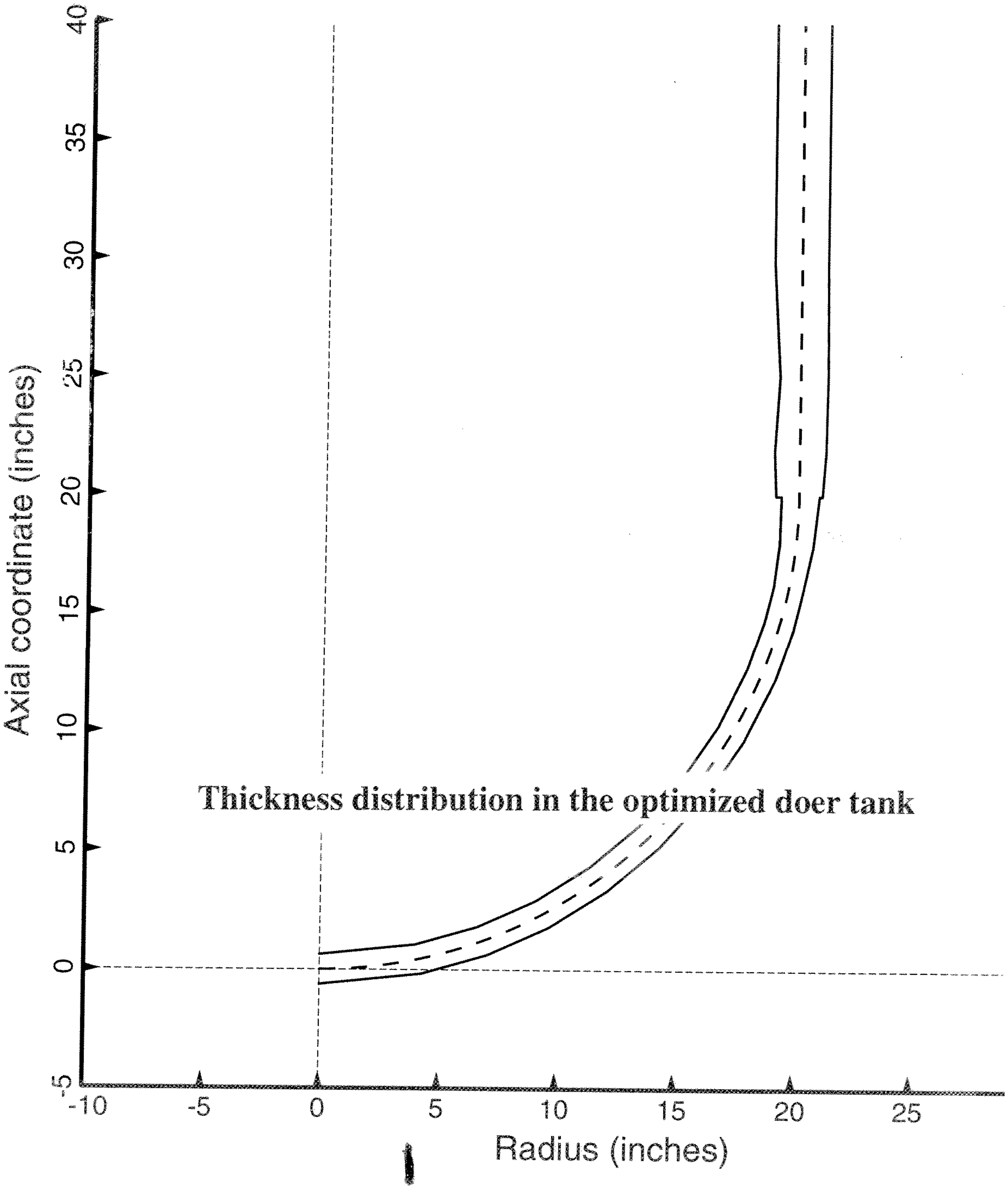
Use of GENOPT and BIGBOSOR4 to obtain optimum designs of a deep submergence tank

David Bushnell

July 1, 2009

ABSTRACT

The GENOPT/BIGBOSOR4 capability [1-3] is used to obtain an optimum design of a titanium cylindrical tank with hemispherical ends. The tank is subjected to 15000 psi uniform external pressure. The objective of the optimization is to minimize the weight of the tank subject to stress and buckling design constraints. The decision variables establish the distribution of the shell wall material in a wide neighborhood of the junction between the hemispherical and cylindrical segments of the tank. The titanium is assumed to remain elastic. Creep is not included. The maximum allowable effective stress is assumed to be 120000 psi. A factor of safety of 1.3 is used for buckling and a factor of safety of 1.0 is used for stress. Enough detail is given so that an engineer or researcher other than the writer will be able to optimize similar shell structures with GENOPT/BIGBOSOR4.



Thickness distribution in the optimized doer tank

INTRODUCTION:

The GENOPT/BIGBOSOR4 capability [1-3] is used to obtain an optimum design of a titanium cylindrical tank with hemispherical ends. The tank is subjected to 15000 psi uniform external pressure. The objective of the optimization is to minimize the weight of the tank subject to stress and buckling design constraints. The decision variables establish the distribution of the shell wall material in a wide neighborhood of the junction between the hemispherical and cylindrical segments of the tank. The titanium is assumed to remain elastic. Creep is not included. The maximum allowable effective stress is assumed to be 120000 psi. A factor of safety of 1.3 is used for buckling and a factor of safety of 1.0 is used for stress.

The GENERIC case described in this report is called "submarine". The SPECIFIC case is called "doer".

Some references are listed in Table 1. The work in this paper was motivated by Grant Cutler of the DOER Company in Alameda, California. The DOER Company wants to build a deep submergence vehicle for operation anywhere in the ocean. This vehicle will include several different geometries of shells that must withstand the high pressure that exists in the deepest parts of the ocean. The example presented here might represent a shell that protects a battery. However, no dimensions or material properties were obtained from the DOER Company during the generation of the results presented here. This is merely an example that the reader can use as a guide for producing analogous results for configurations of particular interest to him or her.

Another, perhaps stronger, motivation to do this work is the desire to encourage engineers and researchers at NASA and elsewhere to use GENOPT at their own facilities. The results presented and the discussion in this "submarine" report provide yet another example that will, it is hoped, make it easier for others to learn how to set up GENOPT cases.

BIGBOSOR4:

BIGBOSOR4 is executed multiple times inside the optimization loop. As is described in [2], the originally "stand alone" version of BOSOR4 [8] has been divided into subroutines: B4READ (the preprocessor), B4MAIN (the mainprocessor), and B4POST (the postprocessor) for execution in an optimization context. The capability of BOSOR4 [8] has been expanded to handle many more shell segments [2]. This expanded version is called "BIGBOSOR4" here and in other reports. BIGBOSOR4 can now handle up to 295 shell segments [5]. There now exists a "stand alone" version of BIGBOSOR4 as well as the version of BIGBOSOR4 designed to be used in connection with GENOPT. This "stand alone" version of BIGBOSOR4 is used to produce plots such as those shown in Figs. 1 - 4.

The models used here are BIGBOSOR4 models. Therefore, the discretization is one-dimensional, which causes solution times on the computer to be much less than for the usual two-dimensionally discretized models such as those used in connection with the STAGS computer program [9 - 11].

SUPEROPT:

SUPEROPT is a script by means of which GENOPT attempts to find a "global" optimum design. A SUPEROPT execution generates about 470 design iterations. SUPEROPT is described in [12]. A SUPEROPT run consists of many optimizations starting from different points in design space. Each new "starting" design in a single SUPEROPT execution is generated randomly but consistent with upper and lower bounds and equality and inequality constraints. The results of a single SUPEROPT execution are displayed in Fig. 5. Each major "spike" in that plot represents a new, randomly obtained, "starting" design.

ABOUT GENOPT:

GENOPT [1 - 5] is a system by means of which one can convert any analysis into a user-friendly analysis and into an optimization capability. GENOPT is not limited to the field of structural mechanics.

In the GENOPT "universe" there are considered to be two types of user: 1. the "GENOPT user", and 2. the "end user". The GENOPT user creates the user-friendly analysis and optimization capability for a class of problems, and the end user uses that capability to find optimum designs for a member of that class. (In this "submarine" case the GENOPT user and the end user are the same person: the writer).

GENOPT user:

It is the duty of the GENOPT user to create user-friendly names, one-line definitions, and "help" paragraphs for the variables to be used in the analysis or analyses [1]. The GENOPT user must also supply software (subroutines and/or FORTRAN statements) that perform the analysis or analyses [1-5]. The GENOPT user must decide what behaviors will constrain the design during optimization cycles, behaviors such as general buckling, local buckling, stress, vibration, etc. While establishing problem variables, the GENOPT user must decide which of 7 roles each of these variables plays [1]. The 7 possible roles are:

1. decision variable candidate (such as a structural dimension)
2. parameter that is not a decision variable candidate (such as a material property)
3. environmental variable (such as a load)
4. behavioral variable (such as a stress)
5. allowable variable (such as a maximum allowable effective stress)
6. factor of safety (such as a factor of safety for stress)
7. objective (such as weight)

End user:

It is the duty of the end user to provide a starting design, loads, and material properties, to choose decision variables, lower and upper bounds, equality constraints, and inequality constraints, and to choose whether to optimize or simply to analyze an existing design or both.

Some advice:

Please read [1] first, followed by the first part of [3], which contains many details about how to use GENOPT. Tables 2 and 3 (taken from [4]) contain some information on the use of GENOPT. In Table 3 a generic name, "cylinder" frequently appears. In [4] the generic name specified by the writer is "weldland". In this report the generic name specified by the writer is "submarine". When studying Table 3 and setting up the proper files at his or her facility, the reader should substitute the generic name, "submarine" for the generic name, "cylinder" used in Table 3. The "setting up" operation in this case called "submarine" is completely analogous to that described for the generic cases called "weldland" in [4] and "trusscomp" in [5].

The optimizer tool:

GENOPT uses the optimizer, ADS, created by Vanderplaats and his colleagues many years ago [6,7]. In GENOPT the ADS software is "hard-wired" in the "1-5-7" mode: a modified steepest descent method.

PRODUCTION OF THE PROGRAM SYSTEM TO OPTIMIZE AN EXTERNALLY PRESSURIZED TANK CONSISTING OF A CYLINDRICAL SHELL WITH HEMISPHERICAL ENDS. THE GENERIC CASE IS CALLED "submarine"

Table 4 lists the run stream used to obtain the results presented here. The reader should use Table 4 as a guide if he or she wants to reproduce the results in this report, or investigate further into other optimum designs in the "submarine" class, or obtain optimum designs of any other shells of revolution in which the intention is to use GENOPT in connection with BIGBOSOR4.

The GENOPT user first provides input during a long GENTEXT interactive session. Reference [3] provides good examples of how this GENTEXT interactive session goes and what GENTEXT does as the interactive session proceeds. GENTEXT records the GENOPT user's input for the long interactive session in a file called *.INP, where "*" represents the GENOPT user's GENERIC name for the case. If the GENOPT user enters a wrong input by mistake, he or she can terminate the GENTEXT interactive session, edit the end of the *.INP file to eliminate the mistake, and then repeat the GENTEXT session, initially using the *.INP file as input rather than tediously repeating the interactive session up to the point where the mistake was made. At the point where the mistake was made GENTEXT returns control to the GENOPT user, and he or she then continues the session in an interactive mode. Table 5 lists the *.INP file after completion of the entire GENTEXT interactive session for the "submarine" case.

GENTEXT automatically produces several files. They are described in [3]. Among them is a file called *.DEF (submarine.DEF), which the GENOPT user should inspect. The *.DEF file includes a glossary of the GENOPT-user-established variables and one-line definitions of the variables. Table 6 is this glossary for the case, "submarine". Additional sections of the submarine.DEF file are repeated at the beginning of the behavior.new library (Table 9), a "skeletal" version of which is automatically produced by GENTEXT. One of the most important files automatically produced by GENTEXT is the prompting file, *.PRO (Table 7). The prompting file, called "submarine.PRO" in this "submarine" case, contains the GENOPT-user-established variable names, one-line definitions, and "help" paragraphs. It is these items that make the GENOPT-user-created system "user-friendly", because the end user will depend on them for his or her input to the GENOPT-created processor, BEGIN.

GENTEXT also produces FORTRAN fragments, submarine.*, analogous to those listed on page 1 of Table 4 in [4] and described on pages 2 and 3 of Table 6 in [4]. GENTEXT automatically assembles these FORTRAN fragments into various programs (BEGIN.NEW, STOGET.NEW, STRUCT.NEW, BEHAVIOR.NEW, CHANGE.NEW) described on page 2 of Table 6 in [4]. BEGIN.NEW, STOGET.NEW, and CHANGE.NEW are complete programs and subroutines, created automatically entirely by GENOPT. The GENOPT user does not have to be concerned about them at all.

It is a different matter in the case of STRUCT.NEW and BEHAVIOR.NEW. These are "skeletal" subroutine libraries either or both of which must be "fleshed out" by the GENOPT user. In this particular application the GENOPT user adds merely three statements, CALL OPNGEN, CALL RWDGEN, and CALL CLSGEN (open, rewind, and close BIGBOSOR4 files) to the version of STRUCT.NEW automatically created by GENOPT. (See p. 2 of Tables 4 and Table 8). In this particular application the GENOPT user does more to "flesh out" the BEHAVIOR.NEW library. (See Table 9).

During the GENTEXT interactive session the GENOPT user here decided to introduce two behaviors: 1. overall buckling and 2. stress in two shell segments. Corresponding to these two behaviors, GENTEXT automatically created two skeletal "behavioral" subroutines, SUBROUTINE BEHX1 and SUBROUTINE BEHX2. The GENOPT user had to "flesh out" each of these two "behavioral" subroutines, as listed in Table 9.

The GENOPT user also had to "flesh out" the subroutine that computes the objective, SUBROUTINE OBJECT in Table 9. In this case the automatically produced skeletal version of SUBROUTINE OBJECT was modified only by the GENOPT user's adding the three lines:

```
     COMMON/TOTMAX/TOTMAS
     WEIGHT = TOTMAS
     OBJGEN = WEIGHT
```

just before the end of the subroutine.

Notice that in the "fleshed out" versions of SUBROUTINE BEHX1 and SUBROUTINE BEHX2 there are calls to SUBROUTINE BOSDEC. SUBROUTINE BOSDEC must be written by the GENOPT user. For the present application SUBROUTINE BOSDEC ("BOSerDECk") is listed in Table 10. SUBROUTINE BOSDEC creates a valid input file for BIGBOSOR4. For a general guideline on how to create SUBROUTINE BOSDEC, see the file, ...genopt/case/cylinder/howto.bosdec.

The best way for the GENOPT user to produce a valid BOSDEC routine is first to generate a valid input file for BIGBOSOR4 that represents an exact example of the geometry, etc. that GENOPT/BIGBOSOR4 will later attempt to optimize. Table 11, which is the same as Table 16 in [13], is a valid input file for BIGBOSOR4. This file contains the proper input data for BIGBOSOR4 for the specific case called "doer" in this report. SUBROUTINE BOSDEC, as listed in Table 10, produces BIGBOSOR4 input files in the form listed in Table 12, for example. Although the file listed in Table 12 looks quite different from that listed in a neat annotated form in Table 11, it contains the same input data in the same order as those listed in Table 11. Table 12 represents valid input for BIGBOSOR4.

**************** IMPORTANT WARNING ***************************
As a GENOPT user you will usually spend a considerable time creating "fleshed out" versions of BEHAVIOR.NEW and maybe also STRUCT.NEW. You must save these "fleshed out" versions with use of some other names. In this application the writer uses the names "behavior.submarine" and "struct.submarine". (also "bosdec.submarine"). You must save these important files for possible future use because execution of the GENOPT processor called GENTEXT destroys behavior.new and struct.new, replacing them with new "skeletal" versions of behavior.new and struct.new. During the development of this "submarine" capability the writer made it a habit always to work with the files, behavior.submarine and bosdec.submarine, then copy them as follows:

```
cp behavior.submarine behavior.new
cp bosdec.submarine bosdec.src
```

just before typing the command "genprograms". (Please read carefully the "digressions" in Table 4).
****

The present application is similar to the GENOPT cases called "weldland" [4] and "trusscomp" [5]. In the case, "trusscomp", more is done to "flesh out" the BEHAVIOR.NEW library than is done to "flesh out" the STRUCT.NEW library. In contrast, in the application described in [3] the BEHAVIOR.NEW library is not "fleshed out" at all, but instead a very long and elaborate "fleshed out" version of STRUCT.NEW is produced. In reading the very long paper [3], one should concentrate on the first part of [3], in which the role of the GENOPT user predominates.

OPTIMIZATION OF THE SPECIFIC CASE CALLED "doer"

Tables 13 - 26 and Figs. 1 - 17 pertain to this section.

The GENOPT user has completed his tasks and now the end user takes over and performs his tasks.

Input for BEGIN:

Table 13 lists the input for the "BEGIN" processor. The starting design is the design listed in Table 16 of [13].

Input for DECIDE:

Table 14 lists the input for the "DECIDE" processor. Note that there are three linking expressions and no inequality constraints. The linking expressions are explained in Table 14.

Analysis of the starting design:

Table 15 lists the input for the "MAINSETUP" processor for a case in which a fixed design is being analyzed (not optimization, that is, ITYPE=2, not ITYPE=1). The intention here is first to check the GENOPT user's "fleshed out" coding in SUBROUTINE BEHX1, SUBROUTINE BEHX2 and SUBROUTINE BOSDEC before doing any optimization.

Table 16 lists the ~~test~~ doer.OPM file corresponding to the starting design established in BEGIN. Figures 1 - 4, generated by executions of BIGBOSOR4 with use of the doer.BEHX1 and doer.BEHX2 files created by SUBROUTINE BEHX1 and SUBROUTINE BEHX2, respectively, show the geometry of the case (Fig. 1), the axisymmetrically deformed prebuckled state of the starting design (Fig.2), the critical buckling mode shape (Fig. 3), and the distribution of effective stress at the inner and outer fibers of the shell wall (Fig.4).

Input for MAINSETUP for optimization (NPRINT = 0, ITYPE = 1):

Table 17 lists the input for the "MAINSETUP" processor for a case in which optimization is desired, that is, ITYPE = 1 and NPRINT = 0. Some of the strategy input data is rather opaque. Therefore these data are explained in the following paragraphs:

IDESIGN = 2 is the preferred value for IDESIGN, and IMOVE = 1 is the preferred value for IMOVE. These input variables are described in the file,
.../genopt/execute/URPROMPT.DAT
as follows:

For IDESIGN:

```
-----------------------------------------------------------
 725.1 Choose 1 or 2 or 3 or 4 or 5 for IDESIGN
 725.2
       IDESIGN controls the quality of the best acceptable design,
       as follows:

       IDESIGN  accept only the best "---" design   minimum allowable
                                                     design margin
        1      "FEASIBLE"                               -0.01
        2      "FEASIBLE or ALMOST FEASIBLE"            -0.05
        3      "FEASIBLE or ALMOST FEASIBLE or
                MILDLY UNFEASIBLE"                      -0.10
        4      "FEASIBLE or ALMOST FEASIBLE or
                MILDLY UNFEASIBLE or MORE UNFEASIBLE"   -0.15
        5      "FEASIBLE or ALMOST FEASIBLE or
                MILDLY UNFEASIBLE or MORE UNFEASIBLE    -0.20
                or MOSTLY UNFEASIBLE"

       These choices are permitted because there are many cases
       for which design iterations "wallow" in a region of design
       space for which the design is in the range from "ALMOST
       FEASIBLE" to "MOSTLY UNFEASIBLE". The best "MOSTLY
       UNFEASIBLE" design may be a lot better (e.g. weigh much less)
       than the best "ALMOST FEASIBLE" design, and the GENOPT user
       may be willing to accept a few "MOSTLY UNFEASIBLE" margins,
       depending upon what particular behavior(s) are "MOSTLY
       UNFEASIBLE". For example, in the design of a shell structure
       for which the maximum stress is generated mostly from bending,
       the GENOPT user may feel that there is considerable residual
       strength in the shell even if its extreme fibers are stressed
       well beyond their elastic limit. Hence, if the behavioral
       constraint is violated because the maximum allowable elastic
       stress has been exceeded, this GENOPT user may feel that the
       optimized design will still be safe.
------------------------------------------------------------
```

For IMOVE:

```
------------------------------------------------------------
 730.0
       Next, choose a control for move limits to be used during
       optimization cycles. By "move limits" we are referring to
       the size of the boxes that appear in Fig. 2 of the paper,
       "GENOPT - a program that writes user-friendly optimization code",
       Int. J. Solids and Structures, Vol. 26, pp 1173- 1210, 1990.
       You are given five choices: IMOVE = 1 or 2 or 3 or 4 or 5:

       IMOVE = 1 means SMOVE = 0.10
       IMOVE = 2 means SMOVE = 0.50
       IMOVE = 3 means SMOVE = 0.01
       IMOVE = 4 means SMOVE = 0.02
       IMOVE = 5 means SMOVE = 0.05

       Small SMOVE (initial move limit) keeps the boxes small and
       leads to the requirement for many "OPTIMIZE" commands to
       obtain an optimum design; the "conservative" approach may
       be boring, but it may be the most reliable.  "Liberal" move
       limits allow bigger boxes, generally leading to the need for
       fewer "OPTIMIZEs".  However, the decision variables may
       jump around a lot and have difficulty converging to those
       corresponding to an optimum design.

       THE BEST CHOICE INITIALLY IS TO USE IMOVE = 1

       For early optimization cycles you can choose "liberal" move
       limits, changing to more "conservative" move limits after
       several "OPTIMIZEs".

       In practical problems (such as realistic design problems
       as opposed to mathematical "toy" problems) it is best to
       choose "conservative" move limits.

  740.1 Choose 1 or 2 or 3 or 4 or 5 for move limits, IMOVE
  740.2
       IMOVE = 1 means that decision variables will generally
                 change by less than 10 percent of their current
                 values in each optimization cycle (except for
                 occasional "jumps" that may occur on the initial
                 cycle corresponding to each "OPTIMIZE" command).
                 **** Ordinarily you should use this choice. ****

       IMOVE = 2 means that decision variables will generally
                 change by less than 50 percent of their current
                 values in each optimization cycle (except for
                 occasional "jumps" that may occur on the initial
                 cycle corresponding to each "OPTIMIZE" command).

       IMOVE = 3 means that decision variables will generally
                 change by less than 1.0 percent of their current
                 values in each optimization cycle (except for
                 occasional "jumps" that may occur on the initial
                 cycle corresponding to each "OPTIMIZE" command).
                 You may want to use this choice: 1. if you already
                 have a "global" optimum design from a SUPEROPT run,
                 and 2. you want to explore more in the immediate
                 neighborhood of the "global" optimum that you have
                 already determined from your previous SUPEROPT run.

       IMOVE = 4 means that decision variables will generally
                 change by less than 2.0 percent in each optimization
                 cycle. See "IMOVE = 3" for more.

       IMOVE = 5 means that decision variables will generally
                 change by less than 5.0 percent in each optimization
                 cycle. See "IMOVE = 3" for more. You may want to
                 use this option if the margins are "jumpy" from
                 optimization cycle to cycle.
-----------------------------------------------------------------
```

The same input data prompt file, ...genopt/execute/URPROMPT.DAT, has plenty to say about the next three MAINSETUP input data in Table 17:

For the response to the prompt in Table 20:

```
      y        $ Do you want default (RATIO=10) for initial move limit jump?
----------------------------------------------------------------------
 742.1 Do you want default (RATIO=10) for initial move limit jump?
 742.2
      In the first optimization cycle following each "OPTIMIZE"
      command the upper and lower bounds for each decision variable
      (x) for that cycle may be expanded ("jumped").  Whether or not
      this "move limit jump" occurs depends on the RATIO of the
      absolute values of the upper (xmax) and lower (xmin) bounds
      that were established by the user in "DECIDE" to the current
      value of the decision variable:

      If abs(xmax/x)/2**k > RATIO the current upper bound is expanded.
      If abs(xmin/x)/2**k > RATIO the current lower bound is expanded.

      in which k represents the number of times a "jump" has occurred
      in previous executions of "OPTIMIZE" since the last time "DECIDE"
      or "CHANGE" were used.  The default value of RATIO is 10.

      The purposes of the "move limit jump" are: (1) to enable decision
```

variables that are near zero to escape this neighborhood, and (2) to permit exploration of an expanded segment of the domain of the decision variable in the search for an optimum.

If you want to prevent the "jump" set RATIO very large.

743.1 Provide a value for the "move limit jump" ratio, RATIO
743.2

If zero is included in the domain of any decision variable it may be best to use the default value, RATIO = 10.

If any of your decision values has lower and upper bounds that span many orders of magnitude, it may be best to set RATIO to a large number.

If in doubt, use the default value.

In response to the prompt in Table 17:

```
      Y          $ Do you want the default perturbation (dx/x = 0.05)?
```

745.1 Do you want the default perturbation (dx/x = 0.05)?
745.2

See Fig. 1 and associated discussion on p. 1179 of the paper "GENOPT - a program that writes user-friendly optimization code", Int. J. of Solids and Structures, Vol. 26, pp 1173- 1210, 1990. In order to get gradients of the behavioral constraints the decision variables for the current design are perturbed one at a time and the behavior is calculated for each perturbation. The default perturbation is five per cent of the value of each decision variable, x(i), i = 1, 2, 3... NDV.

