trusscamp.tit

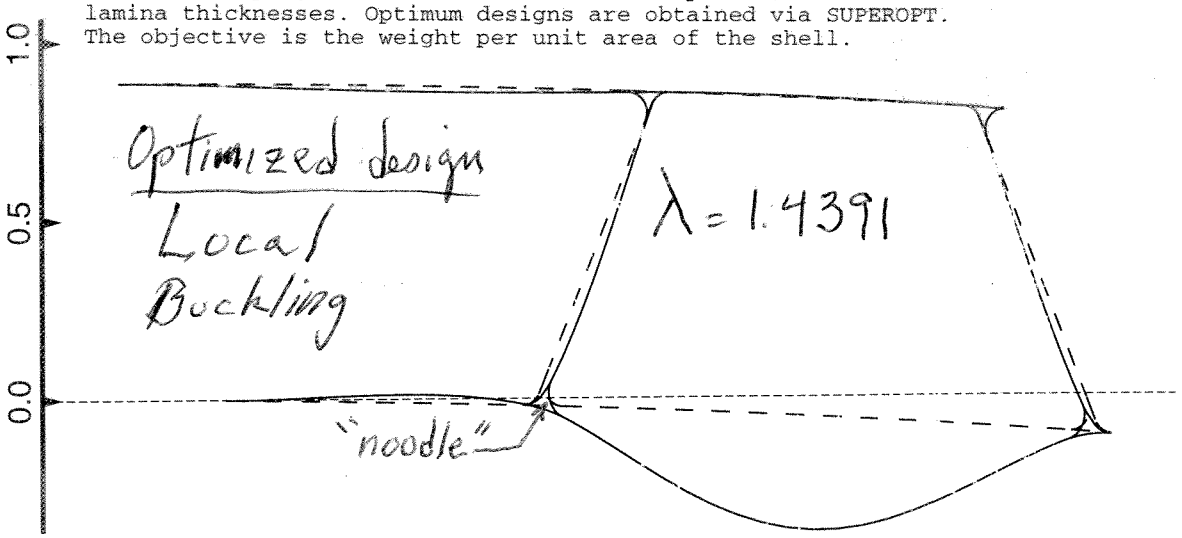
Use of GENOPT and BIGBOSOR4 to obtain optimum designs of a cylindrical shell with a composite truss-core sandwich wall

David Bushnell

June 20, 2009

ABSTRACT

This GENOPT/BIGBOSOR4 case is analogous to the case called "weldland". (See the directory,...genopt/case/weldland, in particular the "weldland" report contained in the three files, 2009weldlandopt.vol1.pdf,2009weldlandopt.vol2.pdf, 2009weldlandopt.vol3.pdf, in that directory). Here, GENOPT/BIGBOSOR4 is applied to the problem of an axially compressed perfect elastic cylindrical shell the wall of which is a composite truss-core sandwich. The design constraints are local buckling, general buckling, and five stress constraints for each material. Local and general buckling are computed from BIGBOSOR4 models in which the "huge torus" representation of the cylindrical shell is employed. In the "huge torus" representation of the cylindrical shell, what was the axial coordinate in the cylindrical shell becomes the circumferential coordinate in the "huge torus" model, and what was the circumferential coordinate in the cylindrical shell becomes the meridional coordinate in the "huge torus" model. In both the local and general buckling models the "huge torus" representation of the cylindrical shell consists of a number of identical modules strung together along the curved meridian of the "huge torus". The local buckling model usually consists of a single module that has 22 shell segments (although multiple-22-segment-module local buckling models are permitted). The rather elaborate 22-segment single-module model used for local buckling includes the small curved segments that occur at the corners of the trapezoidal tool around which the truss-core is wrapped during the fabrication process. The presence of "noodles" that fill the prismatic triangular-like gaps between adjacent trapezoids is accounted for. These "noodles" are represented in the "huge torus" model as rings attached to appropriate shell segments. The single module used in the general buckling model is much simpler. It consists of six straight shell segments analogous to those used in the truss-core sandwich model employed in PANDA2. The effect of the "noodles" is accounted for, however, which is not possible in the PANDA2 model. The number of modules used in the model for general buckling is computed such that 90 degrees of the circumference of the cylindrical shell is included in the model. Stress constraints are computed in a way completely analogous to that used in PANDA2 for composite laminates. The computation of stress constraints is much simpler here than in PANDA2 because the prebuckled state of the "huge torus" is a uniform membrane state. The decision variables are the width of a single module, the width of the crown of one trapezoid, the height of the trapezoid, the small radius from the base to the sloping side of a trapezoid, the small radius from the sloping side to the crown of a trapezoid, and several lamina thicknesses. Optimum designs are obtained via SUPEROPT. The objective is the weight per unit area of the shell.



