NASA Technical Memorandum 83253

AN IMPLEMENTATION OF THE DISTRIBUTED PROGRAMING STRUCTURAL SYNTHESIS SYSTEM (PROSSS)

JAMES L. ROGERS, JR.

DECEMBER 1981

National Aeronautics and Space Administration

Langley Research Center
Hampton, Virginia 23665

AN IMPLEMENTATION OF THE DISTRIBUTED PROGRAMING STRUCTURAL SYNTHESIS SYSTEM (PROSSS)

James L. Rogers, Jr.
NASA Langley Research Center
Hampton, Virginia

INTRODUCTION

Numerous approaches have been documented for combining optimization techniques with an analysis capability (e.g., refs. 1-4). The PROgraming Structural Synthesis System (PROSSS), described in reference 4, was first implemented entirely on a CDC CYBER computer. This implementation is described in reference 5. PROSSS was then distributed between a mainframe and a minicomputer (ref. 6) to provide the user easier access to intermediate results, more control over the flow of the problem, and an interactive modeling and data generation capability (ref. 7).

This document describes the implementation of the distributed version of PROSSS, the second step in a series of implementations. After an overview which explains PROSSS in general with respect to this implementation, the remainder of this document is devoted to describing (1) each component of the system, (2) how to examine the intermediate results, (3) how to restart the system, and (4) results from a sample problem exercising the options of this implementation. Appendices contain listings to clarify the descriptions of the components of the system contained in the body of this paper.

It is not the purpose of this paper to be a self-contained user's guide for PROSSS. Its purpose is to demonstrate a method for implementing a flexible software system that combines large complex programs with small,

user-supplied, problem-dependent programs and distributes their execution between a mainframe and a minicomputer.

OVERVIEW OF THIS IMPLEMENTATION

This distributed implementation of the PROgraming Structural Synthesis System (PROSSS) combines a general purpose finite-element computer program for structural analysis (SPAR, ref. 8), a state-of-the-art optimization program (CONMIN, ref. 9), and several user-supplied, problem-dependent computer programs. The results are flexibility of the optimization procedure organization and versatility of the formulation of constraints and design variables.

Figure 1 is a flow chart of the analysis-optimization process for this implementation. The process results in a minimized objective function, typically the mass. 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 is referred to as the repeatable part of PROSSS. However, some of the analysis, such as model definition, need only be done one time and the results are 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, simple FORTRAN programs to interface between the analysis and optimization programs. One of these programs, the front processor, converts the design variables output from the optimizer into a suitable format for input into the analyzer. Another, the end processor, retrieves the behavior variables and, optionally, their gradients from the analysis program and evaluates the objective function and constraints and optionally 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 which finite elements are being used in the model.

The analysis is always performed on the mainframe, in this implementation a CDC 6600, because the CPU on the mainframe is much faster than that of the minicomputer. The end processor is also executed on the mainframe to reduce the large amounts of data generated by the analysis program to the small amount of data (e.g., objective function and constraints) required by the optimizer. This limits the amount of data transmitted between the two computers.

The optimization is always performed on the minicomputer, in this implementation a PRIME 750, because the optimizer requires a significant memory allocation and the minicomputer has virtual memory. The front processor is also executed on the minicomputer and, as in the case of the end processor, the amount of data transmitted between the two computers is reduced to the minimum. The data transmitted from the front processor to the analysis program consist of updated design variable information.

Five options exist for organization of optimization procedures comprising nonlinear and piecewise linear programing with analytical and finite-difference gradients. These options are shown in Table I. One of the primary differences between the distributed implementation of PROSSS and the first implementation described in reference 5 is the method of controlling the flow of the options. In the first implementation, each option is controlled by a complex sequence of Control Data Corporation-Cyber Control Language (CDC-CCL (ref. 10)), while the flow of the options in the distributed implementation of PROSSS is controlled by FORTRAN programs using FORTRAN-callable system subroutines (ref. 11). Although presented in the context of structural analysis,

these organizations are independent of the type of analysis. The same system concept could be used for aerodynamic optimization of, for example, a wing, if the analyzer had a capability for computational aerodynamics.

The system is intended to be used in the following three basic ways:

1. as a research tool for development of optimization techniques where it offers an interface to efficient analysis and flexibility of execution and sequencing,
2. as a test bed for trying new structural analysis techniques tailored toward efficient use in optimization loop, and
3. as an application tool that can be adopted to a very wide scope of different types of problems.

Before trying to implement this system, the user should be thoroughly familiar with references 4, 5, 6, 8 and 9.

COMPONENTS OF THE SYSTEM

There are seven primary components for the distributed implementation of PROSSS including (1) the driver program, (2) the front processor, (3) CONMIN-the optimizer, (4) procedures transmitting jobs between the PRIME minicomputer and CDC mainframe computer, (5) SPAR-the analyzer, (6) the end processor, and (7) the plotting program. Each component and the files associated with them are explained in detail in the following subsections. Files generated by the system are not shown. Only those files that must be created by the user are shown.

The Driver Program

As previously mentioned, each option in PROSSS is controlled by a FORTRAN program using FORTRAN-callable procedure files. While each option uses its own main driver program, there is a group of eight subroutines common to each

option. A typical flowchart of an option is shown in figure 2. The listings of the program and subroutines are shown in Appendix A. The driver program performs the following functions: (1) initialization, (2) restarting, (3) controlling the flow of the option, and (4) termination.

The driver program first reads one line from a file named NAMEij where ij is the option number without the decimal. For example, a file named NAME11 contains the input data for option 1.1. This first line in NAMEij contains the following variables:

NRSTRT	Ø - not restarting 1 - restarting
ICR	Ø - data not created by a data generator 1 - data created by a data generator
NR	Ø - no nonrepeatable analysis is to be done 1 - perform nonrepeatable analysis
NOPT	the option number (e.g. 11)
NAMES	the name of the file to be sent to the CDC 6600 (four characters in length)
COUNT	the number of passes made through the analysis program (initially one)

The format is (3(I1,1X),I2,1x,A4,1X,I3).

An example of a NAMEij file is as follows:

```
Ø Ø Ø 11 ANOS     1
STRP3             file containing starting design variables
CONP1             file containing CONMIN parameters
CONS              file containing constants for front processors
RRSNG             file containing  repeatable runstream with no analytical
                  gradients
JIM               UFD name
DPROSS            SUBUFD name
SSPROUT           file containing SPAR output
SCNMNIO           file containing end processor output
NRLANG            file containing non-repeatable SPAR library
123456            user number
XYZ               password
123456            charge number
EPFUBIP           file containing end processor
EPFUIN            file containing input to end processor
FPFUS       Ø     file containing front processor (Ø mean compile)
CONMS1      1     file containing CONMIN main program (1 means do not
                  compile)
```

If NRSTRT is equal to zero, subroutine INIT is called to create a temporary procedure file called INITPRC. INIT reads the remainder of file NAMEij and creates edit procedures on file INITPRC to replace general file names in procedure files executing other components of the system with specific file names depending upon the problem being solved. Files for other components of the system are created depending on the option number. (This is transparent to the user.)

If ICR is not equal to zero, then INIT calls subroutine CRSNPT. This subroutine reads data from a file called NAMECRRS, edits a file created by the data generator, and inserts a file containing problem-dependent data. An example of a NAMECRRS file is as follows:

```
  1              number of design variables
SPARIN           file created by a data generator for input to SPAR
TOPSPARIN        file containing problem dependent data for input to SPAR
```

If NR is not equal to zero, then INIT calls subroutine NONRPT. This subroutine reads data from a file called NAMENR and creates a procedure file (TONOSNR, described below) to be sent to the CYBER to perform the nonrepeatable part of the analysis. An example of a NAMENR file is as follows:

```
TSPARIN          non-repeatable SPAR runstream
7ØØØØ            field length required on CDC 6600
NON1             non-repeatable job name
```

If NRSTRT equals 1, then subroutine RSTRT is called instead of subroutine INIT. This subroutine sets up a procedure file named RSTRTPRC to reinitialize the files saved for restarting the program.

If the nonrepeatable part of the analysis is required (NR = 0), a job has been sent to the CDC 6600 to perform this analysis. At this point, subroutine STCHEK is called. This subroutine calls subroutine COMSTA which returns the status of a file that has been transmitted to the CDC 6600 computer. If COMSTA determines that the job has not yet completed executing on the CDC 6600 (because no file has been returned to the PRIME), then it returns a status of zero. If the job has completed executing on the CDC 6600 and a file has been returned to the PRIME, the status is returned as one. When the status cannot be determined, an error has occurred and the status is returned as -1.

Subroutine STCHEK checks the status returned from COMSTA. If the status is one, then control is passed back to the driver program. If the status is -1, subroutine ERRMSG is called to print an error message, and the job is terminated. If the status is zero, a system routine is called to put the job on the PRIME to "sleep" for 2 minutes while waiting for the CDC 6600 to finish processing. After 2 minutes, the job "awakens" and again calls subroutine COMSTA to determine if the file has been returned from the CDC 6600. The "sleep" and "awake" process continues until the CDC 6600 has finished processing the analysis and returned the output file to the PRIME. This concept makes efficient use of the PRIME's resources.

Once the nonrepeatable part of the analysis program has been completed, the driver enters the optimization loop shown in figure 1. It will remain in this loop until an error is encountered, a loop limit (default is 999) is met, or the termination criteria are satisfied. In the optimization loop, procedure files executing the optimization program and front processor program (described below) are called. The front processor creates a file containing updated design variable information to be transmitted to the CDC 6600. The file sent to the CDC 6600 to perform the repeatable part of the analysis must have a unique job name to facilitate the return checking done by subroutine COMSTA. Subroutine UPDSF is called to create a temporary procedure file (UPDTE) for assigning this unique job name. The job name is created from the NAMES variable from the first line of the NAMEij file coupled with the loop counter.

Two temporary files containing the names of files pertaining to the specific problem at hand are created during initialization from two permanent files containing general file names. One temporary file, the names coming from the NAMES variable on the NAMEij file, is created from a permanent file

called TONOS. TONOS is a file containing CDC-CCL statements to execute the repeatable analysis (SPAR) and end processor on the CDC 6600. The second temporary file, called TSENDPRC is created from the permanent file SENDPRC. SENDPRC is a PRIME procedure file for appending the data file created by the front processor to the temporary file created from TONOS and then transmitting that file to the CDC 6600. Once the TSENDPRC file has been executed and processing is transferred to the CDC 6600 for the repeatable part of the analysis, then subroutine STCHEK is again called to test for the status of the file to be returned from the CDC 6600. The same checking, "sleeping," and "awakening" pattern described above for the nonrepeatable analysis is also followed here.

A sample listing of a TONOS file is shown below. This listing contains the general names that are replaced by specific, problem dependent names during initialization.

```
JOB,T1ØØ,CM7ØØØØ.
USER,USERNO,PASSWRD.
CHARGE,CHARGENO,LRC.
DELIVER.$UFD>SUBUFD>TCMNIO
RFL,7ØØØØ.
REDUCE(-)
MAP,OFF.
COPYCR,INPUT,SPFPOUT.
REWIND,SPFPOUT.
GET,RUNSTREAM,MERGFP.
EDIT,RUNSTREAM,,MERGFP.
EDIT,RUNSTREAM,,MERGFP,EDOUT.
REWIND,RUNSTREAM.
GET,SPAR=SPAR14I,DCU=DCU14I.
GET,SPARLA=SPAROUT.
SPAR,RUNSTREAM,SPAROUT.
REWIND,SPAROUT.
REWIND,SPARLD.
GET,EPXXXX,ENDINXX.
GET,SPARLIB.
LDSET(LIB=SPARLIB,PRESET=ZERO)
EPXXXX,ENDINXX,CNMNIO.
REPLACE,SPAROUT=DUMOUT,CNMNIO=DUMIO.
REWIND,CNMNIO.
COPYSBF,CNMNIO,OUTPUT.
DAYFILE(L=DISOUT)
REPLACE,DISOUT.
EXIT.
DAYFILE(L=DISOUT)
REPLACE,DISOUT.
```

A sample SENDPRC file is:

```
SAVE TEMP
DELETE UPDTE
APPEND SENDFILE SPFPOUT
DELETE CNMNIO
RJSEND SENDFILE CDC
DELETE SPFPOUT
R TEMP
```

After the analysis output file, TCMNIO, is returned from the CDC 6600, the driver program checks for the existence of a file called GONOGO. GONOGO is created by the CONMIN main program when the termination criteria have been met. If GONOGO exists, then the optimization process stops. If GONOGO does not exist, then the optimization loop continues by calling the EDPRC procedure file. This procedure file edits TCMNIO to remove any extraneous information, such as the dayfile, returned from the CDC 6600. This leaves only the objective function and constraint data required for input into CONMIN. At this point, the optimization loop begins a new iteration. A sample EDPRC file is:

```
SAVE TEMP
ED TCMNIO
L END CNMNIO
D1ØØØ
T
L BEGIN CNMNIO
D
UNLOAD CNMNIO 1ØØØ
FIL
DELETE TCMNIO
R TEMP
```

The Optimizer

The procedure file called from the driver program to execute CONMIN, the optimizer, is named CNMNPRC1. Whenever any procedure file is called from the driver program, the first instruction in the procedure file is to save the current address. The final instruction in the procedure file is to return control to the saved address. The CNMNPRC1 procedure file performs four functions: (1) saves restart files, (2) opens and closes files for CONMIN input and output, (3) executes CONMIN, and (4) save cumulative plotting and optimization data. A listing of CNMNPRC1 is below:

```
SAVE TEMP
ED PASS
FIL SPASS
ED CNMNIO
FIL SCNMNIO
ED CONREST
FIL SCONREST
OPEN TPLOT 12 2
OPEN PLTDATA 11 3
OPEN PASS 7 3
OPEN CNMNIO 5 3
OPEN CONREST 3 3
OPEN CONOUT 2 2
OPEN CONPAR 1 1
SEG CONMIN
C 1
C 2
C 3
C 5
C 11
C 12
APPEND PLTDATA TPLOT
DELETE TPLOT
APPEND CONOUTHOLD CONOUT
DELETE CONOUT
R TEMP
```

Three files are required to restart CONMIN. File PASS contains the number of passes made through CONMIN--one initially and two on each subsequent pass. File CNMNIO contains the objective function and constraints found in the repeatable analysis. File CONREST contains all of the remaining data saved from the preceding pass through CONMIN. These files are saved as temporary files named SPASS, SCNMNIO, and SCONREST, respectively.

Other files open for CONMIN input and output include TPLOT, PLTDATA, CONOUT, and CONPAR. TPLOT contains the plotting data created in this (and only this) pass through CONMIN. PLTDATA contains a cumulative history for the objective function, design variables, and constraints for the entire run. CONOUT contains the output listing for only this pass through CONMIN. CONPAR contains the output listing for this pass through CONMIN. CONPAR contains the initialization values for the parameters input to CONMIN. This file must be created by the user and is input via a free-field format. A description of these variables, except for JSC, can be found in reference 9. The variable JSC is used to determine the linear constraint identification vector, JSC. If JSC is -999, then all of the constraints are known to be a linear function of the design variables. If JSC is zero, then all of the constaints are nonlinear. If JSC is other than these two numbers, then a file, ISCFILE, must be created by the user containing the constraint numbers (in I4 format) of the linear constraints if JSC is positive or the constraint numbers of the nonlinear constraints if JSC is negative. JSC in this last case is the number of constraints in ISCFILE. ISCFILE is opened and closed within CONMIN. Another file, STARTX, is also opened and closed within CONMIN and must be created by the user. This file contains the number of design variables in I5

format followed on succeeding lines by the starting values of the design variables in 6E13.5 format. A sample CONPAR file is:

```
5 3 2Ø 19Ø Ø 1 Ø
4 Ø Ø Ø.Ø Ø.Ø Ø.Ø Ø.Ø25 Ø.Ø Ø.Ø
Ø.125 5. .Ø5 Ø.ØØØØØØØØØ1 Ø 2Ø 2ØØ 1ØØ 1ØØ 2ØØ
Ø.Ø Ø.Ø Ø Ø.ØØ5 Ø.1 1.Ø 1.Ø 1Ø. 1Ø.
```

A sample STARTX file is:

```
  3
Ø.5ØØØØE=Ø.1    Ø.1ØØØØE Ø1    Ø.4ØØØØE Ø1
```

Once the restart files have been saved and the necessary files have been opened, the CNMNPRC1 executes CONMIN. CONMIN consists of a group of nine subroutines as defined in reference 9. These routines are never changed by the user. The routines are called by a CONMIN main program written by the user dependent upon the problem to be solved. An example of a main program is listed in Appendix B. In addition to calling the CONMIN subroutines, the CONMIN main program is responsible for reading and writing data from and to the above files. After executing CONMIN, CNMNPRC1 closes all files that remain open.