Usually you will answer Y. However, if there is difficulty obtaining convergence to an optimum, or if the constraint conditions jump around a lot from design iteration to design iteration, then you might want to try a smaller perturbation, such as 0.01 or 0.005. Do not use a perturbation larger than the default value of 0.05.

747.1 Amount by which decision variables are perturbed, dx/x
747.2

Try 0.01 or 0.005.

In response to the prompt in Table 17:

```
      n          $ Do you want to have dx/x modified by GENOPT?
```

748.1 Do you want to have dx/x modified by GENOPT?
748.2

For ordinary structures problems you should probably answer N . If you answer Y GENOPT will modify the size of the perturbation, dx/x, by a factor that depends on the history of the evolution of the design during optimization cycles: the perturbation will be increased by the ratio XAVE(IDV)/X(IDV), in which XAVE(IDV) is the average value of the IDVth decision variable over the last several design cycles and X(IDV) is the current value of that decision variable. If XAVE(IDV)/X(IDV) is less than 1.0, then the perturbation dx/x is not modified.

Please do not be overly concerned with the detailed explanations just listed for your convenience only. If you simply use the values given in Table 17 for cases similar to "~~tes~~doert" you will probably be ok.

Optimization:

Figure 5 shows the objective versus design iterations. Each major "spike" in the plot corresponds to a new "starting" design, obtained randomly but consistent with upper and lower bounds, equality constraints, and now inequality constraints. (See Item no. 29 in the file,...genopt/doc/genopt.news). Each new "starting" design is obtained with the processor, AUTOCHANGE. By means of the automated repeating pattern of a new "starting" design generated by AUTOCHANGE followed by several sequential executions of OPTIMIZE, it is hoped that a "global" optimum design will be determined, since each new "starting" design represents a different point in design space. A typical repeating pattern taken from the doer.OPP file generated by SUPEROPT is as follows:

```
-------------------------------------------------------------
 ==============================================================
 ITERATION                                        NUMBER OF
  NUMBER      OBJECTIVE     THE DESIGN IS...   CRITICAL MARGINS
            ------------------------------------------------------------
 -----------------------------------------------------OPTIMIZE
     1          1.5458E+03     ALMOST FEASIBLE          1
     2          1.4199E+03     MORE UNFEASIBLE          2
     3          1.3607E+03     MORE UNFEASIBLE          3
     4          1.3508E+03     MOSTLY UNFEASIB          3
     5          1.3918E+03      NOT FEASIBLE            2
     6          1.3655E+03     MORE UNFEASIBLE          3
 -----------------------------------------------------OPTIMIZE
     7          1.3655E+03     MORE UNFEASIBLE          3
     8          1.3389E+03      NOT FEASIBLE            3
     9          1.3339E+03      NOT FEASIBLE            3
    10          1.2939E+03      NOT FEASIBLE            3
    11          1.3325E+03      NOT FEASIBLE            3
    12          1.3728E+03      NOT FEASIBLE            2
 -----------------------------------------------------OPTIMIZE
    13          1.3728E+03      NOT FEASIBLE            2
    14          1.3693E+03      NOT FEASIBLE            3
    15          1.4196E+03     MORE UNFEASIBLE          2
    16          1.4329E+03      NOT FEASIBLE            2
    17          1.3873E+03     MORE UNFEASIBLE          2
    18          1.4206E+03     MORE UNFEASIBLE          2
 -----------------------------------------------------OPTIMIZE
    19          1.4206E+03     MORE UNFEASIBLE          2
    20          1.3519E+03      NOT FEASIBLE            3
    21          1.3187E+03      NOT FEASIBLE            3
    22          1.2927E+03      NOT FEASIBLE            3
    23          1.3442E+03     MOSTLY UNFEASIB          3
    24          1.3735E+03     MORE UNFEASIBLE          2
 -----------------------------------------------------OPTIMIZE
    25          1.3735E+03     MORE UNFEASIBLE          2
    26          1.3450E+03      NOT FEASIBLE            3
    27          1.3470E+03     MOSTLY UNFEASIB          3
    28          1.3387E+03      NOT FEASIBLE            3
    29          1.3793E+03     MOSTLY UNFEASIB          2
    30          1.3567E+03     MOSTLY UNFEASIB          3
 ---------------------------------------------------AUTOCHANGE
 -----------------------------------------------------OPTIMIZE
    31          1.5658E+03      NOT FEASIBLE            2
    32          1.5921E+03      NOT FEASIBLE            2
    33          1.6214E+03     MOSTLY UNFEASIB          2
    34          1.6361E+03     MORE UNFEASIBLE          2
    35          1.6061E+03     ALMOST FEASIBLE          2
    36          1.5773E+03     ALMOST FEASIBLE          2
 -----------------------------------------------------OPTIMIZE
    37          1.5773E+03     ALMOST FEASIBLE          2
    38          1.4921E+03      NOT FEASIBLE            2
    39          1.5824E+03     ALMOST FEASIBLE          2
    40          1.5044E+03     MORE UNFEASIBLE          2
    41          1.5126E+03     ALMOST FEASIBLE          2
    42          1.4745E+03     ALMOST FEASIBLE          2
 -----------------------------------------------------OPTIMIZE
    43          1.4745E+03     ALMOST FEASIBLE          2
    44          1.4255E+03      NOT FEASIBLE            2
    45          1.5104E+03     MILDLY UNFEASIB          2
    46          1.4638E+03     MILDLY UNFEASIB          2
    47          1.4255E+03     MILDLY UNFEASIB          2
    48          1.4593E+03     MORE UNFEASIBLE          2
 -----------------------------------------------------OPTIMIZE
    49          1.4593E+03     MORE UNFEASIBLE          2
    50          1.3787E+03      NOT FEASIBLE            3
    51          1.4598E+03     ALMOST FEASIBLE          2
    52          1.3898E+03     MORE UNFEASIBLE          3
    53          1.4423E+03     MILDLY UNFEASIB          2
    54          1.4228E+03     MILDLY UNFEASIB          2
 -----------------------------------------------------OPTIMIZE
    55          1.4228E+03     MILDLY UNFEASIB          2
    56          1.3608E+03     MOSTLY UNFEASIB          3
    57          1.3850E+03      NOT FEASIBLE            3
    58          1.4027E+03     MOSTLY UNFEASIB          2
    59          1.3678E+03     MOSTLY UNFEASIB          3
    60          1.3950E+03     MOSTLY UNFEASIB          2
 -------------------------------------------------------AUTOCHANGE
 ---------------------------------------------------------OPTIMIZE

        etc.
```

Analysis of the optimized design obtained from SUPEROPT:

Table 18 lists the ~~test~~ doer.OPM file for the optimized design. This optimum design is preserved by execution of the processor, "CHANGE", which generates an input file for CHANGE called "~~test~~ doer.CHG" (Table 19). The user is urged always to use CHANGE to save optimum designs. By doing so, the user can easily resurrect previously obtained optimum designs in the future by executing CHANGE and using the file, ~~test~~ doer.CHG, as input rather than having tediously to provide the previously obtained optimum design in an interactive mode.

Searching for a better optimum design than that in Table 18:

We felt that the maximum effective stress in Shell Segment 1 (125968.0 psi) and in Shell Segment 2 (123558.8 psi), listed on page 2 of Table 18, are a bit too high. Perhaps with some "refined" optimization we might obtain a better optimum design. This we did by launching another optimization run, this time using the processor called OPTIMIZE rather than the processor called SUPEROPT.

Table 20 lists the input file for MAINSETUP. Notice that we use 15 design iterations instead of the 5 design iterations specified in Table 17. Also, we choose IDESIGN = 1 and IMOVE = 3. These values of IDESIGN and IMOVE are more restrictive than those listed in Table 17 because we now want to explore the region of design space in the immediate neighborhood of the optimum design listed in Table 18. Please see the text above for the exact meanings of IDESIGN = 1 and IMOVE = 3.

The "OPTIMIZE" optimization run yields the results plotted in Figures 6 - 8 and listed in Table 22. Pages 10 - 13 of Table 4 explain how the new "refined" optimum design was obtained. Although this "refined" design weighs more than the optimum design obtained from SUPEROPT and listed in Table 18 (the "refined" design weighs 1403 lb versus 1369 lb listed at the bottom of page 2 of Table 18), we feel that it is a more acceptable design because the maximum effective stress is now 122083 psi compared to 125968 for the optimum design listed in Table 18.

The new "refined" optimum design is preserved by execution of CHANGE. The input data for CHANGE are listed in Table 23. The reader is urged always to use CHANGE in order to preserve optimum designs for possible resurrection in the future.

Using the "x,y" plotting capability called "plotps":

Figures 9 and 10 show how the reference surface location and shell wall thickness vary in the wide neighborhood of the junction between the hemispherical end and the cylindrical shell. These plots are obtained via commands such as:

```
/home/progs/bin/plotps.linux -ss < doer.wall.input > doer.wall.ps
```

In the above command line, /home/progs/bin is the location of the file, plotps.linux, which generates "x,y" plots for LINUX operating systems. The "plotps" tool was created by the writer's son, Bill Bushnell, in the 1990s. The option "-ss" in the command line denotes "same scale" (x-axis and y-axis have the same scale). The file, doer.wall.input, contains the input data for plotps, and the file, doer.wall.ps, is a postscript file that generates the plot shown in Fig. 9.

Obtaining plots from BIGBOSOR4 for the "refined" optimum:

Next, we wish to use BIGBOSOR4 to obtain plots of some of the buckling modes of the "refined" optimized design. As indicated in Table 4, this is done with use of the file, doer.BEHX1. The file, doer.BEHX1, is generated in SUBROUTINE BEHX1 from the following FORTRAN code in SUBROUTINE BEHX1:

```
---------------------------------------------------------------
      IF (ITYPEX.EQ.2) THEN
C      Get CASE.BEHX1 file for input for BIGBOSOR4...
C      CASE.BEHX1 is an input file for BIGBOSOR4 for behavior no. 1:
C      buckling load
         I=INDEX(CASE,' ')
         IF(I.NE.0) THEN
            CASA=CASE(:I-1)//'.BEHX1'
         ELSE
            CASA=CASE//'.BEHX1'
         ENDIF
         OPEN(UNIT=61,FILE=CASA,STATUS='UNKNOWN')
         CALL BOSDEC(1,61,ILOADX,INDIC)
         CLOSE(UNIT=61)
         WRITE(IFILE,'(/,/,A,A,/,A)')
     1 ' BIGBOSOR4 input file for:',
     1 ' buckling load (INDIC=1)',
     1   CASA
      ENDIF
---------------------------------------------------------------
```

doer There is analogous coding in SUBROUTINE BEHX2 to generate the file, ~~tes~~t.BEHX2 for the stress model. These blocks of coding remain almost the same for any case that involves BIGBOSOR4 and therefore a SUBROUTINE BOSDEC. The only differences are the comment line and the line that contains the string, ' buckling load (INDIC=1)'. The GENOPT user only has to change these lines appropriately to reflect the particular "behavior" that the SUBROUTINE BEHXi, i = 1, 2, 3,.., computes.

Figure 12 shows the critical buckling mode predicted by BIGBOSOR4, and Fig. 13 shows the buckling mode corresponding to the lowest eigenvalue for which the hemispherical shell participates significantly. An explanation of why we wish to obtain the information in Fig. 13 is provided on pages 13 and 14 of Table 4.

Figures 11 and 14 - 16 shows results from the BIGBOSOR4 axisymmetric stress analysis conducted with use of the valid input file for BIGBOSOR4, doer.BEHX2.

Optimum design obtained with the use of 10 callout points in each of shell segment 1 and shell segment 2:

Tables 24 - 26 and Fig. 17 pertain to this section. Table 24 is analogous to Table 13, and Table 25 is analogous to Table 14. Figure 17 is analogous to Fig. 5. The optimized design is presented in Table 26. This design does not appear to be as good as the optimum design obtained with the use of only five callout points in each of shell segments 1 and 2 because of the "oscillatory" distribution of shell wall thickness listed on page 2 of Table 26. The decision variables 31 - 35 [cylindrical shell wall thicknesses, THKCYL(1) - THKCYL(5)] have a "zig-zag" distribution which is questionable. Note from the entries in Table 24 that the axial locations of the callout points, ZCYL(1) through ZCYL(7), are spaced at one-inch intervals. The change in shell wall thickness from axial callout to callout is of the same order as the callout spacing. This abruptly changing wall thickness in such a thick shell wall represents a geometry that requires the use of solid finite elements rather than elements based on thin shell theory, which is based on the assumption that variations of thickness along the generator are much more gradual. The actual maximum stress in such a jagged shell wall is doubtless much higher than that predicted from thin shell theory because there probably exist stress concentrations where THKCYL(i) is small. The steep "valleys" in thickness are like circumferentially running notches in the shell wall. In order to avoid obtaining thickness distributions of this type during optimization cycles one would have to introduce inequality constraints that force each thickness, THKDYL(i), to be greater (or less) than its neighbors, THKCYL(i-1) and THKCYL(i+1) by not more than a certain relatively small percentage.

CONCLUSION

Engineers and researchers at the DOER Company and elsewhere are urged to use the "submarine" capability developed here for the generation of optimum designs of other pressure hull configurations for deep-diving vehicles or for any application in which the structure to be optimized is a shell of revolution. Enough detail is given here and in reports on other GENOPT/BIGBOSOR4 applications [1 - 5] to permit development by analogy of optimum designs of other systems.

Table 1

REFERENCES

[1] Bushnell, David, "GENOPT--A program that writes user-friendly optimization code", International Journal of Solids and Structures, Vol. 26, No. 9/10, pp. 1173-1210, 1990. The same paper is contained in a bound volume of papers from the International Journal of Solids and Structures published in memory of Professor Charles D. Babcock, formerly with the California Institute of Technology.

[2] Bushnell, David, "Automated optimum design of shells of revolution with application to ring-stiffened cylindrical shells with wavy walls", AIAA paper 2000-1663, 41st AIAA Structures Meeting, Atlanta, GA, April 2000. Also see Lockheed Martin report, same title, LMMS P525674, November 1999

[3] Bushnell, David, "Minimum weight design of imperfect isogrid-stiffened ellipsoidal shells under uniform external pressure", AIAA paper 2009-2702, 50th AIAA Structures Meeting, Palm Springs, CA, May 4-7, 2009

[4] Bushnell, David, "Use of GENOPT and a BIGBOSOR4 "huge torus" model to optimize a typical weld land and weld land edge stringers in a previously optimized internally stiffened cylindrical shell without weld lands, unpublished report sent to NASA Langley Research Center, May 15, 2009

[5] Bushnell, David, "Use of GENOPT and BIGBOSOR4 to obtain optimum designs of a cylindrical shell with a composite truss-core sandwich wall, unpublished report sent to NASA Langley Research Center, June 20, 2009

[6] Vanderplaats, G. N., "ADS--a FORTRAN program for automated design synthesis, Version 2.01", Engineering Design Optimization, Inc, Santa Barbara, CA, January, 1987

[7] Vanderplaats, G. N. and Sugimoto, H., "A general-purpose optimization program for engineering design", Computers and Structures, Vol. 24, pp 13-21, 1986

[8] Bushnell, David, "Stress, stability and vibration of complex, branched shells of revolution", Computers & Structures, Vol. 4, pp 399-435 (1974)

[9] Almroth, B. O., Brogan, F. A., "The STAGS Computer Code", NASA CR-2950, NASA Langley Research center, Hampton, VA (1979)

[10] Rankin, C. C., Stehlin, P., and Brogan, F. A., "Enhancements to the STAGS computer code", NASA CR-4000, NASA LRC, November 1986

[11] Riks, E., Rankin, C. C., and Brogan, F. A., "On the solution of mode jumping phenomena in thin walled shell structures", First ASCE/ASM/SES Mechanics Conference, Charlottesville, VA, June 6-9, 1993; in: Computer Methods in Applied Mechanics and Engineering, Vol. 136, 1996

[12] Bushnell, David, "Recent enhancements to PANDA2", AIAA Paper 96-1337-CP, Proceedings AIAA 37th Structures Meeting,, pp 126-182, April, 1996

[13] Bushnell, David, "The use of BIGBOSOR4 to obtain predictions of stress and buckling of deep submergence shells", unpublished report to the DOER Company, Alameda, California, June 23, 2009

Table 2 (2 pages)

This table is taken from Ref. [4].

```
THE FOLLOWING LIST IS PART OF THE *.DEF FILE PRODUCED BY
THE GENOPT PROCESSOR CALLED "GENTEXT"
==============================================================
C YOU ARE USING WHAT I HAVE CALLED "GENOPT" TO GENERATE AN
C OPTIMIZATION PROGRAM FOR A PARTICULAR CLASS OF PROBLEMS.
C THE NAME YOU HAVE CHOSEN FOR THIS CLASS OF PROBLEMS IS: weldland

C "GENOPT" (GENeral OPTimization) was written during 1987-1988
C by Dr. David Bushnell, Dept. 93-30, Bldg. 251, (415)424-3237
C    Lockheed Missiles and Space Co., 3251 Hanover St.,
C    Palo Alto, California, USA  94304

C The optimizer used in GENOPT is called ADS, and was
C written by G. Vanderplaats [3]. It is based on the method
C of feasible directions [4].

C                           ABSTRACT

C "GENOPT" has the following purposes and properties:
C      1. Any relatively simple analysis is "automatically"
C         converted into an optimization of whatever system
C         can be analyzed with fixed properties. Please note
C         that GENOPT is not intended to be used for problems
C         that require elaborate data-base management systems
C         or large numbers of degrees of freedom.

C      2. The optimization problems need not be in fields nor
C         jargon familiar to me, the developer of GENOPT.
C         Although all of the example cases (See the cases
C         in the directories under genopt/case)
C         are in the field of structural analysis, GENOPT is
C         not limited to that field.


C      3. GENOPT is a program that writes other programs. These
C         programs, WHEN AUGMENTED BY USER-SUPPLIED CODING,
C         form a program system that should be user-friendly in
C         the GENOPT-user"s field. In this instance the user
C         of GENOPT must later supply FORTRAN coding that
C         calculates behavior in the problem class called "weldland".

C      4. Input data and textual material are elicited from
C         the user of GENOPT in a general enough way so that
C         he or she may employ whatever data, definitions, and
C         "help" paragraphs will make subsequent use of the
C         program system thus generated easy by those less
C         familiar with the class of problems "weldland" than
C         the GENOPT user.

C      5. The program system generated by GENOPT has the same
C         general architecture as previous programs written for
C         specific applications by the developer [7 - 16]. That
C         is, the command set is:

C           BEGIN      (User supplies starting design, loads,
C                       control integers, material properties,
C                       etc. in an interactive-help mode.)

C           DECIDE     (User chooses decision and linked
C                       variables and inequality constraints
C                       that are not based on behavior.)

C           MAINSETUP  (User chooses output option, whether
C                       to perform analysis of a fixed design
C                       or to optimize, and number of design
C                       iterations.)

C           OPTIMIZE   (The program system performs, in a batch
C                       mode, the work specified in MAINSETUP.)

C           SUPEROPT   (Program tries to find the GLOBAL optimum
C                       design as described in Ref.[11] listed
C                       below (Many OPTIMIZEs in one run.)

C           CHANGE     (User changes certain parameters)
```

Table 2 (p. 2 of 2)

```
C           CHOOSEPLOT (User selects which quantities to plot
C                       vs. design iterations.)

C           DIPLOT     (User generates plots)

C           CLEANSPEC (User cleans out unwanted files.)

C       A typical runstream is:
C         GENOPTLOG    (activate command set)
C         BEGIN        (provide starting design, loads, etc.)
C         DECIDE       (choose decision variables and bounds)
C         MAINSETUP    (choose print option and analysis type)
C         OPTIMIZE     (launch batch run for n design iterations)
C         OPTIMIZE     (launch batch run for n design iterations)
C         OPTIMIZE     (launch batch run for n design iterations)
C         OPTIMIZE     (launch batch run for n design iterations)
C         OPTIMIZE     (launch batch run for n design iterations)
C         CHANGE       (change some variables for new starting pt)
C         OPTIMIZE     (launch batch run for n design iterations)
C         OPTIMIZE     (launch batch run for n design iterations)
C         OPTIMIZE     (launch batch run for n design iterations)
C         OPTIMIZE     (launch batch run for n design iterations)
C         OPTIMIZE     (launch batch run for n design iterations)
C         CHOOSEPLOT   (choose which variables to plot)
C         DIPLOT       (plot variables v. iterations)
C         CHOOSEPLOT   (choose additional variables to plot)
C         DIPLOT       (plot more variables v design iterations)
C         CLEANSPEC    (delete extraneous files for specific case)
```

It's better now to use SUPEROPT to do the optimization.

```
C   IMPORTANT:  YOU MUST ALWAYS GIVE THE COMMAND "OPTIMIZE"
C               SEVERAL TIMES IN SUCCESSION IN ORDER TO OBTAIN
C               CONVERGENCE! AN EXPLANATION OF WHY YOU MUST DO
C               THIS IS GIVEN ON P 580-582 OF THE PAPER "PANDA2,
C               PROGRAM FOR MINIMUM WEIGHT DESIGN OF STIFFENED,
C               COMPOSITE LOCALLY BUCKLED PANELS", Computers and
C               Structures, Vol. 25, No. 4, pp 469-605 (1987).
```

Use SUPEROPT now.

```
C Due to introduction of a "global" optimizer, SUPEROPT,
C described in Ref.[11], you can now use the runstream

C       BEGIN        (provide starting design, loads, etc.)
C       DECIDE       (choose decision variables and bounds)
C       MAINSETUP    (choose print option and analysis type)
C       SUPEROPT     (launch batch run for "global" optimization)
C       CHOOSEPLOT   (choose which variables to plot)
C       DIPLOT       (plot variables v. iterations)

C "Global" is in quotes because SUPEROPT does its best to find
C a true global optimum design. The user is strongly urged to
C execute SUPEROPT/CHOOSEPLOT several times in succession in
C order to determine an optimum that is essentially just as
C good as the theoretical true global optimum. Each execution
C of the series,
C       SUPEROPT
C       CHOOSEPLOT

C does the following:

C 1. SUPEROPT executes many sets of the two processors,
C    OPTIMIZE and AUTOCHANGE (AUTOCHANGE gets a new random
C    "starting" design), in which each set does the following:

C       OPTIMIZE           (perform k design iterations)
C       OPTIMIZE           (perform k design iterations)
C       OPTIMIZE           (perform k design iterations)
C       OPTIMIZE           (perform k design iterations)
C       OPTIMIZE           (perform k design iterations)
C       AUTOCHANGE         (get new starting design randomly)

C    SUPEROPT keeps repeating the above sequence until the
C    total number of design iterations reaches about 270.
C    The number of OPTIMIZEs per AUTOCHANGE is user-provided.

C 2. CHOOSEPLOT allows the user to plot stuff and resets the
C    total number of design iterations from SUPEROPT to zero.
C    After each execution of SUPEROPT the user MUST execute
C    CHOOSEPLOT: before the next execution of SUPEROPT the
C    total number of design iterations MUST be reset to zero.
```

Table 3 (6 pages)

FROM THE FILE ...genopt/doc/getting.started

This Table is taken from Ref. [4].

```
******************** GETTING STARTED ********************

 ...genopt/doc/getting.started

 Getting started with GENOPT using BIGBOSOR4

 ********************** NOTE *************************
 In the following the string, "/home/progs", frequently
 occurs. This is the PARENT directory of BOSOR4, BIGBOSOR4,
 BOSOR5, PANDA2, and GENOPT on the writer's computer. You
 must replace the string, "/home/progs", with whatever is
 the PARENT directory of BOSOR4, BIGBOSOR4, BOSOR5, PANDA2,
 and GENOPT at your facility.
 ******************** END NOTE ***********************

 0. Read the following:

 [0] Introduction to the file,

 /home/progs/genopt/case/cylinder/behavior.cylinder .

 Also read the files:

  ...genopt/case/cylinder/howto.bosdec
  ...genopt/case/cylinder/howto.struct
  ...genopt/case/cylinder/howto.behavior
  ...genopt/case/torisph/howto.stags.pdf
  ...genopt/case/torisph/readme.equivellipse
  ...genopt/case/wavycyl/readme.wavycyl

 [1] Bushnell, D., "GENOPT--A program that writes
 user-friendly optimization code", International
 Journal of Solids and Structures, Vol.26, No. 9/10,
 pp. 1173-1210, 1990. Also appeared in a
 bound volume of papers from the International Journal of
 Solids and Structures published in the memory of Professor
 Charles D. Babcock, formerly with the California Institute
 of Technology.

  (lines skipped to save space)
 ------------------------------------------------------------------------
```

← original GENOPT reference

```
********* FOUR STEPS FOR HOW TO SET UP AND RUN GENOPT ***********************

****** STEP 1 ******
1. Set up a directory, /home/progs/genoptcase
"genoptcase" is where you will do all your setting up of a generic case
and running of one or more specific cases. When you have run a case successfully
you should save the following files in the directory "genoptcase" (The following
list pertains to a case with generic name "cylinder" and specific name "cyl",
but this instruction pertains to files with any user-specified names):

cylinder.INP       = contains input data for GENTEXT, which sets up the
                     generic case.

behavior.new         (save it in a file called, for example, "behavior.cylinder".
                      If you are overwriting a "saved" file, make sure that your
                      latest version of "behavior.new" is valid. You may have
                      expended a lot of effort creating "behavior.new" and you
                      don't want to lose it!)

behavior.cylinder = contains the "BEHX1", "BEHX2", "BEHX3",..."BEHXn",
                    "USRCON", "USRLNK", "OBJECT", which are the "behavior"
                    subroutines, user-written constraint and linking routines,
                    and subroutine for computation of the objective function.
                    Save behavior.new by: cp behavior.new behavior.cylinder.
                    (Make sure behavior.new is correct!)

struct.new          (save it in a file called, for example, "struct.cylinder".
                     If you are overwriting a "saved" file, make sure that your
                     latest version of "struct.new" is valid. You may have
                     expended a lot of effort creating "struct.new" and you
                     don't want to lose it!)
```

In the weld land case the generic case name is "weldland" instead of "cylinder".

