td>
</tr>
</tbody>
</table>
provides a quick reference for each data file by providing the following information about each file:

1. the name of the file
2. the type of file (e.g., input or model)
3. whether the file is created by the user, program, or external to PROSSS; files created external to PROSSS, are created one time, stored on a permanent file, and then accessed by any user independent of the application
4. whether the file is local or permanent
5. whether the file is problem dependent or independent
6. the programs that use this file
7. a brief comment about the file's function

4.4.1 Input Data Files

There are seven input files that primarily provide initialization data to the various programs in PROSSS. (See app. E for sample listings.) These files are as follows:

1. PCNPR (CONPAR) initializes variables used in the optimization program. (Note: PCNPR is the name used in the generalized input files. See app. A.) It is replaced by the specific problem dependent file name before the option is used. CONPAR is the local file name into which the specific file is stored. It is in NAMELIST format with the NAMELIST name CONPAR. A description of the variables can be found in reference 6.

2. PSTRT (STARTX) is also read by the optimization program to initialize the design variables. (See note about PCNPR and CONPAR.) It, too, is in NAMELIST format with the NAMELIST name STARTX.

3. PASS contains one number. In the initialization process, the number is set to 1 using the EDPASS1 file in section 4.4.4, signifying the first pass through the system. After the first pass, the value is changed to 2 and different paths are taken in the programs using PASS as an input variable. Options 1.2 and 2.2 (table I) reset the value to 1 within the system using the EDPASS2 file also described in section 4.4.4. PASS is also in NAMELIST format with the NAMELIST name PASSAGE.

4. INPT defines values used in analytical gradient calculation. INPT has two card types. The first type has the format (6(1X,I4),1X,A4), consists of one card, and contains the following variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOLC</td>
<td>number of load cases</td>
</tr>
<tr>
<td>NODV</td>
<td>number of design variables</td>
</tr>
<tr>
<td>ISNOLC</td>
<td>new starting number for load cases with derivatives</td>
</tr>
<tr>
<td>JOINTS</td>
<td>number of joints in the model</td>
</tr>
</tbody>
</table>
NDF  number of degrees of freedom
NOEL  number of different element types in the model
VORB  a four character word to determine the type of 
analysis (e.g., STAT, VIBR, BUCK)

The second type has the format (6(1X, A3, 1X, I3, 1X, I3)), consists of one or 
more cards, and contains the following variables (one set for each element):

EL(I)  element type (e.g., E21, E43)
NSECT(I)  largest section property number used for that element type
NODVPE(I)  number of design variables per element type

5. CNT is a file that initializes parameters for terminating the optimization 
process. CNT is in NAMELIST form with the NAMELIST name CNT.

6. ENDN is a file that defines limits imposed on the design variables in the end 
processor. ENDN is in NAMELIST format with the NAMELIST name EPIN.

7. CONS is a file supplied by the user to define certain constants such as the 
cross-sectional dimensions of a beam used in analytical gradient calculation. Since 
the file is only read by user-supplied subroutines, the format is also defined by the 
user. See section 4.3.6 for more information.

4.4.2 Model Data Files

Model data files contain finite-element-model input data for the analysis 
program SPAR. Two model data files, called SPAR runstreams, are used in PROSSS. The 
first, NRRS, defines the model in terms of nodes and elements connecting those 
nodes. This file is only used in the nonrepeatable part of PROSSS. The second model 
file, NGRS (no gradients) or RGS (gradients), updates the data for elements used as 
design variables and contains data for performing the analysis in the repeatable part 
of PROSSS. Sample listings of NRRS, NGRS, and RGS are contained in appendix E.

4.4.3 Transfer Data Files

Transfer data files pass data among the various programs in PROSSS. These are 
local files created within the system, and they are returned when the analysis- 
optimization process is completed. The following transfer files exist in PROSSS:

1. PCONRST is a file in NAMELIST format with the NAMELIST name SAVE that takes 
initial data from PCNPR and updates these data in each new pass through the 
optimization program. In two options from table I (2.2 and 2.3), the BLOCK file 
described next) is used in place of PCONRST.

2. BLOCK or BLK is a file in NAMELIST format with the NAMELIST name BLK and 
contains objective function and constraint data.
3. PCNMNIO or TCNMNIO, CNMNIO (see note on PCNPR in section 4.4.1) is a file in NAMELIST format with two NAMELIST names. Name LINKE contains the number of design variables and their values. It is written by the end processor and read by the optimization program. Name LINKF contains objective function and gradient information. It is written by the optimization program and read by the front and end processors. Two options (2.2 and 2.3) replace PCNMNIO with CNT file discussed in section 4.4.1.

4. CHECK is created in options 1.2 and 2.2 when analysis has been performed for each design variable and their combination, if this file exists (i.e., local), then the optimization program is executed.

5. GONOGO is created when the objective function has not changed more than a specified tolerance (see CNT file in section 4.4.1) in three passes. If this file exists (i.e., local), the process is terminated.

6. SPFPOUT is created by the front processor and contains the updated design variables. This file is merged into the NGRS or RGS file using the MERGFP files described in section 4.4.4. The NGRS and RGS files are described in section 4.4.2.

7. RSOUT is an input file (runstream) for the analysis program similar to those found in section 4.4.2. RSOUT is created twice in each of two options (2.2 and 2.3), once in the nonrepeatable part and once in the repeatable part, and is used in calculating the analytical gradients.

8. SO is identical to the PSTRT file discussed in section 4.4.1.

9. SPARLA, SPARLB, SPARLC, SPARLD are data libraries created by the analysis program.

Sample listings of the transfer files are presented in appendix E. Files CHECK, GONOGO, SO, SPARLA, SPARLB, SPARLC, and SPARLD are not listed, either because of their length or because of their binary format.

4.4.4 Edit Data Files

Edit data files contain Text Editor (ref. 10) commands primarily for editing files input to the analysis program. None are created by the user. The following edit data files exist in PROSSS:

1. EDPASS1 creates the PASS file described in section 4.4.1 and initializes the variable to 1.

2. EDPASS2 reinitializes the number in the PASS file to 1.

3. EDIT1 edits the NRRS file described in section 4.4.2 to remove all but the element connection data.

4. EDIT2 edits the RSOUT file in the nonrepeatable part described in section 4.4.3 to prepare it for input to the analysis program.

5. MERGFP merges the SPFPOUT file described in section 4.4.3 into the NGRS or RGS files described in section 4.4.2.
6. EDGRDS edits the RSOUT file in the repeatable part described in section 4.4.3 to prepare it for input to the analysis program.

Sample listings of edit data files are contained in appendix E.

4.4.5 Saved Data Files

Three files are automatically saved for listing (SAVCOUT) and postprocessing or restart purposes (NRLA, SAVSPLD). File SAVCOUT contains a cumulative list of all the information output from the optimization program. File NRLA is a library of data output by the analysis program in the nonrepeatable part of the process. File SAVSPLD is a library of objective function, stress, and/or stress derivative information output by the analysis program in the nonrepeatable part of the process. File SAVSPLD is a library of objective function, stress, and/or stress derivative information output by the analysis program in the repeatable part of the process. These files are either too long or in a binary format and, therefore, are not listed in appendix E.

5 SAMPLE EXECUTIONS OF PROSSS

Each option was executed, using the input files shown in appendix A, to determine the final objective function (min. mass, kg) of the finite element model of the fuselage shown in figure 4. The model is composed of 80 joints, 58 rods, 76 beams, and 56 membranes. There are 352 degrees of freedom. The three design variables are the cross-sectional area of the transverse stringers (beams), the

![Figure 4.- Fuselage model used for testing.](image-url)
cross-sectional area of the longitudinal stringers (rods), and the thickness of the panels (membranes). The design variables are handled by the optimizer in reciprocal form to improve the convergence. Their initial reciprocal starting values are 0.05 cm$^2$, 1.0 cm$^2$, and 4.0 cm, respectively. The initial objective function is 5460.12 kg. Shown in table VII is a comparison of actual results (not reciprocals)

<table>
<thead>
<tr>
<th>Option (table I)</th>
<th>1.1</th>
<th>1.2</th>
<th>1.3</th>
<th>2.2</th>
<th>2.3</th>
<th>FSD 1.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective function, minimized mass, kg</td>
<td>6501.75</td>
<td>6398.91</td>
<td>6574.37</td>
<td>6350.10</td>
<td>6338.20</td>
<td>6362.39</td>
</tr>
<tr>
<td>Design variables: Cross-sectional area, cm$^2$</td>
<td>4.7364</td>
<td>8.4246</td>
<td>12.1567</td>
<td>1.2088</td>
<td>1.6722</td>
<td>1.0</td>
</tr>
<tr>
<td>Thickness, cm</td>
<td>1.6333</td>
<td>1.5713</td>
<td>1.5563</td>
<td>1.5706</td>
<td>1.6215</td>
<td>1.6284</td>
</tr>
<tr>
<td>Run cost, $</td>
<td>121</td>
<td>127</td>
<td>98</td>
<td>68</td>
<td>31</td>
<td>20</td>
</tr>
</tbody>
</table>

for each option. All final objective functions and design variables are within reasonable limits. The piecewise linear approach is 50 to 75 percent less expensive to execute than the nonlinear approach.

Langley Research Center
National Aeronautics and Space Administration
Hampton, VA 23665
September 11, 1981
APPENDIX A

SAMPLE SETUP OF PROSSS FOR A SPECIFIC OPTION

This section contains listings of the source file PROSCRS, sample input files for each option, and the control card and edit files created by the binary file PROSCRS.

PROSCRS

PROGRAM PROSCRS(TAPE8,TAPE9,TAPE10,TAPE11)

C
C THIS PROGRAM CREATES A SPECIFIC OPTION PROCEDURE
C FILE FROM A GENERAL ONE.
C
C THE SPECIFIC NAMES ARE READ IN FROM UNIT 8.
C
READ(8,10)A1,N1
10 FORMAT(A7,3X,I2)

C WRITE CONTROL CARDS ON UNIT 9
C
WRITE(9,20)
20 FORMAT(*.PROC,PROSSS,**GET,PROSOPT/UN=753437N.*
IF(N1.EQ.11) GO TO 40
IF(N1.EQ.12) N=1
IF(N1.EQ.13) N=2
IF(N1.EQ.22) N=3
IF(N1.EQ.23) N=4
WRITE(9,30) N
30 FORMAT(*COPYBR,PROSOPT,DUMMY,*I1,*.*)
40 WRITE(9,50)
50 FORMAT(*COPYBR,PROSOPT,OPTION,**REWIND,OPTION,EDOPTC,EDOPT.*)
WRITE(9,55)
55 FORMAT(*EDIT,EDOPT,EDOPTC,EDOUT,**REWIND,EDOPT.*)
WRITE(9,60)
60 FORMAT(*EDIT,OPTION,EDOPT,EDOUT.*)
WRITE(9,70)
70 FORMAT(*RETURN,EDOPTC,PROSOPT,DUMMY,EDOPT,PROSCRB,PROSSIN.*)
WRITE(9,80) N1
80 FORMAT(*REWIND,OPTION,**BEGIN,OPT*,I2,*,OPTION.*)
C
C WRITE EDIT COMMANDS ON UNIT 10.
C
WRITE(10,100) A1,N1
100 FORMAT(*RS/*,A7,*/,**,I2,*/;100*)
110 READ(8,120) A1,A2
120 FORMAT(A7,3X,A7)
130 IF(EOF(8)) 150,130
140 WRITE(10,140) A1,A2
140 FORMAT(*RS/*,A7,*/,**,A7,*/;100*)
GO TO 110

31
C
C REMOVE BLANKS IN EDIT COMMANDS
C
150 WRITE(11,160)
160 FORMAT(*RS1/ /;100**END*)
       WRITE(10,165)
165 FORMAT(*END*)
       STOP
       END
APPENDIX A

SAMPLE INPUT FILES FOR OPTIMIZATION OPTIONS (TABLE I)

Option 1.1 with nonrepeatable part

<table>
<thead>
<tr>
<th>General</th>
<th>Specific</th>
</tr>
</thead>
<tbody>
<tr>
<td>POPT</td>
<td>11</td>
</tr>
<tr>
<td>NONREPT</td>
<td>1</td>
</tr>
</tbody>
</table>

Nonrepeatable part fully stressed

NRRS  | NRSRPS
FUSD  | 0
CONMIN| CONMB1
ENDP1 | FPFB1
FRNT  | FPFB1
ENDN  | FPFB1
PCNPRI| CONP1
PSRT  | STRP3
CONS  | CONS
NGRS  | RRSNG
PCONNST| CONREST
PCNMNIO| CNMNIO
SAVCOUT| REPCOUT
NSPARLA| NRLANG
FLENGTH| 100000
BLK   | BLOCK
SAVSPLD| REPSPLD

Option 1.1 with fully stressed design

<table>
<thead>
<tr>
<th>General</th>
<th>Specific</th>
</tr>
</thead>
<tbody>
<tr>
<td>POPT</td>
<td>11</td>
</tr>
<tr>
<td>NONREPT</td>
<td>0</td>
</tr>
<tr>
<td>FUSD</td>
<td>1</td>
</tr>
</tbody>
</table>

Use fully stressed design

FSDSUB| FSDSUB
CONMIN| FSB
ENDN  | EPFUIN
PCNPRI| CONP1
CONS  | CONS
NGRS  | RRSNG
PCONNST| CONREST
PCNMNIO| CNMNIO
SAVCOUT| REPCOUT
NSPARLA| NRLANG
FLENGTH| 100000
BLK   | BLOCK
SAVSPLD| REPSPLD

Option 1.2

<table>
<thead>
<tr>
<th>General</th>
<th>Specific</th>
</tr>
</thead>
<tbody>
<tr>
<td>POPT</td>
<td>12</td>
</tr>
<tr>
<td>NONREPT</td>
<td>0</td>
</tr>
<tr>
<td>FUSD</td>
<td>0</td>
</tr>
</tbody>
</table>

CONMIN| CONMB1
ENDP1 | EPFB1
FRNT  | FPFG1
ENDN  | EPFUIN
PCNPRI| CONP1
PSRT  | STRP3
CONS  | CONS
NGRS  | RRSNG
PCONNST| CONREST
PCNMNIO| CNMNIO
SAVCOUT| REPCOUT
NSPARLA| NRLANG
FLENGTH| 100000
BLK   | BLOCK
SAVSPLD| REPSPLD

Option 1.3

<table>
<thead>
<tr>
<th>General</th>
<th>Specific</th>
</tr>
</thead>
<tbody>
<tr>
<td>POPT</td>
<td>13</td>
</tr>
<tr>
<td>NONREPT</td>
<td>0</td>
</tr>
<tr>
<td>FUSD</td>
<td>0</td>
</tr>
</tbody>
</table>

CONMIN| CONMB1
ENDP1 | FPFB1
ENDP2 | EPFB1
SUBS  | 1
BINDEPB| BINDEPB
FRNT  | FPFB1
ENDN  | EPFUIN
PCNPRI| CONP13
PSRT  | STRP3
INPT   | INPT
CONS  | CONS
NGRS  | RRSNG
RGS   | RRSNG
PCONNST| CONREST
PCNMNIO| CNMNIO
SAVCOUT| REPCOUT
NSPARLA| NRLANG
FLENGTH| 100000
BLK   | BLOCK
SAVSPLD| REPSPLD

Option 2.2

<table>
<thead>
<tr>
<th>General</th>
<th>Specific</th>
</tr>
</thead>
<tbody>
<tr>
<td>POPT</td>
<td>22</td>
</tr>
<tr>
<td>NONREPT</td>
<td>1</td>
</tr>
<tr>
<td>FUSD</td>
<td>0</td>
</tr>
</tbody>
</table>

CONMIN| CONMB2
ENDP1 | EPFG1
ENDN  | EPFUIN
PCNPRI| CONP2
PSRT  | STRP4
CONS  | CONS
CNT   | CNT
NGRS  | RRSNG
PCONNST| BLOCK
PCNMNIO| CNT
SAVCOUT| REPCOUT
NSPARLA| NRLANG
FLENGTH| 100000
BLK   | BLOCK
SAVSPLD| REPSPLD

Option 2.3

<table>
<thead>
<tr>
<th>General</th>
<th>Specific</th>
</tr>
</thead>
<tbody>
<tr>
<td>POPT</td>
<td>23</td>
</tr>
<tr>
<td>NONREPT</td>
<td>0</td>
</tr>
<tr>
<td>FUSD</td>
<td>0</td>
</tr>
</tbody>
</table>

CONMIN| CONMB2
ENDP1 | EPFG1
ENDN  | EPFUIN
PCNPRI| CONP2
PSRT  | STRP4
INPT   | INPT
CONS  | CONS
CNT   | CNT
NGRS  | RRSNG
PCONNST| BLOCK
PCNMNIO| CNT
SAVCOUT| REPCOUT
NSPARLA| NRLANG
FLENGTH| 100000
BLK   | BLOCK
SAVSPLD| REPSPLD

Option 2.2

<table>
<thead>
<tr>
<th>General</th>
<th>Specific</th>
</tr>
</thead>
<tbody>
<tr>
<td>POPT</td>
<td>22</td>
</tr>
<tr>
<td>NONREPT</td>
<td>1</td>
</tr>
<tr>
<td>FUSD</td>
<td>0</td>
</tr>
</tbody>
</table>

CONMIN| CONMB2
ENDP1 | EPFG1
ENDN  | EPFUIN
PCNPRI| CONP2
PSRT  | STRP4
INPT   | INPT
CONS  | CONS
CNT   | CNT
NGRS  | RRSNG
PCONNST| BLOCK
PCNMNIO| CNT
SAVCOUT| REPCOUT
NSPARLA| NRLANG
FLENGTH| 100000
BLK   | BLOCK
SAVSPLD| REPSPLD

Option 2.3

<table>
<thead>
<tr>
<th>General</th>
<th>Specific</th>
</tr>
</thead>
<tbody>
<tr>
<td>POPT</td>
<td>23</td>
</tr>
<tr>
<td>NONREPT</td>
<td>0</td>
</tr>
<tr>
<td>FUSD</td>
<td>0</td>
</tr>
</tbody>
</table>

CONMIN| CONMB2
ENDP1 | EPFG1
ENDN  | EPFUIN
PCNPRI| CONP2
PSRT  | STRP4
INPT   | INPT
CONS  | CONS
CNT   | CNT
NGRS  | RRSNG
PCONNST| BLOCK
PCNMNIO| CNT
SAVCOUT| REPCOUT
NSPARLA| NRLANG
FLENGTH| 100000
BLK   | BLOCK
SAVSPLD| REPSPLD

33
APPENDIX A

CONTROL CARD FILE CREATED BY PROSCRB

.PROC,PROSS.
GET,PROSOPT/UN=753437N.
COPYRR,PROSOPT,OPTION.
REWIND,OPTION,EDOPTC,EDOPT.
EDIT,EDOPT,EDOPTC,EDOUT.
REWIND,EDOPT.
EDIT,OPTION,EDOPT,EDOUT.
RETURN,EDOPTC,PROSOPT,DUMMY,EDOPT,PROSCRB,PROSSIN.
REWIND,OPTION.
BEGIN,OPT11,OPTION.

