Table A2 List of the file, **equivellipse.DEF.**
This file is generated automatically by GENOPT
after the GENOPT user completes the "GENTEXT"
interactive session. A copy of this file is
automatically inserted near the beginning of
the skeletal behavior.new file (Table a13).

```
==================================================================
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: equivellipse

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 "equivellipse".

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 "equivellipse" than
C        the GENOPT user.

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

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

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

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

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

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

C           CHANGE    (User changes certain parameters)

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

C           DIPLOT    (User generates plots)

C           CLEANSPEC (User cleans out unwanted files.)

C     A typical runstream is:
C       GENOPTLOG   (activate command set)
C       BEGIN       (provide starting design, loads, etc.)
C       DECIDE      (choose decision variables and bounds)
C       MAINSETUP   (choose print option and analysis type)
C       OPTIMIZE    (launch batch run for n design iterations)
C       OPTIMIZE    (launch batch run for n design iterations)
C       OPTIMIZE    (launch batch run for n design iterations)
C       OPTIMIZE    (launch batch run for n design iterations)
```

```
C       OPTIMIZE    (launch batch run for n design iterations)
C       CHANGE      (change some variables for new starting pt)
C       OPTIMIZE    (launch batch run for n design iterations)
C       OPTIMIZE    (launch batch run for n design iterations)
C       OPTIMIZE    (launch batch run for n design iterations)
C       OPTIMIZE    (launch batch run for n design iterations)
C       OPTIMIZE    (launch batch run for n design iterations)
C       CHOOSEPLOT  (choose which variables to plot)
C       DIPLOT      (plot variables v. iterations)
C       CHOOSEPLOT  (choose additional variables to plot)
C       DIPLOT      (plot more variables v design iterations)
C       CLEANSPEC   (delete extraneous files for specific case)

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

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

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

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

C does the following:

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

C      OPTIMIZE          (perform k design iterations)
C      OPTIMIZE          (perform k design iterations)
```

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

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

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

C                     REFERENCES

C [1] Bushnell, D., "GENOPT--A program that writes
C user-friendly optimization code", International
C Journal of Solids and Structures, Vol. 26, No. 9/10,
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 [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 isogrid-stiffened
C torispherical head", written and placed in the file
C ..genopt/case/torisph/readme.torisph, October 2005

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,
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 = equivellipse)
C        4.  GENOPT.HLP file    (type HELPG)
C=============================================================

C=============================================================
C   TABLE 2   GLOSSARY OF VARIABLES USED IN "equivellipse"
C=============================================================
C  ARRAY  NUMBER OF         PROMPT
C    ?   (ROWS,COLS)  ROLE  NUMBER   NAME                DEFINITION OF
VARIABLE
C                          (equivellipse.PRO)
C=============================================================
C    n   (   0,   0)    2      10   npoint   = number of x-coordinates
C    n   (   0,   0)    2      15   Ixinpu   = vector element number for
xinput in xinput(Ixinpu)
C    y   (  21,   0)    2      20   xinput   = x-coordinates for ends of
segments
C    n   (   0,   0)    2      25   ainput   = length of semi-major axis
C    n   (   0,   0)    2      30   binput   = length of semi-minor axis
of ellipse
C    n   (   0,   0)    2      35   nodes    = number of nodal points per
segment
C    n   (   0,   0)    2      40   xlimit   = max. x-coordinate for x-
coordinate callouts
C    y   (  21,   0)    1      45   THKSKN   = skin thickness at xinput
```