The final task performed by CNMNPRC1 is to append the temporary files containing the plotting data (TPLOT) and CONMIN output listing (CONOUT) to the permanent cumulative files PLTDATA and CONOUTHOLD, respectively. TPLOT and CONOUT are deleted and control is returned to the driver program.

The Front Processor

The procedure file called from the driver program to execute the front processor, FPFUS, is named FRONTPRC. The front processor is supplied by the user and converts the design variables output from CONMIN on file CNMNIO to

a file SPFPOUT formatted suitable for input into the SPAR analysis program. A listing of FRONTPRC follows:

```
SAVE TEMP
OPEN CNMNIO 1 1
OPEN SPFPOUT 2 2
OPEN CONSXX 3 1
R *FPROCESSOR
C 1
C 2
C 3
R TEMP
```

The FRONTPRC procedure file opens the CNMNIO and SPFPOUT files. In addition, a file called CONSXX, supplied by the user if necessary and containing certain constants such as the cross section dimensions of a beam, is also opened. The format of the CONSXX file is determined by the user because the user writes the front processor program dependent upon what elements are used in the finite-element model. An example of a front processor is then executed, all open files are closed, and control is returned to the driver program. A sample CONSXX file for the beam parameters B1Ø, B2Ø and TØ follows:

```
12.        30.        2. (beam parameters)
```

The Analyzer--Nonrepeatable Part

The INITPRC procedure file (described above in the section on the driver program) transmits the TONOSNR procedure file to the CDC 6600 to execute the nonrepeatable part of the analysis. The nonrepeatable SPAR runstream (NRRS) is appended to the TONOSNR file for transmittal to the CDC 6600. NRRS is

copied to a local file and SPAR is then executed using NRRS as input. The results are saved on a SPAR library file (NRLA) for later use in the repeatable part of the analysis. A file named TNREPT is returned to the PRIME indicating that the nonrepeatable analysis has completed execution. A listing of TONOSNR follows.

```
JOB,T100,CMFLX.
USER,USERNO,PASSWRD.
CHARGE,CHARGENO,LRC.
DELIVER.$UFD>SUBUFD>TNREPT
MAP,OFF.
COPYCR,INPUT,NRRS.
REPLACE,NRRS.
.*
.*  THE PROCEDURE CREATES A SPAR LIBRARY
.*    FROM THE NON-REPEATABLE   PART
.*
GET,SPAR=SPAR14I,DCU=DCU14I.
GET,NRRS.
RFL,FLX.
REDUCE,-.
SPAR,NRRS,NSPROUT.
REPLACE,SPARLA=NRLA.
.*
.*  TEST TO SEE IF GRADIENTS ARE REQUIRED
.*
IFE,(NROPT.EQ.13.OR.NROPT.EQ.23),GRADIENTS.
GET,INPT=INPTXX.
GET,EDIT1,EDIT2,BLDELDB.
REWIND,NRRS.
.*
.*  EDIT OUT ALL BUT ELD INPUT IN RUNSTREAM
.*
EDIT,NRRS,EDIT1,EDOUT.
REWIND,NRRS.
.*
.*  CREATE SPAR RUNSREAM TO FIND DERIVATIVES
.*
BLDELDB,INPT,NRRS,RSOUT.
REWIND,RSOUT.
EDIT,RSOUT,,EDIT2,EDOUT.
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,BLDELDB,EDOUT.
RETURN,RSOUT,TAPE21,TAPE22,NSPROUT.
REPLACE,SPARLB-NRLA.
ENDIF,GRADIENTS.
RETURN,SPARLB=NRLA.
ENDIF,GRADIENTS.
RETURN,SPAR,DCU,SPARLA,NRRS.
DAYFILE(L=DISOUT)
REPLACE,DISOUT.
EXIT.
DAYFILE(L=DISOUT)
REPLACE,DISOUT.
```

An example of a non-repeatable SPAR runstream (NRRS) is as follows:

```
[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$CYLINDRICAL COORDINATES
1 400. 0. 0. 400. 337.5 0. 16 1 5
16 400. 0. 800. 400. 337.5 800.
MREF
FORMAT=2
1 -2 0. 0. 10000000.
2 1 0. 0. 10000000.
MATC
1 .72+6 0.3 .0028 22.-6$ 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
[XQT ELD
$START
E23$ROD ELEMENTS
NSECT=1$
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$
$END
$START
E21
NSECT=1$
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$
$END
$START
E41$ MEMBRANE PANELS
NSECT=1$
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 16$
32 48 33 17$
48 64 49 33$
$END
[XQT DCU
 TOC 1
[XQT EXIT
```

The Analyzer--Repeatable Part

The SENDPRC procedure file transmits the TONOS procedure file (both files discussed in the above section on the driver program) to the CDC 6600 to execute the analysis (SPAR) and end processor (EPXXXX) programs. The SPFPOUT file created by the front processor is appended to the TONOS file for transmittal to the CDC 6600. The SPAR library file (SPARNAME) containing data created during the nonrepeatable analysis and the repeatable SPAR runstream file (RUNSTREAM) both reside as permanent files in the CDC 6600 disk storage. SPFPOUT is copied onto a temporary file on the CDC 6600 and merged into RUNSTREAM file using a file named MERGFP consisting of CDC TEXT EDITOR (ref. 12) commands. SPAR is then executed using the RUNSTREAM and SPARNAME files as input. The objective function and stresses are computed in SPAR and saved on a SPAR library file (SPARLD). This file is then used as input to the end processor. A listing of the MERGFP file is as follows:

```
MERGE:/SPFPOUT/,/$ MERGE NEW/;1
R
RS:/E+/,/+/;*
RS:/E-/,/-/;*
END
```

An example of a repeatable SPAR runstream without analytical gradients (RRSNG) is as follows:

```
[XQT TAB
 TITLE"FUSELAGE MODEL
TEXT
"MEMBRANE ROD BEAM FUSELAGE MODEL
"REPEATABLE PART WITHOUT GRADIENTS
 UPDATE=1
$ MERGE NEW PROPERTIES HERE.
UPDATE=0
[XQT TOPO
[XQT E
[XQT EKS
[XQT K
[XQT INV
[XQT M
 RESET G=981.
[XQT AUS
 SYSVEC;APPLIED FORCES 1
  I=1;J=65;69;77; 10000. -20000. 20000.
 SYSVEC;UNIT VEC
  I=1; J=1,80; 1.0
 DEFINE WT=DEM DIAG 0 0
 DEFINE UN=UNIT VEC
 OBJFUN=XTY(UN,WT)
[XQT DCU
 PRINT 1 OBJFUN
[XQT SSOL
[XQT GSF
[XQT PSF
[XQT VPRT
 PRINT APPL FORC 1 1
 PRINT STAT DISP 1 1
[XQT 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
[XQT EXIT
```

The End Processor

The end processor, EPFUS, is also executed on the CDC 6600. The objective function and stress data saved on the SPAR library file, SPARLD, are input to the end processor using FORTRAN-callable subroutines from SPAR (ref. 13). Another file, ENDINXX, is stored on the CDC 6600 in NAMELIST form and input to the end processor. This file contains the number of each different type of element used in the model and the limits imposed on the behavior variables in constraint function evaluation. The format of the ENDINXX file is determined by the user because the user writes the end processor program dependent upon what elements are used in the finite-element model. The end processor then computes the constraints and outputs the objective function and constraints in a suitable format for input to the optimizer. The output is placed on file CNMNIO and transmitted to the PRIME on file TCMNIO. An example of an end processor program is listed in Appendix D. A sample of an ENDINXX file is as follows:

```
$EPIN
E23AL=2ØØØ.,
E21AL=2ØØØ.,
E41AL=2ØØØ.,
NSE23=58,
NSE21=76,
NSE41=56,
$END
```

Gradients

Options 1.3 and 2.3 have the capability for calculating gradients analytically. There are three problem independent programs used

to compute the 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. A listing of BLDELDS is in Appendix E. The program is stored on the CDC 6600 and is called by the TONOSNR (described in the section on the nonrepeatable analysis) sent to the CDC 6600 from the PRIME. The file containing the nonrepeatable SPAR runstream NRRS, is edited using the CDC Text Editor (ref. 12) commands from file EDIT1 to remove all but the element connectivity data. A listing of EDIT1 follows:

```
F:/[XQT ELD/
F:/[XQT/;2
D;*
R
F:/[XQT ELD/
E;*
R
D;*
R
ADD
$
END
```

Another file, INPT, stored on the CDC 6600 and created by the user is input to BLDELDS along with the edited NRRS file. The INPT file contains two card types. The first type (only one card) has the format ((6(1X,I4),1X,A4) and contains the following variables:

NOLC - number of load cases
NODV - number of design variables
ISNOLC - new starting number for load cases with derivatives
JOINTS - number of joints in the model
NDF - number of degrees of freedom
NOEL - number of different elements
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 - element type (e.g., E21, E43)
NSECT - largest section property number used for that element type
NODVPE - number of design variables for that element type

A sample listing of INPT follows.

```
   1     3  1ØØ     8Ø     5    3 STAT
E23    1   1  E21    1    1 E41   1   1
```

BLDELDS creates a SPAR runstream and outputs it on file RSOUT. File RSOUT is then edited using the CDC Text Editor and commands from file EDIT2 to remove any extraneous blanks or characters before being input to SPAR. SPAR is then executed using RSOUT as input. The results are saved on a SPAR library file (NRLA) for later use in the repeatable part of the analysis.

A listing of the EDIT2 file follows:

```
RS:/   /,/ /;*
RS:  /,/ /;*
A:/TOPO/;*
>RESET MAXSUB=25ØØ>
END
```

The other two gradient programs, GRDXXXX and DRVXXXX, are used in the repeatable part of the analysis. Listings of GRDXXXX and DRVXXXX are in Appendices F and G, respectively. Two procedure files SENDPRCG and TONOSG (similar to the SENDPRC and TONOS files previously described) are used to execute these programs in conjunction with the SPAR analysis program on the CDC 6600. The first part of the repeatable analysis with analytical gradients is identical to repeatable analysis described above. The CNMNIO file created by the optimizer is appended to the TONOSG file by SENDPRCG before it is transmitted to the CDC 6600. The GRDXXXX, DRVXXXX, INPTXX, CONSXX, and EDGRDS files are all stored on the CDC 6600. The contents of the INPTXX and CONSXX have already been described. Each of the GRDXXXX, DRVXXXX, INPTXX, and CONSXX general file names are replaced by specific names from the NAMEij file before execution. GRDXXXX uses the repeatable SPAR runstream (RRSG) as with analytical gradients to create a 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 (ex. E21 or E43). The name(s) used depend

upon the elements used in the finite-element model. If a subroutine is not used, it remains as an unsatisfied external. 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 H. The subroutine card for DKDVE21 is as follows:

```
SUBROUTINE DKDVE21(NDVJIM,NDF)
```

where

```
NDVJIM - the number of the design variables (first, second, third, etc.)
NDF    - number of degrees of freedom per joint squared
```

A listing of the SENDPRCG file follows.

```
SAVE TEMP
DELETE UPDTE
APPEND SENDFILE SPFPOUT
APPEND SENDFILE CNMNIO
DELETE CNMNIO
RJSEND SENDFILE CDC
DELETE SPFPOUT
R TEMP
```

A listing of the TONOSG file follows.

```
JOB,T100,CM100000.
USER,USERNO,PASSWRD.
CHARGE,CHARGENO,LRC.
DELIVER.$UFD>SUBUFD>TCMNIO
RFL,100000.
REDUCE(-)
MAP,OFF.
COPYCR,INPUT,SPFPOUT.
REWIND,SPFPOUT.
GET,MERGFP/UN=753437N.
GET,RUNSTREAM.
EDIT,RUNSTREAM,,MERGFP,EDOUT.
REWIND,RUNSTREAM.
GET,SPAR=SPAR14I,DCU=DCU14I/UN=750756N.
GET,SPARLA=SPARNAME.
SPAR,RUNSTREAM,SPAROUT.
RETURN,RUNSTREAM,SPFPOUT,MERGFP,EDOUT.
REWIND,SPARLA,SPARLD.
GET,SPARLIB/UN=319925N.
GET,GRDXXXX,DRVXXXX,INPTXX,CONSXX.
GET,EDGRDS/UN=753437N.
COPYCR,INPUT,CNMNIO.
REWIND,CNMNIO.
GRDXXXX,INPTXX,CONSXX,CNMNIO,RSOUT.
REWIND,RSOUT.
EDIT,RSOUT,,EDGRDS,EDOUT.
REWIND,RSOUT.
SPAR,RSOUT,SPAROUT.
RETURN,EDGRDS,EDOUT,GRDXXXX.
IFE,(NSUBS.NE.0),DERIV.
REWIND,SPARLA,SPARLD,SPARLC,CNMNIO,CONSXX,INPTXX.
LDSET(LIB=SPARLIB,PRESET=ZERO)
DRVXXXX,INPTXX,CONSXX,CNMNIO.
RETURN,DRVXXXX,INPTXX,CONSXX,SPARLA,SPARLC.
ENDIF,DERIV.
REWIND,SPARLD,SPARLIB,CNMNIO.
GET,EPXXXX,ENDINXX.
LDSET(LIB=SPARLIB,PRESET=ZERO)
EPXXXX,ENDINXX,CNMNIO,BLK.
RETURN,SPARLD,EPXXXX,ENDINXX,SPARLIB.
REPLACE,SPAROUT=DUMOUT,CNMNIO=DUMIO.
REWIND,BLK,CNMNIO.
COPYSBF,BLK,OUTPUT.
COPYSBF,CNMNIO,OUTPUT.
DAYFILE(L=DISOUT)
REPLACE,DISOUT.
EXIT.
DAYFILE(L=DISOUT)
REPLACE,DISOUT.
```

An example of a repeatable SPAR runstream with analytical gradients (RRSG) follows.

```
[XQT TAB
 TITLE"FUSELAGE MODEL
TEXT
"MEMBRANE ROD BEAM FUSELAGE MODEL
"REPEATABLE PART WITH GRADIENTS
 UPDATE=1
$ MERGE NEW PROPERTIES HERE.
UPDATE=0
[XQT E
[XQT EKS
[XQT TOPO
[XQT K
[XQT INV
[XQT M
 RESET G=981.
[XQT AUS
 SYSVEC;APPLIED FORCES 1
  I=1;J=65;69;77; 10000. -20000. 20000.
 SYSVEC;UNIT VEC
  I=1; J=1,80; 1.0
 DEFINE WT=DEM DIAG 0 0
 DEFINE UN=UNIT VEC
 OBJF AUS 1 1=XTY(UN,WT)
[XQT DCU
 PRINT 1 OBJF AUS 1 1
[XQT SSOL
[XQT GSF
[XQT PSF
[XQT VPRT
 PRINT APPL FORC 1 1
 PRINT STAT DISP 1 1
[XQT DCU
 CHANGE 1,STRS E21 1 1,FAMS E21 1 1
 TOC 1
COPY 1,4 OBJF AUS 1 1
COPY 1,4 FAMS E21 1 1
COPY 1,4 STRS E23 1 1
COPY 1,4 STRS E41 1 1
TOC 4
[XQT EXIT
```

The SPAR runstream is output on file RSOUT. A file, EDGRDS, containing CDC Text Editor commands is used to remove any extraneous blanks or characters from RSOUT. SPAR is executed to compute the derivatives using RSOUT as input. The output data from SPAR, including stresses, stress derivatives, forces and moments, and derivatives of forces and moments are output on a SPAR library file, SPARLD. A listing of EDGRDS follows.

```
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
```

Program DRVXXXX is called if there is a need to convert forces and moments, and derivatives of forces and moments to stresses and stress derivatives. The program 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 an unsatified external. Four integer parameters are passed to BMSTRS. They are (1) a switch, (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 I. The subroutine 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 (ex. moments of inertia) and make certain computations (ex. derivatives of Y with respect to the design variables) only on the first time BMSTRS is called from DRVXXXX

JCNT - a counter to find the location in blank common for various data times, depends on the number of beam elements

IBEAM - 1 if beam is a contributing factor to stress derivatives
- Ø otherwise

This program also uses FORTRAN-callable SPAR subroutines (ref. 13) to input and output this data to and from the SPAR library, SPARLD. After DRVXXXX completes execution the end processor is executed before control is returned to the PRIME.

EXAMINING THE INTERMEDIATE RESULTS

One of the key features in the distributed PROSSS is the capability for the user to examine the intermediate results. Because of this feature, the user does not have to wait until the end of the complete optimization process to determine whether or not the model, variables, and/or process are behaving as expected. The user, after examining the intermediate results, can let the

process continue; stop the process; or temporarily stop the process, make some changes to the optimization parameters or design variables, and restart the proccess. This gives the user much more control over the flow of the problem and should significantly reduce the total time required to reach the optimal solution.