EDIT FILE CREATED BY PROSCRB

(After extraneous blanks removed)

RS:/POPT /11/;100
RS:/NONRFPT/1;100
RS:/NRRS //NRSPARS/;100
RS:/FUSD /0 /;100
RS:/CONMIN //CONMB1 /;100
RS:/ENDP1 //FPFUB1 /;100
RS:/FRNT //FPFGB1 /;100
RS:/ENDN //FPFUIN /;100
RS:/PCNPR //CONP1 /;100
RS:/PSTRT //STRP3 /;100
RS:/CONS //CONS /;100
RS:/NGRS //RRSNG /;100
RS:/PCONRST//CONREST/;100
RS:/PCNMNIO//CNMNIO /;100
RS:/SAVCOUT//REPCOUT/;100
RS:/NSPARLA//NRLANG /;100
RS:/LENGTH//100000 /;100
RS:/BLK /BLOCK /;100
RS:/SAVSPLD//REPSPLD/;100
END
APPENDIX B

OPTION FILES

The following are listings and flowcharts for each of the optimization options given in table I:

Option 1.1

*PROC,OPT11.*
IFE,(NONREPT.EQ.1),NONREPEAT.
  BEGIN,PRCNRPT,PROSPRC,POPT,NRRS,FLENGTH,INPT,NSPARLA.
ENDIF,NONREPEAT.
BEGIN,PRCINIT,PROSPRC,POPT,CONMIN,ENDP1,ENDP2,SUBS,BINDEPB,FUSD,FSDSUB.
BEGIN,PRCGETF,PROSPRC,POPT,FRNT,ENDN,PCNPR,PSTRT,INPT,CONS,CNT,NGRS,RGS.
WHILE,(.NOT.(FILE(GONOGO,LO))},OPTION11.
  BEGIN,PRCOPTM,PROSPRC,PCONRST,PCNMINO,SAVCOUT.
  BEGIN,PRCFPXX,PROSPRC.
  BEGIN,PRCANAL,PROSPRC,NSPARLA,FLENGTH,SAVSPLD.
  BEGIN,PRCEPXX,PROSPRC,BLK.
ENDW,OPTION11.
BEGIN,PRCEND,PROSPRC.
APPENDIX B

Option 1.2

.PROC,OPT12.
.IFE,(NONREPT,EQ.1),NONREPEAT.
BEGIN,PRCNRT,PROSPRC,POPT,NRRS,FLENGTH,INPT,NSPARL.
ENDIF,NONREPEAT.
.SET(R2=1)
BEGIN,PRCINIT,PROSPRC,POPT,CONMIN,ENDP1,ENDP2,SUBS,BINDEPB,FUSD,FSDSUB.
BEGIN,PRCGETF,PROSPRC,POPT,FRNT,ENDN,PCNPR,PSTRT,INPT,CONS,CNT,NGRS,RGS.
BEGIN,PRCOPTM,PROSPRC,PCONRST,PCNMNIO,SAVCOUT.
BEGIN,PRCFPXX,PROSPRC.
BEGIN,PRCANAL,PROSPRC,NSPARL,FLENGTH,SAVSPLD.
BEGIN,PRCEPXX,PROSPRC,BLK.
WHILE,(.NOT.(FILE(GONOGO,LO))),OUTERLOOP.
.IFE,(.NOT.(FILE(GONOGO,LO))),ENDOUTER.
.RETURN,STARTX.
BEGIN,PRCOPTM,PROSPRC,PCONRST,PCNMNIO,SAVCOUT.
.SET(R3=1)
.IFE,(.NOT.(FILE(STARTX,LO))),SETR2.
.SET(R2=1)
ENDIF,SETR2.
WHILE,(.NOT.(FILE(CHECK,LO))),INNERLOOP.
.IFE,((.NOT.(FILE(STARTX,LO))).OR.(R3.EQ.2)),DONOTSKIP.
BEGIN,PRCFPXX,PROSPRC.
BEGIN,PRCANAL,PROSPRC,NSPARL,FLENGTH,SAVSPLD.
BEGIN,PRCEPXX,PROSPRC,BLK.
REWIND,TCMNIO,TCMNIO.
COPYBF,TCMNIO,TCMNIO.
.IFE,((FILE(GONOGO,LO)).OR.(R2.NE.2)),SKIPTOOUT.
.SKIP,ENDOUTER.
ENDIF,SKIPTOOUT.
ENDIF,DONOTSKIP.
.IFE,(R2.NE.2),RERITE.
REWIND,TCMNIO,RERITEB,SO,PASS,EDPASS2.
.SET(R2=2)
LDSET(PRESET=ZERO)
.RERITEB,TCMNIO,SO.
.EDIT,PASS,EDPASS2,EDOUT.
.RETURN,EDOUT.
ENDIF,RERITE.
.SET(R3=2)
REWIND,PASS,CONPAR,SO,TCMNIO,EVALB,BLOCK.
LDSET(PRESET=ZERO)
.EVALB,CONPAR,SO,TCMNIO,BLOCK,CHECK.
.IFE,(FILE(CHECK,LO)),SELECT.
.RETURN,CHECK.
.RFL,250000.
.REDUCE,-.
.REWIND,SELECTB,BLOCK,CONREST.
SELECTB,BLOCK,CONREST.
.RETURN,BLOCK.
APPENDIX B

REWIND,CNMNIO,TCNMNIO.
COPYBF,TCNMNIO,CNMNIO.
SKIP,ENDOUTER.
ENDIF,SELECT.
ENDW,INNERLOOP.
ENDIF,ENDOUTER.
ENDW,OUTERLOOP.
BEGIN,PRCEND,PROSPRC.
APPENDIX B

1. While file GONOGO not local
   PROCPTM
   If STARTS not local
   Yes Set R2 = 1
   No
   While file CHECK not local
   PROCPPXX PROCAML PROCPPXX
   If GONOGO local or R2 = 2
   No
   If R2 ≠ 2
   Yes R2 = 2 REMITEB
   No
   EVALB
   Edit PASS
   SELECTB
   If CHECK local
   Yes
   No
   If CHECK not local
   No
   If GONOGO not local
   Yes
   No
   PROCEND
APPENDIX B

Option 1.3

.PROC,OPT13.
.IFE,(NONREPT.EQ.1),NONREPEAT.
  BEGIN,PRCNRPT,PROSPRC,POPT,NRRS,FLENGTH,INPT,NSPARLA.
.ENDIF,NONREPEAT.
BEGIN,PRCINIT,PROSPRC,POPT,CONMIN,ENDP1,ENDP2,SUBS,BINDEPB,FUSD,FSDSUB.
BEGIN,PRCGETF,PROSPRC,POPT,FRNT,ENDN,PCNPR,PSTRT,INPT,CONS,CNT,NGRS,RGS.
WHILE,(.NOT.(FILE(GONOGO,LO))),(OPTION13).
  IFE,(RIG.EQ.2),NORETURN.
    RETURN,STARTX.
  ENDIF,NORETURN.
BEGIN,PRCOPTM,PROSPRC,PCONRST,PCNMNIO,SAVCOUT.
BEGIN,PRCFPXX,PROSPRC.
.IFE,(FILE(INFOFIL,LO)),GRADS.
  REWIND,TEMPRS,GRS,ENDG,ENDPROC.
  COPYCF,GRS,TEMPRS.
  COPYBF,ENDG,ENDPROC.
BEGIN,PRCANAL,PROSPRC,NSPARLA,FLENGTH,SAVSPLD.
BEGIN,PRCGRDS,PROSPRC,SUBS,SAVSPLD.
BEGIN,PRCEPXX,PROSPRC,blk.
ENDIF,GRADS.
.IFE,(.NOT.(FILE(INFOFIL,LO))),(NOGRADS.
  REWIND,TEMPRS,NGRS,NGEND,ENDPROC.
  COPYCF,NGRS,TEMPRS.
  COPYBF,NGEND,ENDPROC.
BEGIN,PRCANAL,PROSPRC,NSPARLA,FLENGTH,SAVSPLD.
BEGIN,PRCEPXX,PROSPRC,blk.
ENDIF,NOGRADS.
RETURN,INFOFIL.
.IFE,(.NOT.(FILE(GONOGO,LO))),(END13.
  REWIND,BLOCK,CONREST,SELECTB.
  RFL,250000.
  REDUCE,-.
LDSET(PRESET=ZERO)
SELECTB,BLOCK,,CONREST.
.ENDIF,END13.
.ENDIF,OPTION13.
BEGIN,PRCEND,PROSPRC.
PROCOPT22.
IF (NONREPT.EQ.1), NONREPEAT.
  BEGIN, PRCNRPT, PROSPRC, POPT, NRRS, FLENGTH, INPT, NSPARLA.
ENDIF, NONREPEAT.
SET (R2 = 1)
BEGIN, PRCINIT, PROSPRC, POPT, CONMIN, ENDP1, ENDP2, SUBS, BINDEPB, FUSD, FSDSUB.
BEGIN, PRCGETF, PROSPRC, POPT, FRNT, ENDN, PCNPR, PSTRT, INPT, CONS, CNT, NGRS, RGS.
WHILE, ((R2 .LT. 10).AND. (.NOT. (FILE(GONOGO,LO)))). OPTION22.
  IFE, ((R2 .LT. 10).AND. (.NOT. (FILE(GONOGO,LO)))), END22.
    REWIND, PASS, EDPASS2.
  EDIT, PASS, EDPASS2, EDOUT.
    RETURN, CHECK, EDOUT.
  ENDWHILE, .NOT. (FILE(GONOGO,LO)), KEEPON.
    REWIND, CONPAR, STARTX, PASS, CNMNIO, BLOCK, EVALB.
LDSET (PRESET=ZERO)
EVALB, CONPAR, PASS, STARTX, CNMNIO, BLOCK, CHECK.
  IFE, (FILE (CHECK, LO)), OPTIMIZE.
    BEGIN, PRCOPTM, PROSPRC, PCONRST, PCNMIN, SAVCOUT.
      SET (R2 = R2 + 1)
    SKIP, END22.
  ENDIF, OPTIMIZE.
BEGIN, PRCFPXX, PROSPRC.
BEGIN, PRCANAL, PROSPRC, NSPARLA, FLENGTH, SAVSPLD.
BEGIN, PRCFPXX, PROSPRC, BLK.
ENDWHILE, KEEPON.
ENDIF, END22.
ENDWHILE, OPTION22.
BEGIN, PRCEND, PROSPRC.
Option 2.3

*PROC,OPT23.*
IFE,(NONREPT.EQ.1),NONREPEAT.
  BEGIN,PRCNRPT,PROSPRC,POPT,NRRS,FLENGTH,INPT,NSPARLA.
ENDF,NONREPEAT.
SET(R2=1)
BEGIN,PRCINIT,PROSPRC,POPT,CONMIN,ENDP1,ENDP2,SUBS,BINDEPB,FUSD,FSDSUB.
BEGIN,PRCGETF,PROSPRC,POPT,FRNT,ENDN,PCNPR,PSTRT,INPT,CONS,CNT,NGRS,RGS.
WHILE,(.(NOT.(FILE(GONOGO,LO)))),AND,(R2.LT.10)),OPTION23.
  REWIND,STARTX,CNMNIO,CONPAR,EVALGB.
LDSET(PRESET=ZERO)
EVALGB,CONPAR,STARTX,CNMNIO.
  BEGIN,PRCFPXX,PROSPRC.
BEGIN,PRCANA1,PROSPRC,NSPARLA,FLENGTH,SAVSPLD.
IFE,((.NOT.(FILE(GONOGO,LO)))),AND,(R2.LT.10)),END23.
  BEGIN,PRCGRDS,PROSPRC,SUBS,SAVSPLD.
BEGIN,PRCEPXX,PROSPRC,BLK.
BEGIN,PRCOPTM,PROSPRC,PCONRST,PCNMNIO,SAVCOUT.
  SET(R2=R2+1)
ENDIF,END23.
ENDW,OPTION23.
REWIND,STARTX,CNMNIO,CONPAR,EVALGB.
LDSET(PRESET=ZERO)
EVALGB,CONPAR,STARTX,CNMNIO.
  BEGIN,PRCFPXX,PROSPRC.
BEGIN,PRCANA1,PROSPRC,NSPARLA,FLENGTH,SAVSPLD.
BEGIN,PRCGRDS,PROSPRC,SUBS,SAVSPLD.
BEGIN,PRCEPXX,PROSPRC,BLK.
BEGIN,PRCEND,PROSPRC.
APPENDIX C

PROCEDURE FILES

PRCNRPT

.PROC, PRCNRPT, NROPT, NRRS, FLX, I, NRLA.

*  THE PROCEDURE CREATES A SPAR LIBRARY
* FROM THE NON-REPEATABLE PART

GET, SPAR=SPAR14I, DCU=DCU14I/UN=750756N.
GET, NRRS.
PFL, FLX.
REDUCE, -.
SPAR, NRRS, NSPROUT.
REPLACE, SPARLA=NRLA.

*  TEST TO SEE IF GRADIENTS ARE REQUIRED

IF ((NROPT.EQ.13).OR. (NROPT.EQ.23)), GRADIENTS.
GET, INPT=I.
GET, EDIT1, EDIT2, BLDLB/UN=753437N.
REWIND, NRRS.

*  EDIT OUT ALL BUT ELD INPUT IN RUNSTREAM

EDIT, NRRS, EDIT1, EOUT.
REWIND, NRRS.

*  CREATE SPAR RUNSTREAM TO FIND DERIVATIVES

BLDLDB, INPT, NRRS, RSOUT.
REWIND, RSOUT.
EDIT, RSOUT, EDIT2, EOUT.
REWIND, RSOUT, SPARLA.

*  EXECUTE SPAR AGAIN TO FIND DERIVATIVES OF
*  THE STIFFNESS MATRIX WITH RESPECT TO
*  THE DESIGN VARIABLES

SPAR, RSOUT, NSPROUT.
RETURN, INPT, EDIT1, EDIT2, BLDLB, EOUT.
RETURN, RSOUT, TAPE21, TAPE22, NSPROUT.
REPLACE, SPARLB=NRLA.
ENDIF, GRADIENTS.
RETURN, SPAR, DCU, SPARLA, NRRS.
APPENDIX C

PRCINIT

PROC,PRCINIT,OP,A,B,BB,NSUB,C,FSD,FSUB.

**
THIS PROCEDURE FILE CREATES PROGRAMS USED BY THE
DIFFERENT PROCESS OPTIONS.

**
SET(RIG=1)
MAP,OFF.

**
CREATE CONMIN

GET,A.
GET,CONMINB/UN=753437N.