```
C    y   (  21,   0)    1      50   HIGHST   = height of isogrid members
at xinput
C    n   (   0,   0)    1      55   SPACNG   = spacing of the isogrid
members
C    n   (   0,   0)    1      60   THSTIF   = thickness of an isogrid
stiffening member
C    n   (   0,   0)    2      65   THKCYL   = thickness of the
cylindrical shell
C    n   (   0,   0)    2      70   RADCYL   = radius of the cylindrical
shell
C    n   (   0,   0)    2      75   LENCYL   = length of the cylindrical
segment
C    n   (   0,   0)    2      80   WIMP     = amplitude of the
axisymmetric imperfection
C    n   (   0,   0)    2      85   EMATL    = elastic modulus
C    n   (   0,   0)    2      90   NUMATL   = Poisson ratio of material
C    n   (   0,   0)    2      95   DNMATL   = mass density of material
C    n   (   0,   0)    2     100   IMODE    = strategy control for
imperfection shapes
C    n   (   0,   0)    2     105   NCASES   = Number of load cases
(number of environments)  in PRESS(NCASES)
C    y   (  20,   0)    3     110   PRESS    = uniform external pressure
C    y   (  20,   0)    4     115   CLAPS1   = collapse pressure with
imperfection mode 1
C    y   (  20,   0)    5     120   CLAPS1A  = allowable pressure for
axisymmetric collapse
C    y   (  20,   0)    6     125   CLAPS1F  = factor of safety for
axisymmetric collapse
C    y   (  20,   0)    4     130   GENBK1   = general buckling load
factor, mode 1
C    y   (  20,   0)    5     135   GENBK1A  = allowable general buckling
load factor (use 1.0)
C    y   (  20,   0)    6     140   GENBK1F  = factor of safety for
general buckling
C    n   (   0,   0)    2     145   JSKNBK1  = number of regions for
computing behavior in SKNBK1(NCASES,JSKNBK1)
C    y   (  20,  10)    4     150   SKNBK1   = local skin buckling load
factor, mode 1
C    y   (  20,  10)    5     155   SKNBK1A  = allowable buckling load
factor
C    y   (  20,  10)    6     160   SKNBK1F  = factor of safety for skin
buckling
C    y   (  20,  10)    4     165   STFBK1   = buckling load factor,
isogrid member, mode 1
C    y   (  20,  10)    5     170   STFBK1A  = allowable for isogrid
stiffener buckling (Use 1.)
C    y   (  20,  10)    6     175   STFBK1F  = factor of safety for
isogrid stiffener buckling
```

```
C    y   (  20,  10)    4     180   SKNST1   = maximum stress in the shell
skin, mode 1
C    y   (  20,  10)    5     185   SKNST1A  = allowable stress for the
shell skin
C    y   (  20,  10)    6     190   SKNST1F  = factor of safety for skin
stress
C    y   (  20,  10)    4     195   STFST1   = maximum stress in isogrid
stiffener, mode 1
C    y   (  20,  10)    5     200   STFST1A  = allowable stress in isogrid
stiffeners
C    y   (  20,  10)    6     205   STFST1F  = factor of safety for stress
in isogrid member
C    y   (  20,   0)    4     210   WAPEX1   = normal (axial) displacement
at apex, mode 1
C    y   (  20,   0)    5     215   WAPEX1A  = allowable normal (axial)
displacement at apex
C    y   (  20,   0)    6     220   WAPEX1F  = factor of safety for WAPEX
C    y   (  20,   0)    4     225   CLAPS2   = collapse pressure with
imperfection mode 2
C    y   (  20,   0)    5     230   CLAPS2A  = allowable pressure for
axisymmetric collapse
C    y   (  20,   0)    6     235   CLAPS2F  = factor of safety for
axisymmetric collapse
C    y   (  20,   0)    4     240   GENBK2   = general buckling load
factor, mode 2
C    y   (  20,   0)    5     245   GENBK2A  = allowable general buckling
load factor (use 1.0)
C    y   (  20,   0)    6     250   GENBK2F  = factor of safety for
general buckling
C    n   (   0,   0)    2     255   JSKNBK2  = number of regions for
computing behavior in SKNBK2(NCASES,JSKNBK2)
C    y   (  20,  10)    4     260   SKNBK2   = local skin buckling load
factor, mode 2
C    y   (  20,  10)    5     265   SKNBK2A  = allowable skin buckling
load factor (use 1.0)
C    y   (  20,  10)    6     270   SKNBK2F  = factor of safety for local
skin buckling
C    y   (  20,  10)    4     275   STFBK2   = buckling load factor for
isogrid member, mode 2
C    y   (  20,  10)    5     280   STFBK2A  = allowable for isogrid
stiffener buckling (Use 1.)
C    y   (  20,  10)    6     285   STFBK2F  = factor of safety for
isogrid stiffener buckling
C    y   (  20,  10)    4     290   SKNST2   = maximum stress in the shell
skin, mode 2
C    y   (  20,  10)    5     295   SKNST2A  = allowable stress for the
shell skin
C    y   (  20,  10)    6     300   SKNST2F  = factor of safety for skin
```