trusscomp.text

INTRODUCTION

The generic case described in this report is called "trusscomp" for "composite truss-core sandwich wall construction". The geometry is similar to that used in the "nasatruss" project described in [9].

Some references are listed in Table 1. The work in this paper was motivated by Andrew Lovejoy at NASA Langley, who wondered if it was possible easily to modify PANDA2 [9] to permit approximate modeling of the "noodles" located in the triangular-like gaps between adjacent trapezoids in the truss-core sandwich wall (Fig. 6, bottom). PANDA2 cannot easily be modified in this way. Therefore, after thinking about it for a while, the writer decided to use GENOPT in combination with BIGBOSOR4 to accomplish this task.

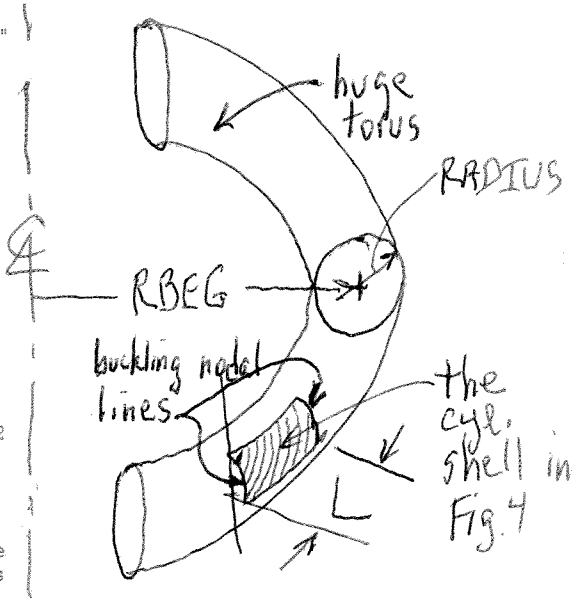
Fig 6 on p. 33

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

← Note!!

"Huge torus" model:

The idea of using the shell-of-revolution code, BOSOR4 [10] (now BIGBOSOR4 [2]), to analyze prismatic structures such as axially stiffened flat panels and axially stiffened cylindrical shells was first published in 1971 [8]. This idea is exploited by the PANDA2 processors called "PANEL" [7] and "PANEL3" [4]. The BIGBOSOR4 "huge torus" model is used for the optimum design of weld lands in [4]. The composite truss-core sandwich case described here (generic case name = "trusscomp") is analogous to the "weld land" case described in [4].



Naturally, the models here are approximate. For example, as with the "weld land" model described in [4], the prebuckled state of the truss-core sandwich is assumed to be a membrane state. There is no prebuckling bending of the axially compressed cylindrical shell with a truss-core sandwich wall construction. The axial compression, Nx, in the various shell segments of the model is assumed to be distributed in proportion to the axial stiffness of each of those segments. 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 prebuckling radial displacement w in the neighborhoods of the two ends of the cylindrical shell caused by restriction of Poisson ratio radial expansion there. No hoop compression is induced by the presence of axial compression. The shell is perfect; there is no general buckling modal imperfection, as is possible in the PANDA2 models as described in [7] and [9].