**
TEST FOR FULLY STRESSED DESIGN REQUIREMENT

IFE,(FSD.EQ.1),GETFSDSUB.
RETURN,CONMINB.
GET,CONMINB,FSUB.
ENDIF,GETFSDSUB.
COPYL,A,CONMINB,CONMIN,RA.

**
CREATE END PROCESSOR

GET,B.
GET,SCOMBLK,SPARLIB/UN=319925N.
COPYBR,B,ENOPROC.
COPYBR,SCOMBLK,ENOPROC.
COPYBF,B,ENOPROC.
IFE,(OP.EQ.13),GEND.
REWIND,SCOMBLK.
GET,BB.
COPYBR,BB,ENDG.
COPYBR,SCOMBLK,ENDG.
COPYBF,BB,ENDG.
RENAME,NGEND=ENOPROC.
ENDIF,GEND.

**
TEST TO SEE IF OPTION USING GRADIENTS WAS CHOSEN

IFE,(OP.EQ.13.OR.OP.EQ.23),GRADIENTS.

**
CREATE PROGRAM FOR GENERATING REPEATABLE SPAR RUNSTREAMS

GET,GNGRDRB/UN=753437N.
COPYBR,GNGRDRB,GREPEAT.

**
TEST FOR BEAM OR PLATE SUBROUTINES

IFE,(NSUB.NE.0),NOSUBS.
GET,C.
COPYBR,C,GREPEAT,NSUB.
ENDIF,NSUB.
COPYBF,GNGRDRB,GREPEAT.
RETURN,GNGRDRB.

*  TEST FOR NEED TO CONVERT FORCES AND MOMENTS TO STRESSES  *
IFE,(NSUB,NE,0),GRADIENTS.

*  CREATE PROGRAM TO CONVERT FORCES AND MOMENTS TO STRESSES  *
GET,DRVSTRB/UN=753437N.
REWIND,SCOMBLK.
COPYBR,DRVSTRB,FAM2STR.
COPYBR,C,FAM2STR,NSUB.
COPYBR,SCOMBLK,FAM2STR.
COPYBF,DRVSTRB,FAM2STR.
RETURN,DRVSTRB.
ENDIF,GRADIENTS.
GET,SPAR=SPAR14I,DCU=DCU14I/UN=750756N.
RETURN,SCOMBLK,A,B,C,CONMINB.
**THIS PROCEDURE FILE GETS ALL THE FILES REQUIRED FOR EXECUTING A PARTICULAR OPTION**

**GET FILES USED IN ALL OPTIONS**

GET EDGRDS, MERGFP, EDPASS1/UN=753437N.
GET, FRTPROC=F, ENDIN=E, CONPAR=CN, STARTX=S.
GET, CONS=C, TEMPRS=RS.

**CHANGE PASS TO 1 FOR FIRST PASS**

EDIT, PASS=EDPASS1, EDOUT.
RETURN, EDPASS1, EDOUT.

**GET ADDITIONAL FILES NEEDED FOR OPTION 1.2**

IFE, (OP EQ.12), OPTION12.
GET, EVALB, RERITEB, SELECTB, EDPASS2/UN=753437N.
ENDIF, OPTION12.

**GET ADDITIONAL FILES NEEDED FOR OPTION 1.3**

IFE, (OP EQ.13), OPTION13.
GET, INPT=I, RGRS=RGS.
RENAME, NGRRS=TEMPRS.
GET, SELECTB/UN=753437N.
ENDIF, OPTION13.

**GET ADDITIONAL FILES NEEDED FOR OPTION 2.2**

IFE, (OP EQ.22), OPTION22.
GET, EVALB, EDPASS2/UN=753437N.
GET, CNT=CT.
ENDIF, OPTION22.

**GET ADDITIONAL FILES NEEDED FOR OPTION 2.3**

IFE, (OP EQ.23), OPTION23.
GET, CNT=CT, INPT=I.
GET, EVALGB/UN=753437N.
ENDIF, OPTION23.
*PROC,PRCFPXX.*

**THIS PROCEDURE FILE EXECUTES THE FRONT PROCESSOR**

REWIND,FRTPROC,CNMNIO,CONS,SPFPOUT.
LDSET(PRESET=ZERO)
FRTPROC,CNMNIO,SPFPOUT,CONS.
**APPENDIX C**

**PRCOPTM**

```
.PROC,PRCOPTM,C,D,F.

*  THIS PROCEDURE FILE EXECUTES CONMIN

*RFL,250000.
REDUCE,--.
REWIND,CONPAR,STARTX,C,D,PASS,GONOGO,CONMIN.
LDSET(PRESET=ZERO)
CONMIN,CONPAR,CONOUT,STARTX,C,D,GONOGO,PASS,INFOFL.
PACK(CONOUT)
REPLACE,CONOUT=F.
SKIPEI(CONOUT)
```
APPENDIX C

PRCANAL

.PROC,PRCANAL,NRLA,FLX,SAVELD.
.*
.* THIS PROCEDURE FILE EXECUTES SPAR FOR THE ANALYSIS
.*
GET,SPARLA=NRLA.
REWIND,SPARLA,RRS,MERGFP,SPFPOUT,SPAROUT,TEMPRS.
COPYCF,TEMPRS,RRS.
REWIND,RRS.
.* MERGE OUTPUT FROM THE FRONT PROCESSOR INTO THE
.* SPAR RUNSTREAM
.*
EDIT,RRS,MERGFP,EDOUT.
REWIND,RRS.
RFL,FLX.
REDUCE,-.
SPAR,RRS,SPAROUT.
REPLACE,SPARLO=SAVELD.
.*
.* SAVE INITIAL SPAR INPUT AND OUTPUT FOR LATER LISTING
.*
IF,E,(RIG.EQ.1),RENAME.
RENAME,SPINI=RRS,SPOUT1=SPAROUT.
SET(RIG=2)
ENDIF,RENAME.
RETURN,EDOUT,SPFPOUT.
APPENDIX C

PRCGRDS

PROC, PRCGRDS, NSUB, SAVELD.
* THIS PROCEDURE FILE CALCULATES GRADIENTS
REWIND, INPT, CONS, CNMNIO, GREPEAT, SPARLA, SPARLD.
* CREATE SPAR RUNSTREAM
GREPEAT, INPT, CONS, CNMNIO, RSOUT.
REWIND, RSOUT, EDGRDS.
EDIT, RSOUT, EDGRDS, EDOU.
REWIND, RSOUT.
* EXECUTE SPAR TO FIND STRESS DERIVATIVES
SPAR, RSOUT, SPAROUT.
REPLACE, SPARLD, SAVELD.
* TEST TO SEE IF BEAM OR PLATE FORCES AND MOMENTS
* NEED CONVERTING TO STRESSES
IFE, (NSUB, NE, O), FAMS.
REWIND, FAM2STR, SPARLIB, CNMNIO, SPARLD, INPT, CONS.
LDSET (LIB = SPARLIB, PRESET = ZERO)
FAM2STR, INPT, CONS, CNMNIO.
REPLACE, SPARLD, SAVELD.
ENDIF, FAMS.
RETURN, SPAROUT, EDOU, RSOUT.
.

PROC,PRCEPXX,BLK.*

* THIS PROCEDURE FILE EXECUTES THE END PROCESSOR *

REWIND,ENDPROC,ENDIN,CNMNIO,BLK,SPARLIB,SPARLD.
LDSET(LIB=SPARLIB,PRESET=ZERO)
ENDPROC,ENDIN,CNMNIO,BLK.
RETURN,SPARLA,SPARLC,SPARLD.
*PROC,PRCEND.*

**THIS PROCEDURE FILE OUTPUTS IMPORTANT FILES**

REWIND,SPIN1,SPOUT1,RRS,SPAROUT,CONOUT,CNMINIO.
COPYSBF,CONOUT,TEMPL.
COPYSBF,CNMINIO,TEMPL.
COPYSBF,SPIN1,TEMPL.
COPYSBF,SPOUT1,TEMPL.
COPYSBF,RRS,TEMPL.
COPYSBF,SPAROUT,TEMPL.
PACK(TEMPL)
REWIND,TEMPL.
REPLACE,TEMPL.
REWIND,TEMPL.
COPYSBF,TEMPL,OUTPUT.
This is a main driver program for the CONMIN subroutine library.

```
PROGRAM CONMS1 (INPUT, OUTPUT, TAPE8, TAPE7, TAPE9, TAPE11, 
1 TAPE10, TAPE12, TAPE5=INPUT, TAPE6=OUTPUT)
C NEW CONMIN SUBROUTINE
COMMON/CNMNI/0ElFUN,DABFUN,fDCH,FDCHM,CT,CTMIN,CTL,CTLMIN,
1ALPHAX,ABOBJJ,THETA,OBJ,NDV,NCON,NSIDE,IPRINT,NFDG,NSCAL,
2LINOBJ,ITMAX,ITRM,ICNDIR,IGOTO,NAC,INFO,INFOG,ITER
COMMON/CNMN2/RDUM(50), IDUM(25)
COMMON X(20), VLB(20), VUB(20), G(400), SCAL(20), DF(20),
1A(20, 200), S(20), G1(400), G2(400), B(200, 200), C(200), ISC(400),
2IC(200), MS1(400)
COMMON/CONSAV/RSAV(50), ISAV(25)
NAMELIST/CONPAR/IPRINT,NOV,ITMAX,NCON,NFDG,NSIDE,ICNDIR,
1NSCAL,LINOBJ,ITRM,FDCH,FDCHM,CT,CTMIN,CTL,CTLMIN,
2THETA,PHI,DELFUN,DABFUN,ISC,N1,N2,N3,N4,N5,
3ALPHAX,ABOBJJ,IGOTO,VLB,VUB
NAMELIST/STARTX/X
NAMELIST/SAVE/IPRINT,NDV,ITMAX,NCON,NSIDE,ICNDIR,NSCAL,NFDG,
1FDCH,FDCHM,CT,CTMIN,CTL,CTLMIN,THETA,PHI,NAC,DELFUN,DABFUN,
2LINOBJ,ITRM,ITER,INFOG,IGOTO,INFO,OBJ,
3RDUM, IDUM,
4X, DF, G, ISC, IC, A, S, G1, G2, C, MS1, B, VLB, VUB, SCAL, RSAV, ISAV, NCOUNT
5, N1, N2, N3, N4, N5, ALPHAX, ABOBJJ
NAMELIST/LINKF/NDV,X
NAMELIST/LINKE/OBJ,G
NAMELIST/PASSAGE/NPASS
READ(10, PASSAGE)
C FIRST PASS, NPASS=1, SUBSEQUENTLY NPASS=2.
GO TO(100, 200), NPASS
100 CONTINUE
READ(5, CONPAR)
READ(8, STARTX)
REWIND 8
GO TO 201
200 CONTINUE
READ(7, SAVE)
REWIND 7
READ(9, LINKE)
REWIND 9
```
APPENDIX D

CONMS1 (Conc.)

201 CONTINUE
NPASS=2
REWIND 10
WRITE(10, PASSAGE)
C SOLVE OPTIMIZATION
CALL CONMIN(X,VLB,VUB,SCAL,DF,A,S,G1,G2,B,C,
*ISC,IC,MS1,N1,N2,N3,N4,N5)
C FUNCTION AND CONSTRAINT VALUES
WRITE(7,SAVE)
WRITE(9, LINKF)
C WRITE CONTROL CARD STORED IN PROFILE GONOGO.
101 FORMAT(*GOTO,2.2*)
IF(IGOTO.EQ.0)WRITE(11,101)
C WRITE ON TAPE8 IF GRADIENTS ARE REQUIRED
IF(INFO.EQ.2) WRITE(12,102)
102 FORMAT(* INFO = 2*)
IF(INFO.EQ.2) WRITE(8,STARTX)
STOP
END
This is a main driver program for the CNMIN subroutine library.

```fortran
PROGRAM CONMS2(INPUT, OUTPUT, TAPE7, TAPE8, TAPE9, TAPE10,
1 TAPE11, TAPE12, TAPE15=INPUT, TAPE6=OUTPUT)
C NEW CONMIN SUBROUTINE
COMMON/CNMN1/DELFUN, DABFUN, FDCH, FDCM, CT, CTMIN, CT, CTL, CTLMIN,
1ALPHAX, ABOBJ1, THETA, OBJ, NDV, NCON, NSIDE, IPRINT, NFDG, NSCAL,
2LINOBJ, ITMAX, ITRM, ICNDIR, IGOTO, NAC, INFO, INFDG, ITER
COMMON X(20), VLB(20), VUB(20), G1(400), G2(400), B(200, 200), C(200), ISC(400),
2IC(200), MS1(400)
DIMENSION XI(20), G1(400), GRDOBJ(20), GRDG(20, 400)
NAMELIST/CONPAR/IPRINT, NDV, NCON, ITMAX, NSIDE, ICNDIR,
1LINOBJ, LINOBJ, ITMAX, FDCH, FDCM, CT, CTMIN, CT, CTL, CTLMIN,
2THETA, PHI, DELFUN, DABFUN, ISC, N1, N2, N3, N4, N5,
3ALPHAX, ABOBJ1, IGOTO, VLB, VUB
NAMELIST/STARTX/X, XINC
NAMELIST/BLK/OBJ, OBJ1, X, XI, G, GI, GRDOBJ, GRDG, ICOUNT
NAMELIST/CNT/OBJ1, OBJ2, OBJ3, TOL
BL = .7
BU = 1.3
Loopct = 50
READ(5, CONPAR)
READ(7, STARTX)
READ(8, BLK)
READ(9, CNT)
REWIND 7
REWIND 8
REWIND 9
NCON = NCON + 1
NCON = NCON + (2 * NDV)
DO 28 NC = NSCON, NCON
ISC(NC) = 1
28 CONTINUE
DO 100 Loop = 1, Loopct
C SOLVE OPTIMIZATION
CALL CONMIN(X, VLB, VUB, G, NSCAL, DF, A, S, G1, G2, B, C,
*ISC, IC, MS1, N1, N2, N3, N4, N5)
C FUNCTION AND CONSTRAINT VALUES
IF (INFO.EQ.2) GO TO 24
OBJ = OBJ1
DO 9 IJ = 1, NCON
9 G(IJ) = GI(IJ)
DO 10 I = 1, NDV
DXI = XI(I) - XI(I)
OBJ = OBJ + GRDOBJ(I) * DXI
DO 20 J = 1, NCON
G(J) = G(J) + GRDG(I, J) * DXI
```

59
APPENDIX D

CONMS2 (Conc.)

20 CONTINUE
10 CONTINUE
NSDV = 0
    DO 15 NC = NSCON,NCON,2
    NSDV = NSDV+1
    GN(C) = 1.-(X(NSDV)/(BN*XI(NSDV))
    GN(C+1) = X(NSDV)/(BN*XI(NSDV))-1.
15 CONTINUE
GO TO 70
24 CONTINUE
DO 23 IDF = 1,NDV
DF(IDF) = GRDOBJ(IDF)
23 CONTINUE
NSDV = 0
    DO 25 NC = NSCON,NCON,2
    NSDV = NSDV+1
    GRD(GNSDV,NC) = -1.0/(BN*XI(NSDV))
    GRD(GNSDV,NC+1) = 1.0/(BN*XI(NSDV))
25 CONTINUE
NAC = 0
    DO 30 J = 1,NCON
    IF(G(J),LT,CT) GO TO 30
    NAC = NAC+1
    IC(NAC) = J
30 CONTINUE
    DO 40 II = 1,NDV
    DO 50 JJ = 1,NAC
    J1 = IC(JJ)
    A(II,JJ) = GRD(II,J1)
50 CONTINUE
40 CONTINUE
70 CONTINUE
IF(IGOTO.EQ.0) GO TO 200
100 CONTINUE
200 CONTINUE
WRITE(7,STARTX)
TOL = DELFUN
OBJ3 = OBJ2
OBJ2 = OBJ1
OBJ1 = OBJ
WRITE(9,CNT)
DA = ABS((OBJ3-OBJ2)/OBJ2)
DB = ABS((OBJ2-OBJ1)/OBJ2)
IF(DA .LE. TOL .AND. DB .LE. TOL) WRITE(10,1)
1 FORMAT(*TERMINATED.*)
STOP
END
APPENDIX D

FPROC

This is an example listing of a front processor program.

```
PROGRAM FPFGS1(INPUT,OUTPUT,TAPE7,TAPE5=INPUT,TAPE6=OUTPUT)
C SPAR FRONT PROCESSOR READS DESIGN VARIABLES AND
C PRINTS THEM IN SPAR SECTION PROPERTY FORMAT
C FUSELAGE MADE OF ROD BEAM MEMBRANE ELEMENTS
DIMENSION X(50)
NAMELIST/LINKF/NDV,X
C NDV=NUMBER OF DESIGN VARIABLES
C X(NDV)=DESIGN VARIABLES
C DV'S ARE X(1)=SECTIONAL AREA OF STRINGER RODS
C X(2)=NONDIMENSIONAL AREA OF BEAM
C X(3)=THICKNESS OF MEMBRANE PANEL
READ(7,10) B10,B20,T0
10 FORMAT(3F10.3)
READ(5,LINKF)
XIN1=1./X(1)
XIN2=1./X(2)
XIN3=1./X(3)
C WRITE E23 ELEMENTS
PRINT 200
200 FORMAT(* E23 SECTION PROPERTIES*)
PRINT 201,XIN1
201 FORMAT(* 1 *,F8.3)
C COMPUTE VALUES FOR DSY CARDS
C WRITE E21 ELEMENTS
PRINT 202
202 FORMAT(* F21 SECTION PROPERTIES*)
AREA0=(2.*B10+B20)*T0
AREA=AREA0*XIN2
SCALE=SQRT(AREA/AREA0)
B1=B10*SCALE
B2=B20*SCALE
T=T0*SCALE
ALPHA1=0.
C*C/AREA
EI2=2.*T*B1**3/12.+2.*T*B1*(B1/2.-C)**2
ALPHA2=0.
F1=0.
Z2=0.
THETA=0.
Q1=0.
Q2=0.
Q3=0.
Y11=-(B1-C)
Y12=.5*B2+T
```
FPROC (Conc.)