All examining of intermediate results is done on the PRIME because of the faster transmission rate (9600 BAUD) to the interactive terminals. Two files, CONOUTHOLD, and PLTDATA, created by the CONMIN driver program are essential for examining these results. The CONOUTHOLD contains a cumulative listing of all information output from CONMIN. This information is originally output on file CONOUT. At the end of the CNMNPRC1 procedure file, CONOUT is appended to CONOUTHOLD to create the cumulative listing. It is best to examine the CONOUTHOLD file with the PRIME editor (ref. 14). The user is usually interested in seeing te progress of the objective function and design variables. Also of interest are the active and violated constraints. It is useful to check for errors and warning messages issued by CONMIN so that the process can be stopped at this point.

If the user desires to see a graphical display of the optimization data, the PLTDATA file should be used in conjunction with program plots. The program plots either the objective function, design variables, or constraints versus the iteration number. The capability of listing these values in tabular form is also available. THe user makes a choice from a menu displayed on the screen (figure 3). The user can also decide whether or not the objective function and design variables are to be normalized. Examples of each type of plot are shown in figures 4-6.

RESTARTING PROSSS

The user may wish to restart PROSSS for a particular point in the process because the computers go down, there is an error in the system, or the examination of the intermediate results revealed a change in the system, or the examination of the intermediate results revealed a change is needed in the model or optimization parameters. To restart PROSSS is quite simple. All of the necessary files required for restarting are saved by the procedure file CNMNPRC1. The user edits the first line of the NAMEij file to restart. The NRSTRT variable is set to 1. The NAMES variable is changed to create a unique name. The COUNT variable may or may not be updated depending upon whether or not the user has need of keeping track of the number of iterations through the system. Other names may be changed in the NAMEij file as needed. After these changes have been made, the user restarts PROSSS. After initialization, processing begins at CONMIN.

SAMPLE PROBLEM

Each option was executed to determine the final objective function (minimum mass, kg) of the finite-element model of the fuselage shown in figure 7 using the NAMEij input files. 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 (1) the cross-sectional area of the transverse stringers (beams), (2) the cross-sectional area of the longitudinal stringers (rods), and (3) 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, 4.0 cm, respectively. The initial objective function is 5460.12 kg. Figure 8 shows the results for each option. All final objective functions are within

reasonable limits, even though some of the design variables differ significantly. The piecewise linear approach is 50 to 75 percent less expensive to execute than the nonlinear approach. Reference 6 describes, in detail, the results of this test compared to the results from an identical model tested on the mainframe-only version of PROSSS. The comparison is done in terms of accuracy, cost, CPU time, and total time.

REFERENCES

1. Haftka, R. T.; and Prasad, B.: Programs for Analysis and Resizing of Complex Structures. Presented at the Symposium on Future Trends in Computerized Structural Analysis and Synthesis, Washington, D.C., October 30 - November 1, 1978. Proceedings entitled "Trends in Computerized Structural Analysis and Synthesis," Pergamon Press, N.Y., 1978, pp 323-330.
2. The NASTRAN Theoretical Manual (Level 16.0). NASA SP-221(03), March 1976.
3. Schmit, L. A., Jr.; and Miura, H.: A New Structural Analysis/Synthesis Capability - ACCESS 1. AIAA J., Vol. 14, No. 5, May 1976, pp 661-671.
4. Sobieszczanski-Sobieski, J.; and Bhat, R. B.: Adaptable Structural Synthesis Using Advanced Analysis and Optimization Coupled by a Computer Operating System. AIAA Paper No. 79-0723, April 1979.
5. Rogers, J. L., Jr.; Sobieszczanski-Sobieski, J.; and Bhat, R. B.: An Implementation of the Programing Structural Synthesis System (PROSSS). NASA TM 83180, 1981.
6. Rogers, J. L., Jr.; Dovi, A. R.; and Riley, K. M.: Distributing Structural Optimization Software Between a Mainframe and a Minicomputer. Presented at the Second International Conference and Exhibition on Engineering Software, London, England, March 24-26, 1981. Proceeding entitled "Engineering Software II," Hobbs the Printers, Southampton, England, 1981, pp 400-415.
7. Manufacturing and Consulting Services, Inc.: AD2000 Reference Manual, 1980.
8. Whetstone, W. D.: SPAR Structural Analysis System Reference Manual, System Level 13A, Vol. I. NASA CR 158970-1, December 1978.

9. Vanderplaats, G. N.: CONMIN--A FORTRAN Program for Constrained Function Minimization, User's Manual. NASA TM X-62282, August 1973.

10. Control Data Corporation: NOS Version 1 Reference Manual. NOS 1.3, CDC No. 60435400, September 1979.

11. PRIME Computer, Inc.: PRIMOS Subroutines Reference Guide. PDR3621, 1980.

12. Control Data Corporation: NOS Version 1 Text Editor, Reference Manual. NOSE 1.3, CDC No. 60436100, September 1979.

13. Giles, G. L.; and Haftka, R. T.: SPAR Data Handling Utilities. NASA TM 78701, September 1978.

14. PRIME Computer, Inc.: The New User's Guide to Editor and Runoff. FDR 3104, 1980.

	GRADIENTS COMPUTED		
	IN OPTIMIZER BY	EXTERNAL TO OPTIMIZER BY	
METHOD	FINITE DIFFERENCE	FINITE DIFFERENCE	ANALYTICAL
NONLINEAR PROGRAMMING	1.1	1.2	1.3
PIECEWISE LINEAR PROGRAMMING	N/A	2.2	2.3

TABLE I.- OPTIONS FOR OPTIMIZATION.

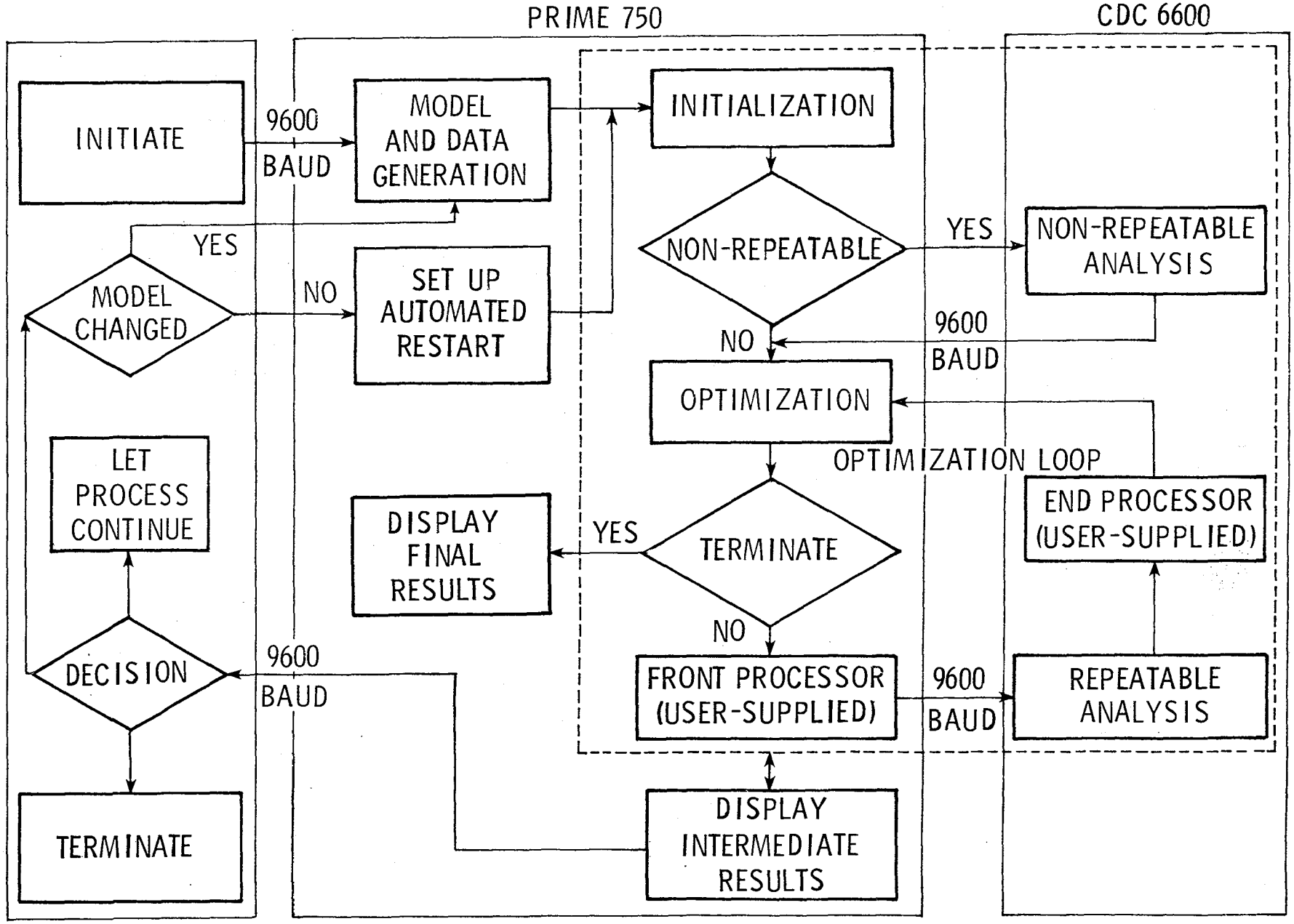

FIGURE 1.- FLOWCHART OF A DISTRIBUTED OPTIMIZATION SOFTWARE SYSTEM.

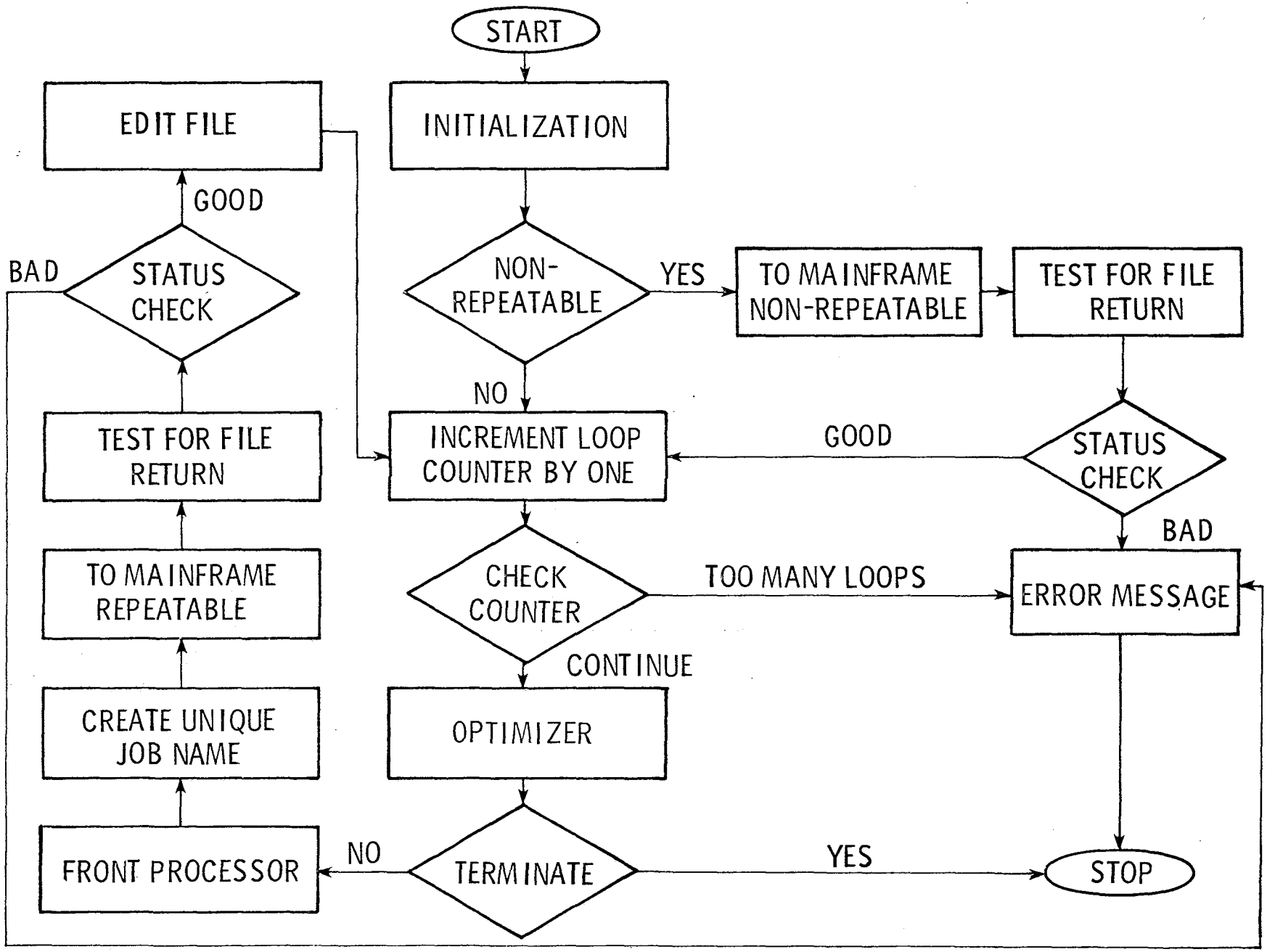

FIGURE 2.- FLOWCHART OF A TYPICAL OPTIMIZATION OPTION.

INPUT NUMBER CORRESPONDING TO TYPE OF PLOT DESIRED

1 - PLOT ITERATIONS VS OBJECTIVE FUNCTION
2 - PLOT ITERATIONS VS DESIGN VARIABLES
3 - PLOT ITERATIONS VS CONSTRANTS
4 - PRINT NUMERICAL RESULTS
5 - END

FIGURE 3.- MENU FOR PLOTTING PROGRAM.

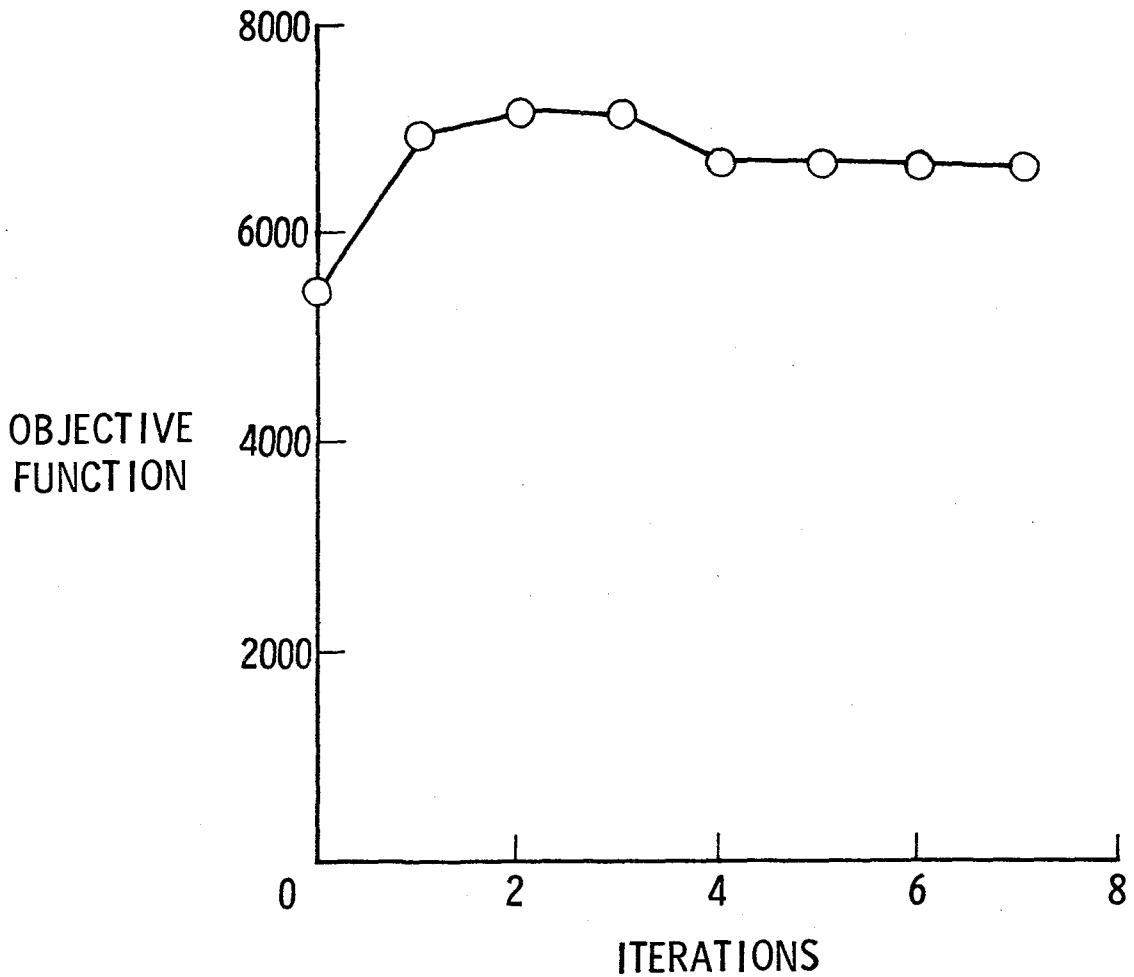

FIGURE 4.- CHANGE IN OBJECTIVE FUNCTION FOR THE FUSELAGE MODEL.