The "huge torus" model of the cylindrical shell is based on the INDIC = 4 branch of BIGBOSOR4. With INDIC = 4 the prebuckling resultants in each shell segment and in each discrete ring are provided as input data in the *.ALL file, which contains all the input data for a BIGBOSOR4 execution.

In this study a cylindrical shell is modeled as a huge torus. More precisely, part of a cylindrical shell (90 degrees of its circumference) is modeled as part of a huge torus. (See Figs 4 and 5 and Appendix 1). The "huge torus" model is a "trick" that permits detailed modeling of the truss-core sandwich wall, including such details as the "noodles" that inhabit the triangular-like gaps between adjacent trapezoids and the small curved shell segments that represent the part of the truss-core that "turns the corners" of the trapezoidal tool about which the truss-core laminate is wrapped (Fig. 1). These details are included while working within the restriction that BIGBOSOR4 only handles shells of revolution.

Figs. 4 & 5 on pp. 31 & 32

Fig 1 on p. 28

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 [4]. 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 [4]. What were stringers in the cylindrical shell and now become rings (circumferentially oriented stiffeners in the huge torus) can be modeled as little flexible shell segments (Figs. 5-7 and Figs. 10-12 in [4]) or as discrete rings with undeformable cross sections with certain area, bending moments of inertia and axial modulus (Figs.2 and 6 here).

Fig. 2 on p. 29
Fig 6 on p. 33

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 1 explains the relationship between the axial length of the cylindrical shell and the circumferential wave number (N = 100, 200, ...) that corresponds to 1, 2, ... half wavelengths that span that axial length of the cylindrical shell. 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. There are buckling mode nodal lines along the two curved ends of the simply supported cylindrical shell.

See the margin sketch on p. 2.

BIGBOSOR4:

BIGBOSOR4 is executed multiple times inside the optimization loop. As is described in [2], the originally "stand alone" version of BOSOR4 [10] has been divided into subroutines: B4READ (the preprocessor), B4MAIN (the mainprocessor), and B4POST (the postprocessor) for execution in an optimization context. The capability of BOSOR4 [10] has been expanded to handle many more shell segments [2]. This expanded version is called "BIGBOSOR4" here and in other reports. BIGBOSOR4 can now handle up to 295 shell segments. There now exists a "stand alone" version of BIGBOSOR4 as well as the version of BIGBOSOR4 designed to be used in connection with GENOPT. This "stand alone" version of BIGBOSOR4 is used to produce plots such as those shown in Figs. 2 - 5.

Figs. 2 & 5 on pp. 29 & 32

BIGBOSOR4 was modified in order to obtain the results presented here. Those who wish to duplicate the results in this report at their facilities must obtain this latest version of BIGBOSOR4. In particular, the size of the common working space set up in BIGBOSOR4 has been doubled, and the sizes of the arrays that contain segment junction conditions and constraints to ground have been more than doubled. The GENOPT software has also been updated. In particular, the program AUTOCHANGE (chauto.src) has been modified to account for inequality constraints, and the INSERT processor, insert.src, has been updated to permit variable names with up to seven characters, which renders INSERT to be consistent with GENPROGRAMS [1]. Details of these modifications to GENOPT are described near the end of the file: ...genopt/doc/genopt.news .
Those who wish to duplicate the results in this report and continue with further investigations using GENOPT/BIGBOSOR4 at their facilities must obtain the latest version of GENOPT, which prints "JUNE, 2009" in the output files, *.OPM and *.OPP .

← Note!

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 [11 - 13]. In spite of this relative simplicity, execution times for optimization via SUPEROPT for the "trusscomp" case are very long, measured in days rather than in minutes or hours. These long computer run times result primarily from the computation

On my LINUX computer.

of general buckling of the multi-module "huge torus" model of the cylindrical shell (Figs. 4 and 5).