Y21 = C
Y22 = .5 * B2 + T
Y31 = C
Y32 = -.5 * B2 + T
Y41 = -(B1 - C)
Y42 = -.5 * B2 + T
J = 1

PRINT 103, J, EI1, ALPHA1, EI2, ALPHA2, AREA
PRINT 1003, F, F1, Z1, Z2, Theta

103 FORMAT (*DSY*, I2, 5E12.4, ***)
1003 FORMAT (1X, 5E13.5)

PRINT 104, Q1, Q2, Q3, Y11, Y12, Y21
104 FORMAT (1X, 6E12.4, ***)

PRINT 1004, Y22, Y31, Y32, Y41, Y42
1004 FORMAT (1X, 5E12.4)

C
WRITE E41 ELEMENTS

PRINT 300, XIN3
300 FORMAT (* SHELL SECTION PROPERTIES** 1*, F8.3)
STOP
END
This is an example listing of an end processor program.

```
PROGRAM EPFUS(INPUT, TAPE6, TAPE8, TAPE5\*INPUT, OUTPUT)
DIMENSION A(4500), B(1400)
DIMENSION A1(400), B1(400), C1(400)
DIMENSION G(400)
NAMELIST/EPIN/E23A1, E21A1, E41A1, NSE23, NSE21, NSE41
NAMELIST/LINKE/OBJ, G
READ(5, EPIN)
CALL DAL(4, 11, A(1), 0, IEA, KADR, IERR, NWDS, NE, LB, ITYPE,
  14H0BJF, 3HAUS, 1, 1)
OBJ = A(1)
CALL DAL(4, 11, A(1), 0, IEA, KADR, IERR, NWDS, NE, LB, ITYPE,
  14HSTRS, 3HE23, 1, 1)
I1 = 1
INL = 6*NSE23
DO 2 IN = 6, INL, 6
  I = I1
  A1(I1) = A(IN)
  I1 = I1 + 1
2 CONTINUE
CALL DAL(4, 11, A(1), 0, IEA, KADR, IERR, NWDS, NE, LB, ITYPE,
  14HSTRS, 3HE21, 1, 1)
J1 = 1
KNL1 = (NSE21-1)*52 + 6
DO 4 IM = 5, KNL1, 52
  B1(J1) = ABS(A(IM))
  A1(J1) = ABS(A(IM+1))
  I = I1 + J1 - 1
  GNUM = B1(J1)
  IF (A1(J1) .GT. B1(J1)) GNUM = A1(J1)
  J1 = J1 + 1
4 CONTINUE
CALL DAL(4, 11, A(1), 0, IEA, KADR, IERR, NWDS, NE, LB, ITYPE,
  14HSTRS, 3HE41, 1, 1)
K1 = 1
LNL1 = (NSE41*23)
DO 10 NP = 21, LNL1, 23
  C1(K1) = A(NP)
  B1(K1) = A(NP+1)
  A1(K1) = A(NP+2)
  I = I1 + J1 + K1 - 2
  GNUM = SQRT(C1(K1)**2+B1(K1)**2-C1(K1)*B1(K1)+3.*A1(K1)**2)
  G(I) = GNUM/E41A1 - 1.
  K1 = K1 + 1
10 CONTINUE
CALL FIN(0, 0)
REWIND 6
WRITE(6, LINKE)
STOP
END
```
PROGRAM EVALS(INPUT,OUTPUT,TAPE5=INPUT,TAPE7,TAPE8,TAPE9,TAPE10,TAPE11)
DIMENSION X(20),XI(20),G(400),GI(400),GRDOBJ(20),GRDG(20,400)
DIMENSION ISC(400),VLB(20),VUB(20)
NAMELIST/CONPAR/IPRINT,NDV,ITMAX,NCON,NFDC,NSIDE,ICNDR,
INSCAL,LINOBJ,ITRM,FDCH,FDCHM,CT,CTMIN,CTL,CTLMIN,
2THETA,PHI,DELFUN,DABFUN,ISC,N1,N2,N3,N4,N5,
3ALPHAX,ABOBJ,IGOTO,VLB,VUB
NAMELIST/PASSAGE/NPASS
NAMELIST/STARTX/X,XINC
NAMELIST/LINKE/OBJ,G
NAMELIST/LINKF/NDV,X
NAMELIST/BLK/OBJ,OBJI,X,XI,G,GI,GRDOBJ,GRDG,ICOUNT
READ(5,CONPAR)
REWIND 5
READ(7,PASSAGE)
REWIND 7
READ(8,STARTX)
REWIND 8
GO TO(100,200),NPASS
100 CONTINUE
ICOUNT=0
DO 1000 I=1,NDV
XI(I)=0.0
G(I)=0.0
GI(I)=0.0
GRDOBJ(I)=0.0
DO 1001 J=1,NCON
GRDG(I,J)=0.0
1001 CONTINUE
1000 CONTINUE
OBJ=0.0
OBJI=0.0
NPASS=2
WRITE(7,PASSAGE)
GO TO 201
200 CONTINUE
READ(10,BLK)
REWIND 10
READ(9,LINKE)
REWIND 9
IF(ICOUNT.NE.1)GO TO 300
OBJI=OBJ
DO 10 J=1,NDV
10 XI(J)=X(J)
DO 20 K=1,NCON
20 GI(K)=G(K)
GO TO 400
300 I=ICOUNT-1
APPENDIX D

EVALS (Conc.)

DELTX = X(I) - XI(I)
GRDOBJ(I) = (OBJ - OBJI) / DELTX
DO 30 L = 1, NCON
GRDG(I, L) = (G(L) - GI(L)) / DELTX
30 CONTINUE
X(I) = XI(I)
400 X(ICOUNT) = X(ICOUNT) * (1. - XINC)
LIM = NDV + 1
IF (ICOUNT .EQ. LIM) WRITE(11, 401)
REWIND 11
401 FORMAT (*GOTO, 7.*)
201 CONTINUE
ICOUNT = ICOUNT + 1
WRITE(10, BLK)
REWIND 10
WRITE(9, LINKF)
REWIND 9
STOP
END
APPENDIX D

FSDS

PROGRAM FSDS(INPUT, OUTPUT, TAPE8, TAPE7, TAPE9, TAPE11, 1 TAPE10, TAPE5=INPUT, TAPE6=OUTPUT)
C NEW CONMIN SUBROUTINE
COMMON/CNMN1/DELFUN, DABFUN, FDCH, FDCHM, CT, CTMIN, CTL, CTLMIN,
1 ALPHAX, ABOBJ1, THETA, OBJ, NDV, NCON, NSIDE, IPRINT, NFDG, NSCAL,
2 LINOBJ, ITMAX, ITRM, ICNDIR, IGOTO, NAC, INFO, INFOG, ITER
COMMON/CNMN2/IDUM(50), IDUM(25)
COMMON X(20), VLB(20), VUB(20), G(400), SCAL(20), DF(20),
1 A(20, 200), S(20), G1(400), G2(400), B(200, 200), C(200), ISC(400),
2 IC(200), MS1(400)
COMMON/CONSAV/RSAV(50), ISA1(25)
COMMON XCZO, VIBCZO, VUB(20), G(400), SCAl(200), DF(ZO),
1 A(ZO, 400), S(ZO), G1(400), G2(400), B(200, 200), C(Z00), ISC(Z00),
2 IC(Z00), MS1(400)
COMMON/CONSAV/RSAV(50), ISA1(25)
NAMELIST/CONPAR/IPRINT, NDV, ITMAX, NCON, NSIDE, ICNDIR,
1 NSCAL, LINOBJ, ITRM, FDCH, FDCHM, CT, CTMIN, CTL, CTLMIN,
2 THETA, PHI, DELFUN, DABFUN, ISC, N1, N2, N3, N4, N5,
3 ALPHAX, ABOBJ1, IGOTO, VLB, VUB
NAMELIST/STARTX/X
NAMELIST/SAVE/IPRINT, NDV, ITMAX, NCON, NSIDE, ICNDIR, NSCAL, NFDG,
1 FDCH, FDCHM, CT, CTMIN, CTL, CTLMIN, THETA, PHI, NAC, DELFUN, DABFUN,
2 LINOBJ, ITRM, ITER, INFOG, IGOTO, INFO, OBJ,
3 RDUM, IDUM,
4 X, DF, G, ISC, IC, A, S, G1, G2, C, MS1, B, VL B, VUB, SCAL, RSAV, ISA1, NCOUNT
5, N1, N2, N3, N4, N5, ALPHAX, ABOBJ1, OBJ1, OBJ2, OBJ3
NAMELIST/LINKF/NOV, X
NAMELIST/LINKE/OBJ, G
NAMELIST/PASSAGE/NPASS
READ(10, PASSAGE)
C FIRST PASS, NPASS=1, SUBSEQUENTLY NPASS=2.
GO TO(10, 200), NPASS
100 CONTINUE
READ(5, CONPAR)
READ(A, STARTX)
DO 300 K=1, NCON
G(I)=0.0
300 CONTINUE
IGOTO=1
OBJ1=1.
OBJ2=1.
OBJ3=1.
GO TO 201
50 CONTINUE
READ(7, SAVE)
REWIND 7
READ(9, LINKE)
OBJ1=OBJ2
OBJ2=OBJ3
OBJ3=OBJ
OINC1=ABS(1-OBJ1/OBJ2)
OINC2=ABS(1-OBJ2/OBJ3)
OINC3=ABS(1-OBJ3/OBJ)
APPENDIX D

FSDS (Conc.)

IF(OINC1.LT.DELFUN.AND.OINC2.LT.DELFUN.AND. OINC3.LT.DELFUN)IGOTO=0
201 CONTINUE
NPASS=2
REWIND 10
WRITE(10,PASSAGE)
C SOLVE FSD PROBLEM
CALL FSDSUB(X,DF,G,ISC,IC,A,S,G1,G2,C,MS1,B,VLB,VUB 1,SCAL,N1,N2,N3,N4,N5)
C FUNCTION AND CONSTRAINT VALUES
WRITE(7,SAVE)
WRITE(9,LINKF)
C WRITE CONTROL CARD STORED IN PROCFILE GONOGO.
101 FORMAT(*GOTO,2.*)
IF(IGOTO.EQ.0)WRITE(11,101)
STOP
END
APPENDIX D

FSDSUBS

SUBROUTINE FSDSUB(X, DF, G, ISC, IC, A, S, G1, G2, C, MS1, B, VLB, VUB,
SCAL, N1, N2, N3, N4, N5)
COMMON/CNMM1/DELFUN, DABFUN, FDCH, FDCHM, CT, CTMIN, CTL, CTLMIN,
1 ALPHAX, ABBJ1, THETA, OBJ, NDV, NCON, NSIDE, IPRINT, NFDG, NSCAL,
2 LINOBJ, J, ITM, ITRM, ICNDIR, IGOTO, NAC, INFO, INFOG, ITER
COMMON/CNMM2/RDUM(50), IDUM(25)
DIMENSION X(20), VLB(20), VUB(20), G(400), SCAL(20), DF(20),
1A(20, 200), S(20), G1(400), G2(400), B(200, 200), C(200), ISC(400),
2IC(200), MS1(400)
COMMON/CONSAV/RSAV(50), ISAV(25)
WRITE(6, 5) (X(I), I=1, NOV)
5 FORMAT(1H1, *DESIGN VARIABLES INTO FSOSUB ARE*, (1X, E13.5))
DO 1 I=1, NCON
  G(I) = G(I) + 1.
  CONTINUE
DO 20 I = 1, NOV
  X(I) = 1./X(I)
20 CONTINUE
XO = 1./VUB(1)
NSE23 = 58
ISTRT = 1
IEND = Istrt + NSE23
DO 2 I = ISTRT, IEND
  XNEW = X(I) * G(I)
  IF(XNEW .LE. XO) GO TO 2
  XO = XNEW
  WRITE(6, 12) I, G(I), XO
2 CONTINUE
X(1) = XO
XO = 1./VUB(2)
NSE23 = 76
ISTRT = NSE23 + 1
IEND = ISTRT + NSE21
DO 4 I = ISTRT, IEND
  XNEW = X(I) * G(I)
  IF(XNEW .LE. XO) GO TO 4
  XO = XNEW
  WRITE(6, 12) I, G(I), XO
4 CONTINUE
X(2) = XO
XO = 1./VUB(3)
NSE21 = 56
ISTRT = NSE23 + NSE21 + 1
IEND = ISTRT + NSE41
DO 6 I = ISTRT, IEND
  XNEW = X(I) * G(I)
  IF(XNEW .LE. XO) GO TO 6
6 CONTINUE
APPENDIX D

FSDSUBS (Cont.)

X0=XNEW
WRITE(6,12) I,G(I),X0
6 CONTINUE
X(3)=X0
DO 30 I = 1,NDV
X(I) = 1./X(I)
30 CONTINUE
WRITE(6,10) (X(I),I=1,NDV)
10 FORMAT(16X,*DESIGN VARIABLES FROM FSDSUB ARE*,/3(1X,E13.5))
RETURN
END
APPENDIX D

SELECTS

PROGRAM SELECTS(INPUT, OUTPUT, TAPE5=INPUT, TAPE6)
DIMENSION X(20), VLB(20), VUB(20), G(400), SCAL(20), DF(20),
IA(20, 200), S(20), GI(400), G2(400), B(200, 200), C(200), ISC(400),
2IC(200), MS1(400)
DIMENSION XI(20), GI(400), GRDOBJ(20), GRDG(20, 400)
DIMENSION RDU~(50), RSAV(50), IDUM(25), ISA(25)
NAMELIST/SAVE/IPRINT, NDV, ITMAX, NCON, NSIDE, INCDIR, NSCAL, NFDG,
FDCG, FDCHM, CT, CTMIN, CTL, CutMIN, THETA, PHI, NAC, DELFUN, DABFUN,
2LINOBJ, ITRM, ITER, INFO, IGOTO, INFO, OBJ,
3RDUM, IDUM,
4X, DF, G, ISC, IC, A, S, GI, G2, C, MS1, B, VLB, VUB, SCAL, RSAV, ISAV, NCOUNT
5, N1, N2, N3, N4, N5, ALPHAX, ABOBJJ
NAMELIST/BLK/OBJ, OBJI, XI, GI, GRDOBJ, GRDG, ICOUNT
READ(6, SAVE)
REWIND 6
READ(5, BLK)
OBJ=OBJI
DO 25 I=1, NDV
X(I)=XI(I)
DF(I)=GRDOBJ(I)
25 CONTINUE
NAC=0
DO 30 J=1, NCON
G(J)=GI(J)
IF(G(J).LT.CTL) GO TO 30
NAC=NAC+1
IC(NAC)=J
30 CONTINUE
DO 40 II=1, NDV
DO 50 JJ=1, NAC
J1=IC(JJ)
A(II, JJ)=GRDG(II, J1)
50 CONTINUE
DO 40
WRITE(6, SAVE)
STOP
END
PROGRAM RERITES(INPUT, TAPE5, INPUT, TAPE6)
DIMENSION X(20)
NAMELIST/LINKF/NDV, X
NAMELIST/STARTX/X, XINC
DATA XINC/0.1/
READ(5, LINKF)
WRITE(6, STARTX)
STOP
END
APPENDIX D

BLDELDs

PROGRAM BLDELDs(TAPE5, TAPE23, TAPE20, TAPE21, TAPE22, OUTPUT)

C THIS PROGRAM CREATES A RUNSTREAM THAT WILL CREATE
C DMDV AND DDKV FOR PARTICULAR ELEMENTS USED DESIGN VARIABLES

DIMENSION EL(999), NSECT(999), TNAME1(8), TNAME2(8), FOR(9)
DIMENSION NODVPE(999)
DATA E21, E22, E23, E41, E43, E44 / 3HE21, 3HE22, 3HE23, 3HE41, 3HE43, 3HE44 /
DATA E31, E33 / 3HE31, 3HE33 /
DATA TNAME1/4HDEF, 4HGD, 4HGTIT, 4HDIR, 4HNS, 3*4HELTS/
DATA TNAME2/5*4H, 4HNAME, 4HNOD, 4HISCT/
DATA START, END, XNSECT/4H$STA, 4H$END, 4HNSECT/
DATA YNSECT/3HNSE/

CALL SUBROUTINE TO REMOVE BEGINNING BLANKS
CALL REMOVE

READ INPUT

NOEL=NUMBER OF ELEMENTS
NODV=NUMBER OF DESIGN VARIABLE ELEMENTS
VORB=TYPE OF ANALYSIS (EX. BUCKLING)
NDF=NUMBER OF DEGREES OF FREEDOM PER JOINT
EL - ELEMENT NAMES CONTAINING DESIGN VARIABLES
(EX. E21)
NSECT - LAST SECTION NUMBER USED FOR EACH DESIGN VARIABLE
NODVPE - NUMBER OF DESIGN VARIABLES PER ELEMENT

READ(5,5) NOEL, NODV, ISNOLC, JOINTS, NDF, NOEL, VORB
5 FORMAT(6(I4), A4)
READ(5,6) (EL(I), NSECT(I), NODVPE(I), I=1, NOEL)
6 FORMAT(6(I4), A3, I4, I3, I1, I3)
NDF*NDF
WRITE(20,2)
2 FORMAT(*[XQT TAB** UPDATE=1*)

LOOP ON NUMBER OF ELEMENTS

DO 30 I = 1, NOEL
30 CALL SUBROUTINE TO UPDATE TAB BY SETTING DESIGN VARIABLES
TO UNITY.
KCNT RETURNS THE NUMBER OF POSSIBLE DESIGN VARIABLES FOR
A PARTICULAR ELEMENT.
KCNT=99 MEANS THE ELEMENT NAME IS BAD

CALL TABNPUT(EL(I), NSECT(I), KCNT)
IF(KCNT.NE.99) GO TO 30
PRINT 29, EL(I)
APPENDIX D

BLDELDs (Cont.)