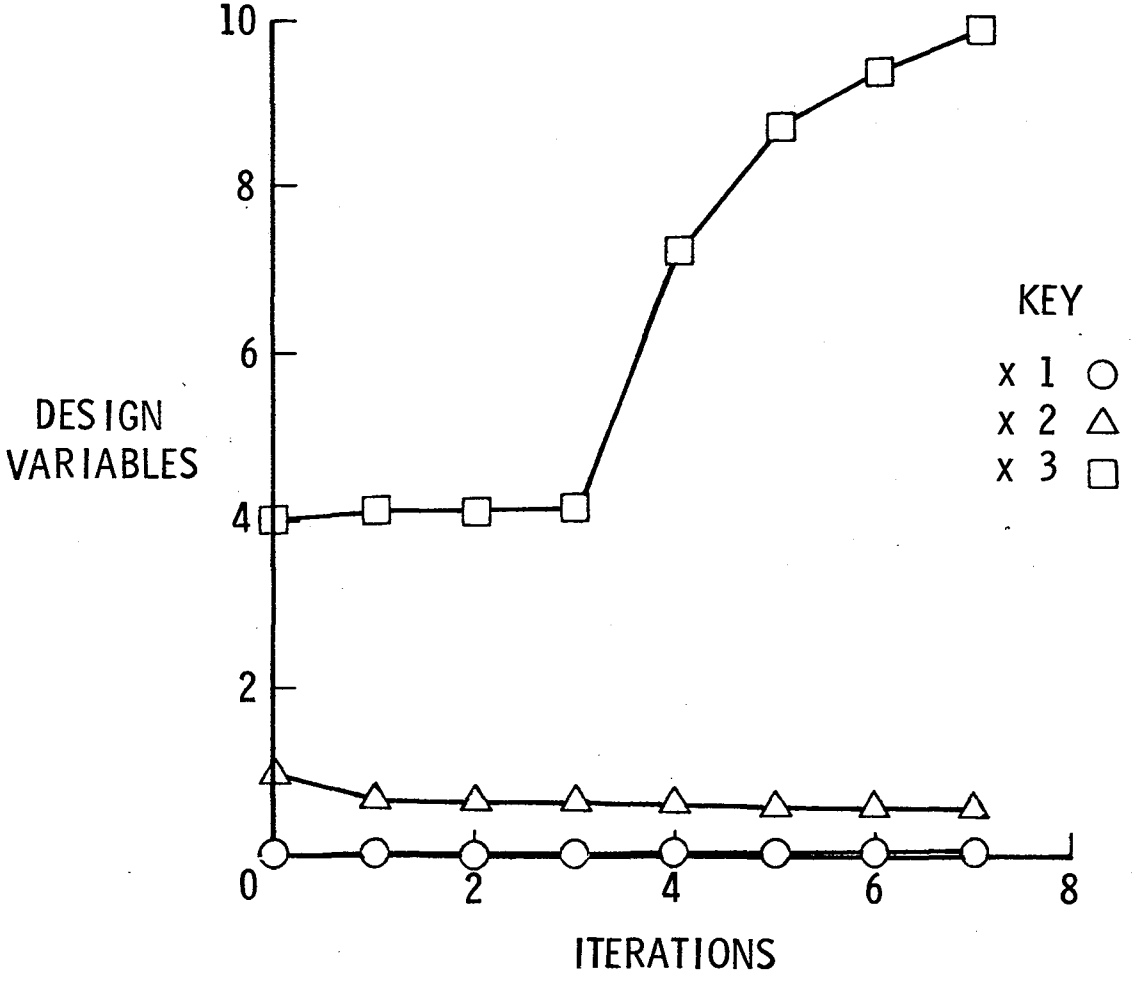

FIGURE 5.- CHANGE IN DESIGN VARIABLES (RECIPROCALS) FOR THE FUSELAGE MODEL.

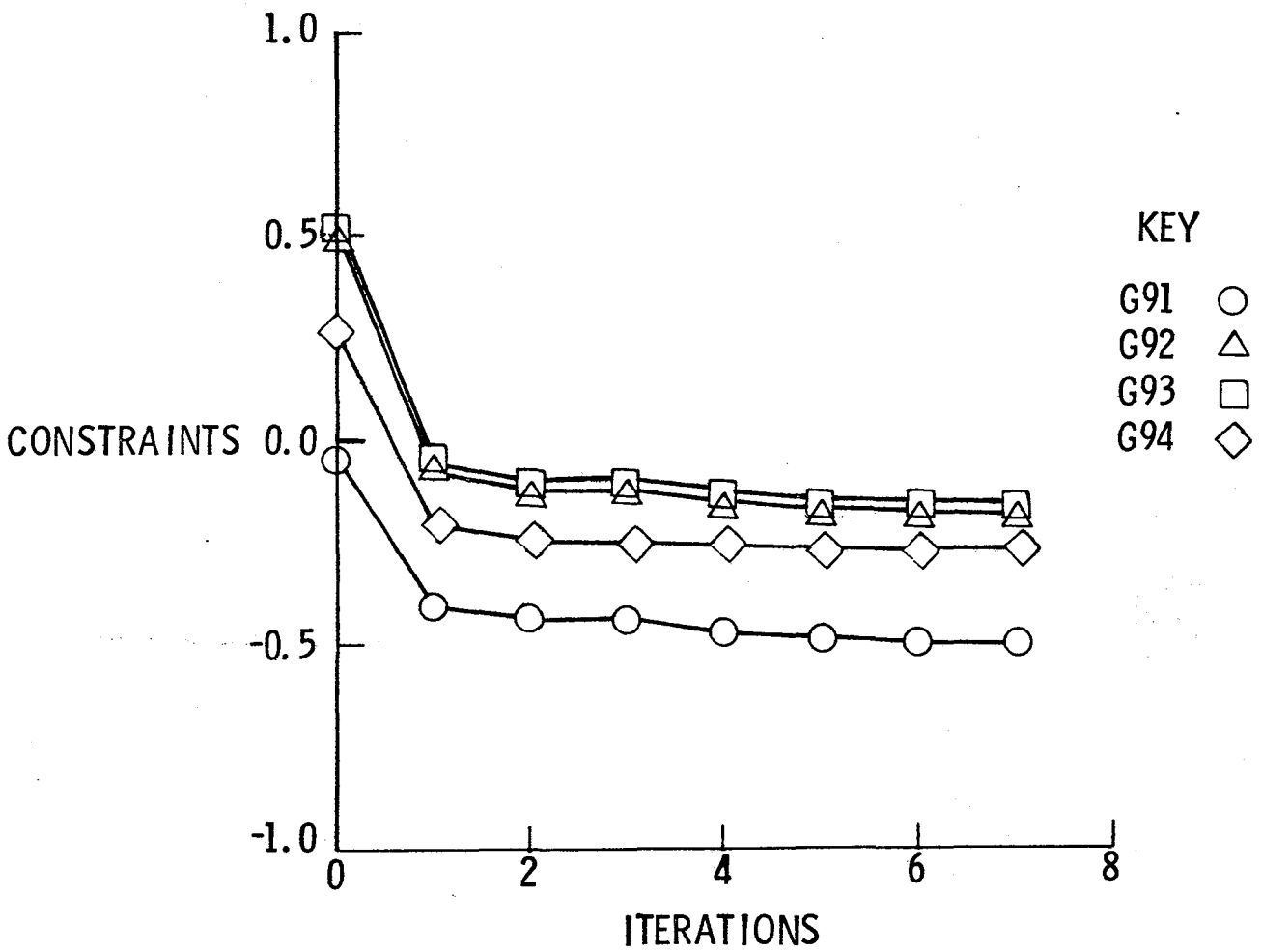

FIGURE 6.- ACTIVE AND VIOLATED CONSTRAINTS FOR THE FUSELAGE MODEL.

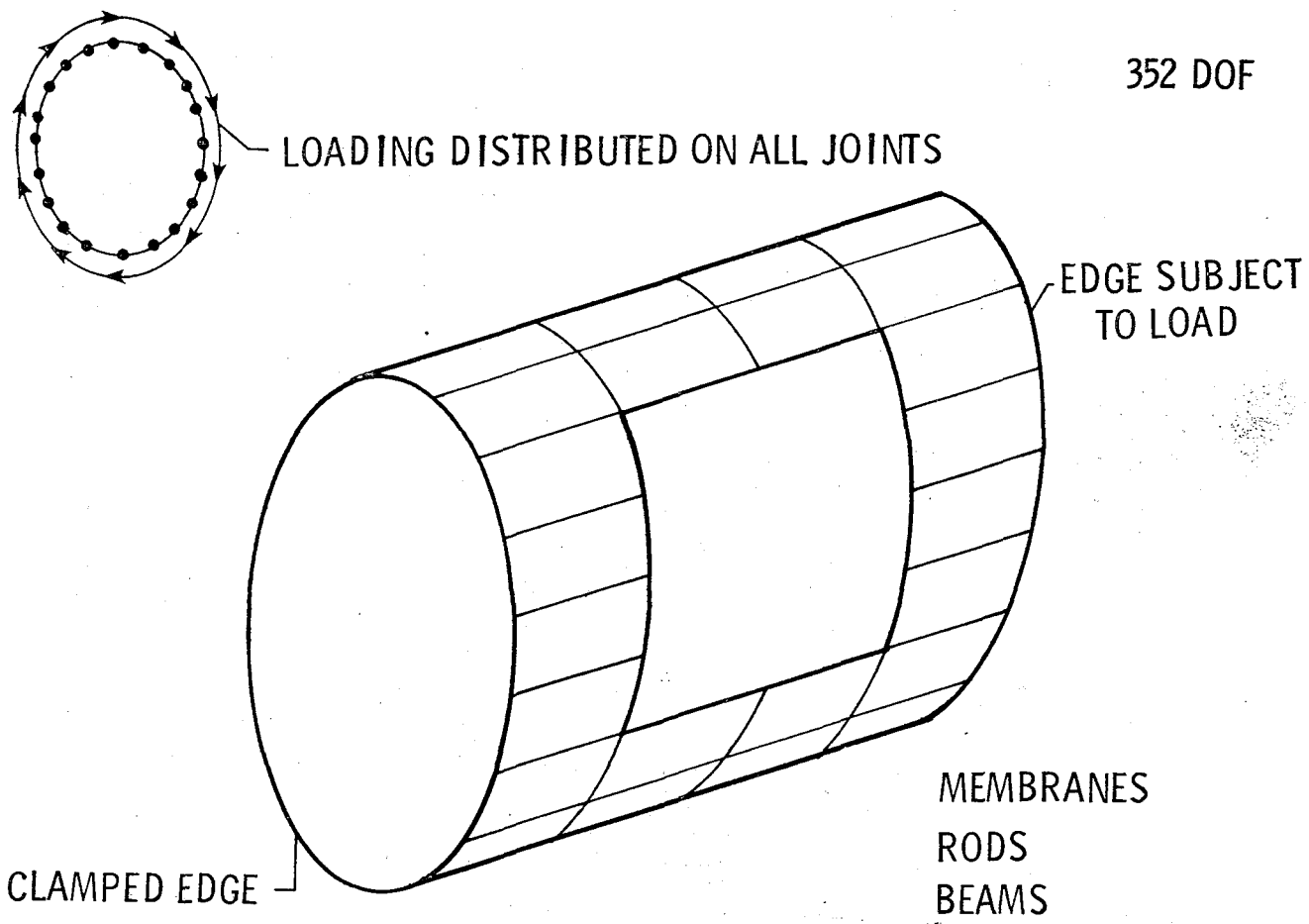

FIGURE 7.- 352 DOF FUSELAGE MODEL USED FOR TESTING.

	CROSS SECTIONAL AREA BEAM (CM^2)		CROSS SECTIONAL AREA ROD (CM^2)		THICKNESS MEMBRANE (CM)	
OPTION	M/O	D	M/O	D	M/O	D
1.1	1.6332	1.6292	4.7341	5.0582	.1000	.1000
1.2	1.5713	1.5681	8.4175	8.3682	.1000	.1000
1.3	1.5458	1.5175	10.6723	11.0132	.1042	.1083
2.2	1.5659	1.5149	1.1770	8.7796	.1746	.2158
2.3	1.6215	1.6134	1.6722	2.8777	.1000	.1000

M/O = MAINFRAME-ONLY SYSTEM D = DISTRIBUTED SYSTEM

FIGURE 8.- COMPARISON OF DESIGN VARIABLES.