Figs. 4 & 5 on pp. 31 & 32

SUPEROPT:

SUPEROPT is a script by means of which GENOPT attempts to find a "global" optimum design. A SUPEROPT execution generates about 470 design iterations. SUPEROPT is described in [14]. A SUPEROPT run consists of many optimizations starting from different points in design space. Each new "starting" design in a single SUPEROPT execution is generated randomly but consistent with upper and lower bounds and equality and inequality constraints. The results of a single SUPEROPT execution are displayed in Fig. 25. Each "spike" in that plot represents a new, randomly obtained, "starting" design.

Fig. 25 is on p. 169

SOME DETAILS OF THE TRUSS-CORE SANDWICH "HUGE TORUS" SHELL

Both general and local buckling constraints are present in the composite truss-core sandwich model, "trusscomp", presented in this report.

General buckling model:

Figures 4 and 5 show an example of 90 degrees of a cylindrical shell with a truss-core sandwich wall. The many small "x" symbols (Fig.4) repesent the "noodles" and surrounding shell wall material in the truss core that are located between adjacent trapezoids (Fig. 6). This is the "huge torus" model used for prediction GENERAL buckling. The 90-degree general buckling model is composed of a number of modules, each module of the type displayed at the top of Fig. 6. Each of the discrete rings ("x" symbols in Fig. 4) represents the material shown at the bottom of Fig. 6: the area of the "noodle", A1 + A2, plus the adjacent shell wall segments that are not included in the module model shown at the top of Fig. 6, minus material that is present in the module model at the top of Fig 6 and that passes through the "noodle" area (the -L3 and the -2d in the formula for ring axial stiffness written at the bottom of Fig. 6).

Figs. 4 & 5 on pp. 31 & 32

Fig. 6 is on p. 33

Local bucking model:

Figures 1 - 3 show the much more elaborate single module model used for LOCAL buckling. The single module displayed in Fig. 1 consists of 22 shell segments. In Fig. 1 the module is shown as if it were flat. Actually, it is curved as shown in Figs. 2 and 3. The shaded areas (Fig.7), A1 and A2, of the triangular-like gaps between adjacent trapezoids are computed as shown in Fig. 7. In the elaborate LOCAL buckling model the area A1 depicted at the top of Fig. 7 is included in a discrete ring attached at the point in the curved shell segment as indicated, with the location of the ring centroid as indicated. An analogous convention is assumed for the smaller area, A2, shown at the bottom of Fig. 7. Hence, in the LOCAL buckling model there are two discrete rings for each triangular-like gap between adjacent trapezoids in the single module model. Figure 2 shows the curved elaborate single module model with the two ring centroids indicated by "x"s in each triangular-like gap. Figure 3 shows the local buckling mode corresponding to the single-module model displayed in Fig. 2. Figures 10 - 13 show how the end points and centers of curvature are computed for the 22 shell segments indicated in the elaborate flat LOCAL buckling single module model displayed in Fig. 1 before it is mapped into a curved single module such as that shown in Fig. 2.

Figs. 1-3 on pp. 28, 29, 30

Fig. 7 is on p. 34

Figs. 10-13 are on pp. 37-40

Curved modules for both general and local buckling models:

The curved model for GENERAL buckling, shown in Figs. 4 and 5, is generated from a single module as depicted in Fig. 9. The same method is used to generate curved multi-module models for LOCAL buckling. The same method is also used to map a flat, single-module model, such as that shown in Fig. 1, into a curved, single-module model, such as that shown in Fig. 2.

Fig. 9 is on p. 36

Decision variables:

The decision variables in the optimization problem presented here are listed in Fig. 8.

Fig. 8 is on p. 35

PREVIOUS WORK ON THE TRUSS-CORE SANDWICH CONFIGURATION FROM [9]

The purpose of this section is to provide a comparison of predictions from PANDA2 [9] (the "nasatruss" project) with those from GENOPT/BIGBOSOR4 (the "trusscomp" project) for the design of the composite truss-core sandwich cylindrical shell. This shell was previously optimized in the "nasatruss" project by PANDA2 [9].