29 FORMAT(* ELEMENT NAME *,A3,* DOES NOT EXIST*)
GO TO 190
30 CONTINUE

C CONCLUDE UPDATE AND SET UP DISABLES

C WRITE(20,31)
31 FORMAT(* UPDATE=0*)
WRITE(20,32)
32 FORMAT(*IXQT DCU/* COPY 1,2*)
DO 35 I = 1,8
IF(I.GT.4) GO TO 33
DO 37 J = 1,NOEL
TNAME2(I)=EL(J)
WRITE(20,34) TNAME1(I),TNAME2(I)
37 CONTINUE
GO TO 35
33 WRITE(20,34) TNAME1(I),TNAME2(I)
34 FORMAT(* DISABLE 1,*,A4,1X,A4)
35 CONTINUE

C SET COUNTER FOR TOTAL NUMBER OF DESIGN VARIABLES

C IELCNT = 0
ICNTDV = 1
ISW=0
ISAVCNT = 0

C READ IN RUNSTREAM AND CHECK FOR START OF A DESIGN VARIABLE

C 39 JCNT = 0
40 READ(21,50) (FOR(J),J=1,9)
50 FORMAT(A4,A6,7A10)
IF(EOF(21)) 170,60
60 IF(FOR(1).EQ.START) GO TO 70
WRITE(20,50) (FOR(J),J=1,9)
GO TO 40
70 IF(ISW.EQ.1) WRITE(20,75)
75 FORMAT(*IXQT ELD*)
ISW=1
READ(21,80) (FOR(J),J=1,9)
80 FORMAT(A3,A7,7A10)

C SET ICNT = NUMBER OF POSSIBLE DESIGN VARIABLES FOR AN ELEMENT
ICNT=99

C DETERMINE POSSIBLE NUMBER OF DESIGN VARIABLES PER ELEMENT
APPENDIX D

BLDELDs (Cont.)

C IF(FOR(1).EO.E23 .OR. FOR(1).EO.E41 .OR. FOR(1).EO.E44) ICNT=1
IF(FOR(1).EO.E31) ICNT=1
IF(FOR(1).EO.E21) ICNT=4
IF(FOR(1).EO.E43 .OR. FOR(1).EO.E33) ICNT=12
IF(FOR(1).EO.E22) ICNT=21
TNAME=FOR(1)
IF(ICNT.EQ.99) TNAME=SAVNAME
JCNT = JCNT+1
IF(ICNT.NE.99) IELCNT=IELCNT+1

C SET UNIT = 20 IF ONLY ONE DESIGN VARIABLE PER ELEMENT
C OTHERWISE SET UNIT = 22 (SCRATCH UNIT)

C IUNIT=22
REWIN 22
IF(ICNT.EQ.1) IUNIT=20
IF(ICNT.EQ.99 .AND. ISAVCNT.EQ.1) IUNIT=20

C CHECK FOR REPEAT OF ELEMENT NAME

C IF(ICNT.NE.99) GO TO 86
WRITE(IUNIT,85) SAVNAME
85 FORMAT(A3,77X)
GO TO 860

C READ DATA FROM UNIT 21 AND WRITE DATA ON UNIT 20 OR 22
C DEPENDING UPON VALUE OF ICNT

86 WRITE(IUNIT,50) (FOR(J), J=1,9)
SAVNAME=FOR(1)
ISAVCNT=ICNT
GO TO 87
860 ICNT=ISAVCNT
GO TO 870
87 READ(21,50) (FOR(J), J=1,9)
870 IF(FOR(1).EO.XNSECT .OR. FOR(1).EO.YNSECT) GO TO 88
IF(FOR(1).EO.END) GO TO 110
WRITE(IUNIT,50) (FOR(J), J=1,9)
GO TO 87
88 IP1=NSECT(IELCNT)+JCNT
WRITE(IUNIT,90) IP1
90 FORMAT(*NSECT=*,I3,71X)
100 READ(21,50) (FOR(J), J=1,9)
IF(FOR(1).EO.END) GO TO 110
WRITE(IUNIT,50) (FOR(J), J=1,9)
GO TO 100

C SKIP THIS IF MORE THAN ONE DESIGN VARIABLE PER ELEMENT
APPENDIX D

BLDELDS (Cont.)

C 110 IF(ICNT .NE. 1) GO TO 120
C C CALL SUBROUTINE TO CREATE REMAINDER OF RUNSTREAM
C C CALL CRRS(ICNTDV,0,0,TNAME,NDF,NODVPE(IELCNT))
C ICNTDV = ICNTDV + NODVPE(IELCNT)
C GO TO 39
C C LOOP ON POSSIBLE NUMBER OF DESIGN VARIABLES PER ELEMENT
C C READ FROM UNIT 22
C C WRITE ON UNIT 20
C C 120 DO 160 I = 1,ICNT
C IF(I .NE. 1) WRITE(20,75)
C REWIND 22
C 130 READ(22,50) (FOR(J), J=1,9)
C IF(EOF(22)) 150,140
C 140 IF((FOR(1).EQ.XNSECT.OR.FOR(1).EQ.YNSECT) GO TO 141
C WRITE(20,50) (FOR(J), J=1,9)
C GO TO 130
C 141 WRITE(20,90) IP1
C GO TO 130
C C CALL SUBROUTINE TO CREATE REMAINDER OF RUNSTREAM
C C 150 CALL CRRS(ICNTDV,ICNT,I,TNAME,NDF,NODVPE(IELCNT))
C IP1=IP1+1
C 160 CONTINUE
C ICNTDV = ICNTDV + NODVPE(IELCNT)
C GO TO 39
C 170 WRITE(20,180)
C 180 FORMAT(* TOC 2*/ *(XQT EXIT*)
C 190 STOP
C END
C SUBROUTINE TABINPUT(ELNAME,NSCT,ICNT)
C C THIS SUBROUTINE CREATES TAB PROCESSOR INPUT FOR A RUNSTREAM
C DEPENDING UPON ELEMENTS USED.
C ALL DESIGN VARIABLES ARE SET TO UNITY.
C ICNT = NUMBER OF DESIGN VARIABLES FOR A PARTICULAR ELEMENT
C ICNT = 99 MEANS THE ELEMENT NAME IS BAD
C
C DIMENSION ELEMENT(21)
C DATA E21,E22,E23,E41,E43,E44/3HE21,3HE22,3HE23,3HE41,3HE43,3HE44/
C DATA E31,E33/3HE31,3HE33/
C DATA BA,BB,BC,SA,SB/2HBA,2HBB,2HBC,2HSA,2HSB/
C IF(ELNAME.EQ.E21) ELID=BA
APPENDIX D

BLDELD5 (Cont.)

IF(ELNAME.EQ.E22) ELID=BB
IF(ELNAME.EQ.E23) ELID=BC
IF(ELNAME.EQ.E41.OR.ELNAME.EQ.E43.OR.ELNAME.EQ.E31.OR.
1 ELNAME.EQ.E33) ELID=SA
IF(ELNAME.EQ.E44) ELID=SB
WRITE(20,10) ELID

10 FORMAT(2X,A2)

ICNT=99
IF(ELNAME.EQ.E23.OR.ELNAME.EQ.E41.OR.ELNAME.EQ.E44) GO TO 30
IF(ELNAME.EQ.E31) GO TO 30
IF(ELNAME.EQ.E21) GO TO 50
IF(ELNAME.EQ.E43.OR.ELNAME.EQ.E33) GO TO 100
IF(ELNAME.EQ.E22) GO TO 150
PRINT 20,ELNAME

20 FORMAT(* ELEMENT NAME *,A3,* DOES NOT EXIST*)

GO TO 210

C ELEMENT NAMES E23, E31, E41, E44
ICNT = 1

30 K=NSCT+1
WRITE(20,40) K

40 FORMAT(1X,I3,* 1.0*)

ICNT=1
GO TO 210

C ELEMENT NAME E21
ICNT = 4

50 DO 90 I = 1,4
DO 60 J = 1,10
ELEMENT(J)=0.0

60 CONTINUE
IF(I.EQ.1) ELEMENT(1)=1.0
IF(I.EQ.2) ELEMENT(3)=1.0
IF(I.EQ.3) ELEMENT(5)=1.0
IF(I.EQ.4) ELEMENT(6)=1.0
K=I+NSCT
WRITE(20,70) K,(ELEMENT(J),J=1,10)

70 FORMAT(* DSY *,I1,10(1X,F3.1))
WRITE(20,80)

80 FORMAT(2X,11(*0.0 *))

90 CONTINUE

ICNT=4
GO TO 210

C ELEMENT NAMES E33, E43
ICNT = 12
C
APPENDIX D

BLDELDs (Cont.)

100  DO 140 I = 1,12
     DO 110 J = 1,12
     ELEMENT(J) = 0.0
110  CONTINUE
     ELEMENT(1) = 1.E-06
     ELEMENT(3) = 1.E-06
     ELEMENT(6) = 1.E-06
     ELEMENT(7) = 1.E-06
     ELEMENT(9) = 1.E-06
     ELEMENT(12) = 1.E-06
     ELEMENT(15) = 1.
     K = I + NSCT
     IF(I.EQ.1) WRITE(20,115)
115  FORMAT(* FORMAT=UNCOPLED*)
     WRITE(20,120) K
120  FORMAT(2X,I2,1X,9(*1.0 *))
     WRITE(20,130) (ELEMENT(J), J = 1,6)
130  FORMAT(1X,6(1X,E10.2))
     WRITE(20,130) (ELEMENT(J), J = 7,12)
140  CONTINUE
     ICNT = 12
     GO TO 210

     C ELEMENT NAME E22
     ICNT = 21

150  DO 200 I = 1,21
     DO 160 J = 1,21
     ELEMENT(J) = 0.0
160  CONTINUE
     K = I + NSCT
     ELEMENT(I) = 1.
     WRITE(20,170) K, ELEMENT(1)
170  FORMAT(2X,I2,1X,E10.2)
     KK = 2
     DO 190 J = 2,6
     L = KK + J - 1
     WRITE(20,180) (ELEMENT(N), N = KK, L)
180  FORMAT(2X,6(F3.1,1X))
     KK = KK + J
190  CONTINUE
200  CONTINUE
     ICNT = 21
210  RETURN
END

SUBROUTINE CRRS(ICNTDV, ICNT, J, TNAME, NDF, NODVEL)

C THIS SUBROUTINE CREATES REMAINDER OF RUNSTREAM
C TO FIND DMDV AND DKDV.
USE NAMES

DMDV DIAG 0 I AND
DKDV SPAR 25 I

WHERE I = 1 TO NUMBER OF POSSIBLE DESIGN VARIABLES

DIMENSION BA(4),BB(21),SAEL(12),TNAME1(9),TNAME2(9)
DATA DK,DM,CK,CM,SK,SM/2HDK,2HDM,2HCK,2HCM,2HSK,2HSM/
DATA DV/2HDV/
DATA BB/2H11,2H21,2H22,2H31,2H32,2H33,2H41,2H42,2H43,
1 2H44,2H51,2H52,2H53,2H54,2H55,2H61,2H62,2H63,2H64,2H65,2H66/
DATA SAEL/2H11,2H12,2H22,2H31,2H33,2H41,2H43,2H44,2H45,2H46,2H51,2H52,
1 2H53,2H54,2H55,2H66/
DATA DEM/10HDEM DIAG 0/
DATA KSP/7HK SPAR /
DATA CHA/9HCHANGE 2, /
DATA COP/9HCOPY 1,2 /
DATA BA/2HIX,2HIY,2HDA,2HJO/
DATA TNAME1/4HDEF ,4HGD ,4HGTIT,4HELTS,4HNS ,4HKMAP,
1 4HAMAP,4HMASK,4HDIR /
DATA TNAME2/3*4H ,4HMASK,3*4H ,4HEFIL,4H /
DATA DMDV,DKDV,SPAR/4HDMDV,4HDKDV,6H SPAR /
DATA DIAG/6H DIAG /
JCNTDV = ICNTDV

WRITE(20,10)
10 FORMAT(*XQT E*I*rXQT EKS*I*rXQT TOPO*I*rXQT K*I*CXQT DCU*)

DISABLE DATA SETS

DO 30 I = 1,9
IF(I.EQ.8) TNAME1(8)=TNAME
IF(I.EQ.1 .OR.I.EQ.2 .OR.I.EQ.3 .OR.I.EQ.9) TNAME2(I)=TNAME
WRITE(20,20) TNAME1(I),TNAME2(I)
20 FORMAT(*DISABLE I,*A4,*2X,A4)
30 CONTINUE

SET UP ELEMENT NAMES FOR CHANGE AND COPY STATEMENTS

DDV*DV
DKD*DK
DDM*DM
IF(ICNT.NE.4) GO TO 31
DDV*BA(J)
GO TO 39
31 IF(ICNT.NE.12) GO TO 32
DDK*CK
APPENDIX D

BLDELS (Conc.)

