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

David Bushnell

May 15, 2009

ABSTRACT

BIGBOSOR4 is used in an optimization loop in which the dimensions of a typical weld land and the dimensions of Tee-shaped stringers along the two straight edges (generators) of the weld land are decision variables. The optimization is carried out in a GENOPT context. Any number of equally spaced weld lands in a 360-degree cylindrical shell can be accommodated. The weld lands are inserted in an internally stiffened cylindrical shell which has been previously optimized by PANDA2. The previously optimized cylindrical shell must have internal stringers and rings of rectangular cross sections. The spacings, heights, and thicknesses of the internal rectangular stiffeners are not decision variables for the optimization problem in which the optimum weld land and "extra" weld land edge stringers are optimized. The design constraints for the cylindrical shell with the weld lands are: 1. general buckling, 2. inter-ring buckling, and 3. stress. The prebuckled state is assumed to be uniform end shortening, with the axial compression in each segment of the structure proportional to the axial membrane stiffness of that segment. The entire structure is assumed to be fabricated of the same material. In the model for general buckling the previously optimized rectangular rings and stringers are smeared out and the cylindrical shell is simply supported at its ends. In the model for inter-ring buckling adjacent rings are replaced by simple supports, a length of shell equal to the ring spacing is analyzed, and the previously optimized stringers are smeared out. The maximum stress in the weld-land-edge-stringer region is computed as if there were no prebuckling bending (membrane compression). The cylindrical shell is modeled as a 180-degree segment of a huge torus, with symmetry conditions applied along the generators at the bottom and at the top of the huge toroidal segment. Weld lands in the previously optimized cylindrical shell called "nasacoldbend" are optimized for the case in which there are weld lands with weld land edge Tee-shaped stringers spaced at 120-degree intervals around the circumference of the internally stiffened cylindrical shell.

GENOPT is used.

No prebuckling bending!

See Appendix Z

$N_x = -2219$ lb/in
Length = 68.75"
Radius = 48.0"

INTRODUCTION

Some references are listed in Table 1. The work in this paper was motivated by Robert Thornburgh's paper on the buckling behavior of axially oriented weld lands in internally ring and stringer stiffened cylindrical shells [1]. A capability to optimize an axially oriented weld land with "extra" stringers added to its edges was needed. The models used in this paper are much simpler than those used in Thornburgh's paper. (Thornburgh used STAGS, a general-purpose finite element code). Naturally, the models here are approximate. For example, the prebuckled state is assumed to be a membrane state: there is no prebuckling bending of the axially compressed cylindrical shell with multiple axially oriented weld lands. The axial compression, Nx, in the various segments of the model is assumed to be distributed in proportion to the axial stiffness of those segments, and this distribution of Nx is prismatic, that is, it does not vary along the axis of the cylindrical shell. There is no "boundary layer" nonuniformity of radial displacement w in the neighborhoods of the two ends of the cylindrical shell caused by restriction of Poisson ratio radial expansion there.

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.

In this study a cylindrical shell is modeled as a huge torus. More precisely, part of a cylindrical shell (180 degrees of its circumference) is modeled as part of a huge torus. (See Fig. 2 and Appendix 2). This is a "trick" that permits detailed modeling of an axially oriented weld land with added edge stringers while working within the restriction that BIGBOSOR4 only handles shells of revolution. With the "huge torus" model the shell coordinates are exchanged: what was the axial coordinate of the cylindrical shell modeled in the usual way becomes the circumferential coordinate of the huge torus model, and what was the circumferential coordinate of the cylindrical shell modeled in the usual way becomes the meridional coordinate of the huge torus model. The stringers become rings and the rings become stringers. In the usual BIGBOSOR4 model of an axially stiffened (stringer stiffened) cylindrical shell the stringers have to be smeared out because BIGBOSOR4 handles only shells of revolution. In the usual BIGBOSOR4 model of a ring-stiffened cylindrical shell the rings can be treated as little elastic, deformable shell segments; they do not have to be smeared out. In the huge torus model the situation is reversed: what were rings and now become stringers (meridionally oriented stiffeners) have to be smeared out. What were stringers and now become rings (cirumferentially oriented stiffeners in the huge torus) can be modeled as little flexible shell segments. An axially oriented weld land with edge stringers in the cylindrical shell can be modeled in the huge torus as a meridionally thickened segment with edge rings, either with rectangular cross section or with Tee-shaped cross section in the particular application here.

In the "huge torus" model, as in any shell-of-revolution model, the meridional coordinate is discretized and variation of the buckling mode in the circumferential coordinate direction is trigonometric with n (or N) circumferential waves around the huge circumference of the torus. Appendix 2 explains the relationship between the axial length of the cylindrical shell and the circumferential wave number (N = 100) that corresponds to one half wavelength that spans that length. Simple support end conditions on the cylindrical shell are implied because simple support (antisymmetry) is what naturally occurs along the nodal lines of buckling modes.

Figure 1 shows an example of a cylindrical shell with six edge-stiffened weld lands over the 360 degrees of its circumference. Each weld land is reinforced by two Tee-shaped "stringers". The circumferential coordinate in the huge torus model is the

coordinate normal to the plane of the paper, and the meridional coordinate in the huge torus model is the coordinate in the plane of the paper along each of the 19 segments of the BIGBOSOR4 huge torus model displayed in Fig. 1.

The decision variables in the optimization problem presented here are the width and thickness of a typical weld land, the height and thickness of a typical edge stringer web, and the width and thickness of a typical edge stringer outstanding flange.

The "non-weld-land" parts of the huge torus model (Segments 4, 10, and 16 in Fig. 1) represent the stiffened cylindrical shell previously optimized by PANDA2. In the particular case analyzed here the optimized internally ring and stringer stiffened cylindrical shell without any weld lands is taken from the PANDA2 sample case located in the directory, ...panda2/case/nasacoldbend. This cylindrical shell was previously optimized by PANDA2 accounting for the effect of cold bending. (See Appendix 1). The internal stiffeners have rectangular cross sections. The decision variables used during the previous optimization by PANDA2 are the thickness of the shell skin, the spacing, height, and thickness of the internal stringers, and the spacing, height, and thickness of the internal rings. These quantities are not decision variables in the current optimization problem in which we wish to find the best dimensions of a weld land and of the weld land edge stringers. (Except see Table 14 for the comments about TSKIN). Details relating to the "nasacoldbend" case appear in Appendix 1 here.

Figure 2 displays a general buckling mode predicted by BIGBOSOR4 corresponding to the model shown in Fig. 1. This is the kind of BIGBOSOR4 result obtained here, except in the particular configuration to be optimized here the weld lands occur at 120-degree intervals rather than at 60-degree intervals.

ABOUT GENOPT

GENOPT [4 - 6] 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 case the GENOPT user and the end user are the same person, the writer).

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. The GENOPT user must also supply software (subroutines and/or FORTRAN statements) that perform the analysis or analyses. 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 identifying problem variables the GENOPT user must decide which of 7 roles each of these variables plays. The 7 possible roles are:

1. decision variable candidate (such as a structural dimension)
2. parameter that is not a decision variable candidatea (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)

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 to simply analyze an existing design or both.

Please read [4] first, followed by the first part of [6], which contains many details about how to use GENOPT. Tables 2 and 3 contain some information on the use of GENOPT. In Table 3 a generic name, "cylinder", frequently appears. In this paper the generic name specified by the writer is "weldland". When studying Table 3 and setting up the proper files at his or her facility, the reader should substitute the generic

name, "weldland" for the generic name, "cylinder".

PRODUCTION OF THE PROGRAM SYSTEM TO OPTIMIZE A WELD LAND. THE GENERIC CASE IS CALLED "weldland"

Table 4 lists the run stream used to obtain the results presented here.

The GENOPT user first provides input during the long GENTEXT interactive session as listed in Table 5. GENTEXT automatically produces the files weldland.DEF (Table 6 is an abridged version) and weldland.PRO (Table 7), which contains the prompts and "help" paragraphs seen by the end user.

GENTEXT also produces FORTRAN fragments, weldland.*, listed on page 1 of Table 4 and described on pages 2 and 3 of Table 6. GENOPT 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. 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 to the version of STRUCT.NEW automatically created by GENOPT. (See Tables 8 and 9). In this particular application the GENOPT user does more to "flesh out" the BEHAVIOR.NEW library. (See Tables 10 and 11).

During the GENTEXT interactive session the GENOPT user here decided to introduce three behaviors: 1. general buckling, 2 inter-ring buckling, and 3. stress. Corresponding to these three behaviors, GENTEXT automatically created three skeletal "behavioral" subroutines, SUBROUTINE BEHX1, SUBROUTINE BEHX2, and SUBROUTINE BEHX3. The GENOPT user had to "flesh out" each of these three "behavioral" subroutines, as listed in Tables 10 and 11.

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 12. 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.

********** 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 some other name. In this application the writer uses the names "behavior.weldland" and "struct.weldland". (also "bosdec.weldland"). 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.
****

The present application is similar to the GENOPT sample case called "cylinder". In the sample case, "cylinder", 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 [6] 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 [6], one should concentrate on the first part of [6], in which the role of the GENOPT user predominates.

OPTIMIZATION OF THE SPECIFIC CASE CALLED "wcold"

Tables 13 - 23 and Figs. 3 - 9 pertain to this section.

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

Table 13 lists the input for the "BEGIN" processor. In this

specific application the weld land edge stringer has a Tee-shaped cross section and stringer-stiffened weld lands occur at 120-degree intervals. The geometry of the previously optimized cylindrical shell without any weld lands is that listed at the end of Table a8 in Appendix 1. A factor of safety of 2.0 is given for general buckling (GENBUKF) and a factor of safety of 1.5 is given for inter-ring buckling (PANBUKF) because in this application the shell geometry is perfect. It is not possible to introduce a general buckling modal imperfection in this "huge torus" BIGBOSOR4 model of the shell. However, notice that the "non-weld-land" portions of the stiffened cylindrical shell were previously optimized by PANDA2 in the presence of plus and minus general buckling modal imperfections with amplitude, Wimp = 0.125 inch (Figs. a1 and a2 and Table a6 of Appendix 1).

Table 14 lists the input for the "DECIDE" processor. Note that a linking expression is introduced for the weld land eccentricity, ECLAND. This linking expression maintains the flushness of the OUTER surfaces of the weld land and the cylinrical shell skin during optimization cycles. Unfortunately, in the "DECIDE" processor, as currently written for GENOPT, all variables in the linking expression must be decision variables. Therefore, it was necessary to make the thickness, TSKIN, of the cylindrical shell skin a decision variable even though we do not want TSKIN to change significantly from its previously obtained optimum value of 0.058191 (Table 13). Accordingly, we establish very tight lower and upper bounds for TSKIN so that during optimization cycles TSKIN will not change significantly.

Table 15 lists the input for the "MAINSETUP" processor.