In [9] the results of the optimization of an axially compressed cylindrical shell with a composite truss-core sandwich wall are described. The starting design of the "huge torus" model that is the subject of the present "trusscomp" work, is the optimum design obtained by PANDA2 in [9] in the "nasatruss" project. Tables 4-7 and Figs. 14-16, taken from [9] and modified slightly, pertain to this section. Reference [9] is available as the file, .../panda2/case/truss3/nasa2009.trusscorecomposite.pdf
In [9] the case name is "nasatruss".

Tables 4-7 = pp. 41-50

Figs. 14-16 on pp. 45, 46, 51

Table 4, which is similar to Table 26 in [9], lists the design margins and optimum design obtained by PANDA2 for the axially compressed cylindrical shell with the truss-core sandwich wall. The design listed on page 2 of Table 4 is to be used as the starting design in the present "trusscomp" work with GENOPT/BIGBOSOR4. [NOTE: In the "nasatruss" case there is an error in the specification of material density. In the file, "nasatruss.BEG" the material weight density is given a value of 0.0057 lb/in^3 instead of 0.057 lb/in^3. Therefore, the specific weight (weight/area) listed in Table 4 is too small by a factor of 10. It should be 0.012315 lb/in^2 instead of 0.0012315 lb/in^2. the weight should be 178.3 lb instead of 17.83 lb.]

Table 5, which is similar to Table 27 in [9], lists the design margins for the same optimized shell with zero amplitude, Wimp = 0, for the general buckling modal imperfection.

Table 5 is on p. 43

Table 6 lists PANDA2 output from Chapter 26 for general buckling. This output is obtained when the user sets NPRINT = 2 in the PANDA2 *.OPT file, in which "*" represents the user-selected case name. In PANDA2 general buckling of the truss-core sandwich wall is computed from a "closed form" solution in which the wall details are smeared out. The effect of transverse shear deformation (t.s.d.) is handled approximately in PANDA2 by a knockdown factor applied to the approximate "closed form" solution in which t.s.d. is neglected.

Table 6 is on p. 44

In Table 6 note that, according to PANDA2, the knockdown factor to account for transverse shear deformation (t.s.d.) is 5.0738/7.9938. As will be seen, this t.s.d. knockdown factor used by PANDA2 seems to be significantly milder than it should be, as is indicated by the "huge torus" model of general buckling in which the details of the truss-core sandwich wall construction are represented by many small flexible shell segments (Fig. 15).

Fig. 15 is on p. 46

Figure 14 shows the general buckling load factor (eigenvalue) and mode shape from BIGBOSOR4 obtained from the input file, nasatruss.ALL, generated by the PANDA2 processor called "PANEL2". In this BIGBOSOR4 model the truss-core sandwich wall is smeared. BIGBOSOR4 neglects the effect of transverse shear deformation. As pointed out in Fig. 18 of [9], the eigenvalue from BIGBOSOR4, 6.1991, is less than that predicted from the PANDA-type (closed form) solution used in PANDA2 (EIGMNC = 7.99 in Table 6 before knockdowns for transverse shear deformation and smearing) mainly because the axial distribution of the general buckling mode shape predicted by BIGBOSOR4 and displayed in Fig. 14 differs significantly from the simple sine wave that is assumed by PANDA2 in the generation of the approximate closed form solution.

Fig. 14 is on p. 45

Table 6 is on p. 44

Figure 15 from [9] shows the general buckling mode predicted by a "huge torus" model of the optimized truss-core sandwich cylindrical shell. The eigenvalue, 2.00324, is significantly smaller than

Fig. 15 on p. 46

the general buckling load factor, 4.7582, predicted by PANDA2 after knockdowns for transverse shear deformation and smearing, and listed in Table 6. The discrepancy between the PANDA2 prediction (4.7582 after knockdowns) and the BIGBOSOR4 "huge torus" prediction (2.0034) most likely exists because the knockdown factor to account for t.s.d. is too mild in the approximate PANDA2 theory for the composite truss-core sandwich wall configuration in which the wall properties are smeared out.