DDM=CM
DDV=SAEL(J)
GO TO 39
32 IF(ICNT.NE.21) GO TO 39
DDK=SK
DDM=SM
DDV=BB(J)
39 DO 80 IJK = 1,NODVEL
   WRITE(20,40) COP,DEM
40 FORMAT(A9,A10,* 0*)
   WRITE(20,50) CHA,DEM,DDM,DDV,diag,JCNTDV
50 FORMAT(A9,A10,* 0*,2A2,A6,*0 *,I3)
   WRITE(20,60) COP,KSP,NDF
60 FORMAT(A9,A7,I2,* 0*)
   WRITE(20,70) CHA,KSP,NDF,DDK,DDV,SPAR,NDF,JCNTDV
70 FORMAT(A9,A7,I2,* 0*,2A2,A6,I2,I4)
   JCNTDV = JCNTDV+1
80 CONTINUE
RETURN
END

SUBROUTINE REMOVE

C THIS SUBROUTINE REMOVES THE LEADING BLANKS FROM EACH
C LINE IN THE RUNSTREAM.
C
DIMENSION DATIN(80)
DATA BLANK/IH /
REWIND 23
1 READ(23,2) (DATIN(I),I=1,80)
2 FORMAT(80A1)
   IF(EOF(23)) 7,3
3 IF(DATIN(1).NE.BLANK) GO TO 50
   DO 6 I = 2,80
      IF(DATIN(I).NE.BLANK) GO TO 60
6 CONTINUE
60 L=I-1
   K=80-I
   DO 4 J = 1,K
      DATIN(J) = DATIN(J+L)
4 CONTINUE
   K=K+1
   DO 5 J = K,80
      DATIN(J) = BLANK
5 CONTINUE
50 WRITE(21,2) (DATIN(J),J=1,80)
   GO TO 1
7 REWIND 21
RETURN
END
PROGRAM GNGRDRS(INPUT, TAPE30, TAPE31, TAPE10, TAPE5=INPUT, OUTPUT)

C THIS PROGRAM CREATES A REPEATABLE SPAR RUNSTREAM
C FOR CALCULATING DERIVATIVES WITH RESPECT TO
C DESIGN VARIABLES. THE SIZE OF THE RUNSTREAM VARIES
C WITH THE NUMBER OF LOAD CASES (NOLC) AND THE
C NUMBER OF DESIGN VARIABLES (NODV).
C
C THE RUNSTREAM IS OUTPUT ON UNIT 10

DIMENSION EL(999), NSECT(999), NUM(8), MFORM(5), NODVPE(999)
DATA NUM/I1, I1, I1, I1, I1, I1, I1, I1/
DATA BUCK, VIBR/4HBUCK, 4HVIBR/
DATA E21, E22, E33, E43/3HE21, 3HE22, 3HE33, 3HE43/
DATA MFORM(1)/10H(*Z11=UNIOI/
DATA MFORM(4)/10H(I1, I1, *Z*)/
DATA MFORM(5)/9H, I1, *)*/
DATA MFORM7/9HN(Z10, Z*, /
DATA MFORM8/5HN(Z*, /

C READ INPUT

C NOEL = NUMBER OF DIFFERENT ELEMENTS
C NOLC = NUMBER OF LOAD CASES
C NODV = TOTAL NUMBER OF DESIGN VARIABLES
C ISNOLC = STARTING NUMBER FOR DERIVATIVE LOAD CASES
C NDF = NUMBER OF DEGREES OF FREEDOM PER JOINT
C NORB = TYPE OF ANALYSIS (EX. BUCKLING)
C JOINTS = NUMBER OF JOINTS IN THE MODEL
C NODVPE = NUMBER OF DESIGN VARIABLES PER ELEMENT
C EL = NAMES OF ELEMENTS CONTAINING DESIGN VARIABLES
C (EX. E21)

READ(5, 60) NOLC, NODV, ISNOLC, JOINTS, NDF, NOEL, VORB
60 FORMAT(6(I4, I4), I4, A4)
READ(5, 61) (EL(I), NSECT(I), NODVPE(I), I=1, NOEL)
61 FORMAT(6(I4, A3, I4, A3), I4)
ISNOLC=ISNOLC-1
NDF=NDF*NDF

C DETERMINE IF THERE IS A E21, E22, E33, OR E43 ELEMENT

ICNTDV = 1
DO 168 I = 1, NOEL
JJ = NODVPE(I)
DO 160 J = 1, JJ
IF(EL(I).EQ.E21) GO TO 161
IF(EL(I).EQ.E22) GO TO 163
IF(EL(I).EQ.E43) GO TO 164
IF(EL(I).EQ.E33) GO TO 165
GO TO 166

160 CONTINUE
161 CONTINUE
163 CONTINUE
164 CONTINUE
165 CONTINUE
166 CONTINUE
APPENDIX D

GNDRGDRS (Cont.)

C CALL SUBROUTINE TO FIND DK/DV AND DM/DV FOR E21 ELEMENTS
C 161 CALL DKDVE21(ICNTDV,NDF)
       GO TO 166
C CALL SUBROUTINE TO FIND DK/DV AND DM/DV FOR E22 ELEMENTS
C 163 CALL DKDVE22(ICNTDV,NDF)
       GO TO 166
C CALL SUBROUTINE TO FIND DK/DV AND DM/DV FOR E43 ELEMENTS
C 164 CALL DKDVE43(ICNTDV,NDF)
       GO TO 166
C CALL SUBROUTINE TO FIND DK/DV FOR E33 ELEMENTS
C 165 CALL DKDVE33(ICNTDV,NDF)
166 ICNTDV = ICNTDV+1
160 CONTINUE
168 CONTINUE
       IF(VORB.EQ.VIBP) GO TO 320
       WRITE(10,112)
       FORMAT(*[XQT AUS*)
172 WRITE(10,115) JOINTS
175 FORMAT(* SYSVEC;UNIT VEC*/* I=1; J=1,**I8,**; 1.0*/
1 1 * DEFINE UN*UNIT VEC*)
       DO 200 I = 1,NODV
       WRITE(10,180) I,1
60 FORMAT(* DEFINE W*,I3,*-OMDV DIAG 0 *,I3)
200 CONTINUE
       DO 250 I = 1,NODV
       WRITE(10,220) I,1
220 FORMAT(* OBJF G*,I3,* 1 1*XTY(UN,W*,I3,**)*)
250 CONTINUE
       WRITE(10,260)
260 FORMAT(*[XQT DCU*)
       DO 300 I = 1,NODV
       WRITE(10,270) I
270 FORMAT(* PRINT 1 OBJF G*,I3)
300 CONTINUE
C CALL SUBROUTINE TO CREATE RUNSTREAM FOR DERIVATIVE
APPENDIX D

GNGRDRS (Cont.)

C CALCULATION
C
C FIND APPLIED FORCES AND MOMENTS
C
WRITE(10,2)
2 FORMAT(*XQT AUS** OUTLIB=3*)
DO 4 I = 1,NOLC
WRITE(10,3) I,I
3 FORMAT(* DEFINE F*,I2,**STAT DISP *,I2* 1*)
4 CONTINUE
DO 6 I = 1,NODV
WRITE(10,5) I,NDF,I
5 FORMAT(* DEFINE L*,I3,**DKDV SPAR *,I2,I4)
6 CONTINUE
DO 10 I = 1,NOLC
NWNOLC=I+ISNOLC
WRITE(10,7) NWNOLC
7 FORMAT(* ALPHA; CASE TITLE *,I3)
DO 9 J = 1,NODV
WRITE(10,8) J,I,J
8 FORMAT(1X,I3,* "LOAD CASE *,I2,* DERIVATIVE 1 DESIGN VARIABLE *
   1 I3)
9 CONTINUE
10 CONTINUE
IF(NODV.NE.1) WRITE(10,2003)
2003 FORMAT(* OUTLIB-4*)
IF(NODV.EQ.1) WRITE(10,6006)
6006 FORMAT(* OUTLIB=3 *)
DO 1010 I = 1,NOLC
NWNOLC=I+ISNOLC
DO 1005 J = 1,NODV
WRITE(10,1003) NWNOLC,J,J,I
1003 FORMAT(* APPL FORC *,I3,I3,** PRODUCT(-1.0 L*,I3,*1.0 F*,I2,*))
1005 CONTINUE
1010 CONTINUE
DO 15 I = 1,NOLC
NWNOLC=I+ISNOLC
IF(NODV.EQ.1) GO TO 50
IF(I.EQ.1) WRITE(10,2009)
2009 FORMAT(* INLIB=4** OUTLIB=3*)
DO 13 J = 1,NODV,9
ITOP=9
ICHK=NODV-J+1
IF(ICHK.LT.9) ITOP=ICHK
DO 40 K = 1,ITOP
L=K+J-1
WRITE(10,12) K,NWNOLC,L
12 FORMAT(* DEFINE Z*,I1,**APPL FORC *,I3,1X,I3)
APPENDIX D

GNGRDRS (Cont.)

40 CONTINUE
   IF(ITOP.EQ.1) GO TO 41
   MFORM(2)=MFORM7
   IF(J.EQ.1) MFORM(2)=MFORM8
   MFORM(3)=NUM(ITOP-1)
   WRITE(10,MFORM) (K,K=1,ITOP)
41 IF(ITOP.EQ.1) WRITE(10,42)
42 FORMAT(*Z11=UNION(Z10 Z1)*)
   WRITE(10,45)
45 FORMAT(* INLIB=3*/Z12=UNION(Z11)** DEFINE Z10=Z12*/
   1 * INLIB=4*)
13 CONTINUE
   WRITE(10,46) NWNOLC
46 FORMAT(* APPL FORC *,I3,* 1=UNION(Z10)*)
15 CONTINUE

C FIND STRESS AND DISPLACEMENT DERIVATIVES
C
50 DO 32 I = 1,NOLC
   WRITE(10,16) NODV
16 FORMAT(*[XQT SSOL]** RESET L1=1,L2=*,I3)
   NWNOLC=I+ISNOLC
   WRITE(10,17) NWNOLC
17 FORMAT(* RESET QLIB=3*/RESET SET=*,I3)
   WRITE(10,18)
18 FORMAT(* [XQT VPRT]** LB=3*)
   WRITE(10,19) NWNOLC
19 FORMAT(* PRINT APPL FORC *,I3)
   WRITE(10,20) NWNOLC
20 FORMAT(* PRINT STAT DISP *,I3)
   IF(VORB.EQ.BUCK) GO TO 400
   WRITE(10,21) NODV
21 FORMAT(*[XQT GSF]** RESET L1=1,L2=*,I3/RESET QLIB=3*)
   WRITE(10,22) NWNOLC
22 FORMAT(* RESET SET=*,I3)
   WRITE(10,23) NODV
23 FORMAT(*[XQT PSF]** RESET L1=1,L2=*,I3/RESET QLIB=3*)
   WRITE(10,24) NWNOLC
24 FORMAT(* RESET SET=*,I3)
   GO TO 25
C
C SET UP RUNSTREAM FOR BUCKLING ANALYSIS
C
400 IDV = 0
   DO 470 J = 1,NOEL
      NODVEL = NODVPE(J)
   DO 460 K = 1, NODVEL
      IDV = IDV+1
   WRITE(10,425)
APPENDIX D

GNGRDRS (Cont.)

425 FORMAT(*[XQT GSF/** RESET EMBED=1]*)
  WRITE(10,430) IDV,IDV,NWNOLC
430 FORMAT(* RESET L1=*,I3=*,L2=*,I3=*,SET=*,I3=*,QLIB=3*)
  WRITE(10,435)
435 FORMAT(*[XQT PSF]*)
  WRITE(10,430) IDV,IDV,NWNOLC
  WRITE(10,440)
440 FORMAT(*[XQT KG]*)
  WRITE(10,445)
445 FORMAT(*[XQT DCU]*)
  WRITE(10,450) NDF,NDF,IDV
450 FORMAT(* CHANGE 1,KG SPAR *,I3=*, 0,DKG SPAR *,I3=I4)
  WRITE(10,455) NDF,IDV
455 FORMAT(* COPY 1,3 DKG SPAR *,I3=I4)
460 CONTINUE
470 CONTINUE
25 WRITE(10,251)
251 FORMAT(*[XQT DCU]*)
  WRITE(10,26) NWNOLC,NWNOLC
26 FORMAT(* CHANGE 3,STAT DISP *,I3=*, 1,DDIS DISP *,I3=*, 1*)

C CHANGE DATA SET NAMES
C
IDV=0
DO 31 J=1,NOEL
  NSECT = NSECT(J)
  NODVEL = NODVPE(J)
DO 30 K=1,NODVEL
  DO 271 LL = 1,NOSECT
    IDV=IDV+1
  DO 27 KK = 1,NOEL
    IF(EL(KK).EQ.E21) GO TO 29
    IF(EL(KK).EQ.E22) GO TO 29
    IF(EL(KK).EQ.E43) GO TO 29
    IF(EL(KK).EQ.E33) GO TO 29
    WRITE(10,28) EL(KK),NWNOLC,IDV,EL(KK),NWNOLC,IDV
28 FORMAT(* CHANGE 3,STRS *,A3=2I4,*,DSTR *,A3=2I4)
    WRITE(10,285) EL(KK),NWNOLC,IDV
285 FORMAT(* COPY 3,4 DSTR *,A3=2I4)
    GO TO 27
C
C STORE BEAM CROSS DERIVATIVES
C
29 WRITE(10,290) EL(KK),NWNOLC,IDV,EL(KK),NWNOLC,IDV
290 FORMAT(* CHANGE 3,STRS *,A3=2I4,*,DFAM *,A3=2I4)
    WRITE(10,292) EL(KK),NWNOLC,IDV
292 FORMAT(* COPY 3,4 DFAM *,A3=2I4)
27 CONTINUE
271 CONTINUE
APPENDIX D

GNGRDRS (Conc.)

30 CONTINUE
31 CONTINUE
32 CONTINUE
320 IF(VORB.EQ.VIBR) WRITE(10,260)
   DO 333 I = 1,NODV
      WRITE(10,332) I
   332 FORMAT(* COPY 1,4 OBJF G*,I3)
   333 CONTINUE
   WRITE(10,33)
   33 FORMAT(* TOC 3** TOC 1** TOC 4**[XQT EXIT]*)
   STOP
END
SUBROUTINE DKDVE21

SUBROUTINE DKDVE21(NDVJIM, NDF)
DIMENSION X(20)
NAMELIST/LINKF/NDV,X

C THIS SUBROUTINE CREATES A SPAR RUNSTREAM TO CALCULATE
C
C DK DK DA DK DI DK DI DK DJ
X Y O
--- --- + --- + --- + --- + --- + ---
C DV DA DV DI DV DI DV DJ DV
I I X I Y I O I

C DM DM DA DM DI DM DI DM DJ
X Y O
--- --- + --- + --- + --- + --- + ---
C DV DA DV DI DV DI DV DJ DV
I I X I Y I O I

WRITE(10,1)
1 FORMAT(*[XOT AUS*)
WRITE(10,3)NDF,NDVJIM,NDF,NDVJIM
3 FORMAT(* DEFINE A1=DKDA SPAR *,I2,I4/* DEFINE A2=DKIX*
1 * SPAR *,I2,I4)
WRITE(10,4)NDF,NDVJIM,NDF,NDVJIM
4 FORMAT(* DEFINE A3=DKIY SPAR *,I2,I4/* DEFINE A4=DKJO*
1 * SPAR *,I2,I4)
WRITE(10,203)NDVJIM,NDVJIM
203 FORMAT(* DEFINE B1=DMDA DIAG 0 *,I3/* DEFINE B2=DMIX*
1 * DIAG 0 *,I3)
WRITE(10,204)NDVJIM,NDVJIM
204 FORMAT(* DEFINE B3=DMIY DIAG 0 *,I3/* DEFINE B4=DMJO*
1 * DIAG 0 *,I3)

C COMPUTE DA/DV
C
DADV=1.

C READ IN AND SET UP INITIALIZATION VALUES
C READ IN CONSTANTS FROM UNIT 30
C
READ(30,5) B10,B20,TO
5 FORMAT(3F10.3)
C
READ IN DESIGN VARIABLES FROM UNIT 31
C
READ(31,LINKF)
APPENDIX D

SUBROUTINE DKDVE21 (Cont.)

AREAO = (2.*B10 + B20)*T0
AREA = AREAO/X(2)
SCALE = SQRT(AREA/AREAO)
B1 = B10*SCALE
B2 = B20*SCALE
T = T0*SCALE
FTR = 0.5/SQRT(AREA/AREAO)
C = C/AREA

C COMPUTE FACTORS FOR DI1/DV, DI2/DV, DJ0/DV

DB1DV = B10*FTR
DB2DV = B20*FTR
DTDV = T0*FTR
DCDB1 = (B1*(B2 + 2.*T) - B2*B1 - 2.*C*T)/AREA
DCDB2 = (B1**2/2. - (B1-T)*(B1+T)/2. - C*T)/AREA
DCDT = (B1**2 + B2*T - C*(2.*B1 + B2))/AREA
DI2DB1 = (T*B1**2/2.)*(2.*T*(B1/2. - C)**2) +
1 (4.*T*B1*(B1/2. - C)***(5 - DCDB1)) +
2 (2.*B2*T*(C-T/2.)*DCDB1)
DI2DB2 = (-4.*T*B1*(B1/2. - C)*DCDB2) + (T**3/12.) +
1 (T*(C-T/2.)*B1**2/2. - B2**2*(C-T/2.)*DCDB2) -
DI2DT = (B1**3/6.) + (2.*B1*(B1/2. - C)**2) -
1 (4.*T*B1*(B1/2. - C)*DCDT) + (B2*T**2/4.) +
2 (B2*(C-T/2.)*B1**2/2.)*(2.*B2*T*(C-T/2.)*(DCDT - .5))
DJ0DB1 = 2.*T**3/3.
DJ0DB2 = T**3/3.
DJ0DT = (2.*B1 + B2)*T**2

C COMPUTE DI1/DV, DI2/DV, DJ0/DV

DI1DV = DI1DB1*DB1DV + DI1DB2*DB2DV + DI1DT*DTDV
DI2DV = DI2DB1*DB1DV + DI2DB2*DB2DV + DI2DT*DTDV
DJ0DV = DJ0DB1*DB1DV + DJ0DB2*DB2DV + DJ0DT*DTDV

C CREATE RUNSTREAM TO FIND DK/DV

WRITE(10, 106) DADV, DI1DV
WRITE(10, 107) DADV, DADV
106 FORMAT(8X, *S1 = SUM(*,E13.5,1X,*A1,*E13.5,1X,*A2)**)
107 FORMAT(8X, *S2 = SUM(*,E13.5,1X,*A3,*E13.5,1X,*A4)**)
WRITE(10, 108) NDF, NDVJIM
108 FORMAT(* DKDV SPAR *, I2, I4, ** = SUM(S1, S2)*)

C CREATE RUNSTREAM TO FIND DM/DV

87
Subroutine DKDVE21 (Conc.)

C

WRITE(10,206)DADV,DI1DV
WRITE(10,207)DI2DV,DJODV
206 FORMAT(8X,*T1=SUM(*,E13.5,1X,*B1,*E13.5,1X,*B2)*)
207 FORMAT(8X,*T2=SUM(*,E13.5,1X,*B3,*E13.5,1X,*B4)*)
WRITE(10,208)NDVJIM
208 FORMAT(* DMDV DIAG 0*,I3,*=SUM(T1,T2)*)
WRITE(10,209)
209 FORMAT(*XQT DCU*/* TDC 1*)
RETURN
END
APPENDIX D

DRVSTRS

PROGRAM DRVSTRS(INPUT=65,TAPE30,TAPE31,TAPF5(INPUT 1 ,TAPE15,TAPE16,OUTPUT=65,TAPE6(OUTPUT)

C THIS PROGRAM COMPUTES THE STRESSES AND STRESS DERIVATIVES FOR SPAR BEAM ELEMENTS USING SPAR LIBRARIES AS INPUT.