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 or MOSTLY UNFEASIBLE"	-0.20

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 15:

For the response to the prompt in Table 15:

```
      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 15:

```
     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 15:

```
     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 15 for cases similar to "wcold" you will be ok.

After completion of the SUPEROPT run, which requires several hours in this particular application, we want to see a plot of the objective vs the approximately 470 design iterations during the SUPEROPT execution. We use CHOOSEPLOT to select what to plot. In this case (and usually for SUPEROPT) we choose to plot only the objective function, not the design margins nor the decision variables. (Plots of the design margins and the decision variables for 470 design iterations would be very messy. These should be plotted only after a few executions of OPTIMIZE, not after execution of SUPEROPT.)

Table 16 lists input for CHOOSEPLOT and Fig. 3 displays the objective vs design iterations for the SUPEROPT run. Note that there are several "spikes" in this plot. Each "spike" corresponds to a new "starting" design. Each new "starting" design is obtained randomly but consistently with the lower and upper bounds and linking expressions. The aim of the SUPEROPT run is to attempt to find a "global" optimum design by starting from many different points in design space and converging to various optimum designs. In this case the best design obtained after the SUPEROPT run is the design with the minimum weight that is either "FEASIBLE" or "ALMOST FEASIBLE" (IDESIGN = 2). In this particular execution of SUPEROPT the user chose to allow 5 executions of OPTIMIZE for each execution of AUTOCHANGE. The processor, AUTOCHANGE, randomly obtains a new "starting" design.

Table 17 lists the optimum design obtained after a single execution of SUPEROPT. Note that the thickness of the weld land goes to its lower bound, set in this case to 0.1 inch. The width of the weld land stays at its lower bound of 4.0 inches throughout the SUPEROPT run. Perhaps these optimized dimensions are inappropriate for some reason, such as the feasibility of welding thin parts together. If the reader wishes to re-design using different and perhaps more appropriate bounds, he or she should do so. The writer knows very little about welding. Hence, his choice of bounds was fairly arbitrary. The reader should regard this report as a guide as to how to obtain optimum designs, not as a guide about the good design of weld lands.

The design margins of the optimized structure are listed on page 2 of Table 17. Note that the stress margin is not critical, probably because the present application does not permit any prebuckling bending. The user should apply STAGS to the optimum design found here in order to validate these results, which,as previously described, are obtained from an approximate ("quick and dirty") model.

The objective in this application is the weight per axial length of a single Tee-stringer-stiffened weld land, as computed in SUBROUTINE OBJECT. (See the bottom of p. 13 of Table 10 for the formula used for the objective, which is called "WEIGHT").

The optimum design is saved for future use via the "CHANGE" processor. Table 18 lists the input to "CHANGE". The writer has found from long experience that it is best always to save the optimum design in this way.

Whenever OPTIMIZE is executed in a "fixed design mode", that is, with ITYPE = 2 in the *.OPT file (wcold.OPT in this specific case), SUBROUTINE BEHX1 produces two valid input files for BIGBOSOR4: wcold.BEHX0 and wcold.BEHX1. These files can be used outside of the GENOPT context in order to see what the general buckling modes look like for the optimized shell without (wcold.BEHX0) and with (wcold.BEHX1) weld lands. SUBROUTINE BEHX2 produces one valid input file for BIGBOSOR4: wcold.BEHX2. This file can be used outside of the GENOPT context in order to see what the inter-ring buckling mode looks like for the optimized shell with weld lands.

Table 19 lists the wcold.BEHX0 file and Table 20 lists the equivalent wcold.ALL file after application of the BIGBOSOR4 processor called "cleanup". The files, wcold.BEHX1 and wcold.BEHX2, are analogous to

wcold.BEHX0. They are not listed here because these models have several shell segments and therefore the files, wcold.BEHX1 and wcold.BEHX2, are much longer than the file, wcold.BEHX0.

Tables 21 - 23 and Figs. 4 - 9 show results from BIGBOSOR4 for the optimized configuration corresponding to the BIGBOSOR4 input files, wcold.BEHX0 (Table 21, Fig 4), wcold.BEHX1 (Table 22, Figs. 5 - 7), and wcold.BEHX2 (Table 23, Figs. 8 and 9). Note that in order to produce the rather long list of buckling loads vs CIRC. WAVES in Table 22 it was necessary to edit the wcold.ALL file obtained after execution of "cleanup" in order to change the range of circumferential wave numbers, N0B, NMINB, NMAXB, INCRB, in the BIGBOSOR4 input file, wcold.ALL. [Search for the string, "N0B" (that is, N zero B) in the file, wcold.ALL to get to the correct place to do the edit.]

Note that in Table 22 the local buckling of the weld land occurs at a load factor of 1.8624, which is the lowest buckling load factor in the large range of circumferential wave numbers, 100 to 3000. In producing the general buckling margin in Table 17 we searched only over the range from 100 to 1000 (as specified in the input for BEGIN (Table 13: input for MLOWG = 1 and MHIGHG = 10 . MLOWG and MHIGHG are later multiplied by 100 because of the huge torus model, as described in Appendix 2.) Did we miss an important failure mode? The answer is no because this particular high-N mode of buckling is covered by the inter-ring buckling model.

Note that in this case we were lucky to capture the buckling mode in which there is sidesway of the weld land edge stringers (Fig. 6) because, as can be seen by the list of "general buckling" eigenvalues at the bottom of p. 1 of Table 17, this mode happens to occur at the extreme upper end of the range, MLOWGx100 to MHIGHGx100. This implies that perhaps we should have used a higher MHIGHG in BEGIN (Table 13). Maybe, but if we had used a higher MHIGHG, we might have captured what is actually an inter-ring buckling mode, which is associated with a different factor of safety. Care is required in this case. One should always use a large range of circumferential wave numbers in the BIGBOSOR4 model, wcold.BEHX1, (that is, outside the context of optimization in the post-optimization evaluation phase of the effort) in order fully to evaluate the optimum design obtained by SUPEROPT.

USE OF THE OPTIMIZED WELD LAND IN THE CASE "nasacoldbend"

Tables 24 - 29 and Figs. 10 - 18 pertain to this section.

The purpose is to determine the effect of not smearing the stringers in the previously optimized acreage of the stiffened cylindrical shell. This capability exists because of recent work reported in Ref. [2].

Again we use the "huge torus" model, but in this case there is always only a single weld land and only 90 degrees of the circumference of the cylindrical shell is included in the "huge torus" model. The BIGBOSOR4 models are generated by the new PANDA2 processor called "bospn3". "bospn3" is executed via the new PANDA2 command, "panel3". (See [2] for details.)

The reader should inspect the results listed in the tables and figures pertaining to this section and compare them to the results listed and plotted for the specific case "wcold". Note that the dimensions of the cylindrical shell and acreage stiffeners are the same in the two cases, "nasacoldbend" and "wcold".

Note from Fig. 18 that the buckling load associated with the lowest load factor, lambda = 1.3515, is not present in the "wcold" application. That is because in the "wcold" application the shell acreage stringers are smeared out. Therefore, the "wcold" model cannot determine local buckling outside the weld land. However, the shell acreage away from any weld land was previously optimized by PANDA2 including a general buckling modal imperfection with amplitude, Wimp = plus and minus 0.125 inch. The local buckling mode depicted in Fig. 18 (and in Fig 17) does not involve the weld land at all. Therefore, we do not need to be concerned

that the "wcold" model is incapable of predicting local buckling away from any weld land, even though the local buckling load factor is lower of any of the buckling load factors determined by "wcold". In previously optimizing the stiffened cylindrical shell without any weld lands, we previously decided that inclusion of the general buckling modal imperfection with amplitude, Wimp = plus and minus 0.125 inch would lead to a conservative optimum design. The present optimization of the weld land does not change that previously made decision about the adequacy of the shell acreage away from any weld land.

CONCLUSION

Only one optimum design has been obtained here. The reader should apply this technology to various configurations that fit within the generic class called "weldland" and should verify the optima he or she obtains by the application of a general-purpose computer program such as STAGS to the optimized configurations.

It is emphasized that in the simplified model here there is no prebuckling bending. The prebuckled state is characterized by the uniform membrane compression that would occur in an ideal situation if the cylindrical shell were subjected to uniform end shortening. There is no local prebuckling bending near the two ends of the axially compressed cylindrical shell, and the nature of the "huge torus" model is such that only simple support end conditions apply for the buckling modal deflection.

It is hoped that NASA Langley and colleagues of NASA Langley will use GENOPT to convert existing analyses into user-friendly analyses and, by means of GENOPT, automatically to add optimization capability to those existing analyses. Guidelines are given here and in the referenced literature (Table 1) as to how to do this.

← NOTE!!!

Table 1

REFERENCES

[1] Thornburgh, Robert P., "Axial weld land buckling in compression-loaded orthogrid cylinders',

[2] Bushnell, David, "Use of PANDA2 and BIGBOSOR4 to obtain linear buckling loads of an orthogrid stiffened cylindrical shell with and without a weld land", unpublished report to NASA Langley, April 30, 2009.

[3] Bushnell, David, The file, .../panda2/case/nasacoldbend/coldbending.pdf included with the PANDA2 documentation

[4] 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. 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.

[5] Bushnell, D., "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

[6] Bushnell, D., "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

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

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

"thorndesign1"

"nasacoldbend" (Appendix I)

GENOPT

GENOPT, BIGBOSOR4

GENOPT

optimizer in GENOPT & in PANDA2

This reference has lots of details about how to use GENOPT.

This is the original GENOPT reference. Read it first.

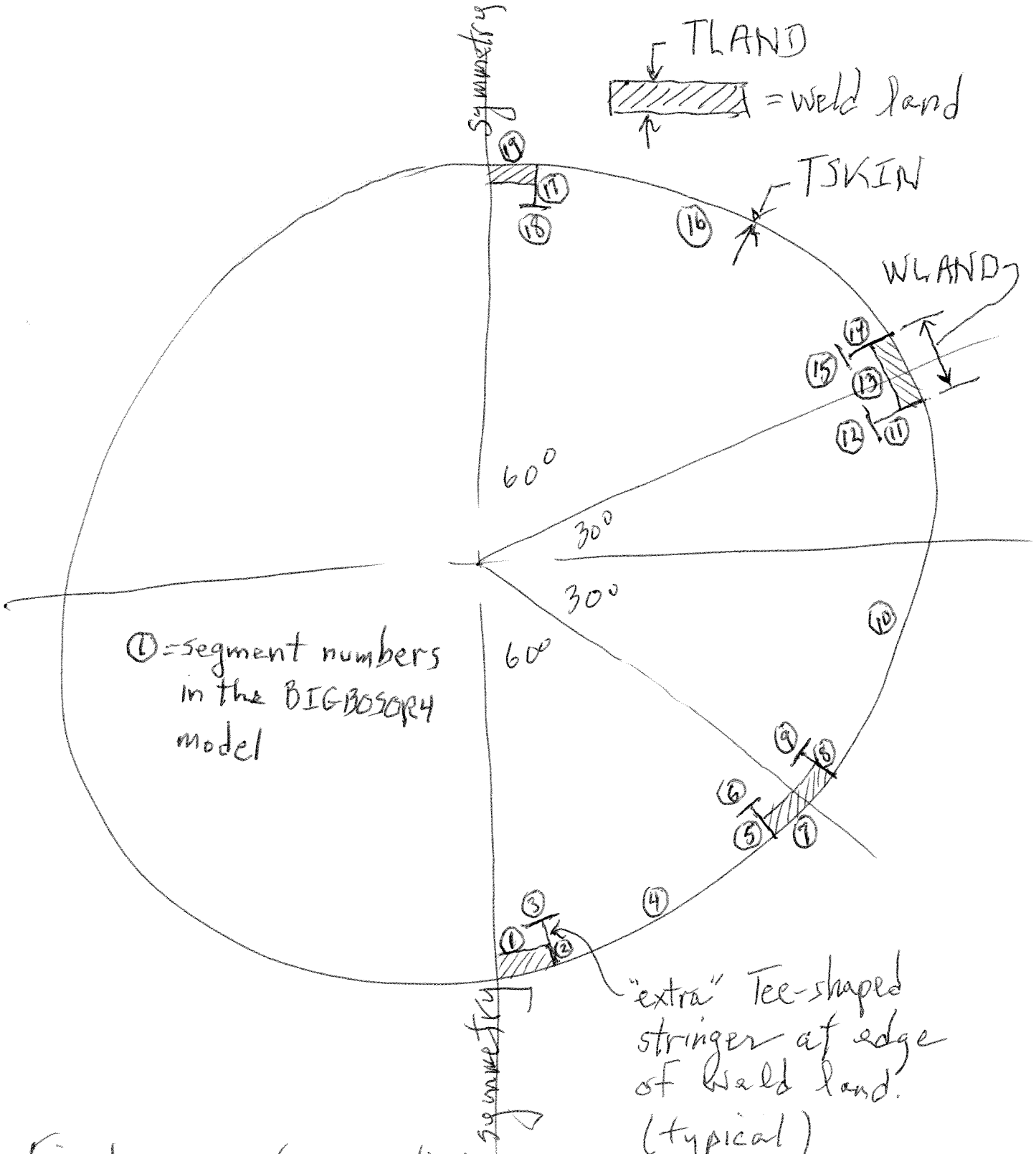

Fig. 1 Example in which there are 6 weld lands in 360 degrees. (12)

3 axial halfwaves; see Appendix 2

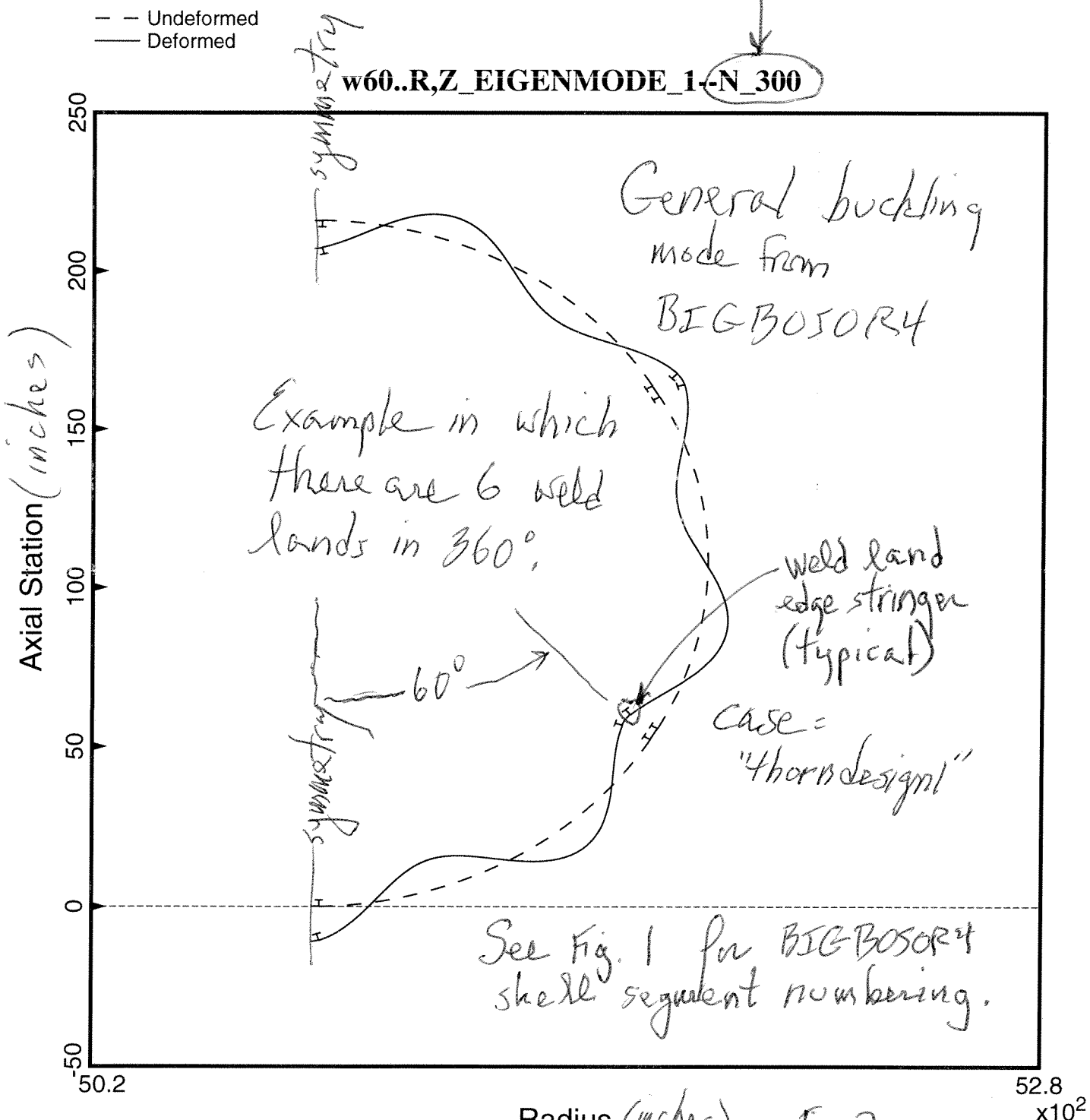

Fig. 2

Table 2 (2 pages)