Table 6 on p. 44

Table 7 lists PANDA2 results from Chapters 13 and 14 of the PANDA2 output. This output pertains to LOCAL buckling of the truss-core sandwich wall as predicted from a single-module model of the wall consisting of six shell segments in which each shell segment is discretized in the meridional coordinate. The module configuration and the stess resultants in each discretized shell segment are listed in Chapter 13, and the buckling load factors and critical local buckling mode shape are listed in Chapter 14. According to PANDA2 the critical local buckling mode has 38 axial halfwaves over the 96-inch length of the cylindrical shell, which corresponds to a critical axial half-wavelength of 2.626 inches. The critical buckling load factor is 1.30959 according to PANDA2 (top of p. 3 of Table 7). The PANDA2 output listed in Table 7 is generated when the user specifies NPRINT = 2 in the PANDA2 *.OPT file, which contains the input data for the MAINSETUP processor of PANDA2.

Table 7 on pp. 47-50
See sketch on p. 47.

The prebuckling axial resultants, Nx(var), listed for the "nasatruss" project [9] at the bottom of p. 1 and at the top of p. 2 in Table 7 should be compared with the prebuckling stress resultants, FN20, listed for the "trusscomp" project in Tables 18 and 19. Note that in the "trusscomp" project some of the total axial load is absorbed by the "noodles", which are not present in the "nasatruss" model [9] but which are accounted for in the "trusscomp" model (Fig. 6, bottom). In the "huge torus" trusscomp model, FN20, which is the circumferential stress resultant, is analogous to the axial stress resultant, Nx(var), in the PANDA2 model, values for which are listed at the bottom of page 1 and in the top half of page 2 in Table 7.

pp. 47 & 48
Tables 18 & 19 are on pp. 137-149
FN20 on p. 137
pp. 47 & 48

Figure 16 shows the local buckling mode from a BIGBOSOR4 "huge torus" model of the "nasatruss" design optimized by PANDA2. The eigenvalue, 1.28773, from BIGBOSOR4 is reasonbly close to the eigenvalue, 1.30959, predicted by PANDA2 for local buckling in the "nasatruss" project (top of page 3 of Table 7).

Fig. 16 on p. 51
p. 49

For the "trusscomp" project that is the subject of this report the predictions for local and general buckling of the same truss-core design as that listed in Table 4 appear in Figs. 3 and 5, respectively.

Fig. 3 on p. 30
Fig. 5 on p. 32

The "trusscomp" critical local buckling load factor, 1.6694 in Fig. 3, should be compared to the "nasatruss" project's [9] prediction, 1.28773, in Fig. 16. It is logical that the "trusscomp" model yields a somewhat higher prediction for local buckling than the "nasatruss" model: the "trusscomp" model includes the little curved segments shown in Figs. 1 - 3. These little curved segments shorten the widths of the segments that participate in the local buckling deflections. Also, the "trusscomp" model includes the "noodles", which absorb some of the axial load. The little curved segments and the "noodles" do not exist in the "nasatruss" project model [9].

Fig. 16 on p. 51

The general buckling load factor from the "trusscomp" project is 2.3655 in Fig. 5. This buckling load factor should be compared with the "nasatruss" project's [9] prediction of 2.0034 in Fig. 15. The critical general buckling mode shapes from the two projects are approximately the same: one half wave over the 96-inch length and three half waves over the 90-degree circumference included in the "huge torus" models. The mode shape plotted in Fig. 5 is approximately the negative of the mode shape plotted in Fig. 15. The "nasatruss" "huge torus" model predicts a somewhat lower general buckling load factor, 2.0034 compared to 2.3644 for the "trusscomp" model, possibly because the two little face sheet shell segments at the uppermost left-hand side of Fig. 15 (near axial station = 0) deform independently of eachother, allowing more freedom of normal displacement of the neighboring modules than should be allowed were simple support conditions applied to both the upper and lower face sheets and not just to the lower face sheet.