C THE STRESSES ARE COMPUTED FROM SPARLA USING DATA SET FAMS MASK I 1 WHERE I = 1 TO NOLC

C THE STRESSES ARE WRITTEN BACK ONTO SPARLA USING DATA SET STRS MASK I 1 WHERE I = 1 TO NOLC

C THE STRESS DERIVATIVES ARE COMPUTED FROM SPARLC USING DATA SET DFAM MASK I J WHERE I = ISTRTLCL TO ISTRTLCL+NOLC AND J = 1 TO NODV

C THE STRESS DERIVATIVES ARE WRITTEN BACK ONTO SPARLC USING DATA SET DSTR MASK I J WHERE I = ISTRTLCL TO ISTRTLCL+NOLC AND J = 1 TO NODV

COMMON KORE,KEVEN,Z(1)
DIMENSION NAME1(2),NAME2(2),EL(999),NODVPE(999)
DATA E21,E22,E33,E43/3HE21,3HEZ2,3HE33,3HE43/
DATA NAME1/4HFAMS,4HOFAMI
DATA NAME2/4HSTRS,4HDFAMI
DATA NAME3/4HSTRS,4HDFAMI

C READ INPUT VALUES

C NOEL IS THE NUMBER OF DIFFERENT ELEMENTS
C NDF IS THE DEGREES OF FREEDOM PER JOINT
C VORB IS THE TYPE OF ANALYSIS (EX. BUCKLING)
C JOINTS IS THE NUMBER OF JOINTS IN THE MODEL
C NOLC IS THE NUMBER OF LOAD CASES
C NODV IS THE NUMBER OF DESIGN VARIABLES
C ISTRTLCL IS THE STARTING NUMBER FOR THE DERIVATIVE LOAD CASES
C EL = NAMES OF ELEMENTS USED AS DESIGN VARIABLES
   (EX. E21)

C READ(5,1) NOLC,NODV,ISNOLC,JOINTS,NDF,NOEL,VORB
1 FORMAT(6(lX,I4),lX,A4)
   READ(5,2) (EL(I),NSECT,NODVPE(I),I=1,NOEL)
2 FORMAT(6(lX,A3,lX,I3,lX,I3))
   ISTRTLCL=ISNOLC-1
   CALL RSET(IL,0,0)

C LOOP ON THE NUMBER OF LOAD CASES
C DO 4 NCNTLC = 1,NOLC
C LOOP ON NUMBER OF ELEMENTS
C DO 10 ISTRS = 1,NOEL

89
APPENDIX D

DRVSTRES (Cont.)

C CHECK FOR E21, E22, E33, E43 ELEMENT
C
GO TO 10
9 NODVEL = NODVPE(ISTR)
C SET NUMBER OF STRESS OR STRESS DERIVATIVES TO BE WRITTEN
C
IF(EL(ISTR).EQ.E21. OR. EL(ISTR).EQ.E22) IWRDCNT = 8
IF(EL(ISTR).EQ.E43. OR. EL(ISTR).EQ.E33) IWRDCNT = 4
C LOOP ON NUMBER OF DESIGN VARIABLES PER ELEMENT
C
DO 90 JK = 1, NODVEL
REWIND 15
REWIND 16
C READ FAMS MASK FROM SPARLA AND COMPUTE STRS MASK
C
CALL RDATSET(4, NAME1(I), NCNTLC, 1, NEI, 1, 0, EL(ISTR))
REWIND 15
REWIND 16
C CALL SUBROUTINE TO WRITE STRESSES ON SPAR LIBRARY
C
C WRITE STRS MASK ON SPARLA
C
CALL WRTDATA(4, NAME2(I), NCNTLC, 1, NEI, EL(ISTR), IWRDCNT)
90 CONTINUE
10 CONTINUE
ICNTDV = 1
C LOOP ON NUMBER OF ELEMENTS
C
DO 3 ISTRS=1, NOEL
GO TO 3
19 DO 30 ICNTDV = 1, NODV
IBEAM = 0
NODVEL = NODVPE(ICNTDV)
C SET SWITCH FOR BEAM ELEMENT
C
IF(EL(ICNTDV).EQ.E21. OR. EL(ICNTDV).EQ.E22) IBEAM = 1
N3 = NCNTLC + ISTRTLC
APPENDIX D

DRVSTRS (Cont.)

C SET NUMBER OF STRESSES AND STRESS DERIVATIVES TO BE WRITTEN
C
IF(EL(ICNTDV).EQ.E21) IWRDCNT = 8
IF(EL(ICNTDV).EQ.E43) IWRDCNT = 4
C
C LOOP ON NUMBER OF DESIGN VARIABLES PER ELEMENT
C
DO 20 JK = 1,NODVEL
C
C READ DFAM MASK FROM SPARLC AND COMPUTE DSTR MASK
C
REWIND 15
CALL RDATSETC4,NAME1(2),N3,ICNTDV,NE1,0,IBEAM,EL(ISTR))
REWIND 15
C
C WRITE DSTR MASK ON SPARLC
C
CALL WRTDATAC4,NAME2(2),N3,ICNTDV,NE1,EL(ISTR),IWRDCNT)
20 CONTINUE
30 CONTINUE
3 CONTINUE
4 CONTINUE
CALL FIN(0,0)
STOP
SUBROUTINE WRTDATACNU,N1,N3,ISTRs,NE1,X2,IWRDCNT)
C
C SUBROUTINE TO WRITE STRESS AND STRESS DERIVATIVE
C DATA SETS BACK INTO SPAR LIBRARIES
C
COMMON KORE,KEVEN,Z(1)
C
C SET UP BLOCK SIZES
C IF NWDO GT OPEN CORE SIZE , USE MORE THAN ONE BLOCK
C
NWDO = NE1*IWRDCNT
LB3 = NWDO
IF(NWDO.LT.KORE) GO TO 61
LB3 = NWDO/2
61 ISW=0
IF(NWDO.LT.KORE) ISW=1
CALL DALCNU(0,0,Z(1),KORE,1,KADR3,IERR,NWDO,NE1,LB3,-1,
1 N1,X2,N3,ISTR)
62 IN=1
63 IN=1+IWRDCNT-1
IF(IN.GT.LB3) GO TO 66
C
C READ STRESSES OR STRESS DERIVATIVES OFF UNIT 15
C
APPENDIX D

DRVSTRS (Conc.)

READ(15) (Z(I),I=I1,I2)
IF(EOF(15)) 66,65
   I1=I1+1WRDCT
GO TO 63
C
WRITE STRESSES OR STRESS DERIVATIVES ONTO SPAR LIBRARY
C
66 CALL RIOCNU,10,2,KADR3,Z(1),LB3)
   IF(ISW.EQ.1) GO TO 67
   ISW=1
   LB3=NWD3-LB3
GO TO 62
67 RETURN
END
SUBROUTINE RDATSETCNU,N1,N3,ISTRS,NE1,ISW,IBEAM,X2)
C
C SUBROUTINE TO READ SPAR LIBRARY, STORE DATA, AND
C COMPUTE STRESSES OR STRESS DERIVATIVES
C
COMMON KORE,KEVEN,Z(1)
DATA E21,E22,E33,E43/3HE21,3HE22,3HE33,3HE43/
C
C SET UP BLOCK SIZE
C
CALL DAL(NU,10,Z(1),KORE,1,KADR1,IERR,NWD1,NE1,LB1,ITYPE,N1,
   1 X2,N3,ISTRS)
   NI1=LB1/NE1
   KCNT=0
   NBLK1=NWD1/LB1
   IF(NBLK1*LB1,NE,NWD1) NBLK1=NBLK1+1
   DO 6 J = 1,NBLK1
     NLB1=LB1
     IF(J.EQ.NBLK1) NLB1=NWD1-(NBLK1-1)*LB1
C
READ DATA FROM SPAR LIBRARY
C
CALL RIO(NU,20,2,KADR1,Z(1),NLB1)
   DO 30 JCNT=1,NLB1,NI1
     KCNT=KCNT+1
C
CALL SUBROUTINE TO COMPUTE STRESSES AND STRESS DERIVATIVES
C
IF(X2.EQ.E21.OR.X2.EQ.E22) CALL BMSTRS(ISW,KCNT,JCNT,IBEAM)
   IF(X2.EQ.E43.OR.X2.EQ.E33) CALL PLTSTRS(ISW,KCNT,JCNT)
30 CONTINUE
6 CONTINUE
RETURN
END
SUBROUTINE BMSTRS(ISW, KCNT, JCNT, IBEAM)
COMMON KORE, KEVEN, Z(1)
DIMENSION DY1DV(4), DY2DV(4), Y(4, 2), S(8), F3(X), XM1(2), XM2(2)
DIMENSION X(20)
NAMELIST/LINKF/NDV, X
JM1 = JCNT-1
C
C STORE FORCES AND MOMENTS IF ISTRS EQ 1
C STORE DERIVATIVES OF FORCES AND MOMENTS IF ISTRS EQ 2
C
F3(1) = -Z(JM1+13)
XM1(1) = -Z(JM1+14)
XM2(1) = -Z(JM1+15)
F3(2) = Z(JM1+19)
XM1(2) = Z(JM1+20)
XM2(2) = Z(JM1+21)
IF(KCNT .NE. 1) GO TO 31
IF(ISW .NE. 1) GO TO 31
C
C STORE AREA, MOMENTS OF INERTIA, AND Y VALUES IF
C FIRST TIME THROUGH
C
XI1 = Z(JM1+25)
XI2 = Z(JM1+27)
ICNT = 38
DO 3 K = 1, 4
DO 2 L = 1, 2
ICNT = ICNT + 1
Y(K, L) = Z(JM1+ICNT)
2 CONTINUE
3 CONTINUE
C
C FIND DERIVATIVE OF Y VALUES IF NEEDED
C
C READ IN CONSTANT NUMBERS FROM UNIT 30
C
READ(30, 10) B10, B20, TO
APPENDIX D

SUBROUTINE BMSTRS (Cont.)

10 FORMAT(3F10.3)
C READ IN DESIGN VARIABLES FROM UNIT 31
C READ(31, LINKF)
C CALCULATE DY/DV
C
A0=(2.*B10+B20)*T0
A=A0/X(2)
SCALE=SQRT(A/A0)
B1=B10*SCALE
B2=B20*SCALE
T=T0*SCALE
FTR=0.5/SQRT(A*A0)
C=C/A
DB1DV=B10*FTR
DB2DV=B20*FTR
DTDV=T0*FTR
DCDB1=(B1*(B2+2.*T)-B2*B1-2.*C*T)/A
DCDB2=(B1**2/2.-B1-T)*(B1+T)/2.-C*T)/A
DCDT=(B1**2+B2*T-C*(2.*B1+B2))/A
DCDV=(DCDB1*DB1DV)+(DCDB2*DB2DV)+(DCDT*DTDV)
DY1DV(1)=-DB1DV+DCDV
DY1DV(2)=DCDV
DY1DV(3)=DCDV
DY1DV(4)=DY1DV(1)
DY2DV(1)=.5*DB2DV+DTDV
DY2DV(2)=DY2DV(1)
DY2DV(3)=-DY2DV(1)
DY2DV(4)=DY2DV(3)
31 ICNT=0
DO 200 II = 1,2
DO 100 I = 1,4
ICNT=ICNT+1
C COMPUTE STRESSES OR STRESS DERIVATIVES
C
S(ICNT)=(F3(II)/A)+((XM1(II)/XI1)*Y(I,2))-((XM2(II)/XI2)*Y(I,1))
C IF AREA OF BEAM IS A CONTRIBUTING FACTOR TO STRESS
C DERIVATIVES AND ISTRS EQ 1 THEN CALCULATE THE FACTOR
C AND STORE ON UNIT 16
C
IF(ISW.NE.1) GO TO 95
DFAC=((XM1(II)/XI1)*DY2DV(I))-((XM2(II)/XI2)*DY1DV(I))
WRITE(16) DFAC

94
APPENDIX D

SUBROUTINE BMSTCRS (Conc.)

GO TO 100

C
C IF AREA OF BEAM IS A CONTRIBUTING FACTOR TO STRESS
C DERIVATIVES AND ISTRS EQ 2 THEN READ FACTOR OFF UNIT 16
C AND ADD IT TO STRESS DERIVATIVE
C
95 IF(IBEAM.EQ.0) GO TO 100
READ(16) DFAC
S(ICNT)*S(ICNT)+DFAC
100 CONTINUE
200 CONTINUE

C
C WRITE STRESSES ON UNIT 15
C
WRITE(15) S
RETURN
END
APPENDIX E

DATA FILES

The following are sample listings of data files.

INPUT DATA FILES

PCONPAR, CONPAR

$CONPAR
IPRINT=5,
NDV=3,
ITMAX=20,
NCON=190,
NFGD=0,
NSIDE=1,
ICNDIR=0,
NSCAL=4,
LB0BJ=0,
ITRM=0,
FDCH=0.,
FDCHM=0.,
CT=0.,
CTMIN=0.025,
CTL=0.,
CTLMIN=0.,
THETA=0.125,
PH1=5.,
DELFUN=0.05,
DABFUN=0.0000000001,
ISC(1)=190*0.0,
N1=20,
N2=400,
N3=200,
N4=200,
N5=400,
ALPHAX=0.0,
ABOBJ1=0.0,
IGOTO=0,
VLB(1)=0.005,
VLB(2)=0.1,
VLB(3)=1.0,
VUB(1)=1.0,
VUB(2)=10.0,
VUB(3)=10.0,
$END

(See ref. 6 for a description of the parameters.)
APPENDIX E

PSTART,STARTX

\$STARTX
X(1)=0.05,
X(2)=1.0,
X(3)=4.0,
\$END

Starting values for three design variables.