```
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)

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.
```

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

Use SUPEROPT now.

Table 3 (6 pages)

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

```
******************** 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)
------------------------------------------------------------------------------

********* 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!)
```

← original GENOPT reference

In the weld land case the generic case name is "weldland" 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-----   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. 3-16)

```
-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. 4-16)

********* 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 (5 pages)

RUNSTREAM USED TO PRODUCE THE OPTIMUM DESIGN
OF A TYPICAL WELD LAND WITH "EXTRA" TEE-SHAPED
EDGE STRINGERS

```
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 ("weldland"), variable names,
                  roles, one-line definitions, help paragraphs, etc.
                  Here the input data for GENTEXT are in the file,
                 "weldland.INP". Also, see the files, "weldland.DEF"
                  and "weldland.PRO".)
(After execution of GENTEXT the following "weldland" files
exist in the directory where GENTEXT was executed (..genoptcase):
------------------------------------------------------------------
bush-> ls -al weldland.*
-rw-r--r--  1 bush bush  2632 May 15 10:43 weldland.CHA
-rw-r--r--  1 bush bush   792 May 15 10:43 weldland.COM
-rw-r--r--  1 bush bush  4152 May 15 10:43 weldland.CON
-rw-r--r--  1 bush bush 27959 May 15 10:43 weldland.DAT
-rw-r--r--  1 bush bush 30253 May 15 10:43 weldland.DEF
-rw-r--r--  1 bush bush 28083 May 14 08.39 weldland.INP
-rw-r--r--  1 bush bush 12104 May 15 10:43 weldland.NEW
-rw-r--r--  1 bush bush  8006 May 15 10:43 weldland.PRO
-rw-r--r--  1 bush bush  1148 May 15 10:43 weldland.REA
-rw-r--r--  1 bush bush   392 May 15 10:43 weldland.SET
-rw-r--r--  1 bush bush  9862 May 15 10:43 weldland.SUB
-rw-r--r--  1 bush bush  1148 May 15 10:43 weldland.WRI
------------------------------------------------------------

(Next, create software that computes the design constraints
 and the objective. In this example create bosdec.weldland
 and "flesh out" the skeletal struct.new and behavior.new
 automatically created by GENOPT, that is, create
 behavior.weldland from behavior.new  and struct.weldland
 from struct.new. This activity might well require
 most of your effort on your project.)

genprograms      (compiles the software created by GENOPT
                  and by the GENOPT user)

(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 71092 May 15 10:44 autochange.linux
```

Table 5 → "weldland.INP"
Table 7 → "weldland.PRO"
Table 6 → "weldland.DEF"

p. 2 & 3 of Table 6 → .CHA, .COM, .CON
Table 6 → .DAT
Table 5 → .DEF
Table 7 → .PRO
p. 2 & 3 of Table 6 → .REA, .SET, .SUB, .WRI

Table 12 → bosdec.weldland
Tables 8 & 9 → behavior.weldland, struct.weldland
Tables 10 & 11

IMPORTANT!: make sure to save "behavior" & "struct" because GENTEXT destroys earlier versions of behavior.new and struct.new

Table 4 (p. 2 of 5)

```
-rwxr-xr-x  1 bush bush 134844 May 15 10:44 begin.linux
-rwxr-xr-x  1 bush bush 123245 May 15 10:44 change.linux
-rwxr-xr-x  1 bush bush 151551 May 15 10:44 chooseplot.linux
-rwxr-xr-x  1 bush bush 150821 May 15 10:44 decide.linux
-rwxr-xr-x  1 bush bush 98005 May 15 10:44 mainsetup.linux
-rwxr-xr-x  1 bush bush 1525173 May 15 10:44 optimize.linux
-rwxr-xr-x  1 bush bush 113172 May 15 10:44 store.linux
```

← GENOPT creates these automatically (and autochange)

```
Next, type the command BEGIN to input data for a new case.
-----------------------------------------------------------

begin           (provide starting design, material, loading,
                 allowables and factors of safety for the
                 three "behaviors": 1. general buckling,
                 2. inter-ring ("panel") buckling, 3. stress;
                 wcold.BEG, in which "wcold" = specific case name.)
```

Table 13

```
(The command, BEGIN, starts an interactive session, the beginning
 of which presents the following to your computer screen:)
-----------------------------------------------------------------
GENOPT = /home/progs/genopt

 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: weldland
```

← GENERIC case name

```
 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: wcold
```

← SPECIFIC case name must be different from the GENERIC name.

```
 ****************   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 wcold.BEG, in which wcold is a name that you
 have chosen for the specific case. (The name, wcold 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 wcold.BEG as input.  Rather than answer all the questions
 interactively, you can use wcold.BEG or an edited version of
 wcold.BEG as input to BEGIN.  BEGIN also generates an output
 file called wcold.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.

 ************************************************

 Are you correcting, adding to, or using an existing file?=
```

See Table 13 for input file for BEGIN.

```
-----------------------------------------------------------------

(When you have completed BEGIN you will have the file, wcold.BEG,
 which can be used in some future execution of BEGIN.)
```

Table 13

```
decide          (provide decision variables, bounds, equality constraint;
                 wcold.DEC)
```

Table 14

```
mainsetup       (strategy, analysis type, etc.; wcold.OPT)
```

Table 15

```
superopt        (use 5 OPTIMIZEs per AUTOCHANGE.)

(Inspect the wcold.OPP file. Then give the following commands:)

chooseplot      (plot only the objective vs design iterations.)
```

Table 16 : wcold.CPL

```
diplot          (get the postscript file, wcold.5.ps. Fig.3)

(Edit the file, wcold.OPT, to change ITYPE from 1 to 2, that is,
 change the analysis type from optimization to analysis of a fixed
```

Table 4 (p. 3 of 5)

design, the optimum design found from SUPEROPT.)

mainsetup (strategy, analysis type, etc.; wcold.OPT) — Table 15 with ITYPE=2

optimize (get results for the optimum design; wcold.OPM) — Table 17

change (save the optimum design, wcold.CHG) — Table 18

(After execution of begin, decide, mainsetup, chooseplot, diplot, optimize, and change, you will have the following "wcold" files:)

```
-rw-r--r--  1 bush bush  10323 May 15 13:58 wcold.ALL
-rw-r--r--  1 bush bush   2170 May 14 09:10 wcold.BEG
-rw-r--r--  1 bush bush   1761 May 15 13:58 wcold.BEHX0
-rw-r--r--  1 bush bush  10714 May 15 13:58 wcold.BEHX1
-rw-r--r--  1 bush bush  10323 May 15 13:58 wcold.BEHX2
-rw-r--r--  1 bush bush 439796 May 15 13:58 wcold.BLK
-rw-r--r--  1 bush bush 114460 May 15 13:58 wcold.CBL
-rw-r--r--  1 bush bush   1795 May 14 09:04 wcold.CHG
-rw-r--r--  1 bush bush    299 May 14 08:08 wcold.CPL
-rw-r--r--  1 bush bush   2579 May 13 17:25 wcold.DEC
-rw-r--r--  1 bush bush  45989 May 15 13:58 wcold.DOC
-rw-r--r--  1 bush bush     30 May 15 13:57 wcold.NAM
-rw-r--r--  1 bush bush   4204 May 15 13:57 wcold.OPB
-rw-r--r--  1 bush bush   4216 May 15 13:57 wcold.OPC
-rw-r--r--  1 bush bush   5055 May 15 13:57 wcold.OPD
-rw-r--r--  1 bush bush   8203 May 15 13:58 wcold.OPM
-rw-r--r--  1 bush bush      0 May 15 13:58 wcold.OPP
-rw-r--r--  1 bush bush    841 May 14 09:11 wcold.OPT
```

"wcold" is the specific case name.