APPENDIX A.- LISTING OF OPTION 1.1 DRIVER PROGRAM AND GENERAL SUBROUTINES.

```
C  DISTRIBUTED PROSSS OPT11
C
      INTEGER*4 COUNT,NAMES
      INTEGER*2 STAT,SW,ERROR,NRSTRT
      SW = 0
C
C  SET UP INITIALIZATION PROCEDURE
C
      CALL SRCH$$(1,'NAME11',6,1,0,ICODE)
      READ(5,2) NRSTRT,ICR,NR,NOPT,NAMES,COUNT
2     FORMAT(3(I1,1X),I2,1X,A4,1X,I3)
      IF(NRSTRT.EQ.0) CALL INIT(NR,ICR,NAMES,NOPT)
      COUNT = COUNT-1
C
C  SET UP FOR RESTART IF NRSTRT = 1
C
      IF(NRSTRT.EQ.0) GO TO 4
      CALL SRCH$$(4,'NAME11',6,1,0,ICODE)
      CALL RSTRT(NOPT,NAMES)
      CALL COMI$$('RSTRTPRC',8,6,ICODE)
      CALL EXIT
      CALL SRCH$$(4,'RSTRTPRC',8,1,0,ICODE)
      CALL SRCH$$(5,'RSTRTPRC',8,1,0,ICODE)
      GO TO 6
C
C  INITIALIZATION PROCEDURE FILE
C


4     CALL SRCH$$(4,'NAME11',6,1,0,ICODE)
      CALL COMI$$('INITPRC',7,5,ICODE)
      CALL EXIT
      CALL SRCH$$(4,'INITPRC',7,1,0,ICODE)
      CALL SRCH$$(5,'INITPRC',7,1,0,ICODE)
C
C  SKIP IF NONREPEATABLE PART NOT REQUIRED
C
      IF(NR.EQ.0) GO TO 5
C
C  CHECK STATUS OF RETURNED FILE
C
      CALL STCHEK(STAT,1)
      CALL SRCH$$(5,'TNREPT',6,1,0,ICODE)
      IF(STAT.EQ.0) GO TO 5
      ERROR = 2
      GO TO 40
```

```
C
C   CONMIN PROCEDURE FILE
C
    5   COUNT = COUNT+1
        IF(COUNT.LT.999) GO TO 6
        ERROR = 1
        GO TO 40
    6   CALL COMI$$('CNMNPRC1',8,6,ICODE)
        CALL EXIT
C
C   FRONT PROCESSOR PROCEDURE FILE
C
        CALL COMI$$('TFRTPRC',7,6,ICODE)
        CALL EXIT
C
C   SUBROUTINE TO UPDATE SEND FILE NAME (UNIQUELY)
C
        CALL UPDSF(COUNT,SW,INFO)
        SW = SW+1
C
C   UPDATE NAME PROCEDURE FILE
C
        CALL COMI$$('UPDTE',5,6,ICODE)
        CALL EXIT
C
C   SEND TO NOS PROCEDURE FILE
C
        CALL COMI$$('TSENDPRC',8,6,ICODE)
        CALL EXIT
C
C   CHECK STATUS OF RETURNED FILE
C


        CALL STCHEK(STAT,0)
        IF(STAT.GT.0) GO TO 35
        ERROR = 2
        GO TO 40
C
C   TEST FOR SYSTEM TERMINATION
C
   35   CALL SRCH$$(6,'GONOGO',6,1,0,ICODE)
        IF(ICODE.EQ.0) GO TO 100
C
C   EDIT CNMNIO FILE RETURNED FROM NOS PROCEDURE FILE
C
        CALL COMI$$('EDPRC',5,6,ICODE)
        CALL EXIT
        GO TO 5
C
C   PRINT ERROR MESSAGE AND STOP PROGRAM
C
   40   CALL ERRMSG(ERROR)
  100   STOP
        END
```

```
      SUBROUTINE INIT(NR,ICR,NAME1,NOPT)
C
C   CREATES A PROCEDURE FILE (INITPRC) FOR
C       INTIALIZING FILES
C
      INTEGER*4 NAME,NAME1,NAME2,STARTX,CONPAR,UFD,SUBUFD
      INTEGER*4 SPLA,USERNO,PASSW,CHARGE,INPT,CONS
      DIMENSION NAME(2),STARTX(2),CONPAR(2),UFD(2),SUBUFD(2),SPLA(2)
      DIMENSION USERNO(2),CHARGE(2),INPT(2),CONS(2)
C
C   OPEN PROCEDURE FILE
C
      CALL SRCH$$(2,'INITPRC',7,2,0,ICODE)
      WRITE(6,7)
    7 FORMAT('SAVE TEMP'/'OPEN CONOUTHOLD 1 2'//'C 1')
C
C   SET UP TEMPORARY FILES FOR CNMNIO, CONREST, AND BLK
C
      IF(NOPT.EQ.11.OR.NOPT.EQ.12.OR.NOPT.EQ.13) WRITE(6,8)
    8 FORMAT('ED'/'X'//'FIL CNMNIO'/'ED'/'X'//'FIL CONREST')
      IF(NOPT.EQ.22.OR.NOPT.EQ.23) WRITE(6,9)
    9 FORMAT('ED'/'X'//'FIL BLK')
C
C   INITIALIZE PASS, GONOGO, AND CNT
C
```

```
      WRITE(6,10)
   10 FORMAT('ED'/'1'//'FIL PASS')
      IF(NOPT.EQ.12.OR.NOPT.EQ.22.OR.NOPT.EQ.23) WRITE(6,16)
   16 FORMAT('ED'/'   1.0E 00      1.0E 00      1.0E 00      0.5E-01 '
     1  //'FIL CNT')
C
C  INITIALIZE SENDPRC
C
      IF(NOPT.EQ.11.OR.NOPT.EQ.12.OR.NOPT.EQ.22) WRITE(6,45)
   45 FORMAT('ED SENDPRC')
      IF(NOPT.EQ.13.OR.NOPT.EQ.23) WRITE(6,46)
   46 FORMAT('ED SENDPRCG')
      WRITE(6,61) NAME1
   61 FORMAT('C/SENDFILE/',A4,'/100'/'FIL TSENDPRC')
C
C  STORE NAMES FOR STARTX AND CONPAR
C
      READ(5,30) STARTX
   30 FORMAT(2A4)
      WRITE(6,40) STARTX
   40 FORMAT('ED ',2A4/,'FIL STARTX')
      READ(5,30) CONPAR
      WRITE(6,43) CONPAR
   43 FORMAT('ED ',2A4/,'FIL CONPAR')
C
C  INITIALIZE FRONT PROCESSOR
C
      READ(5,30) CONS
      WRITE(6,44) CONS
   44 FORMAT('ED FRONTPRC'/'C/CONSXX/',2A4,'/100'/'FIL TFRTPRC')
C
C  INTIALIZE TONOS
C
      READ(5,30) NAME
      IF(NOPT.EQ.11.OR.NOPT.EQ.12.OR.NOPT.EQ.22) WRITE(6,65) NAME
   65 FORMAT('ED TONOS'/'C/RUNSTREAM/',2A4,'/100'/'T')
      IF(NOPT.EQ.23) WRITE(6,64) NAME
   64 FORMAT('ED TONOSG'/'C/RUNSTREAM/',2A4,'/100'/'T')
      IF(NOPT.NE.13) GO TO 644
      WRITE(6,640) NAME
  640 FORMAT('ED TONOS13'/'C/RSGRAD/',2A4,'/100'/'T')
      READ(5,30) NAME
      WRITE(6,641) NAME
  641 FORMAT('C/RSNGRAD/',2A4,'/100'/'T')
      READ(5,30) NAME
      WRITE(6,642) NAME
  642 FORMAT('C/EPYYYY/',2A4,'/100'/'T')
  644 READ(5,30) UFD
      WRITE(6,66) UFD
```

```
   66  FORMAT('C/UFD/',2A4,'/100'/'T')
       READ(5,30) SUBUFD
       WRITE(6,67) SUBUFD
   67  FORMAT('C/SUBUFD/',2A4,'/100'/'T')
       READ(5,30) NAME
       WRITE(6,68) NAME
   68  FORMAT('C/DUMOUT/',2A4,'/100',/'T')
       READ(5,30) NAME
       WRITE(6,69) NAME
   69  FORMAT('C/DUMIO/',2A4,'/100',/'T')
       READ(5,30) SPLA
       WRITE(6,70) SPLA
   70  FORMAT('C/SPARNAME/',2A4,'/100'/'T')
       WRITE(6,71) NAME1
   71  FORMAT('C/JOB/',A4,'/100'/'T')
       READ(5,30) USERNO
       WRITE(6,72) USERNO
   72  FORMAT('C/USERNO,/',2A4,'/100'/'T')
       READ(5,30) PASSW
       WRITE(6,720) PASSW
  720  FORMAT('C/PASSWRD/',A4,'/100'/'T')
       READ(5,30) CHARGE
       WRITE(6,73) CHARGE
   73  FORMAT('C/,CHARGENO,/',2A4,'/100'/'T')
       READ(5,30) NAME
       WRITE(6,75) NAME
   75  FORMAT('C/EPXXXX/',2A4,'/100',/'T')
       READ(5,30) NAME
       WRITE(6,74) NAME
   74  FORMAT('C/ENDINXX/',2A4,'/100',/'T')
C
C   SKIP IF NO GRADIENTS ARE REQUIRED
C
       IF(NOPT.EQ.11.OR.NOPT.EQ.12.OR.NOPT.EQ.22) GO TO 755
       READ(5,30) NAME
       WRITE(6,750) NAME
  750  FORMAT('C/DRVXXXX/',2A4,'/100'/'T')
       READ(5,30) NAME
       WRITE(6,751) NAME
  751  FORMAT('C/GRDXXXX/',2A4,'/100'/'T')
       READ(5,79) NSUBS
   79  FORMAT(I1)
       WRITE(6,752) NSUBS
  752  FORMAT('C/NSUBS/',I1,'/100'/'T')
       READ(5,30) INPT
       WRITE(6,753) INPT
  753  FORMAT('C/INPTXX/',2A4,'/100'/'T')
       WRITE(6,754) CONS
  754  FORMAT('C/CONSXX/',2A4,'/100'/'T')
```

```
C
C  REMOVE EXTRANEOUS BLANKS
C
 755  WRITE(6,76)
  76  FORMAT('C/     //100'/'T'/'C/    //100'/'T'/'C/   //100')
      WRITE(6,77) NAME1
  77  FORMAT('T'/'C/  //100'/'T'/'C/ //100'/'FIL ',A4)
C
C  INITIALIZE STATIN
C
      WRITE(6,78) NAME1
  78  FORMAT('ED'/A4//'FIL STATIN')
C
C  IF NOCR EQ 0 JUST CHANGE NAMES FOR PROGRAMS
C
C  IF NOCR NE 0 EDIT FILES TO COMPILE PROGRAMS
C
C
C                 FRONT PROCESSOR
C
      READ(5,115) NAME,NOCR
 115  FORMAT(2A4,1X,I1)
      IF(NOCR.NE.0) WRITE(6,80) NAME
  80  FORMAT('ED CRFPROC'/'C/FPXXXX/',2A4,'/1234'/'FIL TCRFPROC')
      IF(NOCR.NE.0) WRITE(6,120)
 120  FORMAT('CO TCRFPROC 6'/'C 6'/'DELETE TCRFPROC')
      IF(NOCR.EQ.0) WRITE(6,800) NAME
 800  FORMAT('CN *',2A4,' *FPROCESSOR')
C
C                    EVAL
C
      IF(NOPT.EQ.11.OR.NOPT.EQ.13) GO TO 860
      READ(5,115) NAME,NOCR
      IF(NOCR.NE.0) WRITE(6,85) NAME
  85  FORMAT('ED CREVAL'/'C/EVALXX/',2A4,'/1234'/'FIL TCREVAL')
      IF((NOPT.EQ.12.OR.NOPT.EQ.22.OR.NOPT.EQ.23).AND.NOCR.NE.0)
     1 WRITE(6,140)
 140  FORMAT('C 6'/'CO TCREVAL'/'C 6'/'DELETE TCREVAL')
      IF(NOCR.EQ.0) WRITE(6,850) NAME
 850  FORMAT('CN ',2A4,' EVAL')
C
C                   CONMIN
C
860  READ(5,115) NAME,NOCR
     IF(NOCR.NE.0) WRITE(6,90) NAME
      IF(NOCR.NE.0) WRITE(6,130)
 130  FORMAT('C 6'/'CO TCRCONMIN '/'C 6'/'DELETE TCRCONMIN')
      IF(NOCR.EQ.0) WRITE(6,900) NAME
 900  FORMAT('CN ',2A4,' CONMIN')
```

```
      IF(NOPT.NE.12)  GO TO 960
C
C                  RERITES
C
      READ(5,115) NAME,NOCR
      IF(NOCR.NE.0) WRITE(6,950)
950   FORMAT('CO CRRITE 6'/'C 6')
        GO TO 145
C
C                     SELECTS
C
 970  READ(5,115) NAME,NOCR
      IF(NOCR.NE.0) WRITE(6,951)
 951  FORMAT('CO CRSEL 6'/'C 6')
C
C  TEST TO DETERMINE IF MODEL CREATED WITH DATA GENERATOR
C
 145  IF(ICR.NE.0) CALL CRSNPT
C
C  TEST TO DETERMINE IF NONREPEATABLE PART IS TO BE DONE
C
      IF(NR.NE.0) CALL NONRPT(NR,SPLA,INPT,USERNO,PASSW,CHARGE,
     1 UFD,SUBUFD,NOPT)
      WRITE(6,150)
 150  FORMAT('R TEMP')
      CALL SRCH$$(4,'INITPRC',7,2,0,ICODE)
      RETURN
      END
      SUBROUTINE RSTRT(NOPT,NAMES)
C
C  THIS SUBROUTINE SETS UP A PROCEDURE FILE FOR
C  RESTARTING THE PROGRAM
C
      INTEGER*4 NAMES,NAMERS,NAMOLD
      DIMENSION NAMERS(2)
      CALL SRCH$$(2,'RSTRTPRC',8,2,0,ICODE)
      WRITE(6,10)
  10  FORMAT('SAVE TEMP')
      IF(NOPT.EQ.11.OR.NOPT.EQ.12.OR.NOPT.EQ.13) GO TO 20
      IF(NOPT.EQ.22.OR.NOPT.EQ.23) GO TO 60
C
C  SET UP FILES FOR OPTIONS 1.1, 1.2, AND 1.3
C
  20  WRITE(6,30)
  30  FORMAT('DELETE PASS'/'CN SPASS PASS')
      WRITE(6,40)
  40  FORMAT('DELETE CNMNIO'/'CN SCNMNIO CNMNIO')
      WRITE(6,50)
  50  FORMAT('DELETE CONREST'/'CN SCONREST CONREST')
```

```
      GO TO 100
C
C  SET UP FILES FOR OPTIONS 2.2 AND 2.3
C
   60 WRITE(6,70)
   70 FORMAT('DELETE CNT'/'CN SCNT CNT')
      WRITE(6,80)
   80 FORMAT('DELETE BLK'/'CN SBLK BLK')
      WRITE(6,90)
   90 FORMAT('DELETE STARTX'/'CN SSTARTX STARTX')
C
C  REINITIALIZE SENDPRC
C
  100 WRITE(6,101)
  101 FORMAT('DELETE TSENDPRC')
      IF(NOPT.EQ.11.OR.NOPT.EQ.12.OR.NOPT.EQ.22) WRITE(6,102)
  102 FORMAT('ED SENDPRC')
      IF(NOPT.EQ.13.OR.NOPT.EQ.23) WRITE(6,103)
  103 FORMAT('ED SENDPRCG')
      WRITE(6,107) NAMES
  107 FORMAT('C/SENDFILE/',A4,'/100'/'FIL TSENDPRC')
C
C  REINITIALIZE STATIN
C
      CALL SRCH$$(1,'STATIN',6,1,0,ICODE)
      READ(5,110) NAMOLD
  110 FORMAT(A4)
      CALL SRCH$$(4,'STATIN',6,1,0,ICODE)
      WRITE(6,115) NAMOLD,NAMES
  115 FORMAT('CN ',A4,1X,A4)
      WRITE(6,120) NAMES
  120 FORMAT('DELETE STATIN'/'ED'/A4//'FIL STATIN')
      WRITE(6,130)
  130 FORMAT('R TEMP')
      CALL SRCH$$(4,'RSTRTPRC',8,2,0,ICODE)
      RETURN
      END
      SUBROUTINE NONRPT(NR,SPLA,INPT,USERNO,PASSW,CHARGE,UFD,SUBUFD,
     1 NOPT)
C
C  CREATES PROCEDURE FILE TO SEND TO CYBER FOR NONREPEATABLE PART
C  OF ANALYSIS
C
      INTEGER*4 NAME,INPT,SPLA,NROPT,NRRS,USERNO,CHARGE,UFD,SUBUFD,PASSW
     1 ,TNREPT
      DIMENSION NRRS(2),SPLA(2),INPT(2),NAME(2),USERNO(2),CHARGE(2)
     1 ,UFD(2),SUBUFD(2)
      CALL SRCH$$(1,'NAMENR',6,3,0,ICODE)
      WRITE(6,10)
```

```
   10  FORMAT('ED TONOSNR')
       READ(5,20) NRRS
   20  FORMAT(2A4)
       WRITE(6,30) NRRS
   30  FORMAT('C/NRRS/',2A4,'/100'/'T')
       WRITE(6,50) NOPT
   50  FORMAT('C/NROPT/',I2,'/100'/'T')
       WRITE(6,60) INPT
   60  FORMAT('C/INPTXX/',2A4,'/100'/'T')
       READ(5,20) NAME
       WRITE(6,70) NAME
   70  FORMAT('C/FLX/',2A4,'/100'/'T')
       WRITE(6,71) USERNO
   71  FORMAT('C/USERNO,/',2A4,'/100'/'T')
       WRITE(6,72) PASSW
   72  FORMAT('C/PASSWRD/',A4,'/100'/'T')
       WRITE(6,73) CHARGE
   73  FORMAT('C/,CHARGENO,/',2A4,'/100'/'T')
       WRITE(6,74) UFD
   74  FORMAT('C/UFD/',2A4,'/100'/'T')
       WRITE(6,75) SUBUFD
   75  FORMAT('C/SUBUFD/',2A4,'/100'/'T')
       READ(5,76) TNREPT
   76  FORMAT(A4)
       WRITE(6,77) TNREPT
   77  FORMAT('C/JOB/',A4,'/100'/'T')
       WRITE(6,80) SPLA
   80  FORMAT('C/NRLA/',2A4,'/100'/'T')
       WRITE(6,84)
   84  FORMAT('C/   //100G'/'T'/'C/  //100G'/'T'/'C/ //100G')
       WRITE(6,85) TNREPT
   85  FORMAT('FIL ',A4)
       WRITE(6,86) TNREPT
   86  FORMAT('ED'/A4//'FIL NRSTAT')
       WRITE(6,90) TNREPT
   90  FORMAT('APPEND ',A4,' TSPARIN'/'DELETE TSPARIN')
       WRITE(6,110) TNREPT
  110  FORMAT('RJSEND ',A4,' CDC')
       CALL SRCH$$(4,'NAMENR',6,3,0,ICODE)
       RETURN
       END
       SUBROUTINE CRSNPT
C
C  EDITS FILE CREATED BY DATA GENERATOR AND LOADS IN FILE CONTAINING
C  PROBLEM DEPENDENT DATA
C
       DIMENSION FNAME(2),LNAME(2)
       INTEGER*4 NDV,LNAME,FNAME
       CALL SRCH$$(1,'NAMECRRS',8,3,0,ICODE)
```

```
C
C   READ IN NUMBER OF DESIGN VARIABLES
C
      READ(5,1) NDV
    1 FORMAT(I3)
C
C   READ IN NAME OF FILE CREATED BY DATA GENERATOR
C
      READ(5,2) FNAME
    2 FORMAT(2A4)
      WRITE(6,3) FNAME
    3 FORMAT('ED ',2A4/,'L JLOC'/'N'/'D'/'L CMD')
      WRITE(6,4)
    4 FORMAT('D'/'L CMD'/'C/CMD/[XQT/'/'C>5/13>'///'$START'//)
      NDVM1 = NDV-1
      IF(NDVM1.EQ.0) GO TO 8
      DO 7 I = 1,NDVM1
      WRITE(6,6)
    6 FORMAT('L E'/'N-1'//'$END'/'$START'//)
    7 CONTINUE
    8 WRITE(6,9)
    9 FORMAT('L CMD'/'D'/'N-1'//'$END'/'[XQT EXIT'//'T')
C
C   READ IN FILE NAME CONTAINING PROBLEM DEPENDENT DATA TO BE LOADED
C   IN AT THE TOP OF THE FILE
C
      READ(5,2) LNAME
      WRITE(6,10) LNAME
   10 FORMAT('LOAD ',2A4/,'FIL TSPARIN')
      CALL SRCH$$(4,'NAMECRRS',8,3,0,ICODE)
      RETURN
      END
      SUBROUTINE UPDSF(CNT,SW,INFO)
C
C   THIS SUBROUTINE CREATES A PROCEDURE FILE (UPDTE)
C      TO CREATE UNIQUE JOB NAMES TO SEND TO NOS
C
      INTEGER*2 NME(2),SW
      INTEGER*4 CNT,CNTM1
C
C   MODIFY STATIN
C
      CALL SRCH$$(1,'STATIN',6,1,0,ICODE)
      READ(5,10) NME
   10 FORMAT(2A2)
      CALL SRCH$$(4,'STATIN',6,1,0,ICODE)
      CALL SRCH$$(2,'UPDTE',5,2,0,ICODE)
      WRITE(6,20) NME,NME,CNT
   20 FORMAT('SAVE TEMP'/'ED STATIN'/'N'/'C/',2A2,'/',2A2,I3)
```

```
      WRITE(6,30)
   30 FORMAT('C/   ///'/'C/  ///'/'C/ ///'/'FIL TSTATIN')
      CNTM1 = CNT-1
C
C  CHECK FOR FIRST TIME THROUGH
C
      IF(SW.EQ.0) GO TO 41
C
C  MODIFY SENDPRC DEPENDING UPON PASS COUNTER
C
      IF(CNT.LT.10) WRITE(6,35) NME,CNTM1,NME,CNT
   35 FORMAT('ED TSENDPRC'/'C/',2A2,I1,'/',2A2,I1,'/100'/'T')
      IF(CNT.EQ.10) WRITE(6,360) NME,CNTM1,NME,CNT
  360 FORMAT('ED TSENDPRC'/'C/',2A2,I1,'/',2A2,I2,'/100'/'T')
      IF(CNT.GT.10.AND.CNT.LT.100) WRITE(6,36) NME,CNTM1,NME,CNT
   36 FORMAT('ED TSENDPRC'/'C/',2A2,I2,'/',2A2,I2,'/100'/'T')
      IF(CNT.GE.100.AND.CNT.LT.1000) WRITE(6,37) NME,CNTM1,NME,CNT
   37 FORMAT('ED TSENDPRC'/'C/',2A2,I3,'/',2A2,I3,'/100'/'T')
      WRITE(6,38) NME,CNTM1
   38 FORMAT('L DEL'//'DELETE ',2A2,I3//)
      IF(SW.NE.1) WRITE(6,39)
   39 FORMAT('N'/'D'/'T')
      IF(SW.EQ.1) WRITE(6,40)
   40 FORMAT('T')
      GO TO 49
   41 WRITE(6,45)NME,NME,CNT
   45 FORMAT('ED TSENDPRC'/'C/',2A2,'/',2A2,I3,'/100'/'T')
   49 WRITE(6,50) NME,NME,NME,NME
   50 FORMAT('C/'2A2,'   /'2A2,'/99'/'T'/'C/'2A2,'  /'2A2,'/99',/'T')
      WRITE(6,60) NME,NME
   60 FORMAT('C/',2A2,' /',2A2,'/99'/'FIL TSENDPRC')
C
C  MODIFY FILE TO BE SENT TO NOS
C
      IF(CNT.LT.10) WRITE(6,70) NME,NME,NME,CNT,INFO,NME,CNT
   70 FORMAT('ED '2A2,/'N',/'C/'2A2,'/'2A2,I1,'/100'/'T'/
     1 'C/INFO/',I1,'/100'/'FIL ',2A2,I1)
      IF(CNT.GE.10.AND.CNT.LT.100) WRITE(6,80) NME,NME,NME,CNT,
     1 INFO,NME,CNT
   80 FORMAT('ED '2A2,/'N',/'C/'2A2,'/'2A2,I2,'/100'/'T'/
     1 'C/INFO/',I1,'/100'/'FIL ',2A2,I2)
      IF(CNT.GE.100.AND.CNT.LT.1000) WRITE(6,90) NME,NME,NME,CNT,
     1 INFO,NME,CNT
   90 FORMAT('ED '2A2,/'N',/'C/'2A2,'/'2A2,I3,'/100'/'T'/
     1 'C/INFO/',I1,'/100'/'FIL ',2A2,I3)
      WRITE(6,100)
  100 FORMAT('R TEMP')
      CALL SRCH$$(4,'UPDTE',5,2,0,ICODE)
      RETURN
```

```
      END
      SUBROUTINE ERRMSG(ERROR)
      CALL SRCH$$(2,'ERRFILE',7,2,0,ICODE)
      IF(ERROR.NE.1) GO TO 15
      WRITE(6,10)
   10 FORMAT(' MORE THAN 999 ITERATIONS. STOP PROGRAM.')
      GO TO 100
   15 IF(ERROR.NE.2) GO TO 100
      WRITE(6,20)
   20 FORMAT(' CANNOT FIND COMET. STOP PROGRAM.')
      GO TO 100
  100 CALL SRCH$$(4,'ERRFILE',7,2,0,ICODE)
      RETURN
      END
      SUBROUTINE STCHEK(STAT,ICHK)
C
C  THIS SUBROUTINE CHECKS STATUS UNTIL FILE IS RETURNED
C       FROM NOS
C
      INTEGER*2 SNAME(3),RNAME(3),UFD(3),STAT
      INTEGER*4 TIMER,CNTR
      TIMER = 120000
      CNTR = 0
C
C  PUT JOB TO SLEEP FOR TWO MINUTES
C
      CALL SLEEP$(TIMER)
C
C  DETERMINE FILE NAME FOR NONREPEATABLE PART
C
      IF(ICHK.EQ.0) GO TO 5
      CALL SRCH$$(1,'NRSTAT',6,1,0,ICODE)
      READ(5,10) SNAME
      CALL SRCH$$(4,'NRSTAT',6,1,0,ICODE)
      CALL SRCH$$(5,'NRSTAT',6,1,0,ICODE)
      GO TO 20
C
C  DETERMINE FILE NAME FOR REPEATABLE PART
C
    5 CALL SRCH$$(1,'TSTATIN',7,1,0,ICODE)
      READ(5,10) SNAME
   10 FORMAT(3A2)
      CALL SRCH$$(4,'TSTATIN',7,1,0,ICODE)
C
C  CHECK STATUS UNTIL STAT NE 0
C
   20 CALL COMSTA(SNAME,RNAME,UFD,STAT)
      IF(STAT.NE.0) GO TO 30
      CALL SLEEP$(TIMER)
```

```
C
C   INCREASE SLEEP PERIOD EVERY 5 ITERATIONS
C
      CNTR = CNTR+1
      IF(CNTR.LT.5) GO TO 20
      CNTR = 0
      TIMER = TIMER*2
      GO TO 20
   30 RETURN
      END
       SUBROUTINE COMSTA(SNAME,RNAME,UFD,STAT)
C
C    COMSTAT -COMET STATUS. RETURNS STATUS OF FILE TRANSMITTED
C                TO THE HOST CDC COMPUTER.
C
C   CALLING SEQQUENCE :
C
C   SNAME -THREE WORD ARRAY IN CALLING PROGRAM CONTAING
C          NAME OF FILE SENT TO HOST. LEFT JUSTIFIED,BLANK FILL.
C   RNAME -THREE WORD ARRAY IN CALLING PROGRAM WHERE NAME
C          OF FILE RETURNED FROM HOST IS STORED. LEFT
C          JUSTIFIED,BLANK FILL.
C
C   UFD -THREE WORD ARRAY IN CALLING PROGRAM WHERE UFD OF RETURNED FILE IS STORED
C        THE CALLER'S LOGIN UFD,OTHERWISE , COMET.
C
C          6CHARACTERS LEFT JUSTFIED,BLANK FILL.
C
C   STAT -STATUS OF FILE SEND TO HOST.
C         ONE WORD INTEGER IN CALLING PROGRAM.
C
C         -1 = ERROR HAS OCCURED,STATUS CANNOT BE DETERMINED.
C              THE CONTENTS OF RNAME AND UFD ARE NOT VALID.
C         0 = FILE HAS NOT BEEN RETURNED.
C             THE CONTENTS OF RNAME AND UFD ARE NOT VALID.
C          1 = FILE HAS BEEN RETURNED.
C            THE CONTENTS OF RNAME AND UFD ARE VALID.
       INTEGER*2 SNAME(3),RNAME(3),UFD(3),RJDATA(30),ERR
       INTEGER*2 STAT,CODE,RNW,COMUFD(3),BLANKS(3)
       INTEGER*4 POS,EOFPOS
$INSERT SYSCOM>KEYS.F
       LOGICAL NAMEQV,FOUND
       DATA FOUND/.FALSE./
       DATA COMUFD/'COMET'/
       DATA BLANKS/'      '/
C     ATTACH TO UFD COMET
       CALL ATTACH(COMUFD,INTS(:100000),BLANKS,0,$900)
       CALL TNOU('ATT TO COM',10)
C      WRITE(1,777)STAT
```

```
777   FORMAT('STAT = ',I5)
C     TRY TO OPEN STATUS FILE ON UNIT 12
5     CALL OPRDWT('STATUS',12,CODE)
      IF(CODE.NE.0) GO TO 950
C          OPRDWT CYCLES UNTIL 'STATUS' IS UPDATED DURING RE-WRITE
      CALL TNOU('OPEN STATUS',11)
C          SET POS TO EOF AND THEN
C          REWIND SO WE CAN HAVE
C          A VALID EOF POSITION
      CALL PRWF$$(K$POSN,12,LOC(0),0,10000000,RNW,CODE)
      CALL PRWF$$(K$RPOS,12,LOC(0),0,POS,RNW,CODE)
      EOFPOS=POS
C
C
      CALL PRWF$$(K$POSN,12,LOC(0),0,-10000000,RNW,CODE)
      CALL PRWF$$(K$RPOS,12,LOC(0),0,POS,RNW,CODE)
C
C
C     READ A STATUS ENTRY AND LOOK FOR MATCH WITH SNAME
10    CALL PRWF$$(K$RPOS,12,LOC(0),0,POS,RNW,CODE)
      IF(CODE.NE.0) GO TO 100
      IF(POS.GE.EOFPOS) GO TO 100
      CALL I$AD07(12,RJDATA,30)
      IF(.NOT.(NAMEQV(SNAME(1),RJDATA(1)))) GO TO 10
C    NAME FOUND,CHECK STATUS
      CALL TNOU('FOUND NAME',10)
      FOUND=.TRUE.
      WRITE(1,444) RJDATA(30)
      IF(RJDATA(30).LE.0)GO TO 800
444   FORMAT('RJDATA 30  = ',I5)
C
C   CHECK FILE NAME FOR T$ PREFIX,IF PREFIX NOT
C   PRESENT,AN ERROR HAS OCCURED.
C
C     IF(RJDATA(17).NE.'T$')GO TO 800
C
C  CHECK FOR REPEATABLE NAME TCMNIO
C
      IF(RJDATA(17).EQ.'TC'.AND.RJDATA(18).EQ.'MN'.AND.
     1 RJDATA(19).EQ.'IO') GO TO 19
C
C  CHECK FOR NONREPEATABLE NAME TNREPT
C
      IF(RJDATA(17).EQ.'TN'.AND.RJDATA(18).EQ.'RE'.AND.
     1 RJDATA(19).EQ.'PT') GO TO 19
      GO TO 800
C   FILE HAS BEEN RETURNED,COPY NAME TO RNAME.
 19   DO 20 I=1,3
      RNAME(I)=RJDATA(I+16)
```

```
20     CONTINUE
C     GET UFD STATUS
       IF(RJDATA(30).EQ.1)GO TO 50
       DO 25 I=1,3
            UFD(I)=COMUFD(I)
25          CONTINUE
       GO TO 60
50     DO 55 I=1,3
       UFD(I)=RJDATA(I+6)
55     CONTINUE
60     CONTINUE
C    SET RETURN STATUS TO FOUND
       STAT = 1
       GO TO 10
100    CONTINUE
       IF(FOUND) STAT=1
       CALL ATTACH(0,0,0,0,0)
       CALL SEARCH(4,0,12)
       RETURN
800    CONTINUE
C     SNAME NOT FOUND,SET STATUS AND RETURN
       STAT = 0
       DO 810 I=1,3
            RNAME(I)=BLANKS(I)
            UFD(I)=BLANKS(I)
810         CONTINUE
       CALL TNOU('NAME NOT FOUND',14)
       FOUND=.FALSE.
       GO TO 100
900    CONTINUE
C       CANNOT FIND COMET,SET ERROR STATUS AND RETURN
       STAT = -1
       GO TO 100
950    CONTINUE
C         CANNOT OPEN STATUS,IF BUSY,TRY AGAIN.
       CALL GETERR(ERR,1)
       IF(ERR.EQ.'SI')GO TO 5
C    FILE NOT FOUND MEANS NOT RETURNED YET.
       CALL TNOU('CAN''T OPEN STATUS',17)
       GO TO 900
       END
C OPRDWT->OPEN READ AND WAIT
       SUBROUTINE OPRDWT(NAME,UNIT,RCODE)
C
       INTEGER*2 NAME(3),UNIT,TYPE,CODE,I,RCODE
C
$INSERT SYSCOM>KEYS.F
       DATA I/0/
       RCODE=0
```

```
C
10    CALL SRCH$$(K$EXST,'T$0001',6,1,0,CODE)
C
C          IF T$0001 DOES EXIST THEN STATUS IS BEING RE-WRITTEN
C          SO LOOP UNTIL THRU WITH UPDATE FUNCTION
C
      IF(CODE.EQ.0) GO TO 999
15    DO 200 I=1,25 /*KEEP TRYING*/
      CALL SRCH$$(K$READ,NAME,6,UNIT,TYPE,CODE)
      IF(CODE.EQ.0) RETURN /*SUCCESSFULL*/
      IF(CODE.EQ.4.OR.CODE.EQ.5) GO TO 100
      RCODE=1
      RETURN
100   CALL SLEEP$(INTL(5000))
C
C          SLEEP FOR 5 SECONDS
C
200   CONTINUE
      RCODE=1
      RETURN
999   CALL SLEEP$(INTL(5000))
      GO TO 10
      END
```