Fig. 5 on p. 32
Fig. 15 on p. 46

ABOUT GENOPT

GENOPT [1,2,3,4] 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 [1]. The GENOPT user must also supply software (subroutines and/or FORTRAN statements) that perform the analysis or analyses [1-4]. The GENOPT user must decide what behaviors will constrain the design during optimization cycles, behaviors such as general buckling, local buckling, stress, vibration, etc. While establishing problem variables, the GENOPT user must decide which of 7 roles each of these variables plays [1]. The 7 possible roles are:

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

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

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

Tables 2 & 3 are on pp. 20-27

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

PRODUCTION OF THE PROGRAM SYSTEM TO OPTIMIZE AN AXIALLY COMPRESSED CYLINDRICAL SHELL WITH A COMPOSITE TRUSS-CORE SANDWICH WALL. THE GENERIC CASE IS CALLED "trusscomp"

Note!

Table 8 lists the run stream used to obtain the results presented here. The reader should use this table as a guide if he or she wants to reproduce the results in this report, or investigate further into other optimum designs in the "trusscomp" class, or obtain optimum designs of any other shells of revolution in which the intention is to use GENOPT in connection with BIGBOSOR4.

Table 8 is on pp. 52-69

The GENOPT user first provides input during a long GENTEXT interactive session. Reference [3] provides good examples of how this GENTEXT interactive session goes and what GENTEXT does as the interactive session proceeds. GENTEXT records the GENOPT user's input for the long interactive session in a file called *.INP, where "*" represents the GENOPT user's GENERIC name for the case. If the GENOPT user enters a wrong input by mistake, he or she can terminate the GENTEXT interactive session, edit the end of the *.INP file to eliminate the mistake, and then repeat the GENTEXT session, initially using the *.INP file as input rather than tediously repeating the interactive session up to the point where the mistake was made. At the point where the mistake was made GENTEXT returns control to the GENOPT user, and he or she then continues the session in an interactive mode. Table 9 lists the *.INP file after completion of the entire GENTEXT interactive session for the "trusscomp" case.

Table 9 is on pp. 70-83

GENTEXT automatically produces several files. They are described in [3]. Among them is a file called *.DEF (trusscomp.DEF), which the GENOPT user should inspect. The *.DEF file includes a glossary of the GENOPT-user-established variables and one-line definitions of the variables. Table 10 is this glossary for the case, "trusscomp". Additional sections of the trusscomp.DEF file are repeated at the beginning of the behavior.new library (Table 12), a "skeletal" version of which is automatically produced by GENTEXT. One of the most important files automatically produced by GENTEXT is the prompting file, *.PRO (Table 11). The prompting file, called "trusscomp.PRO" in this "trusscomp" case, contains the GENOPT-user-established variable names, one-line definitions, and "help" paragraphs. It is these items that make the GENOPT-user-created system "user-friendly", because the end user will depend on them for his or her input to the GENOPT-created processor, BEGIN.

Table 10 on p. 89 (see pp. 90-96) Table 11 is on pp. 85-90

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

It is a different matter in the case of STRUCT.NEW and BEHAVIOR.NEW. These are "skeletal" subroutine libraries either or both of which must be "fleshed out" by the GENOPT user. In this particular application the GENOPT user adds merely three statements, CALL OPNGEN, CALL RWDGEN, and CALL CLSGEN (open, rewind, and close BIGBOSOR4 files) to the version of STRUCT.NEW automatically created by GENOPT. (See p. 2 of Tables 8). In this particular application the GENOPT user does more to "flesh out" the BEHAVIOR.NEW library. (See Table 12).

(see p. 53) Table 12 on pp. 91-109

During the GENTEXT interactive session the GENOPT user here decided to introduce four behaviors: 1. local buckling, 2. general buckling, 3. five stress components for material type 1, and 4. five stress components for material type 2. Corresponding to these four behaviors, GENTEXT automatically created four skeletal "behavioral" subroutines, SUBROUTINE BEHX1, SUBROUTINE BEHX2, SUBROUTINE BEHX3, and SUBROUTINE BEHX4. The GENOPT user had to "flesh out" each of these four "behavioral" subroutines, as listed in Table 12.