TABLE 19

(The three files, wcold.BEHX0, wcold.BEHX1, wcold.BEHX2, contain the following:)

wcold.BEHX0 = valid BIGBOSOR4 input file for the 180-degree huge torus model without any weld lands. ← Table 19

wcold.BEHX1 = valid BIGBOSOR4 input file for the 180-degree torus for general buckling of the shell including the weld lands and "extra" weld land edge stringers.

wcold.BEHX2 = valid BIGBOSOR4 input file for the 180-degree torus for inter-ring buckling of the shell including the weld lands and "extra" weld land edge stringers.

(The five files, wcold.BEG, wcold.DEC, wcold.OPT, wcold.CHG, wcold.CPL, contain the following:)

wcold.BEG = input data for BEGIN — Table 13

wcold.DEC = input data for DECIDE — Table 14

wcold.OPT = input data for MAINSETUP and OPTIMIZE — Table 15

wcold.CHG = input data for CHANGE — Table 18

wcold.CPL = input data for CHOOSEPLOT — Table 16

(The two files, wcold.OPM and wcold.OPP contain the following:)

wcold.OPM = output from OPTIMIZE — Table 17

wcold.OPP = output from OPTIMIZE or SUPEROPT for ITYPE = 1 (optimization type of analysis)

****

(Next, we want to get BIGBOSOR4 output and plots corresponding to the three valid BIGBOSOR4 input files, wcold.BEHX0, wcold.BEHX1, and wcold.BEHX2.)

(First, copy the three files, wcold.BEHX0, wcold.BEHX1, wcold.BEHX2, into a directory from which you want to execute BIGBOSOR4:
cp wcold.BEHX0 /home/progs/bigbosor4/workspace/.
cp wcold.BEHX1 /home/progs/bigbosor4/workspace/.
cp wcold.BEHX2 /home/progs/bigbosor4/workspace/.)

(Next, go to that directory:
cd /home/progs/bigbosor4/workspace)

Table 4 (p. 4 of 5)

```
(Next, activate the BIGBOSOR4 commands, copy one of the *BEHX* files
 into wcold.ALL, and execute bigbosorall.)

bigbosor4log    (activate the BIGBOSOR4 commands)

cp wcold.BEHX0 wcold.ALL  (torus general buckling model with NO weld land)
bigbosorall     (execute BIGBOSOR4)

(Inspect the wcold.OUT file. Search for the string, "EIGENVALUE(".
 Then get a plot of the general buckling mode for the critical
 general buckling mode for the BIGBOSOR4 torus model with NO weld land.)

bosorplot       (choose which buckling mode to plot. Fig.4)
cleanup         (clean up the "wcold" files pertaining to BIGBOSOR4)

(Repeat the above for several of the general buckling modes for
 the BIGBOSOR4 model that includes the weld lands and "extra"
 weld land edge Tee-shaped stringers:)

cp wcold.BEHX1 wcold.ALL  (torus general buckling model with weld lands)
bigbosorall     (execute BIGBOSOR4)

(Inspect the wcold.OUT file. Search for the string, "EIGENVALUE(".
 Then get a plot of the general buckling mode for the critical
 general buckling mode for the BIGBOSOR4 torus model WITH weld lands.)

bosorplot       (choose which buckling mode to plot. Fig.5)
cleanup         (clean up the "wcold" files pertaining to BIGBOSOR4)

(Repeat the above for several of the inter-ring buckling modes for
 the BIGBOSOR4 model that includes the weld lands and "extra"
 weld land edge Tee-shaped stringers:)

cp wcold.BEHX2 wcold.ALL  (torus inter-ring buckling model with weld lands)
bigbosorall     (execute BIGBOSOR4)

(Inspect the wcold.OUT file. Search for the string, "EIGENVALUE(".
 Then get a plot of the general buckling mode for the critical
 general buckling mode for the BIGBOSOR4 torus model WITH weld lands.)

bosorplot       (choose which buckling mode to plot. Fig.8)
cleanup         (clean up the "wcold" files pertaining to BIGBOSOR4)
*************************************************************************

(Next, we want to find out what a different weld land model, the one
 in which there is only a single weld land and in which the shell acreage
 stringers are modeled as little shell branches, produces for general
 and inter-ring buckling of the shell with the optimized weld land and
 optimized "extra" weld land edge Tee-shaped stringers. This model is
 produced by the new PANDA2 processor called "bospn3", so we have to
 resurrect the "nasacoldbend" case from the PANDA2 cases, execute PANDA2
 followed by execution of BIGBOSOR4 for various models produced by "bospn3".)

(First, we go to a directory from which we wish to execute PANDA2, for example:)

cd /home/progs/panda2/workspace

(Then, we type the following commands:)

panda2log       (activate the PANDA2 command set)

cp /home/progs/panda2/case/nasacoldbend/nasacoldbend.tar .
tar xvf nasacoldbend.tar

begin           (get the starting design: nasacoldbend.BEG; Table a3)
change          (resurrect the optimum design: nasacoldbend.CHG; Table a5)
setup           (PANDA2 sets up matrix templates)
decide          (choose decision variables: nasacoldbend.DEC; Table a4)
(Edit the nasacoldbend.OPT file to make sure that ITYPE = 2; top of Table a8)
mainsetup       (establish loading, imperfection, strategy, etc.; top of Table a8)
pandaopt        (do the analysis for the fixed, that is, the optimum design.
                 See the bottom part of Table a8)
panel3          (execute "bospn3" for general buckling with shell acreage
                 stringers modeled as little shell branches; nasacoldbend.PAN=Table24)

(Next, run BIGBOSOR4 "huge torus" model corresponding to the nasacoldbend.ALL
 file produced by the previous execution of "panel3":)
```

Table 4 (p.5 of 5)

```
cp nasacoldbend.ALL /home/progs/bigbosor4/workspace/.
cd /home/progs/bigbosor4/workspace
bigbosor4log     (activate BIGBOSOR4 command set)
bigbosorall      (execute BIGBOSOR4 with nasacoldbend.ALL as input)
(Inspect the nasacoldbend.OUT file. Search for "EIGENVALUE("; Table 25) ← Table 25
bosorplot        (Fig. 10)
bosorplot        (Fig. 11)
bosorplot        (Fig. 12)        Figs 10-12
cleanup          (clean up nasacoldbend files pertaining to BIGBOSOR4)
--------------------------------------------------------------------------------

cd /home/progs/panda2/workspace
panel3           (execute "bospn3" for general buckling with shell acreage
                  stringers smeared out; nasacoldbend.PAN=Table 26) — Table 26

(Next, run BIGBOSOR4 "huge torus" model corresponding to the nasacoldbend.ALL
 file produced by the previous execution of "panel3":)

cp nasacoldbend.ALL /home/progs/bigbosor4/workspace/.
cd /home/progs/bigbosor4/workspace
bigbosorall      (execute BIGBOSOR4 with nasacoldbend.ALL as input)
(Inspect the nasacoldbend.OUT file. Search for "EIGENVALUE("; Table 27) ← Table 27
bosorplot        (Fig. 13)
bosorplot        (Fig. 14)        Figs 13-15
bosorplot        (Fig. 15)
cleanup          (clean up nasacoldbend files pertaining to BIGBOSOR4)
--------------------------------------------------------------------------------

cd /home/progs/panda2/workspace
panel3           (execute "bospn3" for inter-ring buckling with shell acreage
                  stringers modeled as little shell branches; nasacoldbend.PAN=Table 28)

(Next, run BIGBOSOR4 "huge torus" model corresponding to the nasacoldbend.ALL
 file produced by the previous execution of "panel3":)

cp nasacoldbend.ALL /home/progs/bigbosor4/workspace/.
cd /home/progs/bigbosor4/workspace
bigbosorall      (execute BIGBOSOR4 with nasacoldbend.ALL as input)
(Inspect the nasacoldbend.OUT file. Search for "EIGENVALUE("; Table 29) ← Table 29
bosorplot        (Fig. 16)
bosorplot        (Fig. 17)        Figs 16-18.
bosorplot        (Fig. 18)
cleanup          (clean up nasacoldbend files pertaining to BIGBOSOR4)
********************************************************************************
```

Tables weldland.INP (7 pages)

```
      5  $ starting prompt index in the file weldland.PRO
      5  $ increment for prompt index
      0  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
This GENOPT case is for optimization of a weld land
y        $ Are there more lines in the "help" paragraph?
in a stiffened cylindrical shell previously optimized by
y        $ Are there more lines in the "help" paragraph?
PANDA2 without any weld land. The decision variables are:
y        $ Are there more lines in the "help" paragraph?
WLAND = width of the weld land
y        $ Are there more lines in the "help" paragraph?
TLAND = thickness of the weld land
y        $ Are there more lines in the "help" paragraph?
and, if there is are stringers at the edges of the weld land:
y        $ Are there more lines in the "help" paragraph?
TWLAND = thickness of the web of the edge stringer
y        $ Are there more lines in the "help" paragraph?
HWLAND = height of the web of the edge stringer
y        $ Are there more lines in the "help" paragraph?
TFLAND = thickness of the outstanding flange of the stringer
y        $ Are there more lines in the "help" paragraph?
WFLAND = thickness of the width of the outstanding flange
y        $ Are there more lines in the "help" paragraph?
The stiffened cylindrical shell with the weld land is
y        $ Are there more lines in the "help" paragraph?
subjected to uniform axial compression, PX, and possibly
y        $ Are there more lines in the "help" paragraph?
also to uniform external or internal pressure, PY.
y        $ Are there more lines in the "help" paragraph?
The design constraints are:
y        $ Are there more lines in the "help" paragraph?
GENBUK = general buckling
y        $ Are there more lines in the "help" paragraph?
PANBUK = "panel" buckling (buckling between rings)
y        $ Are there more lines in the "help" paragraph?
STRESS = stress in the weld land or edge stringer
y        $ Are there more lines in the "help" paragraph?
The objective is minimum weight of the weld land and any
y        $ Are there more lines in the "help" paragraph?
edge stringers.
n        $ Are there more lines in the "help" paragraph?
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt

WLAND    $ Name of a variable in the users program (defined below)
      1  $ Role of the variable in the users program
n        $ Is the variable  WLAND  an array?
width of the weld land
y        $ Do you want to include a "help" paragraph?
WLAND is the circumferential dimension of the weld land
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
TLAND    $ Name of a variable in the users program (defined below)
      1  $ Role of the variable in the users program
n        $ Is the variable  TLAND  an array?
thickness of the weld land
n        $ Do you want to include a "help" paragraph?
y        $ Any more variables for role types  1  or  2   ?    $15
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
ECLAND   $ Name of a variable in the users program (defined below)
      1  $ Role of the variable in the users program
      2  $ type of variable:  1 =integer,  2 =floating point
n        $ Is the variable  ECLAND  an array?
eccentricity of the weld land
y        $ Do you want to include a "help" paragraph?
ECLAND is the distance from the middle surface of the
y        $ Any more lines in the "help" paragraph?
skin of the cylindrical shell to the middle surface of
y        $ Any more lines in the "help" paragraph?
the weld land, positive if the outer surfaces of the
y        $ Any more lines in the "help" paragraph?
weld land and the skin of the cylindrical shell are flush.
n        $ Any more lines in the "help" paragraph?
y        $ Any more variables for role types  1  or  2   ?    $20
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
KLAND    $ 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
```