In this "submarine" case the generic case name is "submarine" instead of "cylinder"

Table 3 (p. 2 of 6)

```
struct.cylinder    = contains a combination of GENOPT-written code and user-
                     written code. Calls the "BEHXn" and "OBJECT" routines.
                     Save struct.new by: cp struct.new struct.cylinder.
                     (Make sure struct.new is correct!)

cyl.BEG = input data for the "begin" processor     (specific case)
cyl.DEC = input data for the "decide" processor    (specific case)
cyl.OPT = input data for the "mainsetup" processor (specific case)

*********************** END OF STEP 1 ***********************************
```

****** STEP 2 ******

2. Set up three directories,

```
/home/progs/bosdec
/home/progs/bosdec/sources
/home/progs/bosdec/objects.linux
```

"bosdec" should have two subdirectories: "objects.linux" and "sources"
"bosdec/sources" must contain the following source libraries:

```
addbosor4.src = BIGBOSOR4 source files (B4READ, B4MAIN, B4POST, etc.)
b4plot.src    = BIGBOSOR4 source files (plotting)
b4util.src    = BIGBOSOR4 source files (BIGBOSOR4 utilities)
bosdec.src    = generic case-dependent source file that creates a valid BOSOR4
                input file for the specific BOSOR4 case. bosdec.src must be
                written by the user for each new generic case. See Ref. [3].
opngen.src    = opens and closes files used in connection with BOSOR4
prompter.src  = BOSOR4 program for prompting input from the user for the specific
                case
resetup.src   = BOSOR4 source file for input for BOSOR4 restarts.
```

"bosdec/sources" should also contain the following files relating to "gasp", which is the block input/output subroutine used throughout BIGBOSOR4:

```
bio.c,   bio_linux.c,   bio_linux.o,   gasp.F,   gasp_linux.o
```

NOTE: The above five files pertain to the version of SUBROUTINE GASP for operation on LINUX workstations. For UNIX workstations, find the appropriate copies of gasp, etc. in the directory, .../genopt/case/sources/othergasps .
You may have to change the permissions on the directory "othergasps" for access.

You will find in the "othergasps" directory the following files:

```
-rw-r--r--   1 bush    bosor     22723 Jul 25  2003 bio.c
-rw-r--r--   1 bush    bosor     22820 Oct  1  1999 bio_alpha.c
-rw-r--r--   1 bush    bosor     41568 Mar  6  2000 bio_alpha.o
-rw-r--r--   1 bush    bosor     22693 Aug  3  2003 bio_hp700.c
-rw-r--r--   1 bush    bosor     11056 Nov  8  2005 bio_hp700.o
-rw-r--r--   1 bush    bosor     31175 Nov  2  2005 bio_linux.c
-rw-r--r--   1 bush    bosor     24596 Nov  2  2005 bio_sgi.o
-rw-r--r--   1 bush    bosor     23628 Nov  2  2005 bio_sgi8.c
-rw-r--r--   1 bush    bosor     24600 Nov  8  2005 bio_sgi8.o
-rw-r--r--   1 bush    bosor     23628 Nov  2  2005 bio_sgiold.c
-rw-r--r--   1 bush    bosor     22723 Jul 25  2003 bio_sol.c
-rw-r--r--   1 bush    bosor     27988 Jul 25  2003 bio_sol.o
-rw-r--r--   1 bush    bush      44856 Mar  6 07:05 gasp.hp700.a
-rw-r--r--   1 bush    bosor     26368 Mar  6  2000 gasp_alpha.o
-rw-r--r--   1 bush    bosor     14592 Jan 17 07:21 gasp_hp700.f
-rw-r--r--   1 bush    bosor     28044 Nov  8  2005 gasp_hp700.o
-rw-r--r--   1 bush    bosor     24760 Nov  2  2005 gasp_sgi.o
-rw-r--r--   1 bush    bosor     35504 Nov  8  2005 gasp_sgi8.o
-rw-r--r--   1 bush    bosor     17008 Jul 25  2003 gasp_sol.o
```

Copy whatever files are appropriate for your workstation into "bosdec/sources" instead of the "linux" versions if your workstation is running UNIX and not LINUX .

Properly initialized, your /home/progs/bosdec/sources directory must contain the following files (for LINUX workstation):

```
-rw-r--r--  1 bush bush 579671 Feb 29 07:19 addbosor4.src
-rw-r--r--  1 bush bush  83175 Feb 22 09:13 b4plot.src
-rw-r--r--  1 bush bush  89671 Feb 28 16:20 b4util.src
-rw-r--r--  1 bush bush  22723 Feb 10 14:27 bio.c
-rw-r--r--  1 bush bush  31175 Feb 10 14:27 bio_linux.c
```

Table 3 (p.3 of 6)

```
-rw-r--r--  1 bush bush  37152 Feb 10 14:27 bio_linux.o
                                           bosdec.src
-rw-r--r--  1 bush bush  15650 Feb 10 14:26 gasp.F
-rw-r--r--  1 bush bush  18364 Feb 10 14:26 gasp_linux.o
-rw-r--r--  1 bush bush   6310 Feb 13 10:12 opngen.src
-rw-r--r--  1 bush bush  22440 Feb 10 14:25 prompter.src
-rw-r--r--  1 bush bush  13426 Feb 22 09:14 resetup.src
```

These files can be found in the directory, /home/progs/genopt/case/sources .

Typically, you type a command,

```
cp /home/progs/genopt/case/sources/opngen.src /home/progs/bosdec/sources/.
```

in order to get the "opngen.src" file into the proper location.

or, more simply, type the following:

```
cp /home/progs/genopt/case/sources/* /home/progs/bosdec/sources/.
```

in order to copy all the "bosdec/sources" source files into the proper location.

```
********** NOTE **************
EVEN IF YOUR CASE DOES NOT INVOLVE bigbosor4 or bosor4 YOU MUST INCLUDE
THE bigbosor4 SOURCE FILES IN THE DIRECTORY /home/progs/bosdec/sources
BECAUSE THE COMMAND, genprograms, EMPLOYS THE "MAKE" FILE,
/home/progs/genopt/execute/usermake.linux
AND THIS "MAKE" FILE INCLUDES COMPILATION OF bigbosor4 routines EVEN IF
THESE bigbosor4 ROUTINES ARE NOT USED IN YOUR CASE.
******************************
```

In addition to the files listed above, you need a source file called "bosdec.src". If you want to run one of the sample cases contained in the /home/progs/genopt/case directory, which includes the following subdirectories:

```
drwxr-xr-x   2 bush bush   456 Nov  9  2005 cylinder    <--based on BOSOR4 or BIGBOSOR4
drwxr-xr-x   2 bush bush   272 Oct 16  2005 plate
drwxr-xr-x   2 bush bush  1456 Nov 19  2005 sphere
drwxr-xr-x   2 bush bush 10960 Nov  2  2006 torisph     <--based on BOSOR4 or BIGBOSOR4
drwxr-xr-x   2 bush bush   272 Oct  8  2005 wavycyl     <--based on BOSOR4 or BIGBOSOR4
```

you must copy one or more of the following files into the directory, /home/progs/bosdec/sources:

```
 bush  7246 Sep 20  2005 bosdec.cylinder (in the /home/progs/genopt/case/cylinder directory)
 bush 33098 Dec 19  2005 bosdec.ellipse      | (The three files, bosdec.ellipse,
 bush 33223 Jan 11  2006 bosdec.equivellipse |  bosdec.equivellipse, and bosdec.tori are in
 bush 33191 Dec 19  2005 bosdec.tori         |  the /home/progs/genopt/case/torisph directory)
 bush 75972 Sep 20  2005 bosdec.wavycyl  (in the /home/progs/genopt/case/wavycyl directory)
```

For example, if you want to run the "cylinder" case, you must type the command:

```
cp /home/progs/genopt/case/cylinder/bosdec.cylinder /home/progs/bosdec/sources/bosdec.src
```

```
*****************************************************************************************
```

NOTE: For a new case that involves using BIGBOSOR4 (or BOSOR4) the GENOPT user must generate a new bosdec.src file from scratch. This might seem to be a monumental task. To ease the burden, please read the file,

```
/home/progs/genopt/case/cylinder/howto.bosdec
```

for guidance in this important part of your effort.

Also, it will be necessary to augment the file, struct.new, which is produced by GENTEXT. For guidance with this task, please read the file,

```
/home/progs/genopt/case/cylinder/howto.struct
```

Also, it may well be necessary to augment the file, behavior.new, which is produced by GENTEXT. For guidance with this task, please read the file,

```
/home/progs/genopt/case/cylinder/howto.behavior
```

```
****************************************************************************
********************** END OF STEP 2 ************************************
```

Table 3 (p.4 of 6)

```
********* NOTE ********** NOTE *****************
YOU WON'T HAVE TO DO THE NEXT ITEM, STEP 3, BECAUSE THE INSTALLATION OF genopt
AT YOUR FACILITY ACCOMPLISHES THIS FOR YOU. ITEM 3 IS INCLUDED HERE FOR YOUR
INFORMATION ONLY.
***********************************************


****** STEP 3 ******
3. set up a directory, /home/progs/genopt, which contains the following subdirectories.
   (This will already have been done when you or someone else installed GENOPT
   at your facility.):

bin = contains files for executing genopt:
      autochange.com, begin.com change.com, chooseplot.com, cleangen.com, cleanspec.com,
      decide.com, diplot.com, genprograms.com, genprograms.bat, genprompt.com,
      gentext.com, helpg.com, insert.com, mainsetup.com, optimize.com, optimize.bat,
      superopt.com, superopt.bat

case = contains sample cases and BIGBOSOR4 source files:
       cylinder, plate, sphere, wavycyl, torisph, sources

doc  = contains documentation files:
       genopt.abs, genopt.news, genopt.story, howto.install, howto.update, getting.started

sources = contains the following files:
          addcode1.src, addcode2.src, addcode3.src, addcode4.src, addcode5.src,
          ads.src, begin.tmpl, change.tmpl, chauto.src, chplot.src, conman.src,
          decide.src, diplot.src, felippa.src, genprompt.src, helpg.src, ieeexx.c,
          ieeexx_linux.o, insert.src, main.src, mainsetup.src, prompter.src,
          prompter2.src, sig.f, sig_linux.o, stoget.tmpl, store.src, struct.tmpl,
          util.c, util.h, util.src, util_linux.o
          (NOTE: the *.tmpl files are skeletal files that are used by GENOPT,
          which generates corresponding *new files after execution of the
          interactive GENOPT processor,  GENTEXT.)

execute = contains the following executable files, prompt files, and "make" files:
          genprompt.linux, helpg.linux, insert.linux,
          GENOPT.HLP, PROMPT.DAT, PROMPT2.DAT, PROMPT3.DAT, PROMPT4.DAT, URPROMPT.DAT,
          makefile.linux, usermake.linux

libraries.linux = contains archive libraries for genopt processors called
                  genprompt, helpg, insert (*.a)

objects.linux   = contains object libraries for genopt libraries called
                  ads, chauto, chplot, conman, decide, felippa, genprompt,
                  helpg, insert, main, mainsetup, prompter, prompter2, store, util  (*.a)

*********************** END OF STEP 3 ***********************************


****** STEP 4 ******
4. In order to rerun a case already done previously (for example, the case "cylinder")
do the following:

 Go to the directory:

 /home/progs/bosdec/sources

 and type the command:

 cp /home/progs/genopt/case/cylinder/bosdec.cylinder bosdec.src

 if you haven't done this already.


 Go to the directory

 /home/progs/genoptcase .

 If you want to run the test case called "cylinder",
 copy the file, cylinder.INP, as follows:

 cp /home/progs/genopt/case/cylinder/cylinder.INP .
```

Table 3 (p.5 of 6)

Type the following:

```
genoptlog       (activate the GENOPT command set)
```

```
************** NOTE ****************** NOTE ****************** NOTE ***************
MAKE SURE ALWAYS TO SAVE COPIES OF struct.new AND behavior.new THAT YOU HAVE
PUT A LOT OF EFFORT INTO CREATING. THE struct.new AND behavior.new FILES ARE
DESTROYED BY EXECUTION OF "gentext", THE COMMAND YOU TYPE NEXT.
***********************************************************************************
```

```
gentext         (provide input for GENOPT, that is, for the generic case)

Enter generic case name: cylinder

 (give as the name for the generic case = "cylinder")

 ARE YOU CORRECTING, ADDING TO, OR USING cylinder.INP ? (TYPE y  OR  n):y

 (reply "y", for YES, you ARE using or correcting a previously established file;
  in this example the already-existing input file for GENOPT is called "cylinder.INP")

 (The use of the file, cylinder.INP, as input to GENTEXT leads, after execution of
  GENTEXT, to the following files:                          purpose of file
-----------------------------------------------------------------------------------
-rw-r--r--  1 bush bush  1850 Oct  8 15:36 cylinder.CHA  code fragment for "change"
-rw-r--r--  1 bush bush   557 Oct  8 15:36 cylinder.COM  labelled common blocks
-rw-r--r--  1 bush bush  5541 Oct  8 15:36 cylinder.CON  code fragments for constraints
-rw-r--r--  1 bush bush  8734 Oct  8 15:36 cylinder.DAT  a copy of cylinder.INP
-rw-r--r--  1 bush bush 20639 Oct  8 15:36 cylinder.DEF  information for user.
-rw-r--r--  1 bush bush 11160 Oct  8 15:36 cylinder.NEW  code fragment for "begin"
-rw-r--r--  1 bush bush  1343 Oct  8 15:36 cylinder.PRO  prompts for specific case.
-rw-r--r--  1 bush bush   733 Oct  8 15:36 cylinder.REA  read labelled common blocks.
-rw-r--r--  1 bush bush    48 Oct  8 15:36 cylinder.SET  code fragment for SETUPC
-rw-r--r--  1 bush bush 10349 Oct  8 15:36 cylinder.SUB  skeletal BEHX1, BEHX2, etc.
-rw-r--r--  1 bush bush   733 Oct  8 15:36 cylinder.WRI  write labelled common blocks

and

-rw-r--r--  1 bush bush 29778 Oct  8 15:36 begin.new
-rw-r--r--  1 bush bush 24785 Oct  8 15:36 behavior.new
-rw-r--r--  1 bush bush 13726 Oct  8 15:36 change.new
-rw-r--r--  1 bush bush  7234 Oct  8 15:36 stoget.new
-rw-r--r--  1 bush bush 14495 Oct  8 15:36 struct.new
```

The "cylinder.*" files, described in cylinder.DEF, contain fragments of FORTRAN code, definitions of variables, and prompts. For descriptions of the contents of these files, please see Table 5 in the file, cylinder.DEF. (Also Table 5 in the file ..genopt/case/torisph/equivellipse.DEF contains the same descriptions for a generic case called "equivellipse".)

The "*.new" files contain complete FORTRAN source for processors, "begin" and "change" and the subroutine stoget, and "skeletons" of subroutines behavior and struct. It is up to the user to "flesh out" the skeletons, "behavior" and "struct", that is, write FORTRAN code that leads to computation of the various behaviors and objective (buckling, stress, vibration, etc., and objective).

Also, the user must create a file, /home/progs/bosdec/sources/bosdec.src, if this has not already been done.

In the case called "cylinder" all this has been done. The complete FORTRAN coding is contained in the three files,

```
/home/progs/genopt/case/cylinder/bosdec.cylinder,
/home/progs/genopt/case/cylinder/behavior.cylinder
/home/progs/genopt/case/cylinder/struct.cylinder
```

In order to re-run this case, these three files must be copied to the correct locations. If we are already in the directory, /home/progs/genoptcase, we type the following:

```
cp /home/progs/genopt/case/cylinder/bosdec.cylinder /home/progs/bosdec/sources/bosdec.src
(establish the subroutine(s) that generate valid BIGBOSOR4 input files)

cp /home/progs/genopt/case/cylinder/behavior.cylinder  behavior.new
(establish source code for the behavior)
The computer will ask you, "overwrite 'behavior.new'?" and you answer, "y"
```

Table 3 (p. 6 of 6)

because you are overwriting the "skeletal" version of behavior.new with the completed version of behavior.new.

```
cp /home/progs/genopt/case/cylinder/struct.cylinder struct.new
(establish source code that calls the "behavior" subroutines and generates
 corresponding design margins)
```

The computer will ask you, "overwrite 'struct.new'?" and you answer, "y" because you are overwriting the "skeletal" version of struct.new with the completed version of struct.new.

Go to the /home/progs/genoptcase directory if you are not there already.

```
genprograms   (compile the GENOPT-written source code. The
               following processors are generated:)
```

Here is a list of all your newly created executables (provided "genprograms" doesn't bomb!):

```
-rwxr-xr-x  1 bush bush 71562 Oct  8 15:56 autochange.linux
-rwxr-xr-x  1 bush bush 139553 Oct  8 15:56 begin.linux
-rwxr-xr-x  1 bush bush 124383 Oct  8 15:56 change.linux
-rwxr-xr-x  1 bush bush 156054 Oct  8 15:56 chooseplot.linux
-rwxr-xr-x  1 bush bush 161231 Oct  8 15:56 decide.linux
-rwxr-xr-x  1 bush bush 104222 Oct  8 15:56 mainsetup.linux
-rwxr-xr-x  1 bush bush 1691559 Oct  8 15:56 optimize.linux
-rwxr-xr-x  1 bush bush 95653 Oct  8 15:56 store.linux
```

If you want to use input from the specific case, "cyl", type the commands (assuming you are now in the /home/progs/genoptcase directory):

```
cp /home/progs/genopt/case/cylinder/cyl.BEG cyl.BEG
cp /home/progs/genopt/case/cylinder/cyl.DEC cyl.DEC
cp /home/progs/genopt/case/cylinder/cyl.OPT cyl.OPT
```

Next, type the command BEGIN to input data for a new (specific) case.

(lines skipped in order to save space.
 See the file.../genopt/doc/getting.started)
==

Table 4 (15 pages) RUN STREAM

```
submarine.runstream

RUNSTREAM USED TO PRODUCE THE OPTIMUM DESIGN
OF THE DEEP SUBMERGENCE TANK

[In this case the GENOPT user (the writer) selected
 "submarine" for the GENERIC case name and "doer"
 for the SPECIFIC case name. The case is run in the
 directory, ../home/progs/genoptcase, in which
 "/home/progs" is the PARENT directory of GENOPT,
 BIGBOSOR4, PANDA2, etc. At your facility, replace
 the string, /home/progs, with whatever directory is
 the PARENT directory of GENOPT, BIGBOSOR4, etc. where
 you are using GENOPT.]

cd /home/progs/genoptcase

genoptlog        (activate GENOPT command set)

(The command, "genoptlog", produces the following screen:
-------------------------------------------------------------
GENOPT commands have been activated.

  gentext      GENOPT user generates a prompt file.
  genprograms  GENOPT user generates (makes) executables:
               begin, decide, mainsetup, optimize,
               change, chooseplot, and diplot.
  begin        End user provides starting data.
  decide       End user chooses decision variables, bounds,
               linked variables, and inequality constraints.
  mainsetup    End user sets up strategy parameters.
  optimize     End user performs optimization.
  change       End user changes some parameters.
  autochange   New values for decision variables randomly
  superopt     End user find global optimum (autochange/optimize)...
  chooseplot   End user chooses which variable to plot vs.
               iterations.
  diplot       End user plots variables vs. iterations.
  insert       GENOPT user adds parameters to the problem.
  cleangen     GENOPT user cleans up GENeric case files.
  cleanspec    End user cleans up SPECific case files.
---------------------------------------------------------------

gentext          [provide generic case name ("submarine"),
                  variable names, roles, one-line definitions,
                  "help" paragraphs, etc. The input data from
                  the long GENTEXT interactive session are saved
                  in the file, "submarine.INP" (Table 5). Also, see
                  the files produced by GENTEXT called "submarine.DEF"
                  (the first part of Table 9, Table 6) and
                   "submarine.PRO" (Table 7).]

************* A SMALL DIGRESSION FROM THE RUN STREAM ***************
 After execution of GENTEXT the following "submarine" files
 exist in the directory where GENTEXT was executed, that is,
 in the directory, /home/progs/genoptcase :]
------------------------------------------------------------------
-rw-r--r--  1 bush bush  1973 Jun 23 09:21 submarine.CHA
-rw-r--r--  1 bush bush   736 Jun 23 09:21 submarine.COM (common blocks)
-rw-r--r--  1 bush bush  2885 Jun 23 09:21 submarine.CON
-rw-r--r--  1 bush bush 29294 Jun 23 09:21 submarine.DAT
-rw-r--r--  1 bush bush 29611 Jun 23 09:21 submarine.DEF (documentation)
-rw-r--r--  1 bush bush 29300 Jun 23 09:21 submarine.INP (input for GENTEXT)
-rw-r--r--  1 bush bush 11757 Jun 23 09:21 submarine.NEW
-rw-r--r--  1 bush bush 11067 Jun 23 09:21 submarine.PRO (prompting file)
-rw-r--r--  1 bush bush   849 Jun 23 09:21 submarine.REA
-rw-r--r--  1 bush bush   556 Jun 23 09:21 submarine.SET
-rw-r--r--  1 bush bush  8789 Jun 23 09:21 submarine.SUB
-rw-r--r--  1 bush bush   849 Jun 23 09:21 submarine.WRI
----------------------------------------------------------------

 The contents of these files is described in the submarine.DEF file.
 See pages 2 and 3 of Table 6 of [4], for example.

 Next, create software that computes the design constraints
```

and the objective. In this example create bosdec.submarine (Table 10), which generates valid input files for the BIGBOSOR4 preprocessor, B4READ, and "flesh out" the skeletal behavior.new file and the struct.new file that are automatically created by GENOPT. That is, create behavior.submarine (Table 9) from behavior.new and struct.submarine (Table 8) from struct.new. The "fleshing out" of struct.new is very simple in this "submarine" application: only THREE lines are added to the version of struct.new created automatically by GENOPT. These three added lines are indicated below:

The following is part of the "fleshed out" version of the struct.new file:

```
-----------------------------------------------------------------------
C  USER: YOU MAY WANT TO INSERT SUBROUTINE CALLS FROM SOFTWARE DEVELOPED
C        ELSEWHERE FOR ANY CALCULATIONS PERTAINING TO THIS LOAD SET.
C
C
      CALL OPNGEN  <--added by the GENOPT user (the writer)
      CALL RWDGEN  <--added by the GENOPT user (the writer)
C
-----------------------------------------------------------------------
```

and

The following is part of the struct.new file:

```
-----------------------------------------------------------------------
C  NEXT, EVALUATE THE OBJECTIVE, OBJGEN:
      IF (ILOADX.EQ.1) THEN
         PHRASE ='weight/area of the truss-core sandwich wall'
         CALL BLANKX(PHRASE,IENDP4)
         CALL OBJECT(IFILE8,NPRINX,IMODX,OBJGEN,
     1    'weight/area of the truss-core sandwich wall')
      ENDIF
      NCONSX = ICONSX
C
      CALL CLSGEN  <--added by the GENOPT user (the writer)
C
      RETURN
      END
----------------------------------------------------------------------
```

The three added statements, CALL OPNGEN, CALL RWDGEN, and CALL CLSGEN, open, rewind, and close various files used by BIGBOSOR4. If you plan to optimize some other shell using GENOPT/BIGBOSOR4 you can "flesh out" struct.new in exactly the same way. To find the places in the "skeletal" version of struct.new that is automatically produced by GENTEXT, search for the string, "YOU MAY WANT" in order to find where you should insert the two lines, CALL OPNGEN and CALL RWDGEN. Search for the string, "NCONSX", in order to find where you should insert the line, CALL CLSGEN. If you want to see what struct.new looks like, see Table 8 . Sometimes, in your other applications of GENOPT, you may want to add more coding to the "skeletal" version of struct.new, as was done in [2] and [3], especially in [3], where all of the computations are done in struct.new and the "skeletal" version of behavior.new produced automatically by GENOPT remains unchanged.

Most of the work in this project was the creation by the GENOPT user (the writer of this report) of the file, bosdec.submarine (Table 10). Some effort was also required to "flesh out" the skeletal "behavior.new" file automatically created by GENOPT. The "fleshed out" version is called "behavior.submarine" (Table 9). There are two "behavior" subroutines: BEHX1 and BEHX2. The GENOPT user must also "flesh out" the subroutine that computes the objective, SUBROUTINE OBJECT. The "BEHXi", i = 1,2, subroutines compute the following:
SUBROUTINE BEHX1 computes the buckling load.
SUBROUTINE BEHX2 computes the effective stress in the shell wall
SUBROUTINE OBJECT computes the objective, which in this case
                  is the weight of the deep submersible tank.