```
stress
C    y   (  20,  10)    4     305   STFST2   = maximum stress in isogrid
stiffener, mode 2
C    y   (  20,  10)    5     310   STFST2A  = allowable stress in isogrid
stiffeners
C    y   (  20,  10)    6     315   STFST2F  = factor of safety for stress
in isogrid member
C    y   (  20,   0)    4     320   WAPEX2   = normal (axial) displacement
at apex, mode 2
C    y   (  20,   0)    5     325   WAPEX2A  = allowable normal (axial)
displacement at apex
C    y   (  20,   0)    6     330   WAPEX2F  = factor of safety for WAPEX
C    n   (   0,   0)    7     335   WEIGHT   = weight of the equivalent
ellipsoidal head
C
C============================================================
C       TABLE 3   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 equivellipse.PRO and five
 FORTRAN source libraries, BEGIN.NEW, STOGET.NEW, STRUCT.NEW,
 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"
 ============================================================
  equivellipse.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 equivellipse.

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
equivellipse.PRO	Prompts and help paragraphs for interactive input to the user-developed optimization code.
equivellipse.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.
equivellipse.INP	Image of interactive input for user-developed program, generated to save time in case you make a mistake during input.
equivellipse.COM	Labelled common blocks generated specifically for the user-developed class of problems.
equivellipse.WRI	Part of subroutine for writing labelled common blocks in SUBROUTINE STORCM (in Library STOGET).
equivellipse.REA	Part of subroutine for reading labelled common blocks in SUBROUTINE GETCOM (in Library STOGET).
equivellipse.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.
equivellipse.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.
equivellipse.SUB	Skeletal subroutines, BEHX1, BEHX2, ..., and the skeletal objective function, OBJECT.
equivellipse.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.

equivellipse.CHA Part of SUBROUTINE NEWPAR (called in the CHANGE processor) in which labelled common values are updated.

equivellipse.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.

==========

WHAT TO DO NEXT (THIS IS REALLY IMPORTANT!):
Next, if necessary, provide the algorithms called for in the skeletal subroutines listed in the library BEHAVIOR.NEW. You may find useful routines, such as a linear interpolator, in the library UTIL.NEW.

And/Or, if necessary, complete the skeletal routines STRUCT and TRANFR. (You may find useful routines in UTIL.NEW). If you are adding subroutine calls to SUBROUTINE STRUCT or SUBROUTINE TRANFR, store the subroutines themselves in the libraries called ADDCODEn.NEW, n = 1,2,3,...5. (Please list one of the ADDCODEn.NEW libraries for instructions.)

After you have done all this, give the command GENPROGRAMS. GENPROGRAMS will generate the absolute elements needed to optimize whatever you have chosen as your objective (see OBJECT routine in BEHAVIOR.NEW) in the presence of whatever behavior or other factors (e.g. clearance) are quantified by user-written subroutines collected in the libraries ADDCODEn.NEW and/or algorithms added to the skeletal routines in the library BEHAVIOR.NEW .

If an error occurs during GENPROGRAMS, check your FORTRAN coding. If you have to change something and rerun, make sure to save the old version under a different file name so that you can efficiently delete all outdated files with names *.NEW without losing a lot of good coding! The writer had fallen more than once into that trap during development of GENOPT.

If GENPROGRAMS runs without bombing, try test examples within the class of problems covered by your FORTRAN contributions to GENOPT before assigning specific design development tasks to individuals who may be more naive in the field covered by your FORTRAN contributions to GENOPT than you are!

Please see the cases under genopt/case for examples and more
information.

USING GENOPT IN GENERAL AND WITH BIGBOSOR4

Please read the file, ..genopt/doc/getting.started.
Please also read the files:
...genopt/case/cylinder/howto.bosdec
...genopt/case/cylinder/howto.struct
...genopt/case/cylinder/howto.behavior
...genopt/case/torisph/howto.stags
...genopt/case/torisph/readme.equivellipse
...genopt/case/wavycyl/readme.wavycyl

The main things you must do are the following:

1. create a file called ..bosdec/sources/bosdec.src, the
purpose of which is to create a BOSOR4 input file, *.ALL .
in which "*" represents the users name for the specific case.
The file, ..genopt/case/torisph/bosdec.equivellipse is a good
example. Make sure to save bosdec.src by copying it into
another file. Example: cp bosdec.src bosdec.equivellipse