APPENDIX B.- LISTING OF MAIN PROGRAM FOR CONMIN.

```
C       PROGRAM CONMS1(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,
C      1TAPE7,TAPE8,TAPE9,TAPE10,TAPE11)
C       NEW CONMIN SUBROUTINE
        COMMON/CNMN1/DELFUN,DABFUN,FDCH,FDCHM,CT,CTMIN,CTL,CTLMIN,
       1ALPHAX,ABOBJ1,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(200),SCAL(20),DF(20),
       1A(20,100),S(20),G1(200),G2(200),B(100,100),C(100),ISC(200),
       2IC(100),MS1(200)
        COMMON/CONSAV/RSAV(50),ISAV(25)
        DIMENSION KSC(200),SAVX(20),SAVG(200)
        INTEGER*2 ICODE
C       READ(10,PASSAGE)
        READ(11,10) NPASS
   10   FORMAT(I1)
C       FIRST PASS,NPASS=1,SUBSEQUENTLY NPASS=2.
        GO TO(100,200),NPASS
   100  CONTINUE
C       READ(5,CONPAR)
        READ(5,*) IPRINT,NDV,ITMAX,NCON,NFDG,NSIDE,ICNDIR,
       1NSCAL,LINOBJ,ITRM,FDCH,FDCHM,CT,CTMIN,CTL,CTLMIN,
       2THETA,PHI,DELFUN,DABFUN,JSC,N1,N2,N3,N4,N5,
```

```
     3ALPHAX,ABOBJ1,IGOTO,(VLB(I),I=1,NDV),(VUB(I),I=1,NDV)
      JITER = 0
      MSC = 0
      IF(JSC.LT.0) MSC = 1
      DO 15 I = 1,NCON
      ISC(I) = MSC
   15 CONTINUE
      IF(JSC.EQ.0.OR.JSC.EQ.-999) GO TO 13
      MSC = 1
      IF(JSC.LT.0) MSC = 0
      IF(JSC.LT.0) JSC = JSC*-1
      CALL SRCH$$(INTS(1),'ISCFILE',INTS(7),INTS(4),INTS(0),ICODE)
      READ(8,12) (KSC(I),I=1,JSC)
   12 FORMAT(10I4)
      CALL SRCH$$(INTS(4),'ISCFILE',INTS(7),INTS(4),INTS(0),ICODE)
      DO 11 I = 1,JSC
      ITEMP = KSC(I)
      ISC(ITEMP) = MSC
   11 CONTINUE
   13 REWIND 13
      WRITE(13,14) JITER,NDV,NCON,(ISC(I),I=1,NCON)
   14 FORMAT(20I5)
C     READ(8,STARTX)
      CALL SRCH$$(INTS(1),'STARTX',INTS(6),INTS(4),INTS(0),ICODE)
      READ(8,30) NDV,(X(I),I=1,NDV)
   30 FORMAT(I5/(6E13.5))
      CALL SRCH$$(INTS(4),'STARTX',INTS(6),INTS(4),INTS(0),ICODE)
      GO TO 201
  200 CONTINUE
C     READ(7,SAVE)
      READ(7) 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,ABOBJ1
      REWIND 7
C     READ(9,LINKE)
      READ(9,40) OBJ,G
   40 FORMAT(E13.5/(6E13.5))
      ITERSV = ITER
      DO 41 I = 1,NCON
      SAVG(I) = G(I)
   41 CONTINUE
      DO 42 I = 1,NDV
      SAVX(I) = X(I)
   42 CONTINUE
      SAVOBJ = OBJ
      SAVCT = CT
```

```
      SAVCTL = CTL
      REWIND 13
      READ(13,14) JITER,NDV,NCON,(ISC(I),I=1,NCON)
      JITER = JITER+1
  201 CONTINUE
      NPASS=2
      REWIND 11
C     WRITE(10,PASSAGE)
      WRITE(11,10) NPASS
C     SOLVE OPTIMIZATION
      CALL CONMIN(X,VLB,VUB,G,SCAL,DF,A,S,G1,G2,B,C,
     *ISC,IC,MS1,N1,N2,N3,N4,N5)
      IF(JITER.EQ.0) GO TO 48
      IF(JITER.EQ.1) GO TO 47
      IF(ITER.EQ.ITERSV) GO TO 48
   47 ITERSV = ITER
      REWIND 13
      WRITE(13,14) JITER,NDV,NCON,(ISC(I),I=1,NCON)
      REWIND 14
      WRITE(14,45) SAVOBJ,(SAVX(I),I=1,NDV),SAVCT,SAVCTL
     1 ,(SAVG(I),I=1,NCON)
   45 FORMAT(6E13.5)
C     FUNCTION AND CONSTRAINT VALUES
C     WRITE(7,SAVE)
   48 WRITE(7) 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,ABOBJ1
C     WRITE(9,LINKF)
      REWIND 9
      WRITE(9,50)NDV,(X(I),I=1,NDV)
   50 FORMAT(I5/(6E13.5))
C     WRITE CONTROL CARD STORED IN PROCFILE GONOGO.
  101 FORMAT('1')
      IF(IGOTO.NE.0) GO TO 301
      CALL SRCH$$(INTS(2),'GONOGO',INTS(6),INTS(8),INTS(0),ICODE)
      WRITE(12,101)
      CALL SRCH$$(INTS(4),'GONOGO',INTS(6),INTS(8),INTS(0),ICODE)
C     WRITE ON TAPE8 IF GRADIENTS ARE REQUIRED
      REWIND 13
      JITER = JITER+1
      WRITE(13,14) JITER,NDV,NCON,(ISC(I),I=1,NCON)
      WRITE(14,45) OBJ,(X(I),I=1,NDV),CT,CTL,(G(I),I=1,NCON)
  301 IF(INFO.NE.2) GO TO 303
      CALL SRCH$$(INTS(2),'STARTX',INTS(6),INTS(4),INTS(0),ICODE)
      WRITE(8,30) NDV,(X(I),I=1,NDV)
      CALL SRCH$$(INTS(4),'STARTX',INTS(6),INTS(4),INTS(0),ICODE)


  303 CALL SRCH$$(INTS(2),'INFO',INTS(4),INTS(4),INTS(0),ICODE)
      WRITE(8,304) INFO
  304 FORMAT(I1)
      CALL SRCH$$(INTS(4),'INFO',INTS(4),INTS(4),INTS(0),ICODE)
      STOP
      END
```