*********************** NOTE ************************************
If you plan to use GENOPT in combination with BIGBOSOR4 for optimizing other shells of revolution which have buckling behavior, you can use the list of SUBROUTINE BEHX1 here as a guide. This is what the writer did in the present "submarine" case. The writer simply took the coding he had added to the "skeletal" version of SUBROUTINE BEHX1 listed in Table 10 of [4] (the "weldland" case) and inserted that coding where it says "INSERT SUBROUTINE STATEMENTS HERE"

in the "skeletal" BEHX1 subroutine in the "submarine" case. Then he edited that "weldland" coding appropriately in order to make it applicable to the buckling behavior in the "submarine" case.

SUBROUTINE BEHX2, which computes the maximum effective stress in Shell segment number JCOL, was "fleshed out" as listed on page 9 of Table 9. BIGBOSOR4 computes the maximum effective stress in each shell segment of a multi-segment model and stores it in the array, SKNMAX(i), i = 1, 2, ... number of segments in the BIGBOSOR4 model. SKNMAX is stored in the BIGBOSOR4 labelled common block called "STRCON". Therefore we need to include that labelled common block in SUBROUTINE BEHX2. Also, we need the other BIGBOSOR4 labelled common blocks listed in SUBROUTINE BEHX2 following the line that reads, "INSERT SUBROUTINE STATEMENTS HERE". which is where the GENOPT user is supposed to "flesh out" each of the skeletal BEHXi, i = 1,2,...,routines automatically produced by GENTEXT. The last statement in SUBROUTINE BEHX2:

```
     STRESS(ILOADX,JCOL) = SKNMAX(JCOL)
```

is where the relevant quantity obtained from BIGBOSOR4 is stored in the array, STRESS(ILOADX,JCOL), which is an array established by the GENOPT user during the GENTEXT interactive session.

```
 ********************* END NOTE ***********************************]
```

NOTE: make it a habit to develop the "behavior", "struct", and "bosdec" files using different suffices than ".new" or ".src". By this practice you will not lose work should you execute GENTEXT again after you have already added FORTRAN coding to behavior.new and to struct.new, which are over-written by GENTEXT. In this "submarine" case the writer INITIALLY copied behavior.new and struct.new (the "skeletal" versions created automatically by GENOPT after completion of the GENTEXT interactive session) to behavior.submarine and struct.submarine, then "fleshed out" the behavior.submarine and struct.submarine files. Also, the writer developed his version of "bosdec" in a file called "bosdec.submarine". Then, just before execution of "genprograms", the writer did the following:

```
--------------------------------------------------------------------------------
 cp behavior.submarine behavior.new  (behavior.submarine and struct.submarine are
 cp struct.submarine struct.new       developed in the directory: /home/progs/genoptcase)
 cd /home/progs/bosdec/sources       (NOTE: "bosdec" is stored in a different directory)
 cp bosdec.submarine bosdec.src
 cd /home/progs/genoptcase           (return to the "genoptcase" directory before
                                      executing "genprograms")
--------------------------------------------------------------------------------
************* END OF A SMALL DIGRESSION FROM THE RUN STREAM **********


genprograms     (compiles the software created by GENOPT and
                 "fleshed out" and added by the GENOPT user)


*********** ANOTHER SMALL DIGRESSION FROM THE RUN STREAM *************
 If compilation is successful, the following is listed on
 your computer screen:
-------------------------------------------------------------
Congratulations!  Your code compiled successfully.  You should
now check to make sure that you get correct results from a
simple test case with a known answer before attempting a more
complicated case.

Here is a list of all your newly created executables:
-rwxr-xr-x  1 bush bush 83839 Jun 12 11:17 autochange.linux
-rwxr-xr-x  1 bush bush 190917 Jun 12 11:17 begin.linux
-rwxr-xr-x  1 bush bush 135079 Jun 12 11:17 change.linux
-rwxr-xr-x  1 bush bush 157882 Jun 12 11:17 chooseplot.linux
-rwxr-xr-x  1 bush bush 160553 Jun 12 11:17 decide.linux
-rwxr-xr-x  1 bush bush 105947 Jun 12 11:17 mainsetup.linux
-rwxr-xr-x  1 bush bush 1603626 Jun 17 14:24 optimize.linux
-rwxr-xr-x  1 bush bush 124325 Jun 12 11:17 store.linux

Next, type the command BEGIN to input data for a new SPECIFIC case.
-------------------------------------------------------------
 NOTE: You may see the lines above even when you still have errors
       in your newly "fleshed out" and created FORTRAN coding. You
       will doubtless discover additional errors when you first execute
       "OPTIMIZE". Make your corrections to the behavior.submarine,
       struct.submarine, and bosdec.submarine files, then again copy
```

them to behavior.new, struct.new, and bosdec.src as specified above, and then give the command, "genprograms" again. Keep doing this until you are satisfied that there are no more errors in your FORTRAN coding. Go through this "error elimination loop" before you try to do any optimization. That is, specify ITYPE = 2 and NPRINT = 2 in the *.OPT file (input for MAINSETUP, listed below) while you are in this error elimination phase of your work. Only when you are satisfied that behavior.submarine, struct.submarine, and bosdec.submarine are correct should you attempt to do any optimization (ITYPE = 1 and NPRINT = 0 in the *.OPT file).

Concerning the use of the GENOPT processor called "INSERT":

It may well happen that, after you have already developed behavior.submarine, struct.submarine, and bosdec.submarine as just described, you may want to add one or more variables to your generic case. You can use "INSERT" to do this. However, note that if you add (or take away) any variables, the labelled common blocks produced automatically by GENTEXT will change. These new common blocks will be present in the new "skeletal" versions of behavior.new and struct.new generated automatically by your re-run of GENTEXT. Also, if you add one or more "behaviors", GENTEXT creates additional FORTRAN coding in the behavior.new and struct.new libraries.

Concerning "behavior" and "struct" after you use "insert" or otherwise change the *.INP file:

With regard to the "behavior" and the "struct" libraries, You now have a choice between the following item 1 or item 2:

1. You can add your "fleshed out" FORTRAN coding now contained only in behavior.submarine, struct.submarine, to the latest "skeletal" versions, behavior.new and struct.new, then type the commands:

```
cp behavior.new behavior.submarine  (Be careful! you may be destroying
cp struct.new struct.submarine       your previous work if you have not
                                     updated behavior.new and/or
                                     struct.new correctly.)
```

This is almost always the best choice, as explained next in item 2.

2. You can replace the old, GENTEXT-created, common blocks with the new common blocks located in the file, *.COM (e.g. "*" = "submarine") in your behavior.submarine and struct.submarine files. This is NOT generally the best choice because you may have added new "behaviors". In that case it is not just the common blocks that change but also the GENTEXT-created FORTRAN coding in behavior.new and struct.new. Also, you may have changed the wording in one or more of the one-line definitions of the variables. These changes in wording of the one-line definitions of the variables exist only in the new "skeletal" behavior.new and struct.new files. Therefore, it is almost always best to port your "fleshed out" FORTRAN coding from your behavior.submarine and struct.submarine files to the new "skeletal" behavior.new and struct.new files produced automatically by GENTEXT, then (only after you are certain that you have done everything correctly!) copy the new "fleshed out" behavior.new into behavior.submarine and the new "fleshed out" struct.new into struct.submarine.

Concerning correcting "bosdec" after changing submarine.INP:

In the case of the file, /home/progs/bosdec/sources/bosdec.submarine, You have to copy the new GENTEXT-created common blocks, located in submarine.COM, into the proper place in the bosdec.submarine file and then remove the old GENTEXT-created common blocks. Since bosdec.submarine contains only FORTRAN code produced by you, you don't have to worry about any new GENTEXT-created FORTRAN code there.

Concerning modification of only the "help" paragraphs in the file *.INP (submarine.INP):

After you run BEGIN you will probably come to the conclusion that the end user will need more "help" information than you have provided. You can do this by editing the *.INP (submarine.INP) file. Just be sure to follow the pattern that exists in the *.INP file. For example, for each new line of a "help" paragraph there exists a following line:

```
       y         $ Are there more lines in the "help" paragraph?
```

As you add new "help" lines make sure in your editing that you follow each new "help" line with the line printed above. The last line in the "help" input is always followed by the line:

```
       n         $ Are there more lines in the "help" paragraph?
```

You can, in the same way, add a new "help" paragraph where there was none previously. You would change the line:

```
       n         $ Do you want to include a "help" paragraph?
```

to

```
       y         $ Do you want to include a "help" paragraph?
```

and then proceed as is done elsewhere in the *.INP file for variables that have "help" paragraphs.

```
********************** END OF SMALL DIGRESSION ***********************
```

```
begin           {provide starting design, material, loading,
                 allowables and factors of safety for the
                 following "behaviors":
                  1. buckling (computed in SUBROUTINE BEHX1)
                  2. JCOLth stress constraint, in which JCOL
                     is the shell segment number
                     (computed in SUBROUTINE BEHX2)
                 As a starting design, use the design listed in
                 Table 16 of the document, "The user of BIGBOSOR4
                 to obtain predictions of stress and buckling of
                 deep submergence shells", D. Bushnell, June 23, 2009 [13].
                 BEGIN saves your interactive input in the file, doer.BEG
                 (Table 13). NOTE: The "end user" named the SPECIFIC
                 case "doer".}
```

(The command, BEGIN, starts an interactive session, the beginning of which presents the following to your computer screen:)

```
-----------------------------------------------------------------
 THE NAME OF THE PROMPT FILE ASKED FOR NEXT
 IS THE NAME OF THE CLASS OF PROBLEMS THAT THE GENOPT-USER
 HAS CHOSEN, NOT THE NAME OF THE PARTICULAR CASE BEING
 STUDIED HERE. IT IS THE "NAME" PART OF "NAME".PRO.

  ENTER THE GENERIC CASE NAME: submarine

 FROM HERE ON, WHENEVER THE CASE NAME IS REQUESTED,
 YOU PROVIDE THE NAME OF THE PARTICULAR INSTANCE IN THE CLASS
 OF PROBLEMS THAT YOU ARE NOW STUDYING.  THIS NAME MUST BE
 DIFFERENT FROM THE NAME YOU HAVE JUST PROVIDED ABOVE.

  ENTER THE SPECIFIC CASE NAME: doer


 ****************   BEGIN   ******************

 Purpose of BEGIN is to permit you to provide a starting design
 in an interactive mode. You give starting dimensions, material
 properties, allowables. The interactive session is stored on
 a file called doer.BEG, in which doer is a name that you
 have chosen for the specific case. (The name, doer, must
 remain the same as you use BEGIN, DECIDE, MAINSETUP, OPTIMIZE,
 and CHANGE.)  In future runs of the same or a
 slightly modified case, you will find it convenient to use the
 file doer.BEG as input.  Rather than answer all the questions
 interactively, you can use doer.BEG or an edited version of
 doer.BEG as input to BEGIN.  BEGIN also generates an output
 file called doer.OPB. OPB lists a summary of the case, and if
 you choose the tutorial option, the questions, helps, and your
 answers for each input datum.

 ************************************************
-----------------------------------------------------------------
```

[When you have completed BEGIN you will have the file, doer.BEG,

```
 which can be used in any future execution of BEGIN. (Table 13)]

decide          (provide decision variables, bounds, equality
                 constraints (linking expressions), and inequality
                 constraints. DECIDE saves your interactive input
                 in the file, doer.DEC (Table 14)).

mainsetup       {provide strategy, analysis type, etc.. MAINSETUP
                 saves your interactive input in the file, doer.OPT (Table 15).
                 Initially, set ITYPE = 2 (analysis of fixed design,
                 NOT optimization.

optimize        (Start the "batch" run that computes results for the
                 fixed design, the design listed in Table 16 of the
                 document, "The user of BIGBOSOR4 to obtain predictions
                 of stress and buckling of deep submergence shells"[13].
                 These results are listed in the file, doer.OPM (Table 16).
                 The GENOPT processor, "OPTIMIZE", when executed in the
                 ITYPE = 2 mode, produces two files that contain valid
                 input for BIGBOSOR4:
                 doer.BEHX1 = BIGBOSOR4 input file for buckling (INDIC = 1)
                 doer.BEHX2 = BIGBOSOR4 input file for stress (INDIC = 0).
                 Just below in this runstream doer.BEHX1 and doer.BEHX2
                 are used, as you will see next.)

(Next, we want to execute BIGBOSOR4 to obtain plots corresponding to
 buckling and stress. First, copy the two files,
 doer.BEHX1 and doer.BEHX2, into a directory from which you want to
 execute BIGBOSOR4:)

cp doer.BEHX1 /home/progs/bigbosor4/workspace/.    (buckling input,INDIC=1)
cp doer.BEHX2 /home/progs/bigbosor4/workspace/.    (stress input,  INDIC=2)

(Go to the directory where you want to run BIGBOSOR4.)

cd /home/progs/bigbosor4/workspace

bigbosor4log    (activate BIGBOSOR4 commands)

(The command, "bigbosor4log", presents the following to
 your screen:)
---------------------------------------------------------
The BIGBOSOR4 commands, in the general order in which
you would probably use them (except in GENOPT applications), are:

help4           (get information on BOSOR4.)
input           (you provide segment-by-seg. input)
assemble        (concatenates segment data files)
bigbosorall     (batch run of pre, main, post proc.)
bosorplot       (batch run for generating plot files)
resetup         (input for restart run, same model)
bigrestart              (batch run of main & postprocessors)
cleanup         (delete all except for .DOC file)
getsegs         (generate segment files from .DOC)
modify          (modify a segment file)
-----------------------------------------------------------


(Copy the BIGBOSOR4 input file for buckling (INDIC=1), doer.BEHX1,
 into doer.ALL because BIGBOSOR4 input files must always have
 the three-letter suffix, ".ALL":)

cp doer.BEHX1 doer.ALL

bigbosorall     (Start "batch" run for buckling. The output
                 file that you want to inspect is called doer.OUT.
                 This will be a fairly long file, so search specifically
                 for the string, "EIGENVALUE(", typed with the
                 "(" at the end of the string. You will find the
                 following output there:
-----------------------------------------------------------------
 **** CRITICAL EIGENVALUE AND WAVENUMBER ****
 EIGCRT=  1.8904E+00; NO. OF CIRC. WAVES, NWVCRT=    3
 ***************************************************

 ***** EIGENVALUES AND MODE SHAPES *****
  EIGENVALUE(CIRC. WAVES)
 =======================================
    5.1170E+00(   0)
    5.0881E+00(   1)
    2.3019E+00(   2)
    1.8904E+00(   3) <--critical buckling
    2.9298E+00(   4)
    4.4143E+00(   5)
    5.6151E+00(   6)
    6.1995E+00(   7)
    7.0420E+00(   8)
    8.1151E+00(   9)
    9.3925E+00(  10)
 ========================================
------------------------------------------------------------------

(The critical buckling mode has 3 circumferential waves around
 the circumference of the deep submergence tank, and the critcal
 buckling load factor is 1.890. The buckling mode is shown in Fig. 3.)


bosorplot       (obtain a plot of the critical buckling mode.
                 The postscript file is called "metafile.ps" (Fig. 3).)

(execution of bosorplot presents the following to your computer
 screen:)
----------------------------------------------------------------
Please enter the BIGBOSOR4 case name: doer

Do you want to use Xgraph or create a PostScript file? (Choose X or P) p

One, maybe Two moments please...

Text file(s) have been created containing plot data.  The names of the
files explain to a greater or lesser extent what the data represent.
Some plot files contain data for more than one plot.
1)      doer..R,Z_EIGENMODE_1--N_0
2)      doer..R,Z_EIGENMODE_1--N_1
3)      doer..R,Z_EIGENMODE_1--N_10
4)      doer..R,Z_EIGENMODE_1--N_2
5)      doer..R,Z_EIGENMODE_1--N_3
6)      doer..R,Z_EIGENMODE_1--N_4
7)      doer..R,Z_EIGENMODE_1--N_5
8)      doer..R,Z_EIGENMODE_1--N_6
9)      doer..R,Z_EIGENMODE_1--N_7
10)     doer..R,Z_EIGENMODE_1--N_8
11)     doer..R,Z_EIGENMODE_1--N_9
12)     doer..R,Z_RingLocation
CR)     to QUIT
Please choose the number of the file you wish to plot: 5
Plotting: Undeformed & Deformed Axial Station as a function of Radius

The PostScript file, metafile.ps, has been created.
Please choose one of the three options below:

   1) Rename the PostScript file.  This is useful if
      you don't have access to a PostScript printer on your
      machine, but you wish to save to a file so you can later
      transfer it to a different machine for printing.

         Example:  mv metafile.ps plot1.ps

   2) Enter an "lpr" command.  This is useful if your default
      printer is not PostScript, but there is a PostScript
      printer available on your system.

         Example:  lpr -PApplelaser metafile.ps

   3) Press the return key.  This executes the command:

                   lpr metafile.ps

      This assumes that your default printer is a PostScript
      printer.

Enter your command> <CR>
Printing PostScript plot on the default printer...

Text file(s) have been created containing plot data.  The names of the
files explain to a greater or lesser extent what the data represent.
```

```
Some plot files contain data for more than one plot.
1)      doer..R,Z_EIGENMODE_1--N_0
2)      doer..R,Z_EIGENMODE_1--N_1
3)      doer..R,Z_EIGENMODE_1--N_10
4)      doer..R,Z_EIGENMODE_1--N_2
5)      doer..R,Z_EIGENMODE_1--N_3
6)      doer..R,Z_EIGENMODE_1--N_4
7)      doer..R,Z_EIGENMODE_1--N_5
8)      doer..R,Z_EIGENMODE_1--N_6
9)      doer..R,Z_EIGENMODE_1--N_7
10)     doer..R,Z_EIGENMODE_1--N_8
11)     doer..R,Z_EIGENMODE_1--N_9
12)     doer..R,Z_RingLocation
CR)     to QUIT
Please choose the number of the file you wish to plot: <CR>
---------------------------------------------------------------------

(In order to view the plot of the local buckling mode, type
 the command:)

gv metafile.ps

("gv" stands for "ghost view" a LINUX utility which presents
 the postscript file, metafile.ps, as a plot on your screen.
 If you do not have "ghost view", just send the postscript
 file to your printer with whatever command is appropriate at
 your facility for obtaining plots from postscript files.)

(Next, "clean up" the BIGBOSOR4 files:)

cleanup          (deletes unneeded BIGBOSOR4 files and
                  generates a properly annotated doer.ALL
                  file (Table 11) and a properly annotated doer.DOC file.)


(Next, we wish to obtain a plot of the axisymmetric prebuckled
 state of the shell from execution of BIGBOSOR4 and BOSORPLOT:)

(Copy the BIGBOSOR4 input file for stress (INDIC=0), doer.BEHX2,
 into doer.ALL because BIGBOSOR4 input files must always have
 the three-letter suffix, ".ALL":)

cp doer.BEHX2 doer.ALL

bigbosorall      (Start "batch" run for stress (INDIC=0). The output
                  file that you want to inspect is called doer.OUT.
                  This will be a fairly long file, so search specifically
                  for the string, "STRMAX". You will find the following
                  output there:
----------------------------------------------------------------
 ****** MAXIMUM EFFECTIVE STRESS IN ISOTROPIC WALL ******
 STRMAX=  1.2234E+05
 ******************************************************

 (lines skipped to save space)

 Local skin and smeared stiffener buckling and stress, Seg.   1
 Skin buckling load factor,                     BUCMIN=  0.0000E+00 at nodal point  0
 Smeared stringer/isogrid buckling load factor, BUCMNS=  0.0000E+00 at nodal point  0
 Smeared ring buckling load factor,             BUCMNR=  0.0000E+00 at nodal point  0
 Smeared stringer/isogrid maximum eff. stress,  STFMXS=  0.0000E+00 at nodal point  0
 Smeared ring maximum effective stress,         STFMXR=  0.0000E+00 at nodal point  0
 Shell skin maximum effective stress,           SKNMAX=  1.2234E+05 at nodal point 38

 Local skin and smeared stiffener buckling and stress, Seg.   2
 Skin buckling load factor,                     BUCMIN=  0.0000E+00 at nodal point  0
 Smeared stringer/isogrid buckling load factor, BUCMNS=  0.0000E+00 at nodal point  0
 Smeared ring buckling load factor,             BUCMNR=  0.0000E+00 at nodal point  0
 Smeared stringer/isogrid maximum eff. stress,  STFMXS=  0.0000E+00 at nodal point  0
 Smeared ring maximum effective stress,         STFMXR=  0.0000E+00 at nodal point  0
 Shell skin maximum effective stress,           SKNMAX=  1.0877E+05 at nodal point 18

 Skin buckling load factor,          BUCSKN=  1.0000E+17
 Stiffener buckling load factor,     BUCSTF=  1.0000E+17
 Skin maximum effective stress,      STRMAX=  1.2234E+05
 Stiffener maximum effective stress, STRSTF=  0.0000E+00
 Normal displacement of shell at apex, ENDUV=  1.4438E-01
----------------------------------------------------------------
```

← This is BIGBOSOR4 output

Table 4, p. 9 of 15

```
bosorplot        (obtain a plot of the axisymmetric prebuckling deformation.
                  The postscript file is called "metafile.ps"(Fig. 2).)
```

(execution of bosorplot presents the following to your computer screen:)

```
-----------------------------------------------------------------
Please enter the BIGBOSOR4 case name: doer

Do you want to use Xgraph or create a PostScript file? (Choose X or P) p

One, maybe Two moments please...

Text file(s) have been created containing plot data.  The names of the
files explain to a greater or lesser extent what the data represent.
Some plot files contain data for more than one plot.
1)      doer..AXISYM_LOADSTEP_1
2)      doer..R,Z_LOADSTEP_1
3)      doer..R,Z_RingLocation
4)      doer..STRESS_LOADSTEP_1
CR)     to QUIT
Please choose the number of the file you wish to plot: 2
Plotting: Undeformed & Deformed Axial Station as a function of Radius

etc. etc. (as above in the buckling example).
------------------------------------------------------------------------
```

[NOTE: If you choose "3" in response to the bosorplot prompt, "Please choose the number of the file you wish to plot:" You get the plot displayed in Fig. 1.] if you use the "X" option here

```
bosorplot        (obtain plots of the meridional, circumferential, and
                  effective stress for inner and outer fibers. Use the
                  the "X" option, not the "P" option. Click on "postscript",
                  "file", and name the file something. then click on "ok".
                  Do the same for the next two plots. See Fig. 4 for a
                  plot of effective stress vs Arc length.)
```

(execution of bosorplot presents the following to your computer screen:)

```
-----------------------------------------------------------------
Please enter the BIGBOSOR4 case name: doer

Do you want to use Xgraph or create a PostScript file? (Choose X or P) x

One, maybe Two moments please...

Text file(s) have been created containing plot data.  The names of the
files explain to a greater or lesser extent what the data represent.
Some plot files contain data for more than one plot.
1)      doer..AXISYM_LOADSTEP_1
2)      doer..R,Z_LOADSTEP_1
3)      doer..R,Z_RingLocation
4)      doer..STRESS_LOADSTEP_1
CR)     to QUIT
Please choose the number of the file you wish to plot: 4
3 plots...
Plot 1: Merid. Stress in Left & Right fibers as a func. of Arc Length
Plot 2: Circumf. Stress in Left & Right fibers as a func. of Arc Length
Plot 3: Effective Stress in Left & Right fibers as a func. of Arc Length

etc. etc. (as above in the buckling example).
------------------------------------------------------------------------
```

(Next, "clean up" the BIGBOSOR4 files:)

```
cleanup          (deletes unneeded BIGBOSOR4 files and
                  generates  properly annotated doer.ALL
                  and doer.DOC files.)
```

(Next, return to the "genoptcase" directory, and continue processing the SPECIFIC case called "doer".)

```
cd /home/progs/genoptcase
```

```
mainsetup       (provide input for an optimization. The
                 interactive input for MAINSETUP is saved
                 in the file, doer.OPT (Table 17).)

superopt        (obtain a "global" optimum design. Use
                 5 "OPTIMIZEs" per AUTOCHANGE. This computer
                 run takes about 3 hours on my LINUX machine.)

(About 3 hours later, when the SUPEROPT run is finished,
 inspect the doer.OPP file)

chooseplot      (choose what to plot vs design iterations. The
                 interactive CHOOSEPLOT session is saved in the
                 file, doer.CPL, which is as follows in this
                 particular example:)
---------------------------------------------------------------------
      n         $ Do you want a tutorial session and tutorial output?
      n         $ Any design variables to be plotted v. iterations (Y or N)?
      n         $ Any design margins to be plotted v. iterations (Y or N)?
      n         $ Do you want to get more plots before your next "SUPEROPT"?
---------------------------------------------------------------------

[Note that following a complete SUPEROPT run (about 470 design iterations)
 we make no attempt to plot design variables or margins vs design iterations.
 The plots would be too messy because there are so many design iterations.
 We therefore obtain only a plot of the objective vs design iterations.]

diplot          [obtain the postscript file, doer.5.ps (design
                 objective vs design iterations)]

xprw doer.5.ps  (obtain a hard copy of the postscript plot, doer.5.ps (Fig.5).

(Edit the doer.OPT file in order to obtain the analysis of a fixed
 design, that is, change ITYPE from 1 to 2 and change NPRINT from 1 to 2)

mainsetup       (set up a run for the fixed, previously optimized, design.
                 see the top part of Table 18)

optimize        (obtain the doer.OPM file (Table 18) corresponding to the
                 optimized design.]

change          (Use the processor, CHANGE, to save the optimum design.
                 The interactive CHANGE session is saved in the file,
                 doer.CHG (Table 19).)

[With IDESIGN = 2 SUPEROPT accepts an "ALMOST FEASIBLE" or "FEASIBLE"
 design. An "ALMOST FEASIBLE" design is a design for which the minimum
 margin is greater than -0.05. In this case the accepted design has
 the following margins:]
-----------------------------------------------------------------------
 MARGIN CURRENT
 NO.    VALUE           DEFINITION
  1    3.380E-03  (BUCKL(1 )/BUCKLA(1 )) / BUCKLF(1 )-1; F.S.=  1.30
  2   -4.738E-02  (STRESSA(1 ,1 )/STRESS(1 ,1 )) / STRESSF(1 ,1 )-1; F.S.=1.0
  3   -2.880E-02  (STRESSA(1 ,2 )/STRESS(1 ,2 )) / STRESSF(1 ,2 )-1; F.S.=1.0
-----------------------------------------------------------------------

[The corresponding "behaviors" (buckling, maximum effective stress in shell
 segment 1, and maximum effective stress in shell segment 2) are:]
----------------------------------------------------------------
 BEH.   CURRENT
 NO.     VALUE             DEFINITION
  1    1.304E+00    tank buckling eigenvalue: BUCKL(1 )
  2    1.260E+05    effective stress in shell segment: STRESS(1 ,1 )
  3    1.236E+05    effective stress in shell segment: STRESS(1 ,2 )
-----------------------------------------------------------------

[We would like to obtain an optimum design with a somewhat smaller
 maximum effective stress. Therefore, we do the following:]

cleanspec       (clean up the files with the SPECIFIC name, "doer")
begin           (restart the specific case, "doer". Use Table 13 as input.)
change          (resurrect the optimum design listed in Table 18. Use
                 Table 19 as input)
decide          (use Table 14 as input)

[Edit the doer.OPT file for input to MAINSETUP. Increase the number of
 design iterations from 5 to 15; change IDESIGN from 2 to 1: change
```