2. Flesh out either or both the libraries, struct.new and/or
behavior.new. In the case, ..genopt/case/torisph, only the
library struct.new is fleshed out. The library behavior.new is
not changed from that created automatically by GENOPT. In the
case, genopt/case/cylinder, both struct.new and behavior.new
are changed, struct.new in minor ways and behavior.new in
major ways. Make sure to save struct.new and behavior.new. For
example: cp struct.new   struct.cylinder
         cp behavior.new behavior.cylinder
(You save copies of bosdec.src, struct.new, behavior.new
 because it usually takes quite a bit of effort to modify
 the versions automatically created by GENOPT in order to
 solve your generic class of problems.)
See the following files for examples of modified libraries:

genopt/case/torisph/struct.tori   (behavior.new not modified)
genopt/case/torisph/struct.ellipse (behavior.new not modified)
genopt/case/torisph/struct.equivellipse
                                   (behavior.new not modified)
genopt/case/cylinder/struct.cylinder
genopt/case/cylinder/behavior.cylinder
genopt/case/wavycyl/struct.wavycyl
genopt/case/wavycyl/behavior.wavycyl
genopt/case/plate/behavior.plate (struct.new is not modified)

genopt/case/plate/behavior.plate (struct.new is not modified)
genopt/case/sphere/behavior.plate (struct.new is not modified)

3. Execute the GENOPT script called GENPROGRAMS. This script "makes" the processors for the user-named generic case. The "makefile" called ..genopt/execute/usermake.linux is used. If GENPROGRAMS compiles everything successfully, which is not likely on your first try because you probably did a lot of FORTRAN coding to create bosdec.src, struct.new, behavior.new, GENPROGRAMS will end with a list like the following:

```
Here is a list of all your newly created executables:
-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
```

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

If GENPROGRAMS bombs due to fatal compilation errors, or even if GENPROGRAMS seems to finish successfully, it is best to inspect the file ..genoptcase/usermakelinux.log. If there are compilation errors, revise the appropriate source codes, bosdec.src and/or struct.new and/or behavior.new, and execute GENPROGRAMS again. Keep doing this until everything is okay.

4. Next, think up a good name for your specific case and run BEGIN, DECIDE, MAINSETUP, and OPTIMISE (several times) or SUPEROPT. (See the file ..genopt/doc/getting.started and the directories, genopt/case/cylinder and genopt/case/torisph for examples.) Even though you had a successful "make" via GENPROGRAMS in the previous step, something will doubtless not be satisfactory and you will have to or want to make further changes to one or more of the source files, bosdec.src, struct.new, behavior.new.

THE NEXT STEPS PERTAIN TO THE USE OF GENOPT WITH BIGBOSOR4

5. You must have the BIGBOSOR4 software in the directory, ..bosdec/sources. You need to have the following files there: addbosor4.src, b4util.src, opngen.src, prompter.src, gasp.F, gasp_linux.o, bio_linux.c, bio_linux.o, b4plot.src, as well as the bosdec.src file discussed above.

6. The "make" file, ..genopt/execute/usermake.linux, must include references to the BIGBOSOR4 sofware listed in Step 5. Please see the file ..genopt/execute/usermake.linux, which already exists. (You do not have to do anything about it!)

7. Suppose everything compiles correctly during the GENPROGRAMS execution, but when you try to run a specific case the run bombs. Suppose all of your contributed FORTRAN coding is in ..bosdec/sources/bosdec.src and in
..genoptcase/struct.new (..genoptcase/behavior.new did not need to be modified for your case, as is true for the generic case called "equivellipse" in ..genopt/case/torisph). It is very helpful to insert a "CALL EXIT" statement after one of the analyses performed in struct.new, then to execute GENPROGRAMS again to recompile the temporarily changed struct.new. The reason for doing this is explained in the file ..genopt/case/torisph/struct.equivellipse and also in the file ..genopt/doc/getting.started: you want to be able to make a BIGBOSOR4 run to be certain that:
a. ..bosdec/sources/bosdec.src created a valid BOSOR4 input file, and,
b. the BIGBOSOR4 run did not finish for some reason.

********** 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".
****

==========