Input for "GENTEXT"

Table 5 (p. 2 of 7)

```
n          $ Is the variable  KLAND  an array?
index for weld land edge stringer (0 or 1 or 2)
y          $ Do you want to include a "help" paragraph?
KLAND = 0 means no stringer at the edges of the weld land
y          $ Any more lines in the "help" paragraph?
KLAND = 1 means the weld land edge stringer has a
y          $ Any more lines in the "help" paragraph?
rectangular cross section
y          $ Any more lines in the "help" paragraph?
KLAND = 2 means the weld land edge stringer has a
y          $ Any more lines in the "help" paragraph?
Tee-shaped cross section
n          $ Any more lines in the "help" paragraph?
y          $ Any more variables for role types  1  or  2   ?    $25
       0  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
Next, you will be asked to supply the dimensions
y          $ Are there more lines in the "help" paragraph?
of the cross section of the stringers along the edges
y          $ Are there more lines in the "help" paragraph?
of the weld land: TWLAND, HWLAND, TFLAND, WFLAND
n          $ Are there more lines in the "help" paragraph?
       1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
TWLAND     $ Name of a variable in the users program (defined below)
       1  $ Role of the variable in the users program
n          $ Is the variable  TWLAND  an array?
web thickness of the weld land edge stringer
n          $ Do you want to include a "help" paragraph?
y          $ Any more variables for role types  1  or  2   ?    $35
       1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
HWLAND     $ Name of a variable in the users program (defined below)
       1  $ Role of the variable in the users program
n          $ Is the variable  HWLAND  an array?
height of the web of the weld land edge stringer
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
TFLAND     $ Name of a variable in the users program (defined below)
       1  $ Role of the variable in the users program
n          $ Is the variable  TFLAND  an array?
thickness of the outstanding flange of the stringer
n          $ Do you want to include a "help" paragraph?
y          $ Any more variables for role types  1  or  2   ?    $45
       1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
WFLAND     $ Name of a variable in the users program (defined below)
       1  $ Role of the variable in the users program
n          $ Is the variable  WFLAND  an array?
width of the outstanding flange of the stringer
n          $ Do you want to include a "help" paragraph?
y          $ Any more variables for role types  1  or  2   ?    $50
       1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
NLAND      $ 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  NLAND  an array?
number of weld lands in 360 degrees
y          $ Do you want to include a "help" paragraph?
Probably the maximum to use for NLAND is 6 (weld
y          $ Any more lines in the "help" paragraph?
land every 60 degrees.
n          $ Any more lines in the "help" paragraph?
y          $ Any more variables for role types  1  or  2   ?    $70
       0  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
Next, you will be asked to supply several dimensions
y          $ Are there more lines in the "help" paragraph?
that are not decision variables.
n          $ Are there more lines in the "help" paragraph?
       1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
TSKIN      $ Name of a variable in the users program (defined below)
       1  $ Role of the variable in the users program
       2  $ type of variable:  1 =integer,  2 =floating point
n          $ Is the variable  TSKIN  an array?
thickness of the cylindrical shell skin
y          $ Do you want to include a "help" paragraph?
NOTE: TSKIN is NOT a decision variable because it is
y          $ Any more lines in the "help" paragraph?
assumed here that TSKIN has a value obtained from a
y          $ Any more lines in the "help" paragraph?
previous optimization of the stiffened cylindrical
```

Table 5 (p. 3 of 7)

```
y         $ Any more lines in the "help" paragraph?
shell without a weld land.
n         $ Any more lines in the "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $80
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
RADCYL    $ 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  RADCYL  an array?
radius of the cylindrical shell
n         $ Do you want to include a "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $85
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
LENCYL    $ 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  LENCYL  an array?
length of the cylindrical shell
y         $ Do you want to include a "help" paragraph?
If the cylindrical shell is clamped, use as LENCYL
y         $ Any more lines in the "help" paragraph?
the following: LENCYL = LENMOD * AXIAL
y         $ Any more lines in the "help" paragraph?
in which LENMOD is the value of the "length modifier"
y         $ Any more lines in the "help" paragraph?
from PANDA2 and "AXIAL" is the actual axial length of
y         $ Any more lines in the "help" paragraph?
the cylindrical shell. You can find the value of
y         $ Any more lines in the "help" paragraph?
LENMOD by searching the PANDA2 *.OPM file for the
y         $ Any more lines in the "help" paragraph?
string, "LENMOD". If the cylindrical shell is
y         $ Any more lines in the "help" paragraph?
simply supported along its curved ends, then use
y         $ Any more lines in the "help" paragraph?
the actual length, "AXIAL", for LENCYL.
n         $ Any more lines in the "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $90
      0  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
Next, you will be asked to provide the stiffener
y         $ Are there more lines in the "help" paragraph?
thicknesses and heights. These are the stiffeners that
y         $ Are there more lines in the "help" paragraph?
are in the acreage of the cylindrical shell, not the
y         $ Are there more lines in the "help" paragraph?
"extra" stringers located at the edges of each weld land.
y         $ Are there more lines in the "help" paragraph?
NOTE: These stiffeners are assumed to have rectangular
y         $ Are there more lines in the "help" paragraph?
cross sections:
y         $ Are there more lines in the "help" paragraph?
for the stringers: BSTR, TSTR, HSTR (spacing, thickness,
y         $ Are there more lines in the "help" paragraph?
height of the stringers)
y         $ Are there more lines in the "help" paragraph?
for the rings: BRNG, TRNG, HRNG (spacing, thickness,
y         $ Are there more lines in the "help" paragraph?
height of the rings).
y         $ Are there more lines in the "help" paragraph?
These quantities are NOT decision variables in this
y         $ Are there more lines in the "help" paragraph?
application because it is assumed that they have
y         $ Are there more lines in the "help" paragraph?
optimum values previously obtained from PANDA2 in the
y         $ Are there more lines in the "help" paragraph?
model of the cylindrical shell without the weld land(s).
n         $ Are there more lines in the "help" paragraph?
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
BSTR      $ 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  BSTR  an array?
stringer spacing
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
TSTR      $ Name of a variable in the users program (defined below)
      2  $ Role of the variable in the users program
```

Table 5 (p. 4 of 7)

```
      2  $ type of variable:  1 =integer,  2 =floating point
n         $ Is the variable  TSTR  an array?
stringer thickness
n         $ Do you want to include a "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $105
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
HSTR      $ 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  HSTR  an array?
stringer height
n         $ Do you want to include a "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $110
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
BRNG      $ 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  BRNG  an array?
ring spacing
n         $ Do you want to include a "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $115
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
TRNG      $ 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  TRNG  an array?
ring thickness
n         $ Do you want to include a "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $120
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
HRNG      $ 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  HRNG  an array?
ring height
n         $ Do you want to include a "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $125
      0  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
Next, you will be asked to provide material properties.
y         $ Are there more lines in the "help" paragraph?
It is assumed that the material is elastic and that the
y         $ Are there more lines in the "help" paragraph?
same material is used for the entire structure.
n         $ Are there more lines in the "help" paragraph?
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
EMOD      $ 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  EMOD  an array?
Young's modulus
n         $ Do you want to include a "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $135
      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
n         $ Do you want to include a "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $140
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
DENS      $ 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  DENS  an array?
mass density (aluminum = 0.00025 in English units)
n         $ Do you want to include a "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $145
      0  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
Next, you will be asked to provide ranges of buckling
y         $ Are there more lines in the "help" paragraph?
axial half waves for the search for a critical
y         $ Are there more lines in the "help" paragraph?
number of axial half waves in the buckling mode
y         $ Are there more lines in the "help" paragraph?
first for general buckling, then for "panel" buckling
y         $ Are there more lines in the "help" paragraph?
"panel" buckling = buckling between adjacent rings with
```

Tables (p.5 of 7)

```
y         $ Are there more lines in the "help" paragraph?
the stringers smeared out).
n         $ Are there more lines in the "help" paragraph?
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
MLOWG     $ 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  MLOWG  an array?
low end of the M-range for general buckling
y         $ Do you want to include a "help" paragraph?
You start by useing MLOWG = 1
n         $ Any more lines in the "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $155
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
MHIGHG    $ 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  MHIGHG  an array?
High end of the M-range for general buckling
y         $ Do you want to include a "help" paragraph?
You might have to experiment with this in order
y         $ Any more lines in the "help" paragraph?
to find an appropriate value. The wider the M-range
y         $ Any more lines in the "help" paragraph?
the more computer time is required.
n         $ Any more lines in the "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $160
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
MLOWP     $ 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  MLOWP  an array?
low end of the M-range for "panel" buckling
y         $ Do you want to include a "help" paragraph?
"Panel" buckling is buckling between adjacent rings
y         $ Any more lines in the "help" paragraph?
with the stringers smeared out.
n         $ Any more lines in the "help" paragraph?
y         $ Any more variables for role types  1  or  2   ?    $165
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
MHIGHP    $ 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  MHIGHP  an array?
high end of the M-range for "panel" buckling
y         $ Do you want to include a "help" paragraph?
Don't use too high a number. Computer time increases
y         $ Any more lines in the "help" paragraph?
the wider your M-range for "panel" buckling.
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
Next, supply the loading
n         $ Are there more lines in the "help" paragraph?
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
PX        $ Name of a variable in the users program (defined below)
      3  $ Role of the variable in the users program
total axial load (2*pi*r*Nx)
y         $ Do you want to include a "help" paragraph?
PX = the total axial load, such as 2 x pi x r x Nx, in
y         $ Any more lines in the "help" paragraph?
which Nx is an axial resultant.
y         $ Any more lines in the "help" paragraph?
NEGATIVE FOR AXIAL COMPRESSION.
n         $ Any more lines in the "help" paragraph?
y         $ Any more variables for role type  3 ?              $185
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
PY        $ Name of a variable in the users program (defined below)
      3  $ Role of the variable in the users program
uniform normal pressure
y         $ Do you want to include a "help" paragraph?
PY is the pressure, assumed to be applied only to the
y         $ Any more lines in the "help" paragraph?
curved surface of the cylindrical shell.
y         $ Any more lines in the "help" paragraph?
NEGATIVE FOR COMPRESSION (external pressure).
n         $ Any more lines in the "help" paragraph?
y         $ Any more variables for role type  3 ?              $190
```