INPT

1 3 100 80 5 3 STAT
E23 1 1 E21 1 1 E41 1 1

Card type 1 Number of load cases = 1, Number of design variables = 3,
Starting number for load cases = 100, Number of joints = 80,
Number of nonzero degrees of freedom = 5,
Number of different element types = 3,
Type of analysis = STAT

Card type 2 Element names E23, E21, E41
Last section number for each element 1,1,1
Number of design variables per element 1,1,1
APPENDIX E

CNT

\$CNT
OBJ1 = 10.0,
OBJ2 = 50.0,
OBJ3 = 10.0,
TOL = .5E-01,
\$END

Initial objective functions and tolerance

ENDN

\$EPIN
E23AL=2000.,
E21AL=2000.,
E41AL=2000.,
NSE23=58,
NSE21=76,
NSE41=56,
\$END

E23AL, E21AL, E41AL are limits on the design variables
NSE23, NSE21, NSE41 are the number of each type of element

CONS

Input values to calculate the derivative of the stiffness and mass matrices with respect to the design variable

B1 = 12
B2 = 30
T1 = 2

Cross-sectional dimensions of a beam
(See ref. 2 for more detail.)
APPENDIX E

MODEL DATA FILES

NRRS (nonrepeatable SPAR runstream)

[XQT TAB
START 80, 5% ROTATIONS ABOUT Y EXCLUDED
TITLE" FUSELAGE MODEL, FSPAR1
TEXT
" MEMBRANE-ROD-BEAM FUSELAGE MODEL
" NONREPEATABLE PART
JLOC$ FUSELAGE DIA. 800. CM., LENGTH 800. CM.
FORMAT=2 CYLINDRICAL COORDINATES
1 400. 0. 0. 400. 337.5 0. 16 15
16 400. 0. 800. 400. 337.5 800.
MREF
FORMAT=2
1 -2 0. 0. 10000000.
2 1 0. 0. 10000000.
MATC
1 .72+6 0.3 .0028 22.-6$ AL-ALLOY, METRIC UNITS
E23 SECTION PROPERTIES % ROD ELEMENTS
1 4.168$ AREA OF THE RODS
E21 SECTION PROPERTIES % BEAM ELEMENTS
DSY 1 16804. 0. 1262.7 0. 108. 144. 0. 6.0784 0. 0.
0. 0. 0. -8.7778 17. 3.2222 17. 3.2222 -17. -8.7778 -17.
SHELL SECTION PROPERTIES
1 0.1$ SKIN THICKNESS
CONSTRAINT CASE 1
ZERO 1,2,3;1,16$ CANTILEVER THE FUSELAGE
ZERO 1,2,3,4,6;36,38

Input to TAB processor to tabulate material properties, structural geometry, constraint conditions, etc.
APPENDIX E

Input to ELD processor to establish element connectivity. (See ref. 5 for more detail.)
APPENDIX E

NGRS (repeatable SPAR runstream)

[XOT TAB
  TITLE "FUSELAGE MODEL"
  TEXT
  "MEMBRANE ROD BEAM FUSELAGE MODEL"
  "REPEATABLE PART WITHOUT GRADIENTS"
  UPDATE = 1
  $ MERGE NEW PROPERTIES HERE.
  UPDATE = 0
[XOT TOPO]
[XOT E]
[XOT EKS]
[XOT K]
[XOT INV]
[XOT M]
RESET G=981.
[XOT AUS
  SYSVECEAPPLIED FORCES 1
  I=1; J=65;69;77; 10000. -20000. 20000. }
  Applied loading definition
  SYSVECEUNIT VEC
  I=1; J=80; 1.0
  DEFINE WT=DEM DIAG 0 0
  DEFINE UN=UNIT VEC
  OBJFUN=XTY(UN,WT)
[XOT DCU
  PRINT 1 OBJFUN
[XOT SSOL
[XOT GSF
[XOT PSF ]
[XOT VPRT
  PRINT APPL FORC 1 1
  PRINT STAT DISP 1 1
[XOT DCU
  PRINT 1 STAT DISP 1 1
  PRINT 1 STRS E21 1 1
  PRINT 1 STRS E23 1 1
  PRINT 1 STRS E41 1 1
  TOC 1
COPY 1,4 OBJF AUS 1 1
COPY 1,4 STRS E21 1 1
COPY 1,4 STRS E23 1 1
COPY 1,4 STRS E41 1 1
TOC 4
[XOT EXIT

(See ref. 5 for more details.)
APPENDIX E

BLOCK, BLK (Cont.)
APPENDIX E

BLOCK, BLK (Cont.)
APPENDIX E

SPFPOUT

<table>
<thead>
<tr>
<th>Rod data</th>
<th>E23 SECTION PROPERTIES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.657</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>E21 SECTION PROPERTIES</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSY 1 40393.3008</td>
</tr>
<tr>
<td>0.0000</td>
</tr>
<tr>
<td>3035.1865</td>
</tr>
<tr>
<td>0.0000</td>
</tr>
<tr>
<td>167.4450</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Beam data</th>
<th>SHELL SECTION PROPERTIES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SHELL SECTION PROPERTIES</td>
</tr>
<tr>
<td>1</td>
<td>.125</td>
</tr>
</tbody>
</table>

| Thickness, cm |
|              |

<table>
<thead>
<tr>
<th>Cross-sectional area, cm² (inverse)</th>
</tr>
</thead>
</table>

(See ref. 5 for more details.)
APPENDIX E

RSOUT (nonrepeatable part)

[EXOT TAB
 UPDATE=1
 BC
 2 1.0
 BA
 DSY 2 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 DSY 3 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 DSY 4 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 DSY 5 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 SA
 2 1.0
 UPDATE=0
[EXOT DCU
 COPY 1,2
 DISABLE 1,DEF E23
 DISABLE 1,DEF E21
 DISABLE 1,DEF E41
 DISABLE 1,GD E23
 DISABLE 1,GD E21
 DISABLE 1,GD E41
 DISABLE 1,GTIT E23
 DISABLE 1,GTIT E21
 DISABLE 1,GTIT E41
 DISABLE 1,DIR E23
 DISABLE 1,DIR E21
 DISABLE 1,DIR E41
 DISABLE 1,NS
 DISABLE 1,ELTS NAME
 DISABLE 1,ELTS NNOD
 DISABLE 1,ELTS ISCT
[EXOT ELD
 E23 $ROD EEMENTS
 NSECT= 2
 NREF=2
 1 17 1 4 3 1 $
 4 20$
 5 21$
 6 22$
 52 68$
 53 69$
 54 70$
 7 23 1 4 10 1$
 [EXOT E
 [EXOT EKS
 [EXOT TOPO
APPENDIX E

RSOUT (nonrepeatable part cont.)

RESET MAXSUB=2500
[XOT K
[XOT DCU
  DISABLE 1,DEF E23
  DISABLE 1,GD E23
  DISABLE 1,GTIT E23
  DISABLE 1,ELTS MASK
  DISABLE 1,NS
  DISABLE 1,KMAP
  DISABLE 1,AMAP
  DISABLE 1,E23 EFL
  DISABLE 1,DIR E23
COPY 1,2 DEM DIAG 0 0
CHANGE 2,DFM DIAG 0 0,DMDV DIAG 0 1
COPY 1,2 K SPAR 25 0
CHANGE 2,K SPAR 25 0,DKDV SPAR 25 1
[XOT ELD
E21
NSEC= 2
NREF=1
1 2 2 16 2 16 $16
49 50 2 16 2 16 $16
33 34$
34 35$
39 40$
40 41$
41 42$
42 43$
43 44$
44 45$
45 46$
46 47$
47 48$
48 33$
[XOT E
[XOT EKS
[XOT TOPO
RESET MAXSUB=2500
[XOT K
[XOT DCU
  DISABLE 1,DEF E21
  DISABLE 1,GD E21
  DISABLE 1,GTIT E21
  DISABLE 1,ELTS MASK
  DISABLE 1,NS
  DISABLE 1,KMAP
  DISABLE 1,AMAP
  DISABLE 1,E21 EFL
  DISABLE 1,DIR E21

121
COPY 1,2 DEM DIAG 0 0
CHANGE 2,DEM DIAG 0 0,DMIX DIAG 0 2
COPY 1,2 K SPAR 25 0
CHANGE 2,K SPAR 25 0,DKIX SPAR 25 2
[EXQT ELD
E21
NSECT= 3
NREF=1
1 2 2 16 2 16 $
49 50 2 16 2 16$
33 34$
34 35$
39 40$
40 41$
41 42$
42 43$
43 44$
44 45$
45 46$
46 47$
47 48$
48 33$
[EXQT E
[EXQT EKS
[EXQT TOPO
RESET MAXSUB=2500
[EXQT K
[EXQT DCU
DISABLE 1,DEF E21
DISABLE 1,GD E21
DISABLE 1,GTIT E21
DISABLE 1,ELTS MASK
DISABLE 1,NS
DISABLE 1,KMAP
DISABLE 1,AMAP
DISABLE 1,E21 EFIL
DISABLE 1,DIR E21
COPY 1,2 DEM DIAG 0 0
CHANGE 2,DEM DIAG 0 0,DMIX DIAG 0 2
COPY 1,2 K SPAR 25 0
CHANGE 2,K SPAR 25 0,DKIX SPAR 25 2
[EXQT ELD
E21
NSECT= 4
NREF=1
1 2 2 16 2 16 $
49 50 2 16 2 16$
33 34$
34 35$
APPENDIX E

RSOUT (nonrepeatable part cont.)

39 40$
40 41$
41 42$
42 43$
43 44$
44 45$
45 46$
46 47$
47 48$
48 33$
[XOT E
[XOT EKS
[XOT TOPO
RESET MAXSUB*2500
[XQT K
[XOT DCU
DISABLE 1,DEF E21
DISABLE 1,GO F21
DISABLE 1,GTIT E21
DISABLE 1,ELTS MASK
DISABLE 1,NS
DISABLE 1,KMAP
DISABLE 1,AMAP
DISABLE 1,E21 EFIL
DISABLE 1(DIR E21
COPY 1,2 DEM DIAG 0 0
CHANGE 2,DEM DIAG 0 0,DMDA DIAG 0 2
COPY 1,2 K SPAR 25 0
CHANGE 2,K SPAR 25 0,DKDA SPAR 25 2
[XOT ELD
E21
NSECT= 5
NREF=1
1 2 2 16 2 16$
49 50 2 16 2 16$
33 34$
34 35$
39 40$
40 41$
41 42$
42 43$
43 44$
44 45$
45 46$
46 47$
47 48$
48 33$
[XOT E
[XOT EKS
APPENDIX E

RSOUT (nonrepeateable part conc.)

[XQT TOPO
RESET MAXSUB=2500
[XQT K
[XQT DCU
DISABLE 1,DEF E21
DISABLE 1,GD E21
DISABLE 1,GTIT E21
DISABLE 1,ELTS MASK
DISABLE 1,NS
DISABLE 1,KMAP
DISABLE 1,AMAP
DISABLE 1,E21 EFIL
DISABLE 1,DFF E21
COPY 1,2 DEM DIAG 0 0
CHANGE 2,DEM DIAG 0 0,DMJO DIAG 0 2
COPY 1,2 K SPAR 25 0
CHANGE 2,K SPAR 25 0,DKJO SPAR 25 2
[XQT ELD
E41 $ MEMBRANE PANELS
NSELECT= 2
1 17 18 2 2 16 1 $
49 65 66 50 2 16 1 $
17 33 34 18 1 1 2 2 16 $
23 39 40 24 1 1 9 2 2 16 $
32 48 33 17 $
48 64 49 33 $
[XQT E
[XQT EKS
[XQT TOPO
RESET MAXSUB=2500
[XQT K
[XQT DCU
DISABLE 1,DEF E41
DISABLE 1,GD E41
DISABLE 1,GTIT E41
DISABLE 1,ELTS MASK
DISABLE 1,NS
DISABLE 1,KMAP
DISABLE 1,AMAP
DISABLE 1,E41 EFIL
DISABLE 1,DFF E41
COPY 1,2 DEM DIAG 0 0
CHANGE 2,DEM DIAG 0 0,DMDV DIAG 0 3
COPY 1,2 K SPAR 25 0
CHANGE 2,K SPAR 25 0,DKDV SPAR 25 3
TOC 2
[XQT EXIT
APPENDIX E
RSOUT (repeatable part)

[XQT AUS
DEFINE A1=DKDA SPAR 25 2
DEFINE A2=DKIX SPAR 25 2
DEFINE A3=DKIY SPAR 25 2
DEFINE A4=DKJO SPAR 25 2
DEFINE B1=DMDA DIAG 0 2
DEFINE B2=DMIX DIAG 0 2
DEFINE R3=DMIY DIAG 0 2
DEFINE B4=DMJO DIAG 0 2
S1=SUM( .10000+01 A1, .31119+03 A2)
S2=SUM( .23383+02 A3, .26667+01 A4)
DKDV SPAR 25 2=SUM(S1,S2)
T1=SUM( .10000+01 B1, .31119+03 B2)
T2=SUM( .23383+02 B3, .26667+01 B4)
DMDV DIAG 0 2=SUM(T1,T2)
[XQT DCU
TOC 1
[XQT AUS
SYSVEC;UNIT VEC
T=1; J=1, 80; 1.0
DEFINE UN=UNIT VEC
DEFINE W1=DMDV DIAG 0 1
DEFINE W2=DMDV DIAG 0 2
DEFINE W3=DMDV DIAG 0 3
OBJF G1 1 1=XTY(UN,W1)
OBJF G2 1 1=XTY(UN,W2)
OBJF G3 1 1=XTY(UN,W3)
[XQT DCU
PRINT 1 OBJF G1
PRINT 1 OBJF G2
PRINT 1 OBJF G3
[XQT AUS
OUTLIB=3
DEFINE F1=STAT DISP 1 1
DEFINE L1=DKDV SPAR 25 1
DEFINE L2=DKDV SPAR 25 2
DEFINE L3=DKDV SPAR 25 3
ALPHA; CASE TITLE 100
1 "LOAD CASE 1 DERIVATIVE 1 DESIGN VARIABLE 1
2 "LOAD CASE 1 DERIVATIVE 1 DESIGN VARIABLE 2
3 "LOAD CASE 1 DERIVATIVE 1 DESIGN VARIABLE 3
OUTLIB=4
APPL FORC 100 1=PRODUCT(-1.0 L1,1.0 F1)
APPL FORC 100 2=PRODUCT(-1.0 L2,1.0 F1)
APPL FORC 100 3=PRODUCT(-1.0 L3,1.0 F1)
INLIB=4
OUTLIB=3
DEFINE Z1/APPL FORC 100 1
DEFINE Z2/APPL FORC 100 2
125
APPENDIX E

RSOUT (repeatable part conc.)

DEFINE Z3=APPL FORC 100 3
Z11=UNION(Z1 Z2 Z3)
INLIB=3
Z12=UNION(Z11)
DEFINE Z10=Z12
INLIB=4
APPL FORC 100 1=UNION(Z10)

XQT SSDL
RESET L1=1,L2=3
RESET QLIB=3
RESET SET=100
XQT VPR
LIB=3
PRINT APPL FORC 100
PRINT STAT DISP 100
XQT GSF
RESET L1=1,L2=3
RESET QLIB=3
RESET SET=100
XQT PSF
RESET L1=1,L2=3
RESET QLIB=3
RESET SET=100
XQT DCU
CHANGE 3,STAT DISP 100 1,DDIS DISP 100 1
CHANGE 3,STRS E23 100 1,DSTR F23 100 1
COPY 3,4 DSTR E23 100 1
CHANGE 3,STRS E21 100 1,DFAM E21 100 1
COPY 3,4 DFAM E21 100 1
CHANGE 3,STRS E41 100 1,DSTR F41 100 1
COPY 3,4 DSTR E41 100 1
CHANGE 3,STRS E23 100 2,DSTR F23 100 2
COPY 3,4 DSTR E23 100 2
CHANGE 3,STRS E21 100 2,DFAM E21 100 2
COPY 3,4 DFAM E21 100 2
CHANGE 3,STRS E41 100 2,DSTR F41 100 2
COPY 3,4 DSTR E41 100 2
CHANGE 3,STRS E23 100 3,DSTR F23 100 3
COPY 3,4 DSTR E23 100 3
CHANGE 3,STRS E21 100 3,DFAM E21 100 3
COPY 3,4 DFAM E21 100 3
CHANGE 3,STRS E41 100 3,DSTR F41 100 3
COPY 3,4 DSTR E41 100 3
COPY 1,4 OBJF G1
COPY 1,4 OBJF G2
COPY 1,4 OBJF G3
TOC 3
TOC 1
TOC 4
XQT EXIT
APPENDIX E

EDIT DATA FILES

EDPASS1

A
> $PASSAGE
NPASS=1,
$END>

Creates PASS file, setting NPASS to 1 for first pass

EDPASS2

RS:2//1/
END

Changes NPASS from 2 to 1 when restarting

EDIT1

F://EXQT ELD/
F://EXQT;/2
D;*
R
F://EXQT ELD/
E;*
R
D;*
R
ADD
$
END

Edits out all information from NRBS file except
ELD input in nonrepeatable part

EDIT2

RS: // , , ;*  
RS: // , , ;* 
A://TOPO::* 
>RESET MAXSUB=500> 
END

Removes blanks from RSOUT file
in nonrepeatable part
APPENDIX E

MERGFP

MERGE: /SPFPOUT/, /$ MERGE NEW;/; R
RS: /E+/, /+; *
RS: /E-, /-; *
END

SPFPOUT file is merged into SPARS. E+ and E- are changed to + and - because SPAR does not accept E's in input

EDGRDS

RS: /E+/, /+; *
RS: / . , / . , / ; *
RS: / . , 0 / . , 0 / ; *
RS: / . , - / . , - / ; *
RS: / / , / ; *
RS: /E-, /-; *
RS: / / , / ; *
RS: / / , / ; *
RS: / W, / W, / ; *
RS: / G, / G, / ; *
RS: / W, / W, / ; *
RS: / F, / F, / ; *
RS: / L, / L, / ; *
END

Remove blanks from RSOUT in repeatable part

(See ref. 10 for more detail on edit commands.)
REFERENCES


### 16. Abstract

This paper describes a particular implementation of the programming structural synthesis system (PROSSS). This software system combines a state-of-the-art optimization program, a production-level structural analysis program, and user-supplied, problem-dependent interface programs. These programs are combined using standard command language features existing in modern computer operating systems. PROSSS is explained in general with respect to this implementation along with the steps for the preparation of the programs and input data. Each component of the system is described in detail with annotated listings for clarification. The components include options, procedures, programs and subroutines, and data files as they pertain to this implementation. The paper concludes with an example exercising each option in this implementation to allow the user to anticipate the type of results that might be expected.