See page 2 of Table 18

IMOVE from 1 to 3. IDESIGN = 1 means that "OPTIMIZE", running in the optimization mode (ITYPE = 1), accepts only designs that are defined by GENOPT as being "FEASIBLE". This means that OPTIMIZE only accepts designs the minimum margin for which is greater than -0.01. IMOVE = 3 means that the move limits of the decision variables during design iterations are severely restricted.]

```
mainsetup       (use Table 20 as input)

optimize        (execute OPTIMIZE. Note: we do not use SUPEROPT for
                 this run because here we are not making a general
                 search over a wide region of design space, but we
                 are only refining the design in the immediate
                 neighborhood of an optimum design that we have
                 already determined: the design listed in Table 18.)
```

[This execution of OPTIMIZE leads to the following (abridged) output in the doer.OPP file:]

```
-----------------------------------------------------------------
 *********** MARGINS FOR  12 ITERATIONS **********
 (BUCKL(1 )/BUCKLA(1 )) / BUCKLF(1 )-1; F.S.=  1.30 =
  3.3780E-03  2.7912E-02  4.3949E-02  5.9787E-02  7.2828E-02
  8.3299E-02  9.1773E-02  8.7449E-02  9.2898E-02  9.7283E-02
  1.0081E-01  1.0364E-01
 (STRESSA(1 ,1 )/STRESS(1 ,1 )) / STRESSF(1 ,1 )-1; F.S.=  1.00 =
 -4.7406E-02 -1.6747E-02 -2.4211E-02 -2.2346E-02 -2.1052E-02
 -1.9973E-02 -1.9140E-02 -1.8501E-02 -1.7996E-02 -1.7600E-02
 -1.7289E-02 -1.7042E-02                                        see next page
 (STRESSA(1 ,2 )/STRESS(1 ,2 )) / STRESSF(1 ,2 )-1; F.S.=  1.00 =
 -2.8801E-02 -2.1182E-02 -1.2836E-02 -7.8496E-03 -4.4315E-03
 -1.1914E-03  1.4168E-03 -1.0403E-03  6.3443E-04  1.9782E-03
  3.0552E-03  3.9189E-03
```

(many lines skipped to save space)

```
 ==============================================================
 ITERATION                                        NUMBER OF
  NUMBER      OBJECTIVE     THE DESIGN IS...   CRITICAL MARGINS
 --------------------------------------------------------------
 ------------------------------------------------------OPTIMIZE
     1         1.3688E+03     ALMOST FEASIBLE         3
     2         1.3787E+03     ALMOST FEASIBLE         3
     3         1.3822E+03     ALMOST FEASIBLE         3
     4         1.3875E+03     ALMOST FEASIBLE         2
     5         1.3919E+03     ALMOST FEASIBLE         2
     6         1.3954E+03     ALMOST FEASIBLE         2
     7         1.3982E+03     ALMOST FEASIBLE         2
     8         1.3975E+03     ALMOST FEASIBLE         2
     9         1.3993E+03     ALMOST FEASIBLE         2
    10         1.4008E+03     ALMOST FEASIBLE         2
    11         1.4019E+03     ALMOST FEASIBLE         2
    12         1.4029E+03     ALMOST FEASIBLE         2  <--We want this design.
 ==============================================================
  VALUES OF DESIGN VARIABLES CORRESPONDING TO ALMOST FEASIBLE DESI
 VAR.   CURRENT
 NO.     VALUE              DEFINITION
  1    6.250E-01  location of ref. surf. in the dome: ZREFSP(1 )
  2    6.250E-01  location of ref. surf. in the dome: ZREFSP(2 )
  3    6.669E-01  location of ref. surf. in the dome: ZREFSP(3 )
  4    6.444E-01  location of ref. surf. in the dome: ZREFSP(4 )
  5    7.748E-01  location of ref. surf. in the dome: ZREFSP(5 )
  6    1.250E+00  wall thickness in the dome: THKSPH(1 )
  7    1.250E+00  wall thickness in the dome: THKSPH(2 )
  8    1.309E+00  wall thickness in the dome: THKSPH(3 )
  9    1.250E+00  wall thickness in the dome: THKSPH(4 )
 10    1.568E+00  wall thickness in the dome: THKSPH(5 )
 11    9.942E-01  location of the ref. surf. in the cylinder: ZREFCY(1 )
 12    1.095E+00  location of the ref. surf. in the cylinder: ZREFCY(2 )
 13    8.384E-01  location of the ref. surf. in the cylinder: ZREFCY(3 )
 14    1.090E+00  location of the ref. surf. in the cylinder: ZREFCY(4 )
 15    1.090E+00  location of the ref. surf. in the cylinder: ZREFCY(5 )
 16    1.905E+00  thickness of the cylindrical shell: THKCYL(1 )
 17    2.116E+00  thickness of the cylindrical shell: THKCYL(2 )
 18    1.954E+00  thickness of the cylindrical shell: THKCYL(3 )
 19    2.180E+00  thickness of the cylindrical shell: THKCYL(4 )
 20    2.180E+00  thickness of the cylindrical shell: THKCYL(5 )

 MARGINS CORRESPONDING TO THE DESIGN (F.S.= FACTOR OF SAFETY)
```

Table 4, p.12 of 15

```
 MAR.   CURRENT
 NO.     VALUE             DEFINITION
  1    3.378E-03  (BUCKL(1 )/BUCKLA(1 )) / BUCKLF(1 )-1; F.S.=  1.30
  2   -4.741E-02  (STRESSA(1 ,1 )/STRESS(1 ,1 )) / STRESSF(1 ,1 )-1; F.S.=  1.
  3   -2.880E-02  (STRESSA(1 ,2 )/STRESS(1 ,2 )) / STRESSF(1 ,2 )-1; F.S.=  1.

 ******************* DESIGN OBJECTIVE ******************
  CORRESPONDING VALUE OF THE OBJECTIVE FUNCTION:
 VAR.   CURRENT
 NO.     VALUE             DEFINITION
  1    1.369E+03  weight of the BIGBOSOR4 model: WEIGHT
 ******************* DESIGN OBJECTIVE ******************
----------------------------------------------------------------------
```

[Notice that in the above list there are no "FEASIBLE" designs, only "ALMOST FEASIBLE" designs. This is because the minimum margin, the margin for effective stress in shell segment 1, (STRESSA(1,1)/STRESS(1,1))/STRESSF(1,1)-1;F.S.=1.0=-1.7042E-02, is greater than -0.05, the lower limit for "ALMOST FEASIBLE", but less than -0.01, the lower limit for "FEASIBLE". Therefore, the design in the doer.OPP file listed above, for which the weight is 1369 lb, is not the design we want but the same "ALMOST FEASIBLE" design that is listed in Table 18. In this particular case we want the design that corresponds to the last design iteration listed above, that is, the design corresponding to Iteration No. 12:

```
   12          1.4029E+03     ALMOST FEASIBLE
```

which is associated with design margin, -1.7042E-02. We can find this design near the end of the doer.OPM file. ~~It is listed in Table 22.]~~

see the previous page.

see above (previous page)

important!

[Next, get plots of design margins, design variables, and the objective corresponding to the postSUPEROPT execution of OPTIMIZE in the ITYPE=1 (optimization) mode.]

```
chooseplot       (choose what to plot. The input for CHOOSEPLOT
                  is listed in the doer.CPL file, Table 21.]

diplot           (get the following postscript files:
                  doer.3.ps = design margins   vs iterations: Fig.6,
                  doer.4.ps = design variables vs iterations: Fig.7,
                  doer.5.ps = design objective vs iterations: Fig.8)
```

[Edit the doer.OPT file to do a fixed design analysis (ITYPE=2), then execute "OPTIMIZE" in the ITYPE = 2 mode.]

```
mainsetup        (execute MAINSETUP; input at the top of Table 22)
optimize         (execute OPTIMIZE and generate Table 22)
```

[Figs. 9 and 10 show the distribution of shell wall material with respect to the reference surface of the optimized shell.]

[Next, execute CHANGE in order to save the best design determined after completion of the "postSUPEROPT" optimization. This best design is listed in Table 22.]

```
change           (Use for input the doer.CHG file listed in Table 23)
```

(Next, we want to execute BIGBOSOR4 to obtain plots corresponding to buckling (INDIC=1) and stress (INDIC=0) for the optimized case. First, copy the two files, doer.BEHX1 and doer.BEHX2, into a directory from which you want to execute BIGBOSOR4:)

```
cp doer.BEHX1 /home/progs/bigbosor4/workspace/.    (buckling input, INDIC=1)
cp doer.BEHX2 /home/progs/bigbosor4/workspace/.    (stress input,   INDIC=0)
```

(Go to the directory where you want to run BIGBOSOR4.)

```
cd /home/progs/bigbosor4/workspace

bigbosor4log     (activate BIGBOSOR4 commands)
```

(First, we wish to obtain a plot of the critical buckling mode from execution of BIGBOSOR4 and BOSORPLOT)

Table 8, p. 13-15

(Copy the BIGBOSOR4 input file for buckling, doer.BEHX1, into doer.ALL because BIGBOSOR4 input files must always have the three-letter suffix, ".ALL":)

```
cp doer.BEHX1 doer.ALL

bigbosorall      (Start "batch" run for buckling. The output
                  file that you want to inspect is called doer.OUT.
                  This will be a long file, so search specifically
                  for the string, "EIGENVALUE(", typed with the
                  "(" at the end of the string. You will find the
                  following output there:
----------------------------------------------------------------
 **** CRITICAL EIGENVALUE AND WAVENUMBER ****
 EIGCRT=  1.4347E+00; NO. OF CIRC. WAVES, NWVCRT=    3
 ****************************************************

 ***** EIGENVALUES AND MODE SHAPES *****
  EIGENVALUE(CIRC. WAVES)
 =======================================
    5.0469E+00(   0)
    5.0282E+00(   1)
    1.9429E+00(   2)
    1.4347E+00(   3)←— critical eigenvalue & mode
    2.1787E+00(   4)
    3.2639E+00(   5)
    4.4859E+00(   6)
    5.2346E+00(   7)
    5.7873E+00(   8)
    6.5147E+00(   9)
    7.4054E+00(  10)
 =======================================
-----------------------------------------------------------------


bosorplot        (obtain a plot of the critical buckling mode.
                  The postscript file is called "metafile.ps"(Fig.12).)

cleanup          (clean up BIGBOSOR4 files called "doer" and
                  generate the version of doer.ALL and doer.DOC
                  with proper annotation throughout.)
```

[NOTE: An appropriate factor of safety for buckling of the spherical dome is between 3 and 4, whereas an appropriate factor of safety for buckling of the cylindrical shell is between 1.2 and 1.5. We know that the cylindrical shell is ok for buckling because the critical buckling load factor is 1.4347 (n=3 circ.waves) from the eigenvalues just listed. However, we do not yet know whether the spherical dome is safe.]

[In order to determine whether or not the spherical shell is safe, we have to find the lowest buckling load factor corresponding to a buckling mode in which the spherical dome deforms significantly. We do two things:

1. Look at the buckling modes corresponding to circumferential waves n = 2 - 5 (the values of n for which the lowest eigenvalue is less than 4.0) to make certain these modes mainly involve buckling of the cylindrical shell. Suppose that they all do. (Actually, they all do; I checked.)

2. Next we need to look for the lowest eigenvalue that corresponds to buckling of the spherical dome. In order to do this we need to edit the doer.ALL file and change the input relating to the number of eigenvalues NVEC to be computed per circumferential wave number. We do this and get the following input lines in the doer.ALL file:]

```
------------------------------------------------------------------
   2    $ N0B    = starting number of circ. waves (buckling analysis)
   2    $ NMINB = minimum number of circ. waves  (buckling analysis)
   5    $ NMAXB = maximum number of circ. waves  (buckling analysis)
   1    $ INCRB = increment in number of circ. waves (buckling)
   3    $ NVEC  = number of eigenvalues for each wave number
------------------------------------------------------------------
```

[Then we execute bigbosorall again:]

Table 4, p.14 of 15

```
bigbosorall
```

[We obtain the following lines in the doer.OUT file (edited somewhat):]

```
----------------------------------------------------------
 CIRCUMFERENTIAL WAVE NUMBER, N =        2
 EIGENVALUES =
    1.94287E+00    5.00331E+00    5.56276E+00

 CIRCUMFERENTIAL WAVE NUMBER, N =        3
 EIGENVALUES =
    1.43475E+00    4.99944E+00    5.49431E+00

 CIRCUMFERENTIAL WAVE NUMBER, N =        4
 EIGENVALUES =
    2.17870E+00    4.90940E+00    5.19972E+00

 CIRCUMFERENTIAL WAVE NUMBER, N =        5
 EIGENVALUES =
    3.26393E+00    5.02135E+00    5.23608E+00
------------------------------------------------------------
```

lowest 2nd eigenvalue
See Fig. 13

[We already checked the first eigenvalues for N = 2 to 5, and all of them correspond to buckling primarily of the cylindrical shell, for which the factor of safety between 1.3 and 1.5 applies. The lowest 2nd eigenvalue in the range from N = 2 to 5 corresponds to N = 4 circumferential waves, and the eigenvalue is 4.90940. Already we know, without even looking at the buckling mode shape, that the spherical dome is safe. We know it is safe because there is no eigenvalue less than 4.0 corresponding to buckling of the spherical shell. Out of curiousity, we execute bosorplot to see what the 2nd eigenvector for N = 4 looks like]

```
bosorplot        (The 2nd mode for N = 4 is indeed buckling of
                  the spherical dome, Fig. 13).
```

(Next, we wish to obtain a plot of the axisymmetric prebuckled state from execution of BIGBOSOR4 and BOSORPLOT)

(Copy the BIGBOSOR4 input file for stress, doer.BEHX2, into doer.ALL because BIGBOSOR4 input files must always have the three-letter suffix, ".ALL":)

```
cp doer.BEHX2 doer.ALL

bigbosorall      (Start "batch" run for stress. The output
                  file that you want to inspect is called doer.OUT.
                  This will be a rather long file, so search specifically
                  for the string, "STRMAX". You will find the
                  following output there:
--------------------------------------------------------------------------------
 ****** MAXIMUM EFFECTIVE STRESS IN ISOTROPIC WALL ******
 STRMAX=  1.2208E+05
 *****************************************************
-------------------------------------------------------------------------------


bosorplot        (obtain a plot of the axisymmetric prebuckled state.
                  The postscript file is called "metafile.ps"(Fig. 11).)

bosorplot        (obtain plots of the meridional, circumferential, and
                  effective stress for inner and outer fibers. Use the
                  the "X" option, not the "P" option. Click on "postscript",
                  "file", and name the file something. then click on "ok".
                  Do the same for the next two plots. See Figs. 14-16.)

cleanup          (clean up BIGBOSOR4 files called "doer.*)
```

(Next, return to the "genoptcase" directory, and continue processing the SPECIFIC case called "doer".)

```
cd /home/progs/genoptcase
```

[Next, we optimize using 10 callout stations in each of Segment 1 and Segment 2 for the reference surface location relative to the leftmost surface and for the thickness. We edit doer.BEG and doer.DEC to reflect this change. Also, we use 91 nodal points in the cylindrical shell instead of 51 and we narrow the search for the critical buckling load from n = 0 to 10 circumferential waves to n = 0 to 5 circumferential waves. The new doer.BEG and doer.DEC files are listed in Tables 24 and 25.]

```
begin          (supply starting design, etc. Table 24 is input)
decide         (choose decision variables, etc. Table 25 is input)
mainsetup      (choose analysis type, etc. Table 17 is input)
superopt       (find "global" optimum design)
chooseplot     (choose to plot the objective vs design iterations)
diplot         (get the postscript file, doer.5.ps: Fig. 17)
[Edit the doer.OPT file to do analysis of fixed design: ITYPE=2]
mainsetup      (set up analysis of fixed design]
optimize       (perform the analysis of the fixed design. The
                output is listed in Table 26)
```

[This optimum design is of questionable value. Specification of 10 callout points for ZVAL and TVAL in each of shell segments 1 (the spherical shell) and 2 (the cylindrical shell) is too many for such a thick shell. The optimum design obtained with the use of 5 callout points in each of the two shell segments makes more sense.] ← Note!

end of RUN STREAM

Table 5 (7 pages) Input for GENTEXT

submarine.INP. File

```
      5  $ starting prompt index in the file submarine.PRO
      5  $ increment for prompt index
      0  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
 This application of GENOPT/BIGBOSOR4 is for an
 y         $ Are there more lines in the "help" paragraph?
 externally pressurized tank which has a spherical
 y         $ Are there more lines in the "help" paragraph?
 dome connected to a cylindrical shell. The material is
 y         $ Are there more lines in the "help" paragraph?
 elastic. The tank is subjected to uniform external
 y         $ Are there more lines in the "help" paragraph?
 pressure. Design constraints include stress and buckling.
 y         $ Are there more lines in the "help" paragraph?
 Symmetry conditions are applied at the midlength of the
 y         $ Are there more lines in the "help" paragraph?
 cylindrical shell. The objective is minimum weight.
 y         $ Are there more lines in the "help" paragraph?
 The main aim of the optimization is to determine the
 y         $ Are there more lines in the "help" paragraph?
 best distribution of the material in the neighborhood
 y         $ Are there more lines in the "help" paragraph?
 of the junction between the spherical dome and the
 y         $ Are there more lines in the "help" paragraph?
 cylindrical shell. Hence the position of the reference
 y         $ Are there more lines in the "help" paragraph?
 surface and the wall thickness vary in the neighborhood
 y         $ Are there more lines in the "help" paragraph?
 of the junction between the spherical dome and the
 y         $ Are there more lines in the "help" paragraph?
 cylindrical shell. In the BIGBOSOR4 model the tank
 y         $ Are there more lines in the "help" paragraph?
 consists of two shell segments. Segment 1 is the
 y         $ Are there more lines in the "help" paragraph?
 spherical dome, and Segment 2 is the cylindrical
 y         $ Are there more lines in the "help" paragraph?
 shell. Nodal points are concentrated in the region near
 y         $ Are there more lines in the "help" paragraph?
 the junction between dome and cylinder. The INDIC=1
 y         $ Are there more lines in the "help" paragraph?
 branch of BIGBOSOR4 is used for the buckling analysis,
 y         $ Are there more lines in the "help" paragraph?
 and the INDIC = 0 branch of BIGBOSOR4 is used for the
 y         $ Are there more lines in the "help" paragraph?
 stress analysis.
 n         $ Are there more lines in the "help" paragraph?
       1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt

 RADIUS    $ Name of a variable in the users program (defined below)
       2  $ Role of the variable in the users program
       2  $ type of variable:  1 =integer,  2 =floating point
 n         $ Is the variable  RADIUS  an array?
 radius of the tank
 y         $ Do you want to include a "help" paragraph?
 The radius is measured to the reference surface.
 n         $ Any more lines in the "help" paragraph?
 y         $ Any more variables for role types  1  or  2   ?    $10
       1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
 LENGTH    $ Name of a variable in the users program (defined below)
       2  $ Role of the variable in the users program
       2  $ type of variable:  1 =integer,  2 =floating point
 n         $ Is the variable  LENGTH  an array?
 total length of the cylinder
 y         $ Do you want to include a "help" paragraph?
 This is twice the length used in the analysis.
 y         $ Any more lines in the "help" paragraph?
 It is the length of cylindrical shell between the two
 y         $ Any more lines in the "help" paragraph?
 end domes. In the BIGBOSOR4 model half this length is
 y         $ Any more lines in the "help" paragraph?
 included, with symmetry conditions imposed at the end
 y         $ Any more lines in the "help" paragraph?
 of the cylindrical shell which is at the symmetry plane
 y         $ Any more lines in the "help" paragraph?
 of the tank.
 n         $ Any more lines in the "help" paragraph?
 y         $ Any more variables for role types  1  or  2   ?    $15
       1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
 EMOD      $ Name of a variable in the users program (defined below)
```

Table 5, p. 2 of 7

```
      2  $ Role of the variable in the users program
      2  $ type of variable:  1 =integer,  2 =floating point
n         $ Is the variable  EMOD  an array?
elastic modulus of the shell wall
n         $ Do you want to include a "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $20
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
NU        $ Name of a variable in the users program (defined below)
      2  $ Role of the variable in the users program
      2  $ type of variable:  1 =integer,  2 =floating point
n         $ Is the variable  NU  an array?
Poisson ratio of the shell wall
n         $ Do you want to include a "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $25
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
DENSTY    $ Name of a variable in the users program (defined below)
      2  $ Role of the variable in the users program
      2  $ type of variable:  1 =integer,  2 =floating point
n         $ Is the variable  DENSTY  an array?
weight density of the shell wall
y         $ Do you want to include a "help" paragraph?
For example, aluminum has a weight density of 0.1 lb/in^3.
n         $ Any more lines in the "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $30
      0  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
Next, you will be asked to provide the number of
y         $ Are there more lines in the "help" paragraph?
nodal points to be used in the dome and the number of
y         $ Are there more lines in the "help" paragraph?
nodal points to be used in the cylinder. Choose odd
y         $ Are there more lines in the "help" paragraph?
integers between 51 and 91 . You might optimize first
y         $ Are there more lines in the "help" paragraph?
with the use of 51 nodal points in each shell segment
y         $ Are there more lines in the "help" paragraph?
(51 nodal points in the spherical dome and 51 nodal
y         $ Are there more lines in the "help" paragraph?
points in the cylindrical shell), then check your optimum
y         $ Are there more lines in the "help" paragraph?
design with the use of 91 nodal points in each shell
y         $ Are there more lines in the "help" paragraph?
segment.
n         $ Are there more lines in the "help" paragraph?
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
NODSPH    $ Name of a variable in the users program (defined below)
      2  $ Role of the variable in the users program
      1  $ type of variable:  1 =integer,  2 =floating point
n         $ Is the variable  NODSPH  an array?
number of nodal points in the dome
n         $ Do you want to include a "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $40
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
NODCYL    $ Name of a variable in the users program (defined below)
      2  $ Role of the variable in the users program
      1  $ type of variable:  1 =integer,  2 =floating point
n         $ Is the variable  NODCYL  an array?
number of nodal points in the cylinder
n         $ Do you want to include a "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $45
      0  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
You will next be asked to provide the number of callout
y         $ Are there more lines in the "help" paragraph?
stations in the dome and the locations of those
y         $ Are there more lines in the "help" paragraph?
stations in the dome, followed by the number of callout
y         $ Are there more lines in the "help" paragraph?
stations in the cylinder, followed by the axial locations
y         $ Are there more lines in the "help" paragraph?
of those stations in the cylinder at which two quantities
y         $ Are there more lines in the "help" paragraph?
are specified:
y         $ Are there more lines in the "help" paragraph?
1. the distance from the leftmost surface of the shell
y         $ Are there more lines in the "help" paragraph?
wall to the reference surface ("leftmost" means the
y         $ Are there more lines in the "help" paragraph?
left-hand surface of the shell wall as you face in
y         $ Are there more lines in the "help" paragraph?
```