Tables (p. 6 of 7)

```
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
PX0       $ Name of a variable in the users program (defined below)
      3  $ Role of the variable in the users program
axial load in Load Set B
y         $ Do you want to include a "help" paragraph?
PX0 affects the stiffness matrix only. It is not an
y         $ Any more lines in the "help" paragraph?
"eigenvalue" load.
y         $ Any more lines in the "help" paragraph?
NEGATIVE FOR AXIAL COMPRESSION.
n         $ Any more lines in the "help" paragraph?
y         $ Any more variables for role type  3 ?              $195
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
PY0       $ Name of a variable in the users program (defined below)
      3  $ Role of the variable in the users program
pressure in Load Set B
y         $ Do you want to include a "help" paragraph?
PY0 = pressure on the curved surface only. This load
y         $ Any more lines in the "help" paragraph?
affects the stiffness matrix only. PY0 is not an
y         $ Any more lines in the "help" paragraph?
"eigenvalue" load.
y         $ Any more lines in the "help" paragraph?
NEGATIVE FOR COMPRESSION (external pressure).
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
Next, you must provide an allowable and a factor of
y         $ Are there more lines in the "help" paragraph?
safety for each Role 4 variable, that is, for each
y         $ Are there more lines in the "help" paragraph?
"behavior" (such as general buckling, "panel" buckling,
y         $ Are there more lines in the "help" paragraph?
and stress).
n         $ Are there more lines in the "help" paragraph?
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
GENBUK    $ Name of a variable in the users program (defined below)
      4  $ Role of the variable in the users program
n         $ Do you want to reset the number of columns in GENBUK ?
general buckling load
y         $ Do you want to include a "help" paragraph?
In the model for the general buckling both stringers and
y         $ Any more lines in the "help" paragraph?
rings are smeared out. NOTE: The "extra" stringers along
y         $ Any more lines in the "help" paragraph?
eadh edge of the weld land are ALWAYS modeled as
y         $ Any more lines in the "help" paragraph?
flexible shell segments.
n         $ Any more lines in the "help" paragraph?
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
GENBUKA   $ Name of a variable in the users program (defined below)
      5  $ Role of the variable in the users program
allowable for general buckling
n         $ Do you want to include a "help" paragraph?
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
GENBUKF   $ Name of a variable in the users program (defined below)
      6  $ Role of the variable in the users program
factor of safety for general buckling
n         $ Do you want to include a "help" paragraph?
      2  $ Indicator (1 or 2 or 3) for type of constraint
y         $ Any more variables for role type  4 ?              $220
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
PANBUK    $ Name of a variable in the users program (defined below)
      4  $ Role of the variable in the users program
n         $ Do you want to reset the number of columns in PANBUK ?
"panel" buckling
y         $ Do you want to include a "help" paragraph?
PANBUK is buckling between adjacent rings with the
y         $ Any more lines in the "help" paragraph?
stringers smeared out.
n         $ Any more lines in the "help" paragraph?
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
PANBUKA   $ Name of a variable in the users program (defined below)
      5  $ Role of the variable in the users program
allowable for "panel" buckling
n         $ Do you want to include a "help" paragraph?
      1  $ Type of prompt: 0="help" paragraph, 1=one-line prompt
PANBUKF   $ Name of a variable in the users program (defined below)
```

Tables (p. 7 of 7)

```
      6  $ Role of the variable in the users program
factor of safety for "panel" buckling
n         $ Do you want to include a "help" paragraph?
      2  $ Indicator (1 or 2 or 3) for type of constraint
y         $ Any more variables for role type  4 ?              $235
      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
n         $ Do you want to reset the number of columns in STRESS ?
weld land effective stress
y         $ Do you want to include a "help" paragraph?
This is the effective stress in the weld land or
y         $ Any more lines in the "help" paragraph?
in the "extra" stringers at the edges of the
y         $ Any more lines in the "help" paragraph?
weld land.
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
n         $ Do you want to include a "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 effective stress
n         $ Do you want to include a "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, provide the objective.
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 weld land+"extra" edge stringers
y         $ Do you want to include a "help" paragraph?
WEIGHT = weight of the weld land of width WLAND and
y         $ Any more lines in the "help" paragraph?
thickness TLAND plus the weight of the extra
y         $ Any more lines in the "help" paragraph?
stringers that run along the two axial edges of
y         $ Any more lines in the "help" paragraph?
the weld land.
n         $ Any more lines in the "help" paragraph?
```

input for GENTEXT

Table 6 (3 pages)

from weldland.DEF

```
C=============================================================
C   TABLE 6   GLOSSARY OF VARIABLES USED IN "weldland"
C=============================================================
C  ARRAY  NUMBER OF          PROMPT
C    ?   (ROWS,COLS)  ROLE  NUMBER   NAME                DEFINITION OF VARIABLE
C                          (weldland.PRO)
C=============================================================
C    n   (   0,   0)    1      10   WLAND    = width of the weld land
C    n   (   0,   0)    1      15   TLAND    = thickness of the weld land
C    n   (   0,   0)    1      20   ECLAND   = eccentricity of the weld land
C    n   (   0,   0)    2      25   KLAND    = index for weld land edge stringer (0 or 1 or 2)
C    n   (   0,   0)    1      35   TWLAND   = web thickness of the weld land edge stringer
C    n   (   0,   0)    1      40   HWLAND   = height of the web of the weld land edge stringer
C    n   (   0,   0)    1      45   TFLAND   = thickness of the outstanding flange of the string»
er
C    n   (   0,   0)    1      50   WFLAND   = width of the outstanding flange of the stringer
C    n   (   0,   0)    2      55   NLAND    = number of weld lands in 360 degrees
C    n   (   0,   0)    1      65   TSKIN    = thickness of the cylindrical shell skin
C    n   (   0,   0)    2      70   RADCYL   = radius of the cylindrical shell
C    n   (   0,   0)    2      75   LENCYL   = length of the cylindrical shell
C    n   (   0,   0)    2      85   BSTR     = stringer spacing
C    n   (   0,   0)    2      90   TSTR     = stringer thickness
C    n   (   0,   0)    2      95   HSTR     = stringer height
C    n   (   0,   0)    2     100   BRNG     = ring spacing
C    n   (   0,   0)    2     105   TRNG     = ring thickness
C    n   (   0,   0)    2     110   HRNG     = ring height
C    n   (   0,   0)    2     120   EMOD     = Young's modulus
C    n   (   0,   0)    2     125   NU       = Poisson ratio
C    n   (   0,   0)    2     130   DENS     = mass density (aluminum = 0.00025 in English units»
)
C    n   (   0,   0)    2     140   MLOWG    = low end of the M-range for general buckling
C    n   (   0,   0)    2     145   MHIGHG   = High end of the M-range for general buckling
C    n   (   0,   0)    2     150   MLOWP    = low end of the M-range for "panel" buckling
C    n   (   0,   0)    2     155   MHIGHP   = high end of the M-range for "panel" buckling
C    n   (   0,   0)    2     165   NCASES   = Number of load cases (number of environments)  in»
 PX(NCASES)
C    y   (  20,   0)    3     170   PX       = total axial load (2*pi*r*Nx)
C    y   (  20,   0)    3     175   PY       = uniform normal pressure
C    y   (  20,   0)    3     180   PX0      = axial load in Load Set B
C    y   (  20,   0)    3     185   PY0      = pressure in Load Set B
C    y   (  20,   0)    4     195   GENBUK   = general buckling load
C    y   (  20,   0)    5     200   GENBUKA  = allowable for general buckling
C    y   (  20,   0)    6     205   GENBUKF  = factor of safety for general buckling
C    y   (  20,   0)    4     210   PANBUK   = "panel" buckling
C    y   (  20,   0)    5     215   PANBUKA  = allowable for "panel" buckling
C    y   (  20,   0)    6     220   PANBUKF  = factor of safety for "panel" buckling
C    y   (  20,   0)    4     225   STRESS   = weld land effective stress
C    y   (  20,   0)    5     230   STRESSA  = maximum allowable effective stress
C    y   (  20,   0)    6     235   STRESSF  = factor of safety for effective stress
C    n   (   0,   0)    7     245   WEIGHT   = weight of the weld land+"extra" edge stringers
C
C=============================================================
C       TABLE 7   SEVEN ROLES THAT VARIABLES 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 ============================================================

 The purpose of GENTEXT is to generate a file of
 prompting phrases and helps called weldland.PRO and five
 FORTRAN source libraries, BEGIN.NEW, STOGET.NEW, STRUCT.NEW,
```

Table 6 (p. 2 of 3)

BEHAVIOR.NEW, and CHANGE.NEW. The purposes of these files are as follows:

TABLE 4 FILE OF PROMPTING PHRASES AND HELPS AND SOURCE CODE LIBRARIES GENERATED BY "GENTEXT"

weldland.PRO	= prompt file for input data for the problem class that you wish to set up for optimization. When BEGIN asks you for the name of the generic file, you should respond in this case with weldland. The Prompt Numbers listed in TABLE 2 correspond to the prompts in this file.
BEGIN.NEW	= source library for FORTRAN program which will be used to set up the starting design, material properties, and any other data you wish.
STOGET.NEW	= source library for FORTRAN subroutines which are used to transfer labelled common blocks. These labelled common blocks are the data base.
STRUCT.NEW	= source library for FORTRAN subroutines that perform the analysis for each iterate in the set of optimization iterations. You may have to complete this routine (add dimension statements, subroutine calls, output statements, etc.). The library, STRUCT.NEW, also contains a skeletal routine, SUB. TRANFR, that you can complete in order to translate data names from from those just established by you (TABLE 2) to other names used by the developer of previously written code that you may plan to incorporate into SUBROUTINE STRUCT and/or SUBROUTINES BEHX1, BEHX2, BEHX3,...BEHXn (described next).
BEHAVIOR.NEW=	a library of subroutine skeletons, BEHX1,BEHX2, BEHX3,...BEHXn, that, upon completion by you, will calculate behavior for a given design or design perturbation. Skeletal subroutines for a user-written constraint condition, USRCON, and a skeletal routine for the objective function, OBJECT, are also generated and are included in the BEHAVIOR.NEW library.
CHANGE.NEW	= FORTRAN program that permits you to change certain program parameters without having to go back to BEGIN and run a case from scratch.

TABLE 5: CONTENTS OF SMALL FILES CREATED BY "GENTEXT"

FILE NAME	DEFINITION OF FILE CONTENTS
weldland.PRO	Prompts and help paragraphs for interactive input to the user-developed optimization code.
weldland.NEW	Part of BEGIN.NEW that contains calls to SUBROUTINE DATUM and SUBROUTINE GETVAR. This coding sets up the interactive input for the starting design in the user-generated design code.
weldland.INP	Image of interactive input for user-developed program, generated to save time in case you make a mistake during input.
weldland.COM	Labelled common blocks generated specifically for the user-developed class of problems.
weldland.WRI	Part of subroutine for writing labelled common blocks in SUBROUTINE STORCM (in Library STOGET).