APPENDIX C.- LISTING OF FRONT PROCESSOR.

```
C       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)
C       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)
C       READ(5,LINKF)
        READ(5,15)NDV,(X(I),I=1,NDV)
   15   FORMAT(I5/(6E13.5))
        XIN1=1./X(1)
        XIN2=1./X(2)
        XIN3=1./X(3)
C       WRITE E23 ELEMENTS
        WRITE(6,200)
   200  FORMAT(' E23 SECTION PROPERTIES')
        WRITE(6,201) XIN1
   201  FORMAT(' 1 ',F8.3)
C       COMPUTE VALUES FOR DSY CARDS
C       WRITE E21 ELEMENTS
        WRITE(6,202)
   202  FORMAT(' E21 SECTION PROPERTIES')
        AREA0=(2.*B10+B20)*T0
        AREA=AREA0*XIN2
        SCALE=SQRT(AREA/AREA0)
        B1=B10*SCALE
        B2=B20*SCALE
        T=T0*SCALE
        EI1=B1*(B2+2.*T)**3/12.-(B1-T)*B2**3/12.
        ALPHA1=0.
        C=(B2+2.*T)*B1**2/2.-B2*(B1-T)*(B1+T)/2.
        C=C/AREA
        EI2=2.*T*B1**3/12.+2.*T*B1*(B1/2.-C)**2
       1+B2*T**3/12.+B2*T*(C-T/2.)**2
```

```
      ALPHA2=0.
      F=(2.*B1+B2)*T**3/3.
      F1=0.
      Z1=((B1*B2)**2)*T/(4.*EI1)+C-T/2.
      Z2=0.
      THETA=0.
      Q1=0.
      Q2=0.
      Q3=0.
      Y11=-(B1-C)
      Y12=.5*B2+T
      Y21=C
      Y22=.5*B2+T
      Y31=C
      Y32=-(.5*B2+T)
      Y41=-(B1-C)
      Y42=-(.5*B2+T)
      J=1
      WRITE(6,103) J,EI1,ALPHA1,EI2,ALPHA2,AREA
      WRITE(6,1003) F,F1,Z1,Z2,THETA
  103 FORMAT('DSY',I2,5F12.4,'%')
 1003 FORMAT(1X,5F13.5)
      WRITE(6,104) Q1,Q2,Q3,Y11,Y12,Y21
  104 FORMAT(1X,6F12.4,'%')
      WRITE(6,1004) Y22,Y31,Y32,Y41,Y42
 1004 FORMAT(1X,5F12.4)
C     WRITE E41 ELEMENTS
      WRITE(6,300) XIN3
  300 FORMAT(' SHELL SECTION PROPERTIES'/' 1',F8.3)
      WRITE(6,301)
  301 FORMAT('@')
      STOP
      END
```

APPENDIX D.- LISTING OF END PROCESSOR.

```
      PROGRAM EPFUS(INPUT,OUTPUT,TAPE8,TAPE5=INPUT,TAPE6=OUTPUT)
      DIMENSION A(4500),B(1400)
      DIMENSION A1(400),B1(400),C1(400)
      DIMENSION G(400)
      NAMELIST/EPIN/E23AL,E21AL,E41AL,NSE23,NSE21,NSE41
C     NAMELIST/LINKE/OBJ,G
      READ(5,EPIN)
      CALL DAL(4,11,A(1),0,IEA,KADR,IERR,NWDS,NE,LB,ITYPE,
     14HOBJF,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)
      G(I)=ABS(A1(I1))/E23AL-1.
      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)
      G(I)=GNUM/E21AL-1.
      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/E41AL-1.
      K1=K1+1
   10 CONTINUE
      CALL FIN(0,0)
      REWIND 6
C     WRITE(6,LINKE)
      WRITE(6,232) OBJ,G
  232 FORMAT(*BEGIN CNMNIO*/E13.5/(6E13.5))
      WRITE(6,233)
  233 FORMAT(*END CNMNIO*)
      STOP
      END
```

APPENDIX E.- LISTING OF BLDELDS.

```
      PROGRAM BLDELDS(TAPE5,TAPE23,TAPE20,TAPE21,TAPE22,OUTPUT)
C
C   THIS PROGRAM CREATES A RUNSTREAM THAT WILL CREATE
C   DMDV AND DKDV FOR PARTICULAR ELEMENTS USED DESIGN VARIABLES
C
      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,4HNNOD,4HISCT/
      DATA START,END,XNSECT/4H$STA,4H$END,4HNSEC/
      DATA YNSECT/3HNSE/
C
C   CALL SUBROUTINE TO REMOVE BEGINNING BLANKS
C
      CALL REMOVE
C
C   READ INPUT
C
C     NOEL=NUMBER OF ELEMENTS
C     NODV=NUMBER OF DESIGN VARIABLE ELEMENTS
C     VORB=TYPE OF ANALYSIS (EX. BUCKLING)
C     NDF=NUMBER OF DEGREES OF FREEDOM PER JOINT
C     EL - ELEMENT NAMES CONTAINING DESIGN VARIABLES
C                  (EX. E21)
```

```
C       NSECT - LAST SECTION NUMBER USED FOR EACH DESIGN VARIABLE
C       NODVPE - NUMBER OF DESIGN VARIABLES PER ELEMENT
C
        READ(5,5) NOLC,NODV,ISNOLC,JOINTS,NDF,NOEL,VORB
    5   FORMAT(6(1X,I4),1X,A4)
        READ(5,6) (EL(I),NSECT(I),NODVPE(I),I=1,NOEL)
    6   FORMAT(6(1X,A3,1X,I3,1X,I3))
        NDF=NDF*NDF
        WRITE(20,2)
    2   FORMAT(*[XQT TAB*/* UPDATE=1*)
C
C   LOOP ON NUMBER OF ELEMENTS
C
        DO 30 I = 1,NOEL
C
C   CALL SUBROUTINE TO UPDATE TAB BY SETTING DESIGN VARIABLES
C   TO UNITY.
C   KCNT RETURNS THE NUMBER OF POSSIBLE DESIGN VARIABLES FOR
C   A PARTICULAR ELEMENT.
C   KCNT=99 MEANS THE ELEMENT NAME IS BAD
C
        CALL TABNPUT(EL(I),NSECT(I),KCNT)
        IF(KCNT.NE.99) GO TO 30
        PRINT 29,EL(I)
   29   FORMAT(* ELEMENT NAME *,A3,* DOES NOT EXIST*)
        GO TO 190
   30   CONTINUE
C
C   CONCLUDE UPDATE AND SET UP DISABLES
C
        WRITE(20,31)
   31   FORMAT(* UPDATE=0*)
        WRITE(20,32)
   32   FORMAT(*[XQT 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
C   SET COUNTER FOR TOTAL NUMBER OF DESIGN VARIABLES
C
        IELCNT = 0
        ICNTDV = 1
```

```
      ISW=0
      ISAVCNT = 0
C
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(*[XQT ELD*)
      ISW=1
      READ(21,80) (FOR(J),J=1,9)
   80 FORMAT(A3,A7,7A10)
C
C  SET ICNT = NUMBER OF POSSIBLE DESIGN VARIABLES FOR AN ELEMENT
C
      ICNT=99
C
C  DETERMINE POSSIBLE NUMBER OF DESIGN VARIABLES PER ELEMENT
C
      IF(FOR(1).EQ.E23.OR.FOR(1).EQ.E41.OR.FOR(1).EQ.E44) ICNT=1
      IF(FOR(1).EQ.E31) ICNT=1
      IF(FOR(1).EQ.E21) ICNT=4
      IF(FOR(1).EQ.E43.OR.FOR(1).EQ.E33) ICNT=12
      IF(FOR(1).EQ.E22) ICNT=21
      TNAME=FOR(1)
      IF(ICNT.EQ.99) TNAME=SAVNAM
      JCNT = JCNT+1
      IF(ICNT.NE.99) IELCNT=IELCNT+1
C
C  SET UNIT = 20 IF ONLY ONE DESIGN VARIABLE PER ELEMENT
C  OTHERWISE SET UNIT = 22 (SCRATCH UNIT)
C
      IUNIT=22
      REWIND 22
      IF(ICNT.EQ.1) IUNIT=20
      IF(ICNT.EQ.99.AND.ISAVCNT.EQ.1) IUNIT=20
C
C CHECK FOR REPEAT OF ELEMENT NAME
C
      IF(ICNT.NE.99) GO TO 86
      WRITE(IUNIT,85) SAVNAM
   85 FORMAT(A3,77X)
      GO TO 860
C
```

```
C  READ DATA FROM UNIT 21 AND WRITE DATA ON UNIT 20 OR 22
C  DEPENDING UPON VALUE OF ICNT
C
  86  WRITE(IUNIT,50) (FOR(J),J=1,9)
      SAVNAM=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).EQ.XNSECT.OR.FOR(1).EQ.YNSECT) GO TO 88
      IF(FOR(1).EQ.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).EQ.END) GO TO 110
      WRITE(IUNIT,50) (FOR(J),J=1,9)
      GO TO 100
C
C  SKIP THIS IF MORE THAN ONE DESIGN VARIABLE PER ELEMENT
C
 110  IF(ICNT.NE.1) GO TO 120
C
C  CALL SUBROUTINE TO CREATE REMAINDER OF RUNSTREAM
C
      CALL CRRS(ICNTDV,0,0,TNAME,NDF,NODVPE(IELCNT))
      ICNTDV = ICNTDV+NODVPE(IELCNT)
      GO TO 39
C
C  LOOP ON POSSIBLE NUMBER OF DESIGN VARIABLES PER ELEMENT
C          READ FROM UNIT 22
C
C          WRITE ON UNIT 20
C
 120  DO 160 I = 1,ICNT
      IF(I.NE.1) WRITE(20,75)
      REWIND 22
 130  READ(22,50) (FOR(J),J=1,9)
      IF(EOF(22)) 150,140
 140  IF(FOR(1).EQ.XNSECT.OR.FOR(1).EQ.YNSECT) GO TO 141
      WRITE(20,50) (FOR(J),J=1,9)
      GO TO 130
 141  WRITE(20,90) IP1
      GO TO 130
C
C  CALL SUBROUTINE TO CREATE REMAINDER OF RUNSTREAM
```

```
C
 150  CALL CRRS(ICNTDV,ICNT,I,TNAME,NDF,NODVPE(IELCNT))
      IP1=IP1+1
 160  CONTINUE
      ICNTDV = ICNTDV+NODVPE(IELCNT)
      GO TO 39
 170  WRITE(20,180)
 180  FORMAT(* TOC 2*/*[XQT EXIT*)
 190  STOP
      END
      SUBROUTINE TABNPUT(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
      DIMENSION ELEMENT(21)
      DATA E21,E22,E23,E41,E43,E44/3HE21,3HE22,3HE23,3HE41,3HE43,3HE44/
      DATA E31,E33/3HE31,3HE33/
      DATA BA,BB,BC,SA,SB/2HBA,2HBB,2HBC,2HSA,2HSB/
      IF(ELNAME.EQ.E21) ELID=BA
      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
C   ELEMENT NAMES E23 , E31 , E41 , E44
C           ICNT = 1
C
  30  K=NSCT+1
      WRITE(20,40) K
  40  FORMAT(1X,I3,* 1.0*)
      ICNT=1
      GO TO 210
C
```

```
C  ELEMENT NAME E21
C           ICNT = 4
C
  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
C  ELEMENT NAMES E33 , E43
C           ICNT = 12
C
 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(I)=1.
      K=I+NSCT
      IF(I.EQ.1) WRITE(20,115)
 115  FORMAT(* FORMAT=UNCOUPLED*)
      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
C  ELEMENT NAME E22
C           ICNT = 21
C
```

```
  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
C  THIS SUBROUTINE CREATES REMAINDER OF RUNSTREAM
C  TO FIND DMDV AND DKDV.
C        USE NAMES
C           DMDV DIAG 0 I AND
C           DKDV SPAR 25 I
C        WHERE I = 1 TO NUMBER OF POSSIBLE DESIGN VARIABLES
C
       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,
      2 2H66/
       DATA SAEL/2H11,2H12,2H22,2H13,2H23,2H33,2H44,2H45,2H55,
      1 2H46,2H56,2H66/
       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
C
C  XQT E , EKS , TOPO , K , DCU
```

```
C
      WRITE(20,10)
   10 FORMAT(*[XQT E*/*[XQT EKS*/*[XQT TOPO*/*[XQT K*/*[XQT DCU*)
C
C  DISABLE DATA SETS
C
      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 1,*A4,2X,A4)
   30 CONTINUE
C
C  SET UP ELEMENT NAMES FOR CHANGE AND COPY STATEMENTS
C
      DDV=DV
      DDK=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
      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
C  THIS SUBROUTINE REMOVES THE LEADING BLANKS FROM EACH
C  LINE IN THE RUNSTREAM.
C
      DIMENSION DATIN(80)
```

```
      DATA BLANK/1H /
      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
```