Table 5, p.3 of 7

```
the direction of increasing arc length).
y         $ Are there more lines in the "help" paragraph?
2. the thickness of the shell wall.
y         $ Are there more lines in the "help" paragraph?
The locations of the reference surface and the wall thicknesses
y         $ Are there more lines in the "help" paragraph?
are to be decision variables in the optimization problem.
n         $ Are there more lines in the "help" paragraph?
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
NPTSPH    $ Name of a variable in the users program (defined below)
      2  $ Role of the variable in the users program
      1  $ type of variable:  1 =integer,  2 =floating point
n         $ Is the variable  NPTSPH  an array?
number of axial callouts in the dome
y         $ Do you want to include a "help" paragraph?
Use a number between 5 and 10, probably 5 at first.
n         $ Any more lines in the "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $55
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
ZSPH      $ Name of a variable in the users program (defined below)
      2  $ Role of the variable in the users program
      2  $ type of variable:  1 =integer,  2 =floating point
y         $ Is the variable  ZSPH  an array?
y         $ Do you want to establish new dimensions for ZSPH ?
      1  $ Number of dimensions in the array,  ZSPH
number of axial callouts in the dome
     15  $ Max. allowable number of rows NROWS in the array, ZSPH
axial location of callout in the dome
y         $ Do you want to include a "help" paragraph?
The axial callout is the axial distance from the apex
y         $ Any more lines in the "help" paragraph?
of the dome (where the dome meets the axis of revolution
y         $ Any more lines in the "help" paragraph?
of the tank) to the callout point. NOTE: The first callout
y         $ Any more lines in the "help" paragraph?
point must be ZSPH(1) = 0.0 (the location of the apex), and
y         $ Any more lines in the "help" paragraph?
the last callout point must be the equator of the dome
y         $ Any more lines in the "help" paragraph?
(ZSPH(NPTSPH) = RADIUS). You will want the thickness of the
y         $ Any more lines in the "help" paragraph?
dome to be constant for most of the arc length of the dome,
y         $ Any more lines in the "help" paragraph?
that is, from the dome apex to ZSPH = RADIUS/2, for
y         $ Any more lines in the "help" paragraph?
example. ZSPH = RADIUS/2 corresponds to an angle of 60
y         $ Any more lines in the "help" paragraph?
degrees from the apex to the second axial callout point.
y         $ Any more lines in the "help" paragraph?
Then you provide axial locations of additional callout
y         $ Any more lines in the "help" paragraph?
points located on the dome reference surface between
y         $ Any more lines in the "help" paragraph?
that second callout point at ZSPH(2) = RADIUS/2 to the
y         $ Any more lines in the "help" paragraph?
NPTSPHth callout point, which must correspond to the
y         $ Any more lines in the "help" paragraph?
equator of the dome, that is, ZSPH(NPTSPH) = RADIUS.
n         $ Any more lines in the "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $65
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
NPTCYL    $ Name of a variable in the users program (defined below)
      2  $ Role of the variable in the users program
      1  $ type of variable:  1 =integer,  2 =floating point
n         $ Is the variable  NPTCYL  an array?
Number of axial callouts in the cylinder
y         $ Do you want to include a "help" paragraph?
Use an integer between 5 and 10; probably 5 is best to
y         $ Any more lines in the "help" paragraph?
start with. Then you can use more in a re-optimization
y         $ Any more lines in the "help" paragraph?
if you wish.
n         $ Any more lines in the "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $70
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
ZCYL      $ Name of a variable in the users program (defined below)
      2  $ Role of the variable in the users program
      2  $ type of variable:  1 =integer,  2 =floating point
```

Table 5, p.417

```
y         $ Is the variable  ZCYL  an array?
y         $ Do you want to establish new dimensions for ZCYL ?
      1  $ Number of dimensions in the array,  ZCYL
number of axial callouts in the cylinder
     15  $ Max. allowable number of rows NROWS in the array, ZCYL
axial location of callout in the cylinder
y         $ Do you want to include a "help" paragraph?
The axial location is measured from the apex of the dome
y         $ Any more lines in the "help" paragraph?
to the callout point. As callout points in the cylindrical
y         $ Any more lines in the "help" paragraph?
shell you must include the first point in the cylindrical
y         $ Any more lines in the "help" paragraph?
shell, that is, ZCYL(1) = RADIUS, and the last point in the
y         $ Any more lines in the "help" paragraph?
cylindrical shell, that is, ZCYL(NPTCYL) = RADIUS + LENGTH/2.
y         $ Any more lines in the "help" paragraph?
You want most of the length of the cylindrical shell to
y         $ Any more lines in the "help" paragraph?
have a constant thickness. Therefore, the second-to-last
y         $ Any more lines in the "help" paragraph?
callout point should probably be located at something like
y         $ Any more lines in the "help" paragraph?
ZCYL(NPTCYL-1) = RADIUS + LENGTH/2 - LENGTH/4.
n         $ Any more lines in the "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $80
      0  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
Next, you will be asked to provide the following decision
y         $ Are there more lines in the "help" paragraph?
variables:
y         $ Are there more lines in the "help" paragraph?
ZREFSP = distance from the leftmost shell wall surface to
y         $ Are there more lines in the "help" paragraph?
the reference surface in the spherical shell.
y         $ Are there more lines in the "help" paragraph?
THKSPH = thickness of the shell wall at the callout points
y         $ Are there more lines in the "help" paragraph?
in the spherical shell
y         $ Are there more lines in the "help" paragraph?
ZREFCY = distance from the leftmost shell wall surface to
y         $ Are there more lines in the "help" paragraph?
the reference surface in the cylindrical shell
y         $ Are there more lines in the "help" paragraph?
THKCYL = thickness of the shell wall at the callout points
y         $ Are there more lines in the "help" paragraph?
in the cylindrical shell.
n         $ Are there more lines in the "help" paragraph?
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
ZREFSP    $ Name of a variable in the users program (defined below)
      1  $ Role of the variable in the users program
y         $ Is the variable  ZREFSP  an array?
y         $ Do you want to establish new dimensions for ZREFSP ?
      1  $ Number of dimensions in the array,  ZREFSP
axial callout number in the dome
     15  $ Max. allowable number of rows NROWS in the array, ZREFSP
location of ref. surf. in the dome
y         $ Do you want to include a "help" paragraph?
This is the distance from the shell wall leftmost surface to
y         $ Any more lines in the "help" paragraph?
the reference surface as you face in the direction of
y         $ Any more lines in the "help" paragraph?
increasing arc length in the spherical dome.
n         $ Any more lines in the "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $95
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
THKSPH    $ Name of a variable in the users program (defined below)
      1  $ Role of the variable in the users program
y         $ Is the variable  THKSPH  an array?
n         $ Do you want to establish new dimensions for THKSPH ?
wall thickness in the dome
n         $ Do you want to include a "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $100
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
ZREFCY    $ Name of a variable in the users program (defined below)
      1  $ Role of the variable in the users program
y         $ Is the variable  ZREFCY  an array?
y         $ Do you want to establish new dimensions for ZREFCY ?
      1  $ Number of dimensions in the array,  ZREFCY
```

Table 5, p.5-17

```
callout number in the cylinder
     15  $ Max. allowable number of rows NROWS in the array, ZREFCY
location of the ref. surf. in the cylinder
y         $ Do you want to include a "help" paragraph?
This is the distance from the leftmost shell wall surface
y         $ Any more lines in the "help" paragraph?
to the reference surface. "Leftmost" means the shell wall]
y         $ Any more lines in the "help" paragraph?
surface on the left-hand side as you face in the direction
y         $ Any more lines in the "help" paragraph?
of increasing arc length along the shell segment.
n         $ Any more lines in the "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $110
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
THKCYL    $ Name of a variable in the users program (defined below)
      1  $ Role of the variable in the users program
y         $ Is the variable  THKCYL  an array?
n         $ Do you want to establish new dimensions for THKCYL ?
thickness of the cylindrical shell
n         $ Do you want to include a "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $115
      0  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
Next you will be asked to provide the range of
y         $ Are there more lines in the "help" paragraph?
circumferential wave numbers to be used during
y         $ Are there more lines in the "help" paragraph?
the buckling analysis. You will be asked to provide
y         $ Are there more lines in the "help" paragraph?
NBUKLO and NBUKHI, where NBUKLO is the low end of the
y         $ Are there more lines in the "help" paragraph?
range of circumferential wave numbers and NBUKHI is the
y         $ Are there more lines in the "help" paragraph?
high end of the range of circumferential wave numbers.
y         $ Are there more lines in the "help" paragraph?
It is best to use NBUKL0 = 0 and NBUKHI = something like
y         $ Are there more lines in the "help" paragraph?
10
n         $ Are there more lines in the "help" paragraph?
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
NBUKLO    $ Name of a variable in the users program (defined below)
      2  $ Role of the variable in the users program
      1  $ type of variable:  1 =integer,  2 =floating point
n         $ Is the variable  NBUKLO  an array?
low end of range of buckling circ. waves
y         $ Do you want to include a "help" paragraph?
Usually you will want to set NBUKLO = 0
n         $ Any more lines in the "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $125
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
NBUKHI    $ Name of a variable in the users program (defined below)
      2  $ Role of the variable in the users program
      1  $ type of variable:  1 =integer,  2 =floating point
n         $ Is the variable  NBUKHI  an array?
high end of range of buckling circ. waves
y         $ Do you want to include a "help" paragraph?
Use an integer less than 20   . NBUKHI = 10 will probably
y         $ Any more lines in the "help" paragraph?
suffice.
n         $ Any more lines in the "help" paragraph?
n         $ Any more variables for role types  1  or  2   ?    $
      0  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
In this formulation the only environmental parameter (load)
y         $ Are there more lines in the "help" paragraph?
is the external pressure. Use a positive number.
n         $ Are there more lines in the "help" paragraph?
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
PRESS     $ Name of a variable in the users program (defined below)
      3  $ Role of the variable in the users program
uniform external pressure
y         $ Do you want to include a "help" paragraph?
For the deepest part of the ocean (35000 feet) the
y         $ Are there more lines in the "help" paragraph?
pressure is (64 x 35000)/144 psi.
n         $ Any more lines in the "help" paragraph?
n         $ Any more variables for role type  3 ?              $
      0  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
In this formulation there are two "behavioral" design
y         $ Are there more lines in the "help" paragraph?
```

Table 5, p. 6 of 7

```
constraints: 1 = buckling and 2 = stress. The buckling
y         $ Are there more lines in the "help" paragraph?
variable is called BUCKL and the stress variable is
y         $ Are there more lines in the "help" paragraph?
called STRESS. By STRESS is meant, in this case that
y         $ Are there more lines in the "help" paragraph?
involves only isotropic material, "effective stress",
y         $ Are there more lines in the "help" paragraph?
which is the VonMises effective stress:
y         $ Are there more lines in the "help" paragraph?
STRESS = SQRT(sig1^2 +sig2^2 -sig1sig2 +3*sig12^2)
n         $ Are there more lines in the "help" paragraph?
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
BUCKL     $ Name of a variable in the users program (defined below)
      4  $ Role of the variable in the users program
y         $ Do you want to reset the number of columns in BUCKL ?
      1  $ Number of dimensions in the array,  BUCKL
tank buckling eigenvalue
n         $ Do you want to include a "help" paragraph?
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
BUCKLA    $ Name of a variable in the users program (defined below)
      5  $ Role of the variable in the users program
allowable buckling load (Use 1.0)
y         $ Do you want to include a "help" paragraph?
The buckling eigenvalue is a buckling load factor. The
y         $ Any more lines in the "help" paragraph?
load applied to the shell is PRESS, the external pressure.
y         $ Any more lines in the "help" paragraph?
BUCKL is the factor by which PRESS should be multiplied
y         $ Any more lines in the "help" paragraph?
in order to get the buckling pressure. For buckling, the
y         $ Any more lines in the "help" paragraph?
allowable, BUCKLA, should ordinarily be set to unity,
y         $ Any more lines in the "help" paragraph?
because you will be asked to provide a factor of safety
y         $ Any more lines in the "help" paragraph?
next. It is the factor of safety that compensates for
y         $ Any more lines in the "help" paragraph?
initial imperfections, not the allowable.
n         $ Any more lines in the "help" paragraph?
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
BUCKLF    $ Name of a variable in the users program (defined below)
      6  $ Role of the variable in the users program
factor of safety for buckling
y         $ Do you want to include a "help" paragraph?
In order to compensate for initial imperfections,
y         $ Any more lines in the "help" paragraph?
use a factor of safety for buckling of between 1.4 and 3.0.
y         $ Any more lines in the "help" paragraph?
If buckling occurs in the spherical dome, then a factor of
y         $ Any more lines in the "help" paragraph?
safety of 3.0 is appropriate. If buckling occurs in the
y         $ Any more lines in the "help" paragraph?
cylindrical shell, then a factor of safety of 1.4 or
y         $ Any more lines in the "help" paragraph?
something like that is appropriate. Unfortunately, in
y         $ Any more lines in the "help" paragraph?
this case the factor of safety depends on which part of
y         $ Any more lines in the "help" paragraph?
the tank buckles. For this submarine, which is to operate
y         $ Any more lines in the "help" paragraph?
at extreme depths, buckling may well not be critical. It
y         $ Any more lines in the "help" paragraph?
may well be that the stress constraint becomes active
y         $ Any more lines in the "help" paragraph?
whereas the buckling constraint remains inactive. You
y         $ Any more lines in the "help" paragraph?
may have to optimize with one factor of safety, see where
y         $ Any more lines in the "help" paragraph?
the optimized tank buckles, then re-optimize using a
y         $ Any more lines in the "help" paragraph?
different factor of safety.
n         $ Any more lines in the "help" paragraph?
      2  $ Indicator (1 or 2 or 3) for type of constraint
y         $ Any more variables for role type  4 ?           $165
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
STRESS    $ Name of a variable in the users program (defined below)
      4  $ Role of the variable in the users program
```

Table 5, p.7 of 7

```
y         $ Do you want to reset the number of columns in STRESS ?
      2  $ Number of dimensions in the array,  STRESS
shell segment number
     10  $ Max. allowable number of columns NCOLS in the array, STRESS
effective stress in shell segment
y         $ Do you want to include a "help" paragraph?
STRESS(ILOADX,ISEG)=SQRT(sig1^2 +sig2^2 -sig1sig2 +3sig12^2)
y         $ Any more lines in the "help" paragraph?
in which ILOADX=load set no. & ISEG=shell segment no.
n         $ Any more lines in the "help" paragraph?
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
STRESSA   $ Name of a variable in the users program (defined below)
      5  $ Role of the variable in the users program
maximum allowable effective stress
y         $ Do you want to include a "help" paragraph?
It is safest to use the proportional limit of the
y         $ Any more lines in the "help" paragraph?
material as the allowable. For example, with titanium the
y         $ Any more lines in the "help" paragraph?
elastic limit is about 120000 psi.
n         $ Any more lines in the "help" paragraph?
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
STRESSF   $ Name of a variable in the users program (defined below)
      6  $ Role of the variable in the users program
factor of safety for stress
y         $ Do you want to include a "help" paragraph?
Note that in this formulation the only bending stresses
y         $ Any more lines in the "help" paragraph?
are located in the neighborhood of the junction between
y         $ Any more lines in the "help" paragraph?
the spherical dome and the cylindrical shell. There are
y         $ Any more lines in the "help" paragraph?
no bending stresses due to imperfections. Therefore, it
y         $ Any more lines in the "help" paragraph?
might be a good idea to set the factor of safety for
y         $ Any more lines in the "help" paragraph?
the effective stress, STRESS, to some value greater than
y         $ Any more lines in the "help" paragraph?
unity, such as 1.1 to 1.3.
n         $ Any more lines in the "help" paragraph?
      3  $ Indicator (1 or 2 or 3) for type of constraint
n         $ Any more variables for role type  4 ?            $
      0  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
Next, establish the objective. In this case the objective
y         $ Are there more lines in the "help" paragraph?
is the weight of the part of the tank that is included
y         $ Are there more lines in the "help" paragraph?
in the BIGBOSOR4 model, that is, the weight of half of
y         $ Are there more lines in the "help" paragraph?
the tank. The objective is to be called WEIGHT
n         $ Are there more lines in the "help" paragraph?
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
WEIGHT    $ Name of a variable in the users program (defined below)
      7  $ Role of the variable in the users program
weight of the BIGBOSOR4 model
y         $ Do you want to include a "help" paragraph?
In the BIGBOSOR4 model half of the tank is to be included
y         $ Any more lines in the "help" paragraph?
Therefore, WEIGHT is the weight of half of the tank. Since
y         $ Any more lines in the "help" paragraph?
we are using weight density rather than mass density, the
y         $ Any more lines in the "help" paragraph?
objective is in lb at the surface of the earth.
n         $ Any more lines in the "help" paragraph?
```

end of submarine.INP file
(input for GENTEXT)

Table 6 Part of the submarine.DEF file

```
C===============================================================
C      GLOSSARY OF VARIABLES USED IN "submarine"
C===============================================================
C  ARRAY  NUMBER OF         PROMPT
C    ?   (ROWS,COLS)  ROLE  NUMBER   NAME                 DEFINITION OF VARIABLE
C                          (submarine.PRO)
C===============================================================
C    n   (   0,    0)    2       10   RADIUS   = radius of the tank
C    n   (   0,    0)    2       15   LENGTH   = total length of the cylinder
C    n   (   0,    0)    2       20   EMOD     = elastic modulus of the shell wall
C    n   (   0,    0)    2       25   NU       = Poisson ratio of the shell wall
C    n   (   0,    0)    2       30   DENSTY   = weight density of the shell wall
C    n   (   0,    0)    2       40   NODSPH   = number of nodal points in the dome
C    n   (   0,    0)    2       45   NODCYL   = number of nodal points in the cylinder
C    n   (   0,    0)    2       55   NPTSPH   = number of axial callouts in the dome
C    n   (   0,    0)    2       60   IZSPH    = number of axial callouts in the dome in ZSPH(IZSP»
H)
C    y   (  15,    0)    2       65   ZSPH     = axial location of callout in the dome
C    n   (   0,    0)    2       70   NPTCYL   = Number of axial callouts in the cylinder
C    n   (   0,    0)    2       75   IZCYL    = number of axial callouts in the cylinder in ZCYL(»
IZCYL)
C    y   (  15,    0)    2       80   ZCYL     = axial location of callout in the cylinder
C    n   (   0,    0)    2       90   IZREFSP  = axial callout number in the dome in ZREFSP(IZREFS»
P)
C    y   (  15,    0)    1       95   ZREFSP   = location of ref. surf. in the dome
C    y   (  15,    0)    1      100   THKSPH   = wall thickness in the dome
C    n   (   0,    0)    2      105   IZREFCY  = callout number in the cylinder in ZREFCY(IZREFCY)
C    y   (  15,    0)    1      110   ZREFCY   = location of the ref. surf. in the cylinder
C    y   (  15,    0)    1      115   THKCYL   = thickness of the cylindrical shell
C    n   (   0,    0)    2      125   NBUKLO   = low end of range of buckling circ. waves
C    n   (   0,    0)    2      130   NBUKHI   = high end of range of buckling circ. waves
C    n   (   0,    0)    2      140   NCASES   = Number of load cases (number of environments)  in»
 PRESS(NCASES)
C    y   (  20,    0)    3      145   PRESS    = uniform external pressure
C    y   (  20,    0)    4      155   BUCKL    = tank buckling eigenvalue
C    y   (  20,    0)    5      160   BUCKLA   = allowable buckling load (Use 1.0)
C    y   (  20,    0)    6      165   BUCKLF   = factor of safety for buckling
C    n   (   0,    0)    2      170   JSTRESS  = shell segment number in STRESS(NCASES,JSTRESS)
C    y   (  20,   10)    4      175   STRESS   = effective stress in shell segment
C    y   (  20,   10)    5      180   STRESSA  = maximum allowable effective stress
C    y   (  20,   10)    6      185   STRESSF  = factor of safety for stress
C    n   (   0,    0)    7      195   WEIGHT   = weight of the BIGBOSOR4 model
C===============================================================
```

output from GENTEXT

Table 7 (4 pages) submarine.PRO file.

Output from GENTEXT

The "end user" is prompted and "helped" by this file.

```
 5.0
        This application of GENOPT/BIGBOSOR4 is for an
        externally pressurized tank which has a spherical
        dome connected to a cylindrical shell. The material is
        elastic. The tank is subjected to uniform external
        pressure. Design constraints include stress and buckling.
        Symmetry conditions are applied at the midlength of the
        cylindrical shell. The objective is minimum weight.
        The main aim of the optimization is to determine the
        best distribution of the material in the neighborhood
        of the junction between the spherical dome and the
        cylindrical shell. Hence the position of the reference
        surface and the wall thickness vary in the neighborhood
        of the junction between the spherical dome and the
        cylindrical shell. In the BIGBOSOR4 model the tank
        consists of two shell segments. Segment 1 is the
        spherical dome, and Segment 2 is the cylindrical
        shell. Nodal points are concentrated in the region near
        the junction between dome and cylinder. The INDIC=1
        branch of BIGBOSOR4 is used for the buckling analysis,
        and the INDIC = 0 branch of BIGBOSOR4 is used for the
        stress analysis.

10.1 radius of the tank: RADIUS
10.2
        The radius is measured to the reference surface.

15.1 total length of the cylinder: LENGTH
15.2
        This is twice the length used in the analysis.
        It is the length of cylindrical shell between the two
        end domes. In the BIGBOSOR4 model half this length is
        included, with symmetry conditions imposed at the end
        of the cylindrical shell which is at the symmetry plane
        of the tank.

20.1 elastic modulus of the shell wall: EMOD
25.1 Poisson ratio of the shell wall: NU
30.1 weight density of the shell wall: DENSTY
30.2
        For example, aluminum has a weight density of 0.1 lb/in^3.


35.0
        Next, you will be asked to provide the number of
        nodal points to be used in the dome and the number of
        nodal points to be used in the cylinder. Choose odd
        integers between 51 and 91 . You might optimize first
        with the use of 51 nodal points in each shell segment
        (51 nodal points in the spherical dome and 51 nodal
        points in the cylindrical shell), then check your optimum
        design with the use of 91 nodal points in each shell
        segment.

40.1 number of nodal points in the dome: NODSPH
45.1 number of nodal points in the cylinder: NODCYL

50.0
        You will next be asked to provide the number of callout
        stations in the dome and the locations of those
        stations in the dome, followed by the number of callout
        stations in the cylinder, followed by the axial locations
        of those stations in the cylinder at which two quantities
        are specified:
        1. the distance from the leftmost surface of the shell
        wall to the reference surface ("leftmost" means the
        left-hand surface of the shell wall as you face in
        the direction of increasing arc length).
        2. the thickness of the shell wall.
        The locations of the reference surface and the wall thicknesses
        are to be decision variables in the optimization problem.

55.1 number of axial callouts in the dome: NPTSPH
55.2
        Use a number between 5 and 10, probably 5 at first.

60.1 Number IZSPH   of rows in the array  ZSPH: IZSPH
```

Table 7 (p.2 of 4)

```
 65.1 axial location of callout in the dome: ZSPH
 65.2
      The axial callout is the axial distance from the apex
      of the dome (where the dome meets the axis of revolution
      of the tank) to the callout point. NOTE: The first callout
      point must be ZSPH(1) = 0.0 (the location of the apex), and
      the last callout point must be the equator of the dome
      (ZSPH(NPTSPH) = RADIUS). You will want the thickness of the
      dome to be constant for most of the arc length of the dome,
      that is, from the dome apex to ZSPH = RADIUS/2, for
      example. ZSPH = RADIUS/2 corresponds to an angle of 60
      degrees from the apex to the second axial callout point.
      Then you provide axial locations of additional callout
      points located on the dome reference surface between
      that second callout point at ZSPH(2) = RADIUS/2 to the
      NPTSPHth callout point, which must correspond to the
      equator of the dome, that is, ZSPH(NPTSPH) = RADIUS.

 70.1 Number of axial callouts in the cylinder: NPTCYL
 70.2
      Use an integer between 5 and 10; probably 5 is best to
      start with. Then you can use more in a re-optimization
      if you wish.

 75.1 Number IZCYL   of rows in the array  ZCYL: IZCYL
 80.1 axial location of callout in the cylinder: ZCYL
 80.2
      The axial location is measured from the apex of the dome
      to the callout point. As callout points in the cylindrical
      shell you must include the first point in the cylindrical
      shell, that is, ZCYL(1) = RADIUS, and the last point in the
      cylindrical shell, that is, ZCYL(NPTCYL) = RADIUS + LENGTH/2.
      You want most of the length of the cylindrical shell to
      have a constant thickness. Therefore, the second-to-last
      callout point should probably be located at something like
      ZCYL(NPTCYL-1) = RADIUS + LENGTH/2 - LENGTH/4.


 85.0
      Next, you will be asked to provide the following decision
      variables:
      ZREFSP = distance from the leftmost shell wall surface to
      the reference surface in the spherical shell.
      THKSPH = thickness of the shell wall at the callout points
      in the spherical shell
      ZREFCY = distance from the leftmost shell wall surface to
      the reference surface in the cylindrical shell
      THKCYL = thickness of the shell wall at the callout points
      in the cylindrical shell.

 90.1 Number IZREFSP of rows in the array  ZREFSP: IZREFSP
 95.1 location of ref. surf. in the dome: ZREFSP
 95.2
      This is the distance from the shell wall leftmost surface to
      the reference surface as you face in the direction of
      increasing arc length in the spherical dome.

100.1 wall thickness in the dome: THKSPH
105.1 Number IZREFCY of rows in the array  ZREFCY: IZREFCY
110.1 location of the ref. surf. in the cylinder: ZREFCY
110.2
      This is the distance from the leftmost shell wall surface
      to the reference surface. "Leftmost" means the shell wall]
      surface on the left-hand side as you face in the direction
      of increasing arc length along the shell segment.

115.1 thickness of the cylindrical shell: THKCYL

120.0
      Next you will be asked to provide the range of
      circumferential wave numbers to be used during
      the buckling analysis. You will be asked to provide
      NBUKLO and NBUKHI, where NBUKLO is the low end of the
      range of circumferential wave numbers and NBUKHI is the
      high end of the range of circumferential wave numbers.
      It is best to use NBUKL0 = 0 and NBUKHI = something like
      10
```

Table 7 (p. 3 of 4)

```
125.1 low end of range of buckling circ. waves: NBUKLO
125.2
      Usually you will want to set NBUKLO = 0

130.1 high end of range of buckling circ. waves: NBUKHI
130.2
      Use an integer less than 20   . NBUKHI = 10 will probably
      suffice.


135.0
      In this formulation the only environmental parameter (load)
      is the external pressure. Use a positive number.

140.1 Number NCASES  of load cases (environments): NCASES
145.1 uniform external pressure: PRESS
145.2
      For the deepest part of the ocean (35000 feet) the
      pressure is (64 x 35000)/144 psi.


150.0
      In this formulation there are two "behavioral" design
      constraints: 1 = buckling and 2 = stress. The buckling
      variable is called BUCKL and the stress variable is
      called STRESS. By STRESS is meant, in this case that
      involves only isotropic material, "effective stress",
      which is the VonMises effective stress:
      STRESS = SQRT(sig1^2 +sig2^2 -sig1sig2 +3*sig12^2)

155.0 tank buckling eigenvalue: BUCKL
160.1 allowable buckling load (Use 1.0): BUCKLA
160.2
      The buckling eigenvalue is a buckling load factor. The
      load applied to the shell is PRESS, the external pressure.
      BUCKL is the factor by which PRESS should be multiplied
      in order to get the buckling pressure. For buckling, the
      allowable, BUCKLA, should ordinarily be set to unity,
      because you will be asked to provide a factor of safety
      next. It is the factor of safety that compensates for
      initial imperfections, not the allowable.

165.1 factor of safety for buckling: BUCKLF
165.2
      In order to compensate for initial imperfections,
      use a factor of safety for buckling of between 1.4 and 3.0.
      If buckling occurs in the spherical dome, then a factor of
      safety of 3.0 is appropriate. If buckling occurs in the
      cylindrical shell, then a factor of safety of 1.4 or
      something like that is appropriate. Unfortunately, in
      this case the factor of safety depends on which part of
      the tank buckles. For this submarine, which is to operate
      at extreme depths, buckling may well not be critical. It
      may well be that the stress constraint becomes active
      whereas the buckling constraint remains inactive. You
      may have to optimize with one factor of safety, see where
      the optimized tank buckles, then re-optimize using a
      different factor of safety.

170.1 Number JSTRESS of columns in the array, STRESS: JSTRESS
175.0 effective stress in shell segment: STRESS
175.2
      STRESS(ILOADX,ISEG)=SQRT(sig1^2 +sig2^2 -sig1sig2 +3sig12^2)
      in which ILOADX=load set no. & ISEG=shell segment no.

180.1 maximum allowable effective stress: STRESSA
180.2
      It is safest to use the proportional limit of the
      material as the allowable. For example, with titanium the
      elastic limit is about 120000 psi.

185.1 factor of safety for stress: STRESSF
185.2
      Note that in this formulation the only bending stresses
      are located in the neighborhood of the junction between
      the spherical dome and the cylindrical shell. There are
      no bending stresses due to imperfections. Therefore, it
      might be a good idea to set the factor of safety for
```

Table 7 (p. 4 of 4)

```
      the effective stress, STRESS, to some value greater than
      unity, such as 1.1 to 1.3.


190.0
      Next, establish the objective. In this case the objective
      is the weight of the part of the tank that is included
      in the BIGBOSOR4 model, that is, the weight of half of
      the tank. The objective is to be called WEIGHT

195.0 weight of the BIGBOSOR4 model: WEIGHT
195.2
      In the BIGBOSOR4 model half of the tank is to be included
      Therefore, WEIGHT is the weight of half of the tank. Since
      we are using weight density rather than mass density, the
      objective is in lb at the surface of the earth.

999.0 DUMMY ENTRY TO MARK END OF FILE
```

end of submarine.PRO file,
the file for prompting the "end user".