Table 6 (p. 3 of 3)

weldland.REA	Part of subroutine for reading labelled common blocks in SUBROUTINE GETCOM (in Library STOGET).
weldland.SET	Part of SUBROUTINE SETUPC in which new values are installed in labelled common blocks from the array VAR(I), which contains the latest values of all candidates for decision variables.
weldland.CON	Calls to subroutines, BEHX1, BEHX2, BEHX3,..., which calculate behavior such as stresses modal frequencies, buckling loads, etc. Also, calls to CON, which generate the value of the behavioral constraints corresponding to BEHX1, BEHX2, BEHX3,... Also, generates phrases that identify, in the output of the user-generated program, the exact meaning of each behavioral constraint.
weldland.SUB	Skeletal subroutines, BEHX1, BEHX2, ..., and the skeletal objective function, OBJECT.
weldland.DEF	List of user-established variable names, definitions, and roles that these variables play in the user-generated program. Also, contains list of files created by GENTEXT and the functions of these files.
weldland.CHA	Part of SUBROUTINE NEWPAR (called in the CHANGE processor) in which labelled common values are updated.
weldland.DAT	Image of interactive input for user-developed program, generated to save time in case you make a mistake during input. This file is used by the INSERT processor.

weldland.PRO file:

Table 7 (3 pages)

```
 5.0
      This GENOPT case is for optimization of a weld land
      in a stiffened cylindrical shell previously optimized by
      PANDA2 without any weld land. The decision variables are:
      WLAND = width of the weld land
      TLAND = thickness of the weld land
      and, if there is are stringers at the edges of the weld land:
      TWLAND = thickness of the web of the edge stringer
      HWLAND = height of the web of the edge stringer
      TFLAND = thickness of the outstanding flange of the stringer
      WFLAND = thickness of the width of the outstanding flange
      The stiffened cylindrical shell with the weld land is
      subjected to uniform axial compression, PX, and possibly
      also to uniform external or internal pressure, PY.
      The design constraints are:
      GENBUK = general buckling
      PANBUK = "panel" buckling (buckling between rings)
      STRESS = stress in the weld land or edge stringer
      The objective is minimum weight of the weld land and any
      edge stringers.

10.1 width of the weld land: WLAND
10.2
      WLAND is the circumferential dimension of the weld land

15.1 thickness of the weld land: TLAND
20.1 eccentricity of the weld land: ECLAND
20.2
      ECLAND is the distance from the middle surface of the
      skin of the cylindrical shell to the middle surface of
      the weld land, positive if the outer surfaces of the
      weld land and the skin of the cylindrical shell are flush.

25.1 index for weld land edge stringer (0 or 1 or 2): KLAND
25.2
      KLAND = 0 means no stringer at the edges of the weld land
      KLAND = 1 means the weld land edge stringer has a
      rectangular cross section
      KLAND = 2 means the weld land edge stringer has a
      Tee-shaped cross section


30.0
      Next, you will be asked to supply the dimensions
      of the cross section of the stringers along the edges
      of the weld land: TWLAND, HWLAND, TFLAND, WFLAND

35.1 web thickness of the weld land edge stringer: TWLAND
40.1 height of the web of the weld land edge stringer: HWLAND
45.1 thickness of the outstanding flange of the stringer: TFLAND
50.1 width of the outstanding flange of the stringer: WFLAND
55.1 number of weld lands in 360 degrees: NLAND
55.2
      Probably the maximum to use for NLAND is 6 (weld
      land every 60 degrees.


60.0
      Next, you will be asked to supply several dimensions
      that are not decision variables.

65.1 thickness of the cylindrical shell skin: TSKIN
65.2
      NOTE: TSKIN is NOT a decision variable because it is
      assumed here that TSKIN has a value obtained from a
      previous optimization of the stiffened cylindrical
      shell without a weld land.

70.1 radius of the cylindrical shell: RADCYL
75.1 length of the cylindrical shell: LENCYL
75.2
      If the cylindrical shell is clamped, use as LENCYL
      the following: LENCYL = LENMOD * AXIAL
      in which LENMOD is the value of the "length modifier"
      from PANDA2 and "AXIAL" is the actual axial length of
```

The prompting file produced automatically by GENTEXT. The data names, one-line definitions, & "help" paragraphs are created by the "GENOPT user" & seen by the "end user".

Table 7 (p. 2 of 3)

```
      the cylindrical shell. You can find the value of
      LENMOD by searching the PANDA2 *.OPM file for the
      string, "LENMOD". If the cylindrical shell is
      simply supported along its curved ends, then use
      the actual length, "AXIAL", for LENCYL.


 80.0
      Next, you will be asked to provide the stiffener
      thicknesses and heights. These are the stiffeners that
      are in the acreage of the cylindrical shell, not the
      "extra" stringers located at the edges of each weld land.
      NOTE: These stiffeners are assumed to have rectangular
      cross sections:
      for the stringers: BSTR, TSTR, HSTR (spacing, thickness,
      height of the stringers)
      for the rings: BRNG, TRNG, HRNG (spacing, thickness,
      height of the rings).
      These quantities are NOT decision variables in this
      application because it is assumed that they have
      optimum values previously obtained from PANDA2 in the
      model of the cylindrical shell without the weld land(s).

 85.1 stringer spacing: BSTR
 90.1 stringer thickness: TSTR
 95.1 stringer height: HSTR
100.1 ring spacing: BRNG
105.1 ring thickness: TRNG
110.1 ring height: HRNG

115.0
      Next, you will be asked to provide material properties.
      It is assumed that the material is elastic and that the
      same material is used for the entire structure.

120.1 Young's modulus: EMOD
125.1 Poisson ratio: NU
130.1 mass density (aluminum = 0.00025 in English units): DENS

135.0
      Next, you will be asked to provide ranges of buckling
      axial half waves for the search for a critical
      number of axial half waves in the buckling mode
      first for general buckling, then for "panel" buckling
      "panel" buckling = buckling between adjacent rings with
      the stringers smeared out).

140.1 low end of the M-range for general buckling: MLOWG
140.2
      You start by useing MLOWG = 1

145.1 High end of the M-range for general buckling: MHIGHG
145.2
      You might have to experiment with this in order
      to find an appropriate value. The wider the M-range
      the more computer time is required.

150.1 low end of the M-range for "panel" buckling: MLOWP
150.2
      "Panel" buckling is buckling between adjacent rings
      with the stringers smeared out.

155.1 high end of the M-range for "panel" buckling: MHIGHP
155.2
      Don't use too high a number. Computer time increases
      the wider your M-range for "panel" buckling.


160.0
      Next, supply the loading

165.1 Number NCASES  of load cases (environments): NCASES
170.1 total axial load (2*pi*r*Nx): PX
170.2
      PX = the total axial load, such as 2 x pi x r x Nx, in
      which Nx is an axial resultant.
      NEGATIVE FOR AXIAL COMPRESSION.
```

Table 7 (p. 313)

```
175.1 uniform normal pressure: PY
175.2
      PY is the pressure, assumed to be applied only to the
      curved surface of the cylindrical shell.
      NEGATIVE FOR COMPRESSION (external pressure).

180.1 axial load in Load Set B: PX0
180.2
      PX0 affects the stiffness matrix only. It is not an
      "eigenvalue" load.
      NEGATIVE FOR AXIAL COMPRESSION.

185.1 pressure in Load Set B: PY0
185.2
      PY0 = pressure on the curved surface only. This load
      affects the stiffness matrix only. PY0 is not an
      "eigenvalue" load.
      NEGATIVE FOR COMPRESSION (external pressure).


190.0
      Next, you must provide an allowable and a factor of
      safety for each Role 4 variable, that is, for each
      "behavior" (such as general buckling, "panel" buckling,
      and stress).

195.0 general buckling load: GENBUK
195.2
      In the model for the general buckling both stringers and
      rings are smeared out. NOTE: The "extra" stringers along
      eadh edge of the weld land are ALWAYS modeled as
      flexible shell segments.

200.1 allowable for general buckling: GENBUKA
205.1 factor of safety for general buckling: GENBUKF
210.0 "panel" buckling: PANBUK
210.2
      PANBUK is buckling between adjacent rings with the
      stringers smeared out.

215.1 allowable for "panel" buckling: PANBUKA
220.1 factor of safety for "panel" buckling: PANBUKF
225.0 weld land effective stress: STRESS
225.2
      This is the effective stress in the weld land or
      in the "extra" stringers at the edges of the
      weld land.

230.1 maximum allowable effective stress: STRESSA
235.1 factor of safety for effective stress: STRESSF

240.0
      Next, provide the objective.

245.0 weight of the weld land+"extra" edge stringers: WEIGHT
245.2
      WEIGHT = weight of the weld land of width WLAND and
      thickness TLAND plus the weight of the extra
      stringers that run along the two axial edges of
      the weld land.

999.0 DUMMY ENTRY TO MARK END OF FILE
```

Table 8 (5 pages)

```
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/FV01/WLAND,TLAND,ECLAND,TWLAND,HWLAND,TFLAND,WFLAND,TSKIN
       REAL WLAND,TLAND,ECLAND,TWLAND,HWLAND,TFLAND,WFLAND,TSKIN
       COMMON/FV09/RADCYL,LENCYL,BSTR,TSTR,HSTR,BRNG,TRNG,HRNG,EMOD
       REAL RADCYL,LENCYL,BSTR,TSTR,HSTR,BRNG,TRNG,HRNG,EMOD
       COMMON/FV20/PX(20)
       REAL PX
       COMMON/FV26/GENBUK(20),GENBUKA(20),GENBUKF(20)
       REAL GENBUK,GENBUKA,GENBUKF
       COMMON/FV29/PANBUK(20),PANBUKA(20),PANBUKF(20)
       REAL PANBUK,PANBUKA,PANBUKF
       COMMON/FV32/STRESS(20),STRESSA(20),STRESSF(20)
       REAL STRESS,STRESSA,STRESSF
       COMMON/IV01/KLAND,NLAND,MLOWG,MHIGHG,MLOWP,MHIGHP
       INTEGER KLAND,NLAND,MLOWG,MHIGHG,MLOWP,MHIGHP
       COMMON/FV18/NU,DENS,WEIGHT
       REAL NU,DENS,WEIGHT
       COMMON/FV21/PY(20),PX0(20),PY0(20)
       REAL PY,PX0,PY0
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.
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).
```

Except for the encircled statements, this file was created automatically by GENOPT (GENTEXT).

Table 8 (p. 2 of 5)

```
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.
C
         IF (NINEQX.GT.0)
     1        CALL VARCON(WORDIQ,WORDMX,CINEQX,DPWREQ,IDINEQ,
     1        NINEQX,JINEQX,IEQTYP,INUMTT,IMODX,CONMAX,IPOINC,
```

Table 8 (p. 3 of 5)

```
     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
C
C     CALL OPNGEN
C     CALL RWDGEN
C
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 GENBUK:
C  GENBUK = general buckling load
C
        PHRASE =
     1 'general buckling load'
        CALL BLANKX(PHRASE,IENDP4)
        IF (IBEHV(1  ).EQ.0) CALL BEHX1
     1 (IFILE8,NPRINX,IMODX,IFAST,ILOADX  ,
     1 'general buckling load')
        IF (GENBUK(ILOADX  ).EQ.0.)  GENBUK(ILOADX  ) = 1.E+10
        IF (GENBUKA(ILOADX  ).EQ.0.)  GENBUKA(ILOADX  ) = 1.0
        IF (GENBUKF(ILOADX  ).EQ.0.)  GENBUKF(ILOADX  ) = 1.0
        KCONX = KCONX + 1
        CARX(KCONX) =GENBUK(ILOADX  )
        WORDCX= '(GENBUK(''//CIX//'')/GENBUKA(''//CIX//
     1  '')) / GENBUKF(''//CIX//'')'
        CALL CONX(GENBUK(ILOADX  ),GENBUKA(ILOADX  ),GENBUKF(ILOADX  )
     1,'general buckling load',
     1 'allowable for general buckling',
     1 'factor of safety for general buckling',
     1 2,INUMTT,IMODX,CONMAX,ICONSX,IPOINC,CONSTX,WORDCX,
     1 WORDMX,PCWORD,CPLOTX,ICARX)
        IF (IMODX.EQ.0) THEN
           CODPHR =
     1 '  general buckling load: '
           IENDP4 =25
           CODNAM ='GENBUK(''//CIX//'')'
           MLET4 =6 + 4
           WORDBX(KCONX)= CODPHR(1:IENDP4)//CODNAM(1:MLET4)
           IF (NPRINX.GT.0) WRITE(IFILE8,'(I5,6X,G14.7,A,A)')
```

The "GENOPT user" inserted the two lines & one more indicated on p. 5 of this table.