APPENDIX F.- LISTING OF GRDXXXX.

```
      PROGRAM GNGRDRS(INPUT,TAPE30,TAPE31,TAPE10,TAPE5=INPUT,OUTPUT)
C
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
C
      DIMENSION EL(999),NSECT(999),NUM(8),MFORM(5),NODVPE(999)
      DATA NUM/1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8/
      DATA BUCK,VIBR/4HBUCK,4HVIBR/
      DATA E21,E22,E33,E43/3HE21,3HE22,3HE33,3HE43/
      DATA MFORM(1)/10H(*Z11=UNIO/
      DATA MFORM(4)/10H(I1,1X,*Z*/
      DATA MFORM(5)/9H),I1,*)*)/
      DATA MFORM7/9HN(Z10,Z*,/
      DATA MFORM8/5HN(Z*,/
C
C   READ INPUT
C
C     NOEL = NUMBER OF DIFFERENT ELEMENTS
```

```
C     NOLC = NUMBER OF LOAD CASES
C     NODV = TOTALNUMBER 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)
C
      READ(5,60) NOLC,NODV,ISNOLC,JOINTS,NDF,NOEL,VORB
   60 FORMAT(6(1X,I4),1X,A4)
      READ(5,61) (EL(I),NSECT(I),NODVPE(I),I=1,NOEL)
   61 FORMAT(6(1X,A3,1X,I3,1X,I3))
      ISNOLC=ISNOLC-1
      NDF=NDF*NDF
C
C  DETERMINE IF THERE IS A E21 , E22 , E33 , OR E43 ELEMENT
C
      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
C
C  CALL SUBROUTINE TO FIND DK/DV AND DM/DV FOR E21 ELEMENTS
C
  161 CALL DKDVE21(ICNTDV,NDF)
      GO TO 166
C
C  CALL SUBROUTINE TO FIND DK/DV AND DM/DV FOR E22 ELEMENTS
C
  163 CALL DKDVE22(ICNTDV,NDF)
      GO TO 166
C
C  CALL SUBROUTINE TO FIND DK/DV AND DM/DV FOR E43 ELEMENTS
C
  164 CALL DKDVE43(ICNTDV,NDF)
      GO TO 166
C
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.VIBR) GO TO 320
      WRITE(10,172)
 172  FORMAT(*[XQT AUS*)
C
C  FIND OBJECTIVE FUNCTIONS
C
 162  WRITE(10,175) JOINTS
 175  FORMAT(* SYSVEC;UNIT VEC*/*  I=1; J=1,*,I8,*; 1.0*/
     1 * DEFINE UN=UNIT VEC*)
      DO 200 I = 1,NODV
      WRITE(10,180) I,I
  180 FORMAT(* DEFINE W*,I3,*=DMDV DIAG 0 *,I3)
  200 CONTINUE
      DO 250 I = 1,NODV
      WRITE(10,220) I,I
  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
C  CALL SUBROUTINE TO CREATE RUNSTREAM FOR DERIVATIVE
C  CALCULATION
C
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)
   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
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*/* LIB=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)
  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
C  CHANGE DATA SET NAMES
C
```

```
      IDV=0
      DO 31 J=1,NOEL
      NOSECT = 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
   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*/*EXQT EXIT*)
      STOP
      END
```

APPENDIX G.- LISTING OF DRVXXXX.

```
      PROGRAM DRVSTRS(INPUT=65,TAPE30,TAPE31,TAPE5=INPUT
     1 ,TAPE15,TAPE16,OUTPUT=65,TAPE6=OUTPUT)
C
C    THIS PROGRAM COMPUTES THE STRESSES AND STRESS DERIVATIVES
C    FOR SPAR BEAM ELEMENTS USING SPAR LIBRARIES AS INPUT.
C
C    THE STRESSES ARE COMPUTED FROM SPARLA USING DATA SET
C                FAMS MASK  I  1 WHERE I = 1 TO NOLC
C    THE STRESSES ARE WRITTEN BACK ONTO SPARLA USING DATA SET
C                STRS MASK  I  1 WHERE I = 1 TO NOLC
C
C    THE STRESS DERIVATIVES ARE COMPUTED FROM SPARLC USING DATA SET
C         DFAM MASK  I  J WHERE I = ISTRTLC TO ISTRTLC+NOLC AND J = 1 TO NODV
C    THE STRESS DERIVATIVES ARE WRITTEN BACK ONTO SPARLC USING DATA SET
C         DSTR MASK  I  J WHERE I = ISTRTLC TO ISTRTLC+NOLC AND J = 1 TO NODV
C
      COMMON KORE,KEVEN,Z(1)
      DIMENSION NAME1(2),NAME2(2),EL(999),NODVPE(999)
      DATA E21,E22,E33,E43/3HE21,3HE22,3HE33,3HE43/
      DATA NAME1/4HFAMS,4HDFAM/
      DATA NAME2/4HSTRS,4HDSTR/
C
C    READ INPUT VALUES
C
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 VRIABLES
C    ISTRTLC IS THE STARTING NUMBER FOR THE DERIVATIVE LOAD CASES
C    EL =NAMES OF ELEMENTS USED AS DESIGN VARIABLES
```

```
C                (EX. E21)
C
      READ(5,1) NOLC,NODV,ISNOLC,JOINTS,NDF,NOEL,VORB
    1 FORMAT(6(1X,I4),1X,A4)
      READ(5,2) (EL(I),NSECT,NODVPE(I),I=1,NOEL)
    2 FORMAT(6(1X,A3,1X,I3,1X,I3))
      ISTRTLC=ISNOLC-1
      CALL RSET(IL,0,0)
C
C   LOOP ON THE NUMBER OF LOAD CASES
C
      DO 4 NCNTLC = 1,NOLC
C
C   LOOP ON NUMBER OF ELEMENTS
C
      DO 10 ISTRS = 1,NOEL
C
C   CHECK FOR E21,E22,E33,E43 ELEMENT
C
      IF(EL(ISTRS).EQ.E21.OR.EL(ISTRS).EQ.E43.OR.
     1 EL(ISTRS).EQ.E22.OR.EL(ISTRS).EQ.E33) GO TO 9
      GO TO 10
    9 NODVEL = NODVPE(ISTRS)
C
C   SET NUMBER OF STRESS OR STRESS DERIVATIVES TO BE WRITTEN
C
      IF(EL(ISTRS).EQ.E21.OR.EL(ISTRS).EQ.E22) IWRDCNT = 8
      IF(EL(ISTRS).EQ.E43.OR.EL(ISTRS).EQ.E33) IWRDCNT = 4
C
C   LOOP ON NUMBER OF DESIGN VARIABLES PER ELEMENT
C
      DO 90 JK = 1,NODVEL
      REWIND 15
      REWIND 16
C
C   READ FAMS MASK FROM SPARLA AND COMPUTE STRS MASK
C
      CALL RDATSET(4,NAME1(1),NCNTLC,1,NE1,1,0,EL(ISTRS))
      REWIND 15
      REWIND 16
C
C   CALL SUBROUTINE TO WRITE STRESSES ON SPAR LIBRARY
C
C
C   WRITE STRS MASK ON SPARLA
C
      CALL WRTDATA(4,NAME2(1),NCNTLC,1,NE1,EL(ISTRS),IWRDCNT)
   90 CONTINUE
   10 CONTINUE
```

```
      ICNTDV = 1
C
C  LOOP ON NUMBER OF ELEMENTS
C
      DO 3 ISTRS=1,NOEL
      IF(EL(ISTRS).EQ.E21.OR.EL(ISTRS).EQ.E43.OR.
     1 EL(ISTRS).EQ.E22.OR.EL(ISTRS).EQ.E33) GO TO 19
      GO TO 3
   19 DO 30 ICNTDV = 1,NODV
      IBEAM = 0
      NODVEL = NODVPE(ICNTDV)
C
C  SET SWITCH FOR BEAM ELEMENT
C
      IF(EL(ICNTDV).EQ.E21.OR.EL(ICNTDV).EQ.E22) IBEAM=1
      N3=NCNTLC+ISTRTLC
C
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 RDATSET(4,NAME1(2),N3,ICNTDV,NE1,0,IBEAM,EL(ISTRS))
      REWIND 15
C
C  WRITE DSTR MASK ON SPARLC
C
      CALL WRTDATA(4,NAME2(2),N3,ICNTDV,NE1,EL(ISTRS),IWRDCNT)
   20 CONTINUE
   30 CONTINUE
    3 CONTINUE
    4 CONTINUE
      CALL FIN(0,0)
      STOP
      END
      SUBROUTINE WRTDATA(NU,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 NWD3 GT OPEN CORE SIZE , USE MORE THAN ONE BLOCK
C
       NWD3=NE1*IWRDCNT
       LB3=NWD3
       IF(NWD3.LT.KORE) GO TO 61
       LB3=NWD3/2
  61   ISW=0
       IF(NWD3.LT.KORE) ISW=1
       CALL DAL(NU,0,Z(1),KORE,1,KADR3,IERR,NWD3,NE1,LB3,-1,
      1 N1,X2,N3,ISTRS)
  62   I1=1
  63   I2=I1+IWRDCNT-1
       IF(I2.GT.LB3) GO TO 66
C
C    READ STRESSES OR STRESS DERIVATIVES OFF UNIT 15
C
       READ(15) (Z(I),I=I1,I2)
       IF(EOF(15)) 66,65
  65   I1=I1+IWRDCNT
       GO TO 63
C
C    WRITE STRESSES OR STRESS DERIVATIVES ONTO SPAR LIBRARY
C
  66   CALL RIO(NU,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 RDATSET(NU,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
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
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
```

APPENDIX H.- LISTING OF DKDVE21.

```
      SUBROUTINE DKDVE21(NDVJIM,NDF)
      DIMENSION X(20)
      NAMELIST/LINKF/NDV,X
C
C   THIS SUBROUTINE CREATES A SPAR RUNSTREAM TO CALCULATE
C
C          DK     DK    DA     DK     DI     DK     DI     DK     DJ
C                                      X             Y             O
C          --- = -- . --- + --- . --- + --- . --- + --- . ---
C          DV     DA    DV     DI     DV     DI     DV     DJ     DV
C            I           I      X      I      Y      I      O      I
C
C
C
C          DM     DM    DA     DM     DI     DM     DI     DM     DJ
C                                      X             Y             O
C          --- = -- . --- + --- . --- + --- . --- + --- . ---
C          DV     DA    DV     DI     DV     DI     DV     DJ     DV
C            I           I      X      I      Y      I      O      I
C
      WRITE(10,1)
    1 FORMAT(*[XQT 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
C COMPUTE  DA/DV
C
      DADV=1.
C
C   READ IN AND SET UP INITIALIZATION VALUES
C
C   READ IN CONSTANTS FROM UNIT 30
C
      READ(30,5) B10,B20,TO
    5 FORMAT(3F10.3)
C
C   READ IN DESIGN VARIABLES FROM UNIT 31
C
      READ(31,LINKF)
```

```
      AREA0=(2.*B10+B20)*T0
      AREA=AREA0/X(2)
      SCALE=SQRT(AREA/AREA0)
      B1=B10*SCALE
      B2=B20*SCALE
      T=T0*SCALE
      FTR=0.5/SQRT(AREA*AREA0)
      C=(B2+2.*T)*B1**2/2.-B2*(B1-T)*(B1+T)/2.
      C=C/AREA
C
C  COMPUTE FACTORS FOR DI1/DV , DI2/DV , DJ0/DV
C
      DB1DV=B10*FTR
      DB2DV=B20*FTR
      DTDV=T0*FTR
      DI1DB1=(B2+2.*T)**3/12.-B2**3/12.
      DI1DB2=(B2+2.*T)**2*B1/4.-B2**2*(B1-T)/4.
      DI1DT=(B2+2.*T)**2*B1/2.+B2**3/12.
      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.)**2)+(2.*B2*T*(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.)**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
C  COMPUTE DI1/DV , DI2/DV , DJ0/DV
C
      DI1DV=DI1DB1*DB1DV+DI1DB2*DB2DV+DI1DT*DTDV
      DI2DV=DI2DB1*DB1DV+DI2DB2*DB2DV+DI2DT*DTDV
      DJ0DV=DJ0DB1*DB1DV+DJ0DB2*DB2DV+DJ0DT*DTDV
C
C  CREATE RUNSTREAM TO FIND DK/DV
C
      WRITE(10,106)DADV,DI1DV
      WRITE(10,107)DI2DV,DJ0DV
  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
C  CREATE RUNSTREAM TO FIND DM/DV


C
      WRITE(10,206)DADV,DI1DV
      WRITE(10,207)DI2DV,DJ0DV
  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(*EXQT DCU*/* TOC 1*)
      RETURN
      END
```

APPENDIX I.- LISTING OF BMSTRS.

```
      SUBROUTINE BMSTRS(ISW,KCNT,JCNT,IBEAM)
      COMMON KORE,KEVEN,Z(1)
      DIMENSION DY1DV(4),DY2DV(4),Y(4,2),S(8),F3(2),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
```

```
C   READ IN CONSTANT NUMBERS FROM UNIT 30
C
      READ(30,10) B10,B20,TO
   10 FORMAT(3F10.3)
C
C   READ IN DESIGN VARIABLES FROM UNIT 31
C
      READ(31,LINKF)
C
C   CALCULATE DY/DV
C
      AO=(2.*B10+B20)*TO
      A=AO/X(2)
      SCALE=SQRT(A/AO)
      B1=B10*SCALE
      B2=B20*SCALE
      T=TO*SCALE
      FTR=0.5/SQRT(A*AO)
      C=(B2+2.*T)*B1**2/2.-B2*(B1-T)*(B1+T)/2.
      C=C/A
      DB1DV=B10*FTR
      DB2DV=B20*FTR
      DTDV=TO*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
C   COMPUTE STRESSES OR STRESS DERIVATIVES
C
      S(ICNT)=(F3(II)/A)+((XM1(II)/XI1)*Y(I,2))-((XM2(II)/XI2)
     1 *Y(I,1))
C
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
      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
```

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-83253		

4. Title and Subtitle	5. Report Date
An Implementation of the Distributed Programing Structural Synthesis System (PROSSS)	December 1981
	6. Performing Organization Code 505-33-63-02
7. Author(s)	8. Performing Organization Report No.
James L. Rogers, Jr.	
9. Performing Organization Name and Address	10. Work Unit No.
NASA Langley Research Center Hampton, Virginia 23665	11. Contract or Grant No.
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Technical Memorandum
National Aeronautics and Space Administration Washington, DC 20546	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

This distributed implementation of the PROgraming Structural Synthesis System (PROSSS) combines a general purpose finite-element computer program for structural analysis, a state-of-the-art optimization program, and several user-supplied, problem-dependent computer programs. The results are flexibility of the optimization procedure organization and versatility of the formulation of constraints and design variables. The analysis-optimization process results in a minimized objective function, typically the mass. The analysis and optimization programs are executed repeatedly by looping through the system until the process is stopped by a user-defined termination criterion. However, some of the analysis, such as model definition, need only be one time and the results are saved for future use. The user must write some small, simple FORTRAN programs to interface between the analysis and optimization programs. One of these programs, the front processor, converts the design variables output from the optimizer into a suitable format for input into the analyzer. Another, the end processor, retrieves the behavior variables and, optionally, their gradients from the analysis program and evaluates the objective function and constraints and optionally 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 which finite elements are being used in the model. The analysis and end processor programs are always executed on the mainframe computer because the CPU on the mainframe is much faster than that of the minicomputer. The optimization and front processor programs are always executed on the minicomputer because the optimizer requires a significant memory allocation and the minicomputer has virtual memory.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Optimization, Distributed Processing, Finite-Element Analysis, Minicomputer	FEDD Distribution Subject Category 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	89	

Available: NASA's Industrial Applications Centers