Output from GENTEXT

Table 8 (5 pages) Subroutine STRUCT, etc.

```
C=DECK      STRUCT
      SUBROUTINE STRUCT(IMODX,CONSTX,OBJGEN,CONMAX,NCONSX,IPOINC,
     1 PCWORD,CPLOTX,ILOADX,ISTARX,NUSERC,IBEHV,IDV,IFAST,JJJ1)
C
C  PURPOSE IS TO PERFORM THE ANALYSIS FOR A GIVEN DESIGN AND LOADING.
C  CONSTRAINT CONDITIONS ARE ALSO GENERATED.
C
C  Common blocks already present in the struct.tmpl file, that is,
C  in the "skeletal" file possibly to be augmented by the user:
      COMMON/PRMFIL/IFILEX,IFILE2,IOUT,IPRM(5)
      COMMON/PRMOUT/IFILE3,IFILE4,IFILE8,IFILE9,IFIL11
      COMMON/INDAT/INFILE
      COMMON/LWRUPR/VLBX(50),VUBX(50),CLINKX(50,5),VLINKX(50),VBVX(99)
      COMMON/NUMPAR/IPARX,IVARX,IALLOW,ICONSX,NDECX,NLINKX,NESCAP,ITYPEX
      COMMON/PARAMS/PARX(99),VARX(50),ALLOWX(99),CONSXX(99),DECX(50),
     1              ESCX(50)
      COMMON/WORDS1/WORDPX(99),WORDVX(50),WORDAX(99),WORDCC(99),
     1              WORDDX(50)
      COMMON/WORDS2/WORDLX(50),WORDEX(50),WORDIQ(20)
      COMMON/OPTVAR/IDVX(50),ILVX(50),IDLINK(50,5),IEVX(50),JTERMS(20)
      COMMON/NUMPR2/ILARX,ICARX,IOARX,IFLATX,NCASES,NPRINX
      COMMON/PARAM2/FLARX(50),CARX(99),OARX(50),FSAFEX(99),CPWRX(50,5)
      COMMON/PARAM3/CINEQX(15,20),DPWREQ(15,20)
      COMMON/PARAM4/IDINEQ(15,20),NINEQX,JINEQX(20),IEQTYP(20)
      COMMON/WORDS3/WORDFX(50),WORDBX(99),WORDOB(50),WORDSX(99)
      COMMON/WORDS4/WORDMX(99)
      COMMON/PWORD/PHRASE
      COMMON/PWORD2/IBLANK
      COMMON/ISKIPX/ISKIP(30)
      DIMENSION IBEHV(99)
C
C=========================================================================
C  Start of first part of STRUCT written by "GENTEXT"
C  INSERT ADDITIONAL COMMON BLOCKS HERE: (THESE ARE "GENTEXT" VARIABLES)
      COMMON/FV06/ZSPH(15),IZSPH
      REAL ZSPH
      COMMON/FV07/ZCYL(15),IZCYL
      REAL ZCYL
      COMMON/FV08/ZREFSP(15),IZREFSP
      REAL ZREFSP
      COMMON/FV09/THKSPH(15)
      REAL THKSPH
      COMMON/FV10/ZREFCY(15),IZREFCY
      REAL ZREFCY
      COMMON/FV11/THKCYL(15)
      REAL THKCYL
      COMMON/FV12/PRESS(20)
      REAL PRESS
      COMMON/FV15/BUCKL(20),BUCKLA(20),BUCKLF(20)
      REAL BUCKL,BUCKLA,BUCKLF
      COMMON/FV18/STRESS(20,10),JSTRESS,STRESSA(20,10),STRESSF(20,10)
      REAL STRESS,STRESSA,STRESSF
      COMMON/IV01/NODSPH,NODCYL,NPTSPH,NPTCYL,NBUKLO,NBUKHI
      INTEGER NODSPH,NODCYL,NPTSPH,NPTCYL,NBUKLO,NBUKHI
      COMMON/FV01/RADIUS,LENGTH,EMOD,NU,DENSTY,WEIGHT
      REAL RADIUS,LENGTH,EMOD,NU,DENSTY,WEIGHT
C
C
      CHARACTER*80 PHRASE,CODPHR,PCWORD
      CHARACTER*80 WORDPX,WORDVX,WORDAX,WORDCX,WORDDX,WORDLX,WORDEX
      CHARACTER*80 WORDFX,WORDBX,WORDOB,WORDSX,WORDMX,WORDCC,WORDIQ
c     CHARACTER*4 ANSOUT,CHARAC,ANSWER
      CHARACTER*2 CIX
      character*2 CJX
      CHARACTER*13 CODNAM
c     DIMENSION ISUBX(100)
c     LOGICAL ANSL1
C
      DIMENSION CONSTX(*),IPOINC(*),PCWORD(*),CPLOTX(*)
C  End of first part of STRUCT written by "GENTEXT"
C=========================================================================
C
C  INSERT ADDITIONAL DIMENSION AND/OR LABELLED COMMON BLOCKS HERE,
C  IF NECESSARY. THESE WOULD BE STATEMENTS THAT ARE CONSISTENT WITH
C  SUBROUTINES THAT YOU OR OTHERS MAY HAVE WRITTEN THAT ARE REQUIRED
C  FOR WHATEVER ANALYSIS YOU ARE PERSUING.  MAKE SURE THAT YOU DO NOT
C  INTRODUCE NAME CONFLICTS WITH THE "GENTEXT" LABELLED COMMON BLOCKS
C  LISTED ABOVE.
```

This FORTRAN coding is almost entirely produced automatically by GENTEXT. The GENOPT user added only 3 lines!

Table 8 (p.2 of 5)

```
C
C  Please note that you do not have to modify STRUCT.NEW if you would
C  rather provide all of your algorithms via the BEHAVIOR.NEW library.
C  (See instructions in BEHAVIOR.NEW).
C
C  If you are using a lot of software previously written either by
C  yourself or others, or if there are a lot of behavioral constraints
C  that are best generated by looping over array indices (such as
C  occurs, for example, with stress constraints in laminates of
C  composite materials), then it may be best to insert your common
C  blocks and dimension statements here, your subroutine calls
C  below (where indicated), and your subroutines in any of the libraries
C  called ADDCODEn.NEW, n = 1,2,...,5.  Please note that you
C  may also have to add statements to SUBROUTINE TRANFR, the
C  purpose of which is described below (in TRANFR).
C
C  The several test cases provided with GENOPT demonstrate different
C  methods:
C
C  PLATE   : leave STRUCT.NEW unchanged and modify BEHAVIOR.NEW
C  SPHERE  : leave STRUCT.NEW unchanged and modify BEHAVIOR.NEW
C  TORISPH: leave BEHAVIOR.NEW unchanged except possibly for the objective
C           function (SUBROUTINE OBJECT), modify STRUCT.NEW,
C           possibly add a subroutine library called ADDCODE1.NEW, and
C           possibly augment the usermake.linux file to collect object
C           libraries from other directories. In the "TORISPH" case
C           BEHAVIOR.NEW remains unchanged, no ADDCODE1.NEW library is
C           added, and usermake.linux is not changed.  Instead, the
C           BIGBOSOR4 code is added and SUBROUTINE BOSDEC is written
C           by the genopt user. The BIGBOSOR4 code and SUBROUTINE
C           BOSDEC must be stored in /home/progs/bosdec/sources, as
C           follows:
C    BIGBOSOR4 code:
C     -rw-r--r--  1 bush bush 579671 Feb 29 07:19 addbosor4.src
C     -rw-r--r--  1 bush bush  83175 Feb 22 09:13 b4plot.src
C     -rw-r--r--  1 bush bush  89671 Feb 28 16:20 b4util.src
C     -rw-r--r--  1 bush bush  22723 Feb 10 14:27 bio.c
C     -rw-r--r--  1 bush bush  31175 Feb 10 14:27 bio_linux.c
C     -rw-r--r--  1 bush bush  37152 Feb 10 14:27 bio_linux.o
C     -rw-r--r--  1 bush bush  15650 Feb 10 14:26 gasp.F
C     -rw-r--r--  1 bush bush  18364 Feb 10 14:26 gasp_linux.o
C     -rw-r--r--  1 bush bush   6310 Feb 13 10:12 opngen.src
C     -rw-r--r--  1 bush bush  22440 Feb 10 14:25 prompter.src
C     -rw-r--r--  1 bush bush  13426 Feb 22 09:14 resetup.src
C    BOSDEC.src code:
C     -rw-r--r--  1 bush bush  33851 Mar  1 08:34 bosdec.src
C
C  WAVYCYL: both BEHAVIOR.NEW and STRUCT.NEW are both changed. Otherwise
C           the activity is the same as that described for TORISPH,
C           except, of course, that struct.new is different from
C           that used in connection with TORISPH.
C
C  CYLINDER:same as the description for WAVYCYL.
C
C
C  INSERT YOUR ADDITIONAL COMMON BLOCKS FOR THIS GENERIC CASE HERE:
C
C
C  THE FOLLOWING CODE WAS WRITTEN BY "GENTEXT":
C
C=================================================================
C  Start the second portion of STRUCT written by "GENTEXT":
C
      ICARX   = ISTARX
      INUMTT = 0
      ICONSX = 0
      KCONX   = 0
      IF (IMODX.EQ.0) THEN
         CALL MOVERX(0.,0,CONSTX,1,99)
         CALL MOVERX(0, 0,IPOINC,1,1500)
      ENDIF
C
      IF (ILOADX.EQ.1) THEN
C
C  ESTABLISH FIRST ANY CONSTRAINTS THAT ARE INEQUALITY RELATIONSHIPS
C  AMONG THE VARIABLES IN THE ARRAY VARX(*) (THAT IS, VARIABLES THAT
C  ARE EITHER DECISION VARIABLES, LINKED VARIABLES, ESCAPE VARIABLES,
C  OR CANDIDATES FOR ANY OF THESE TYPES OF VARIABLES.
```

Table 8 (p.3-15)

```
C
         IF (NINEQX.GT.0)
     1        CALL VARCON(WORDIQ,WORDMX,CINEQX,DPWREQ,IDINEQ,
     1        NINEQX,JINEQX,IEQTYP,INUMTT,IMODX,CONMAX,IPOINC,
     1        ICONSX,CONSTX,VARX,PCWORD,CPLOTX,ICARX)
C
C  NEXT, ESTABLISH USER-WRITTEN CONSTRAINTS. AT PRESENT, THE PROGRAM
C  ALLOWS ONLY ONE USER-WRITTEN CONSTRAINT. HOWEVER, THE USER CAN
C  EASILY EXPAND THIS CAPABILITY SIMPLY BY ADDING SUBROUTINES THAT
C  ARE ANALOGOUS TO USRCON (WITH NAMES SUCH AS USRCN2, USRCN3, ETC.
C  TO THE  BEHAVIOR.NEW LIBRARY, AND ADD CALLS TO THESE ADDITIONAL
C  SUBROUTINES FOLLOWING THE CALL TO USRCON IMMEDIATELY BELOW.
C
         CALL USRCON(INUMTT,IMODX,CONMAX,ICONSX,IPOINC,CONSTX,WORDCX,
     1            WORDMX,PCWORD,CPLOTX,ICARX,IFILE8)
C
         NUSERC = ICARX - NINEQX
      ENDIF
C
      IF (NPRINX.GT.0) THEN
         WRITE(IFILE8,'(1X,A,I2,A)')
     1 ' BEHAVIOR FOR ',ILOADX,' ENVIRONMENT (LOAD SET)'
         WRITE(IFILE8,'(A)')'  '
         WRITE(IFILE8,'(A)')
     1 ' CONSTRAINT  BEHAVIOR                DEFINITION'
         WRITE(IFILE8,'(A)')
     1 '   NUMBER     VALUE'
      ENDIF
C
      CALL CONVR2(ILOADX,CIX)
      IF (NPRINX.GT.0) THEN
         WRITE(IFILE8,'(1X,A)')'  '
         WRITE(IFILE8,'(1X,A,I2)')
     1 ' BEHAVIOR FOR LOAD SET NUMBER, ILOADX=',ILOADX
      ENDIF
C
C  End of the second portion of STRUCT written by "GENTEXT"
C=======================================================================
C
C  USER: YOU MAY WANT TO INSERT SUBROUTINE CALLS FROM SOFTWARE DEVELOPED
C        ELSEWHERE FOR ANY CALCULATIONS PERTAINING TO THIS LOAD SET.
C
      CALL OPNGEN
      CALL RWDGEN
```

added by the GENOPT User

```
C
C=======================================================================
C  Start of the final portion of STRUCT written by "GENTEXT"
C
C  INSERT THE PROGRAM FILE HERE:
C
C  Behavior and constraints generated next for BUCKL:
C  BUCKL = tank buckling eigenvalue
C
      PHRASE =
     1 'tank buckling eigenvalue'
      CALL BLANKX(PHRASE,IENDP4)
      IF (IBEHV(1   ).EQ.0) CALL BEHX1
     1 (IFILE8,NPRINX,IMODX,IFAST,ILOADX   ,
     1 'tank buckling eigenvalue')
```

buckling analysis performed by SUBROUTINE BEHX1 (see Table 9)

```
      IF (BUCKL(ILOADX   ).EQ.0.)  BUCKL(ILOADX   ) = 1.E+10
      IF (BUCKLA(ILOADX   ).EQ.0.)  BUCKLA(ILOADX   ) = 1.0
      IF (BUCKLF(ILOADX   ).EQ.0.)  BUCKLF(ILOADX   ) = 1.0
      KCONX = KCONX + 1
      CARX(KCONX) =BUCKL(ILOADX   )
      WORDCX= '(BUCKL('//CIX//')/BUCKLA('//CIX//
     1  ')) / BUCKLF('//CIX//')'
      CALL CONX(BUCKL(ILOADX   ),BUCKLA(ILOADX   ),BUCKLF(ILOADX   )
     1,'tank buckling eigenvalue',
     1 'allowable buckling load (Use 1.0)',
     1 'factor of safety for buckling',
     1 2,INUMTT,IMODX,CONMAX,ICONSX,IPOINC,CONSTX,WORDCX,
     1 WORDMX,PCWORD,CPLOTX,ICARX)
      IF (IMODX.EQ.0) THEN
         CODPHR =
     1 '  tank buckling eigenvalue: '
         IENDP4 =28
         CODNAM ='BUCKL('//CIX//')'
         MLET4 =5 + 4
```

Table 8 (p. 4 of 5)

```
         WORDBX(KCONX)= CODPHR(1:IENDP4)//CODNAM(1:MLET4)
         IF (NPRINX.GT.0) WRITE(IFILE8,'(I5,6X,G14.7,A,A)')
      1    KCONX,CARX(KCONX),CODPHR(1:IENDP4),CODNAM(1:MLET4)
      ENDIF
  165 CONTINUE
  166 CONTINUE
C
C  Behavior and constraints generated next for STRESS:
C  STRESS = effective stress in shell segment
C
      IF (JSTRESS.EQ.0) GO TO 186
      IF (NPRINX.GT.0) THEN
         IF (JSTRESS.GT.1) THEN
            WRITE(IFILE8,'(1X,A)')'  '
            WRITE(IFILE8,'(1X,A,$)')' BEHAVIOR OVER J = '
            WRITE(IFILE8,'(1X,A)')
     1      'shell segment number'
         ENDIF
      ENDIF
      DO 185  J=1,JSTRESS
      CALL CONVR2(J,CJX)
      PHRASE =
     1 'effective stress in shell segment'
      CALL BLANKX(PHRASE,IENDP4)
      IF (IBEHV(2  ).EQ.0) CALL BEHX2
     1 (IFILE8,NPRINX,IMODX,IFAST,ILOADX,J,
     1 'effective stress in shell segment')
```

stress analysis performed by SUBROUTINE BEHX2 (See Table 9)

```
      IF (STRESS(ILOADX,J).EQ.0.)  STRESS(ILOADX,J) = 1.E-10
      IF (STRESSA(ILOADX,J).EQ.0.)  STRESSA(ILOADX,J) = 1.0
      IF (STRESSF(ILOADX,J).EQ.0.)  STRESSF(ILOADX,J) = 1.0
      KCONX = KCONX + 1
      CARX(KCONX) =STRESS(ILOADX,J)
      WORDCX= '(STRESSA('//CIX//','//CJX//')/STRESS('//CIX//','//CJX//
     1  ')) / STRESSF('//CIX//','//CJX//')'
      CALL CONX(STRESS(ILOADX,J),STRESSA(ILOADX,J),STRESSF(ILOADX,J)
     1,'effective stress in shell segment',
     1 'maximum allowable effective stress',
     1 'factor of safety for stress',
     1 3,INUMTT,IMODX,CONMAX,ICONSX,IPOINC,CONSTX,WORDCX,
     1 WORDMX,PCWORD,CPLOTX,ICARX)
      IF (IMODX.EQ.0) THEN
         CODPHR =
     1 '  effective stress in shell segment: '
         IENDP4 =37
         CODNAM ='STRESS('//CIX//','//CJX//')'
         MLET4 =6 + 7
         WORDBX(KCONX)= CODPHR(1:IENDP4)//CODNAM(1:MLET4)
         IF (NPRINX.GT.0) WRITE(IFILE8,'(I5,6X,G14.7,A,A)')
     1    KCONX,CARX(KCONX),CODPHR(1:IENDP4),CODNAM(1:MLET4)
      ENDIF
  185 CONTINUE
  186 CONTINUE
C
C  NEXT, EVALUATE THE OBJECTIVE, OBJGEN:
      IF (ILOADX.EQ.1) THEN
         PHRASE ='weight of the BIGBOSOR4 model'
         CALL BLANKX(PHRASE,IENDP4)
         CALL OBJECT(IFILE8,NPRINX,IMODX,OBJGEN,
     1    'weight of the BIGBOSOR4 model')
      ENDIF
      NCONSX = ICONSX
C
      CALL CLSGEN
```

added by the GENOPT user.

```
C
      RETURN
      END
C
C
C
C
C
C  End of the final portion of STRUCT written by "GENTEXT"
C=====================================================================
C
C=DECK      TRANFR
      SUBROUTINE TRANFR(ARG1,ARG2,ARG3,ARG4,ARG5)
C
C  USER:  DO NOT FORGET TO MODIFY THE ARGUMENT LIST OF TRANFR AS
```

Table 8 (p.5 of 5)

```
C         APPROPRIATE FOR YOUR CASE!
C
C  PURPOSE IS TO TRANSFER DATA FROM THE LABELLED COMMON BLOCKS
C  SET UP BY THE GENOPT CODE TO LABELLED COMMON OR ARGUMENTS IN
C  THE SUBROUTINE ARGUMENT LIST THAT MATCH PREVIOUSLY WRITTEN CODE
C  BY YOURSELF OR OTHER PROGRAM DEVELOPERS.  THE USER SHOULD ESTABLISH
C  THE ARGUMENT LIST AND/OR LABELLED COMMON BLOCKS THAT MATCH VARIABLES
C  IN THE PREVIOUSLY WRITTEN CODE.  FOR AN EXAMPLE, SEE THE DISCUSSION
C  OF THE CASE CALLED "PANEL".
C
C==========================================================================
C  Start of part of TRANFR written by "GENTEXT"
C  INSERT ADDITIONAL COMMON BLOCKS HERE: (THESE ARE "GENTEXT" VARIABLES)
      COMMON/FV06/ZSPH(15),IZSPH
      REAL ZSPH
      COMMON/FV07/ZCYL(15),IZCYL
      REAL ZCYL
      COMMON/FV08/ZREFSP(15),IZREFSP
      REAL ZREFSP
      COMMON/FV09/THKSPH(15)
      REAL THKSPH
      COMMON/FV10/ZREFCY(15),IZREFCY
      REAL ZREFCY
      COMMON/FV11/THKCYL(15)
      REAL THKCYL
      COMMON/FV12/PRESS(20)
      REAL PRESS
      COMMON/FV15/BUCKL(20),BUCKLA(20),BUCKLF(20)
      REAL BUCKL,BUCKLA,BUCKLF
      COMMON/FV18/STRESS(20,10),JSTRESS,STRESSA(20,10),STRESSF(20,10)
      REAL STRESS,STRESSA,STRESSF
      COMMON/IV01/NODSPH,NODCYL,NPTSPH,NPTCYL,NBUKLO,NBUKHI
      INTEGER NODSPH,NODCYL,NPTSPH,NPTCYL,NBUKLO,NBUKHI
      COMMON/FV01/RADIUS,LENGTH,EMOD,NU,DENSTY,WEIGHT
      REAL RADIUS,LENGTH,EMOD,NU,DENSTY,WEIGHT
C
C
C  End of part of TRANFR written by "GENTEXT"
C==========================================================================
C  INSERT ADDITIONAL DIMENSION AND/OR LABELLED COMMON BLOCKS HERE,
C  IF NECESSARY. THESE WOULD BE STATEMENTS THAT ARE CONSISTENT WITH
C  SUBROUTINES THAT YOU OR OTHERS MAY HAVE WRITTEN THAT ARE REQUIRED
C  FOR WHATEVER ANALYSIS YOU ARE NOW PERSUING.  MAKE SURE THERE ARE
C  NO NAME CONFLICTS WITH THE "GENTEXT" LABELLED COMMON BLOCKS.
C
C
C  INSERT APPROPRIATE FORTRAN STATEMENTS HERE (DON'T FORGET TO CORRECT
C  THE ARGUMENT LIST OF SUBROUTINE TRANFR!)
C  PROGRAM FILE:
C
C
      RETURN
      END
C
C
C
```

end of the "STRUCT" library

Table 9 (12 pages)   behavior. submarine

```
C=DECK      BEHAVIOR.NEW
C  This library contains the skeletons of
C  subroutines called SUBROUTINE BEHXn, n = 1,
C  2, 3, . . . that will yield predictions
C  of behavioral responses of various systems
C  to environments (loads).
C
C  You may complete the subroutines by writing
C  algorithms that yield the responses,
C  each of which plays a part in constraining
C  the design to a feasible region. Examples
C  of responses are: stress, buckling, drag,
C  vibration, deformation, clearances, etc.
C
C  A skeleton routine called SUBROUTINE OBJECT
C  is also provided for any objective function
C  (e.g. weight, deformation, conductivity)
C  you may wish to create.
C
C  A skeleton routine called SUBROUTINE USRCON
C  is also provided for any user-written
C  constraint condition you may wish to write:
C  This is an INEQUALITY condition that
C  involves any program variables.  However,
C  note that this kind of thing is done
C  automatically in the program DECIDE, so
C  try DECIDE first to see if your particular
C  constraint conditions can be accommodated
C  more easily there.
C
C  Please note that you do not have to modify
C  BEHAVIOR.NEW in any way, but may instead
C  prefer to insert your subroutines into the
C  skeletal libraries ADDCODEn.NEW, n=1,2,...
C  and appropriate common blocks, dimension
C  and type statements and calls to these
C  subroutines in the library STRUCT.NEW.
C  This strategy is best if your FORTRAN
C  input to GENOPT contains quite a bit
C  of software previously written by
C  yourself or others, and/or the generation
C  of behavioral constraints is more easily
C  accomplished via another architecture
C  than that provided for in the
C  BEHAVIOR.NEW library. (See instructions
C  in the libraries ADDCODEn.NEW and
C  STRUCT.NEW for this procedure.)
C
C  The two test cases provided with GENOPT
C  provide examples of each method:
C   PLATE (test case 1): use of BEHAVIOR.NEW
C   PANEL (test case 2): use of ADDCODEn.NEW
C                        and STRUCT.NEW.
C
C   SEVEN ROLES THAT VARIABLES IN THIS SYSTEM OF PROGRAMS PLAY
C
C    A variable can have one of the following roles:
C
C    1 = a possible decision variable for optimization,
C        typically a dimension of a structure.
C    2 = a constant parameter (cannot vary as design evolves),
C        typically a control integer or material property,
C        but not a load, allowable, or factor of safety,
C        which are asked for later.
C    3 = a parameter characterizing the environment, such
C        as a load component or a temperature.
C    4 = a quantity that describes the response of the
C        structure, (e.g. stress, buckling load, frequency)
C    5 = an allowable, such as maximum allowable stress,
C        minimum allowable frequency, etc.
C    6 = a factor of safety
C    7 = the quantity that is to be minimized or maximized,
C        called the "objective function" (e.g. weight).
C ==========================================================
C
C  NAMES, DEFINITIONS, AND ROLES OF THE VARIABLES:

C YOU ARE USING WHAT I HAVE CALLED "GENOPT" TO GENERATE AN
```

This is the version of the skeletal behavior.new library that has been "fleshed out" by the GENOPT user.