Table 8 (p. 4 of 5)

```
     1    KCONX,CARX(KCONX),CODPHR(1:IENDP4),CODNAM(1:MLET4)
       ENDIF
  205 CONTINUE
  206 CONTINUE
C
C  Behavior and constraints generated next for PANBUK:
C  PANBUK = "panel" buckling
C
       PHRASE =
     1 '"panel" buckling'
       CALL BLANKX(PHRASE,IENDP4)
       IF (IBEHV(2  ).EQ.0) CALL BEHX2
     1 (IFILE8,NPRINX,IMODX,IFAST,ILOADX  ,
     1 '"panel" buckling')
       IF (PANBUK(ILOADX  ).EQ.0.)  PANBUK(ILOADX  ) = 1.E+10
       IF (PANBUKA(ILOADX  ).EQ.0.)  PANBUKA(ILOADX  ) = 1.0
       IF (PANBUKF(ILOADX  ).EQ.0.)  PANBUKF(ILOADX  ) = 1.0
       KCONX = KCONX + 1
       CARX(KCONX) =PANBUK(ILOADX  )
       WORDCX= '(PANBUK(''//CIX//'')/PANBUKA(''//CIX//
     1  '')) / PANBUKF(''//CIX//'')'
       CALL CONX(PANBUK(ILOADX  ),PANBUKA(ILOADX  ),PANBUKF(ILOADX  )
     1,'"panel" buckling',
     1 'allowable for "panel" buckling',
     1 'factor of safety for "panel" buckling',
     1 2,INUMTT,IMODX,CONMAX,ICONSX,IPOINC,CONSTX,WORDCX,
     1 WORDMX,PCWORD,CPLOTX,ICARX)
       IF (IMODX.EQ.0) THEN
          CODPHR =
     1 '  "panel" buckling: '
          IENDP4 =20
          CODNAM ='PANBUK(''//CIX//'')'
          MLET4 =6 + 4
          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
  220 CONTINUE
  221 CONTINUE
C
C  Behavior and constraints generated next for STRESS:
C  STRESS = weld land effective stress
C
       PHRASE =
     1 'weld land effective stress'
       CALL BLANKX(PHRASE,IENDP4)
       IF (IBEHV(3  ).EQ.0) CALL BEHX3
     1 (IFILE8,NPRINX,IMODX,IFAST,ILOADX  ,
     1 'weld land effective stress')
       IF (STRESS(ILOADX  ).EQ.0.)  STRESS(ILOADX  ) = 1.E-10
       IF (STRESSA(ILOADX  ).EQ.0.)  STRESSA(ILOADX  ) = 1.0
       IF (STRESSF(ILOADX  ).EQ.0.)  STRESSF(ILOADX  ) = 1.0
       KCONX = KCONX + 1
       CARX(KCONX) =STRESS(ILOADX  )
       WORDCX= '(STRESSA(''//CIX//'')/STRESS(''//CIX//
     1  '')) / STRESSF(''//CIX//'')'
       CALL CONX(STRESS(ILOADX  ),STRESSA(ILOADX  ),STRESSF(ILOADX  )
     1,'weld land effective stress',
     1 'maximum allowable effective stress',
     1 'factor of safety for effective stress',
     1 3,INUMTT,IMODX,CONMAX,ICONSX,IPOINC,CONSTX,WORDCX,
     1 WORDMX,PCWORD,CPLOTX,ICARX)
       IF (IMODX.EQ.0) THEN
          CODPHR =
     1 '  weld land effective stress: '
          IENDP4 =30
          CODNAM ='STRESS(''//CIX//'')'
          MLET4 =6 + 4
          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
  235 CONTINUE
  236 CONTINUE
C
C  NEXT, EVALUATE THE OBJECTIVE, OBJGEN:
       IF (ILOADX.EQ.1) THEN
          PHRASE ='weight of the weld land+"extra" edge stringers'
```

Table 8 (p. 5 of 5)

```
         CALL BLANKX(PHRASE,IENDP4)
         CALL OBJECT(IFILE8,NPRINX,IMODX,OBJGEN,
     1    'weight of the weld land+"extra" edge stringers')
       ENDIF
       NCONSX = ICONSX
C
       CALL CLSGEN
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
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/FV01/WLAND,TLAND,ECLAND,TWLAND,HWLAND,TFLAND,WFLAND,TSKIN
      REAL WLAND,TLAND,ECLAND,TWLAND,HWLAND,TFLAND,WFLAND,TSKIN
      COMMON/FV09/RADCYL,LENCYL,BSTR,TSTR,HSTR,BRNG,TRNG,HRNG,EMOD
      REAL RADCYL,LENCYL,BSTR,TSTR,HSTR,BRNG,TRNG,HRNG,EMOD
      COMMON/FV20/PX(20)
      REAL PX
      COMMON/FV26/GENBUK(20),GENBUKA(20),GENBUKF(20)
      REAL GENBUK,GENBUKA,GENBUKF
      COMMON/FV29/PANBUK(20),PANBUKA(20),PANBUKF(20)
      REAL PANBUK,PANBUKA,PANBUKF
      COMMON/FV32/STRESS(20),STRESSA(20),STRESSF(20)
      REAL STRESS,STRESSA,STRESSF
      COMMON/IV01/KLAND,NLAND,MLOWG,MHIGHG,MLOWP,MHIGHP
      INTEGER KLAND,NLAND,MLOWG,MHIGHG,MLOWP,MHIGHP
      COMMON/FV18/NU,DENS,WEIGHT
      REAL NU,DENS,WEIGHT
      COMMON/FV21/PY(20),PX0(20),PY0(20)
      REAL PY,PX0,PY0
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
```

"GENOPT user" inserted this statement.

Table 9

Results from the command:
diff struct.weldland struct.new > struct.diff
The following is the struct.diff file.

```
==========================================
195,199d194
< C
<       CALL OPNGEN
<       CALL RWDGEN
< C
< C
319d313
<       CALL CLSGEN
```

(1) struct.new is created automatically by GENOPT

(2) The three lines inserted by the "GENOPT user" are required if the BIGBOSOR4 software is to be used in the optimization loop, as is the situation here.

(3) For other applications involving BIGBOSOR4 you can modify ("flesh out") the "skeletal" version of struct.new generated automatically by GENOPT in exactly the same way as is done here. (simple)

Table 10 (13 pages)

```
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
```

(first 5 pages of this Table is Introduction written automatically by GENOPT (GENTEXT).

Table 10 (p. 2 of 13)

```
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)

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

C           DIPLOT    (User generates plots)

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

Table 10 (p. 3 of 13)

```
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 10 (p. 4 of 13)

```
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 [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

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

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

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

C [17] 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,
```

Table 10 (p. 5 of 13)

```
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 = weldland)
C        4.  GENOPT.HLP file     (type HELPG)
C=============================================================

C=================================================================
C   TABLE 2   GLOSSARY OF VARIABLES USED IN "weldland"
C=================================================================
C  ARRAY  NUMBER OF          PROMPT
C    ?   (ROWS,COLS)  ROLE  NUMBER   NAME                DEFINITION OF VARIABLE
C                          (weldland.PRO)
C=================================================================
C    n   (   0,   0)    1      10   WLAND    = width of the weld land
C    n   (   0,   0)    1      15   TLAND    = thickness of the weld land
C    n   (   0,   0)    1      20   ECLAND   = eccentricity of the weld land
C    n   (   0,   0)    2      25   KLAND    = index for weld land edge stringer
C    n   (   0,   0)    1      35   TWLAND   = web thickness of the weld land ed
C    n   (   0,   0)    1      40   HWLAND   = height of the web of the weld lan
C    n   (   0,   0)    1      45   TFLAND   = thickness of the outstanding flan
C    n   (   0,   0)    1      50   WFLAND   = width of the outstanding flange o
C    n   (   0,   0)    2      55   NLAND    = number of weld lands in 360 degre
C    n   (   0,   0)    1      65   TSKIN    = thickness of the cylindrical shel
C    n   (   0,   0)    2      70   RADCYL   = radius of the cylindrical shell
C    n   (   0,   0)    2      75   LENCYL   = length of the cylindrical shell
C    n   (   0,   0)    2      85   BSTR     = stringer spacing
C    n   (   0,   0)    2      90   TSTR     = stringer thickness
C    n   (   0,   0)    2      95   HSTR     = stringer height
C    n   (   0,   0)    2     100   BRNG     = ring spacing
C    n   (   0,   0)    2     105   TRNG     = ring thickness
C    n   (   0,   0)    2     110   HRNG     = ring height
C    n   (   0,   0)    2     120   EMOD     = Young's modulus
C    n   (   0,   0)    2     125   NU       = Poisson ratio
C    n   (   0,   0)    2     130   DENS     = mass density (aluminum = 0.00025
C    n   (   0,   0)    2     140   MLOWG    = low end of the M-range for genera
C    n   (   0,   0)    2     145   MHIGHG   = High end of the M-range for gener
C    n   (   0,   0)    2     150   MLOWP    = low end of the M-range for "panel
C    n   (   0,   0)    2     155   MHIGHP   = high end of the M-range for "pane
C    n   (   0,   0)    2     165   NCASES   = Number of load cases (number of e
C    y   (  20,   0)    3     170   PX       = total axial load (2*pi*r*Nx)
C    y   (  20,   0)    3     175   PY       = uniform normal pressure
C    y   (  20,   0)    3     180   PX0      = axial load in Load Set B
C    y   (  20,   0)    3     185   PY0      = pressure in Load Set B
C    y   (  20,   0)    4     195   GENBUK   = general buckling load
C    y   (  20,   0)    5     200   GENBUKA  = allowable for general buckling
C    y   (  20,   0)    6     205   GENBUKF  = factor of safety for general buck
C    y   (  20,   0)    4     210   PANBUK   = "panel" buckling
C    y   (  20,   0)    5     215   PANBUKA  = allowable for "panel" buckling
C    y   (  20,   0)    6     220   PANBUKF  = factor of safety for "panel" buck
C    y   (  20,   0)    4     225   STRESS   = weld land effective stress
C    y   (  20,   0)    5     230   STRESSA  = maximum allowable effective stres
C    y   (  20,   0)    6     235   STRESSF  = factor of safety for effective st
C    n   (   0,   0)    7     245   WEIGHT   = weight of the weld land+"extra" e
C
```