Table 9 (p. 2 of 12)

```
C OPTIMIZATION PROGRAM FOR A PARTICULAR CLASS OF PROBLEMS.
C THE NAME YOU HAVE CHOSEN FOR THIS CLASS OF PROBLEMS IS: submarine

C "GENOPT" (GENeral OPTimization) was written during 1987-1988
C by Dr. David Bushnell, Dept. 93-30, Bldg. 251, (415)424-3237
C    Lockheed Missiles and Space Co., 3251 Hanover St.,
C    Palo Alto, California, USA  94304

C The optimizer used in GENOPT is called ADS, and was
C written by G. Vanderplaats [3]. It is based on the method
C of feasible directions [4].

C                              ABSTRACT

C "GENOPT" has the following purposes and properties:
C     1. Any relatively simple analysis is "automatically"
C        converted into an optimization of whatever system
C        can be analyzed with fixed properties. Please note
C        that GENOPT is not intended to be used for problems
C        that require elaborate data-base management systems
C        or large numbers of degrees of freedom.

C     2. The optimization problems need not be in fields nor
C        jargon familiar to me, the developer of GENOPT.
C        Although all of the example cases (See the cases
C        in the directories under genopt/case)
C        are in the field of structural analysis, GENOPT is
C        not limited to that field.


C     3. GENOPT is a program that writes other programs. These
C        programs, WHEN AUGMENTED BY USER-SUPPLIED CODING,
C        form a program system that should be user-friendly in
C        the GENOPT-user"s field. In this instance the user
C        of GENOPT must later supply FORTRAN coding that
C        calculates behavior in the problem class called "submarine".

C     4. Input data and textual material are elicited from
C        the user of GENOPT in a general enough way so that
C        he or she may employ whatever data, definitions, and
C        "help" paragraphs will make subsequent use of the
C        program system thus generated easy by those less
C        familiar with the class of problems "submarine" than
C        the GENOPT user.

C     5. The program system generated by GENOPT has the same
C        general architecture as previous programs written for
C        specific applications by the developer [7 - 16]. That
C        is, the command set is:

C           BEGIN     (User supplies starting design, loads,
C                      control integers, material properties,
C                      etc. in an interactive-help mode.)

C           DECIDE    (User chooses decision and linked
C                      variables and inequality constraints
C                      that are not based on behavior.)

C           MAINSETUP (User chooses output option, whether
C                      to perform analysis of a fixed design
C                      or to optimize, and number of design
C                      iterations.)

C           OPTIMIZE  (The program system performs, in a batch
C                      mode, the work specified in MAINSETUP.)

C           SUPEROPT  (Program tries to find the GLOBAL optimum
C                      design as described in Ref.[11] listed
C                      below (Many OPTIMIZEs in one run.)

C           CHANGE    (User changes certain parameters)

C           CHOOSEPLOT (User selects which quantities to plot
C                       vs. design iterations.)

C           DIPLOT    (User generates plots)

C           CLEANSPEC (User cleans out unwanted files.)
```

```
C      A typical runstream is:
C        GENOPTLOG    (activate command set)
C        BEGIN        (provide starting design, loads, etc.)
C        DECIDE       (choose decision variables and bounds)
C        MAINSETUP    (choose print option and analysis type)
C        OPTIMIZE     (launch batch run for n design iterations)
C        OPTIMIZE     (launch batch run for n design iterations)
C        OPTIMIZE     (launch batch run for n design iterations)
C        OPTIMIZE     (launch batch run for n design iterations)
C        OPTIMIZE     (launch batch run for n design iterations)
C        CHANGE       (change some variables for new starting pt)
C        OPTIMIZE     (launch batch run for n design iterations)
C        OPTIMIZE     (launch batch run for n design iterations)
C        OPTIMIZE     (launch batch run for n design iterations)
C        OPTIMIZE     (launch batch run for n design iterations)
C        OPTIMIZE     (launch batch run for n design iterations)
C        CHOOSEPLOT   (choose which variables to plot)
C        DIPLOT       (plot variables v. iterations)
C        CHOOSEPLOT   (choose additional variables to plot)
C        DIPLOT       (plot more variables v design iterations)
C        CLEANSPEC    (delete extraneous files for specific case)

C   IMPORTANT:  YOU MUST ALWAYS GIVE THE COMMAND "OPTIMIZE"
C               SEVERAL TIMES IN SUCCESSION IN ORDER TO OBTAIN
C               CONVERGENCE! AN EXPLANATION OF WHY YOU MUST DO
C               THIS IS GIVEN ON P 580-582 OF THE PAPER "PANDA2,
C               PROGRAM FOR MINIMUM WEIGHT DESIGN OF STIFFENED,
C               COMPOSITE LOCALLY BUCKLED PANELS", Computers and
C               Structures, Vol. 25, No. 4, pp 469-605 (1987).

C Due to introduction of a "global" optimizer, SUPEROPT,
C described in Ref.[11], you can now use the runstream

C       BEGIN        (provide starting design, loads, etc.)
C       DECIDE       (choose decision variables and bounds)
C       MAINSETUP    (choose print option and analysis type)
C       SUPEROPT     (launch batch run for "global" optimization)
C       CHOOSEPLOT   (choose which variables to plot)
C       DIPLOT       (plot variables v. iterations)

C "Global" is in quotes because SUPEROPT does its best to find
C a true global optimum design. The user is strongly urged to
C execute SUPEROPT/CHOOSEPLOT several times in succession in
C order to determine an optimum that is essentially just as
C good as the theoretical true global optimum. Each execution
C of the series,
C       SUPEROPT
C       CHOOSEPLOT

C does the following:

C 1. SUPEROPT executes many sets of the two processors,
C    OPTIMIZE and AUTOCHANGE (AUTOCHANGE gets a new random
C    "starting" design), in which each set does the following:

C       OPTIMIZE          (perform k design iterations)
C       OPTIMIZE          (perform k design iterations)
C       OPTIMIZE          (perform k design iterations)
C       OPTIMIZE          (perform k design iterations)
C       OPTIMIZE          (perform k design iterations)
C       AUTOCHANGE        (get new starting design randomly)

C    SUPEROPT keeps repeating the above sequence until the
C    total number of design iterations reaches about 270.
C    The number of OPTIMIZEs per AUTOCHANGE is user-provided.

C 2. CHOOSEPLOT allows the user to plot stuff and resets the
C    total number of design iterations from SUPEROPT to zero.
C    After each execution of SUPEROPT the user MUST execute
C    CHOOSEPLOT: before the next execution of SUPEROPT the
C    total number of design iterations MUST be reset to zero.

C                         REFERENCES

C [1] Bushnell, D., "GENOPT--A program that writes
C user-friendly optimization code", International
C Journal of Solids and Structures, Vol. 26, No. 9/10,
```

Table 9, p. 4 of 12

```
C pp. 1173-1210, 1990. The same paper is contained in a
C bound volume of papers from the International Journal of
C Solids and Structures published in memory of Professor
C Charles D. Babcock, formerly with the California Institute
C of Technology.

C [2] Bushnell, D., "Automated optimum design of shells of
C revolution with application to ring-stiffened cylindrical
C shells with wavy walls", AIAA paper 2000-1663, 41st
C AIAA Structures Meeting, Atlanta, GA, April 2000. Also see
C Lockheed Martin report, same title, LMMS P525674, November
C 1999

C [2b] Bushnell, D., "Minimum weight design of imperfect
C isogrid-stiffened ellipsoidal shells under uniform external
C pressure", AIAA paper 2009-2702, 50th AIAA Structures
C Meeting, Palm Springs, CA, May 4-7, 2009

C [2c] Bushnell, D., "Use of GENOPT and a BIGBOSOR4 "huge"
C torus" model to optimize a typical weld land and weld land
C edge stringers in a previously optimized internally
C stiffened cylindrical shell without weld lands",
C unpublished report to NASA Langley Research, May 15, 2009

C [2d] Bushnell, D., "Use of GENOPT and BIGBOSOR4 to obtain
C optimum designs of a cylindrical shell with a composite
C truss-core sandwich wall", unpublished report to NASA
C Langley Research Center, Hampton, VA, June 20, 2009

C [2e] Bushnell, D., "Use of GENOPT and BIGBOSOR4 to obtain
C an optimum design of a deep submergence tank", unpublished
C report to the DOER company, Alameda, CA, June 30, 2009

C [3] Vanderplaats, G. N., "ADS--a FORTRAN program for
C automated design synthesis, Version 2.01", Engineering
C Design Optimization, Inc, Santa Barbara, CA, January, 1987

C [4] Vanderplaats, G. N. and Sugimoto, H., "A general-purpose
C optimization program for engineering design", Computers
C and Structures, Vol. 24, pp 13-21, 1986

C [5] Bushnell, D., "BOSOR4: Program for stress, stability,
C and vibration of complex, branched shells of revolution",
C in STRUCTURAL ANALYSIS SYSTEMS, Vol. 2, edited by A.
C Niku-Lari, pp. 25-54, (1986)

C [6] Bushnell, D., "BOSOR5: Program for buckling of complex,
C branched shells of revolution including large deflections,
C plasticity and creep," in STRUCTURAL ANALYSIS SYSTEMS, Vol.
C 2,  edited by A. Niku-Lari, pp. 55-67, (1986)

C [7] Bushnell, D., "PANDA2--program for minimum weight
C design of stiffened, composite, locally buckled panels",
C COMPUTERS AND STRUCTURES, vol. 25, No. 4, pp 469-605, 1987

C [8] Bushnell, D., "Improved optimum design of dewar
C supports", COMPUTERS and STRUCTURES, Vol. 29, No. 1,
C pp. 1-56 (1988)

C [9] Bushnell, D., "SPHERE - Program for minimum weight
C design of isogrid-stiffened spherical shells under uniform
C external pressure", Lockheed Report F372046, January, 1990

C [10] Bushnell, D.,"Optimum design of imperf.isogrid-stiffened
C ellipsoidal shells...", written and placed in the file
C ..genopt/case/torisph/sdm50.report.pdf

C [11] Bushnell, D., "Recent enhancements to PANDA2", AIAA
C paper 96-1337-CP, Proc. 37th AIAA SDM Meeting, April 1996
C pp. 126-182, in particular, pp. 127-130

C [12] Bushnell, D., the file ..genopt/doc/getting.started

C [13] Bushnell, D., the case ..genopt/case/torisph, Ref.[2b]

C [14] Bushnell, D., the case ..genopt/case/cylinder

C [15] Bushnell, D., the case ..genopt/case/wavycyl, Ref.[2]
```

Table 9, p. 5 of 12

```
C [16] Bushnell, D., the case ..genopt/case/plate

C [17] Bushnell, D., the case ..genopt/case/weldland, Ref.[2c]

C [18] Bushnell, D., the case ..genopt/case/trusscomp,Ref.[2d]

C [19] Bushnell, D., the case ..genopt/case/submarine,Ref.[2e]

C [20] Bushnell, D., the case ..genopt/case/sphere

C=============================================================
C                TABLE 1      "GENOPT" COMMANDS
C=============================================================
C     HELPG         (get information on GENOPT.)
C     GENTEXT       (GENOPT user generate a prompt file, program
C                    fragments [see TABLE 5], programs [see
C                    TABLE 4]., and this and other files
C                    [see TABLE 5 and the rest of this file.])
C     GENPROGRAMS   (GENOPT user generate absolute elements:
C                    BEGIN.EXE, DECIDE.EXE, MAINSETUP.EXE,
C                    OPTIMIZE.EXE, CHANGE.EXE, STORE.EXE,
C                    CHOOSEPLOT.EXE, DIPLOT.EXE.)

C     BEGIN         (end user provide starting data.)
C     DECIDE        (end user choose decision variables, bounds,
C                    linked variables,inequality constraints.)
C     MAINSETUP     (end user set up strategy parameters.)
C     OPTIMIZE      (end user perform optimization, batch mode.)
C     SUPEROPT      (Program tries to find the GLOBAL optimum
C                    design as described in Ref.[11] listed
C                    above (Many OPTIMIZEs in one run.)

C     CHANGE        (end user change some parameters.)
C     CHOOSEPLOT    (end user choose which variables to plot v.
C                    design iterations.)
C     DIPLOT        (end user obtain plots.)
C     INSERT        (GENOPT user add parameters to the problem.)
C     CLEANGEN      (GENOPT user cleanup your GENeric files.)
C     CLEANSPEC     (end user cleanup your SPECific case files)

C   Please consult the following sources for more
C   information about GENOPT:
C        1.  GENOPT.STORY  and  HOWTO.RUN  and  GENOPT.NEWS
C        2.  Sample cases: (in the directory, genopt/case)
C        3.  NAME.DEF file, where NAME is the name chosen by
C            the GENOPT-user for a class of problems. (In this
C            case  NAME = submarine)
C        4.  GENOPT.HLP file    (type HELPG)
C============================================================

C============================================================
C   TABLE 2   GLOSSARY OF VARIABLES USED IN "submarine"
C============================================================
C  ARRAY  NUMBER OF          PROMPT
C    ?   (ROWS,COLS)  ROLE  NUMBER   NAME                 DEFINITION OF VARIABLE
C                          (submarine.PRO)
C============================================================
C    n   (   0,    0)    2      10   RADIUS    = radius of the tank
C    n   (   0,    0)    2      15   LENGTH    = total length of the cylinder
C    n   (   0,    0)    2      20   EMOD      = elastic modulus of the shell wall
C    n   (   0,    0)    2      25   NU        = Poisson ratio of the shell wall
C    n   (   0,    0)    2      30   DENSTY    = weight density of the shell wall
C    n   (   0,    0)    2      40   NODSPH    = number of nodal points in the dom
C    n   (   0,    0)    2      45   NODCYL    = number of nodal points in the cyl
C    n   (   0,    0)    2      55   NPTSPH    = number of axial callouts in the d
C    n   (   0,    0)    2      60   IZSPH     = number of axial callouts in the d
C    y   (  15,    0)    2      65   ZSPH      = axial location of callout in the
C    n   (   0,    0)    2      70   NPTCYL    = Number of axial callouts in the c
C    n   (   0,    0)    2      75   IZCYL     = number of axial callouts in the c
C    y   (  15,    0)    2      80   ZCYL      = axial location of callout in the
C    n   (   0,    0)    2      90   IZREFSP   = axial callout number in the dome
C    y   (  15,    0)    1      95   ZREFSP    = location of ref. surf. in the dom
C    y   (  15,    0)    1     100   THKSPH    = wall thickness in the dome
C    n   (   0,    0)    2     105   IZREFCY   = callout number in the cylinder in
C    y   (  15,    0)    1     110   ZREFCY    = location of the ref. surf. in the
C    y   (  15,    0)    1     115   THKCYL    = thickness of the cylindrical shel
C    n   (   0,    0)    2     125   NBUKLO    = low end of range of buckling circ
```

Table 9, p. 6 of 12

```
C    n   (   0,   0)   2     130    NBUKHI    = high end of range of buckling cir
C    n   (   0,   0)   2     140    NCASES    = Number of load cases (number of e
C    y   (  20,   0)   3     145    PRESS     = uniform external pressure
C    y   (  20,   0)   4     155    BUCKL     = tank buckling eigenvalue
C    y   (  20,   0)   5     160    BUCKLA    = allowable buckling load (Use 1.0)
C    y   (  20,   0)   6     165    BUCKLF    = factor of safety for buckling
C    n   (   0,   0)   2     170    JSTRESS   = shell segment number in STRESS(NC
C    y   (  20,  10)   4     175    STRESS    = effective stress in shell segment
C    y   (  20,  10)   5     180    STRESSA   = maximum allowable effective stres
C    y   (  20,  10)   6     185    STRESSF   = factor of safety for stress
C    n   (   0,   0)   7     195    WEIGHT    = weight of the BIGBOSOR4 model
C
C=DECK      BEHX1
      SUBROUTINE BEHX1
     1 (IFILE,NPRINX,IMODX,IFAST,ILOADX,PHRASE)
C
C   PURPOSE: OBTAIN tank buckling eigenvalue
C
C   YOU MUST WRITE CODE THAT, USING
C   THE VARIABLES IN THE LABELLED
C   COMMON BLOCKS AS INPUT, ULTIMATELY
C   YIELDS THE RESPONSE VARIABLE FOR
C   THE ith LOAD CASE, ILOADX:
C
C     BUCKL(ILOADX)
C
C   AS OUTPUT. THE ith CASE REFERS
C   TO ith ENVIRONMENT (e.g. load com-
C   bination).
C
C   DEFINITIONS OF INPUT DATA:
C    IMODX  = DESIGN CONTROL INTEGER:
C     IMODX = 0 MEANS BASELINE DESIGN
C     IMODX = 1 MEANS PERTURBED DESIGN
C     IFAST = 0 MEANS FEW  SHORTCUTS FOR PERTURBED DESIGNS
C     IFAST = 1 MEANS MORE SHORTCUTS FOR PERTURBED DESIGNS
C    IFILE = FILE FOR OUTPUT LIST:
C    NPRINX= OUTPUT CONTROL INTEGER:
C     NPRINX=0 MEANS SMALLEST AMOUNT
C     NPRINX=1 MEANS MEDIUM AMOUNT
C     NPRINX=2 MEANS LOTS OF OUTPUT
C
C     ILOADX = ith LOADING COMBINATION
C     PHRASE = tank buckling eigenvalue
C
C   OUTPUT:
C
C     BUCKL(ILOADX)
C
       CHARACTER*80 PHRASE
C  INSERT ADDITIONAL COMMON BLOCKS:
      COMMON/FV06/ZSPH(15),IZSPH
      REAL ZSPH
      COMMON/FV07/ZCYL(15),IZCYL
      REAL ZCYL
      COMMON/FV08/ZREFSP(15),IZREFSP
      REAL ZREFSP
      COMMON/FV09/THKSPH(15)
      REAL THKSPH
      COMMON/FV10/ZREFCY(15),IZREFCY
      REAL ZREFCY
      COMMON/FV11/THKCYL(15)
      REAL THKCYL
      COMMON/FV12/PRESS(20)
      REAL PRESS
      COMMON/FV15/BUCKL(20),BUCKLA(20),BUCKLF(20)
      REAL BUCKL,BUCKLA,BUCKLF
      COMMON/FV18/STRESS(20,10),JSTRESS,STRESSA(20,10),STRESSF(20,10)
      REAL STRESS,STRESSA,STRESSF
      COMMON/IV01/NODSPH,NODCYL,NPTSPH,NPTCYL,NBUKLO,NBUKHI
      INTEGER NODSPH,NODCYL,NPTSPH,NPTCYL,NBUKLO,NBUKHI
      COMMON/FV01/RADIUS,LENGTH,EMOD,NU,DENSTY,WEIGHT
      REAL RADIUS,LENGTH,EMOD,NU,DENSTY,WEIGHT
C
C
C  INSERT SUBROUTINE STATEMENTS HERE.
      COMMON/TOTMAX/TOTMAS ← note
      COMMON/INSTAB/INDIC
```

buckling analysis

GENTEXT creates this part

GENOPT user creates this part

Table 9, p. 7 of 12

GENOPT user creates this

```
      COMMON/EIGB4M/EIGCOM(200),EIGNEG(200),EIGCRN
      COMMON/WVEB4M/NWVCOM(200),NWVNEG(200),IWAVEB,NWVCRN
      COMMON/EIGBUK/EIGCRT
      COMMON/NWVBUK/NWVCRT
      COMMON/BUCKN/N0BX,NMINBX,NMAXBX,INCRBX
      COMMON/BUCKN0/N0B,NMAXB
      COMMON/PRMOUT/IFILE3,IFILE4,IFILE8,IFILE9,IFIL11
      COMMON/EIGALL/EIG0,EIG1,EIG2,EIG3,EIG4
      COMMON/WAVALL/NWAV0,NWAV1,NWAV2,NWAV3,NWAV4
      COMMON/NUMPAR/IPARX,IVARX,IALLOW,ICONSX,NDECX,NLINKX,NESCAP,ITYPEX
      common/caseblock/CASE
      CHARACTER*28 CASE
      CHARACTER*35 CASA
C
      INDIC = 1
```

means "buckling analysis"

```
C
      N0B = NBUKLO
      NMAXB = NBUKHI
C
      CALL BOSDEC(1,24,ILOADX,INDIC)
```

BOSDEC generates a BIGBOSOR4 input file

Creates *.BEHX1

```
C
      IF (ITYPEX.EQ.2) THEN
C      Get CASE.BEHX1 file for input for BIGBOSOR4...
C      CASE.BEHX1 is an input file for BIGBOSOR4 for behavior no. 1:
C      buckling load
         I=INDEX(CASE,' ')
         IF(I.NE.0) THEN
            CASA=CASE(:I-1)//'.BEHX1'
         ELSE
            CASA=CASE//'.BEHX1'
         ENDIF
         OPEN(UNIT=61,FILE=CASA,STATUS='UNKNOWN')
         CALL BOSDEC(1,61,ILOADX,INDIC)
         CLOSE(UNIT=61)
         WRITE(IFILE,'(/,/,A,A,/,A)')
     1 ' BIGBOSOR4 input file for:',
     1 ' buckling load (INDIC=1)',
     1   CASA
      ENDIF
C
      CALL B4READ
```

BIGBOSOR4 preprocessor

```
      IF (IMODX.EQ.0) THEN
         N0BX = N0B
         NMINBX = N0B
         NMAXBX = NMAXB
         INCRBX = 1
      ELSE
         NOBX = NWAV1
         NMINBX = NWAV1
         NMAXBX = NWAV1
         INCRBX = 1
      ENDIF
      REWIND IFILE9
      CALL STOCM1(IFILE9)
      CALL STOCM2(IFILE9)
      CALL B4MAIN
```

BIGBOSOR4 mainprocessor

```
      CALL GASP(DUM1,DUM2,-2,DUM3)
      IF (IMODX.EQ.0) THEN
         EIG1 = EIGCRT
         NWAV1= NWVCRT
      ENDIF
C
      IF (IMODX.EQ.0) THEN
      WRITE(IFILE,5) TOTMAS
    5 FORMAT(/,' WEIGHT OF THE BIGBOSOR4 MODEL OF THE TANK',/,
     1' TOTMAS =',1P,E12.4,/,
     1' *****************************************************')
        WRITE(IFILE,'(/,A)')
     1 ' BUCKLING LOAD FACTORS AND MODES (BEHX1)'
        DO 10 I = 1,IWAVEB
         WRITE(IFILE,'(A,1P,E12.4,A,I4,A)')
     1  '        ',EIGCOM(I),'(',NWVCOM(I),')'
   10   CONTINUE
        WRITE(IFILE,'(A,1P,E12.4)')
     1' Critical buckling load factor, BUCKL=',EIGCRT
        WRITE(IFILE,'(A,I5)')
     1' Critical number of circumferential waves, NWVCRT=',NWVCRT
      ENDIF
```