The GENOPT user also had to "flesh out" SUBROUTINE OBJECT, which is listed on pp. 108 & 109.

The five stress components for each material type are defined as follows:

Maximum stress components from BEHX3 and BEHX4:	
0 deg. tension	(tension in the fiber direction of a layer)
0 deg. comp.	(compression in the fiber direction of a layer)
90 deg. tension	(tension normal to the fibers of a layer)
90 deg. comp.	(compression normal to the fibers of a layer)
in-plane shear	(shear in the plane of the layer)

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 Tables 13a and 13b (abridged versions; the complete "bosdec.trusscomp" is quite long and is located in the directory, ..genopt/case/trusscomp). 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.

See p. 97, for example. Table 13a on pp. 110-122 Table 13b on pp. 123-125

********** 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.trusscomp" and "struct.trusscomp". (also "bosdec.trusscomp"). You must save these important files for possible future use because execution of the GENOPT processor called GENTEXT destroys behavior.new and struct.new, replacing them with new "skeletal" versions of behavior.new and struct.new. During the development of this "trusscomp" capability the writer made it a habit always to work with the files, behavior.trusscomp and bosdec.trusscomp, then

copy them as follows:
cp behavior.trusscomp behavior.new
cp bosdec.trusscomp bosdec.src
just before typing the command "genprograms". (Please read carefully the "digressions" in Table 8).
****

(see especially pp. 52-56)

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

OPTIMIZATION OF THE SPECIFIC CASE CALLED "test"

Tables 14 - 29 and Figs. 17 - 27 pertain to this section. pp. 126 - 176

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

Input for BEGIN: (p. 5 of Table 8)

Table 14 lists the input for the "BEGIN" processor. The starting design of the composite truss-core sandwich wall is the optimum design found in [9] and listed in Table 26 of [9]. Only one material type is used in this specific application of "trusscomp", the specific application called "test". In the input for "layer type" (LAYTYP(i,j)) the column number j is the shell segment number in Fig. 1 and the row number i is the composite wall layer number in shell segment j. Note that the layer types for the layers in the various shell segments change in a subtle manner. For example, Shell Segment number 2 has 5 layers with the symmetric layup, layer types 1, 2, 3, 2, 1 . In contrast, Shell Segment number 5, which consists of the same layers, has layer types 2, 1, 3, 1, 2 , that is, the +45-degree and -45-degree layer types are reversed. This has to do with the way the trapezoidal tool is wrapped and the layer numbering convention used in BIGBOSOR4: layer numbers increase from left to right as one faces in the direction of increasing nodal point number within a shell segment. Figure 1 shows the directions for increasing nodal point number in each of the 22 shell segments in the single module model used for local buckling.

Table 14 on pp. 126-129

Fig. 1 on p. 28

Input for DECIDE: (p. 6 of Table 8)

Table 15 lists the input for the "DECIDE" processor. Note that there are two linking expressions and six inequality constraints. The two linking expressions cause the -45-degree and +45-degree layers always to have the same thickness during optimization cycles. Hence, the 5-layer truss-core laminate with total thickness called "t(core)" in Fig. 8 is always balanced, and the 7-layer laminate of thickness denoted "t(flat)" in Fig. 8 is always balanced.

Table 15 on pp. 130-131

Fig. 8 on p. 35

The purpose of the six inequality constraints is to prevent errors during the execution of the BIGBOSOR4 preprocessor, SUBROUTINE B4READ. The first inequality keeps the web that includes shell segment 7 in Fig. 1 slanted in the same way as is shown in Fig. 1. The second and third inequalities prevent shell segments such as Segments 1 and 15 in Fig. 1 from having zero or negative lengths. The fourth and fifth inequalities prevent shell segments of the type 7 and 12 in Fig. 1 from having zero or negative lengths. The sixth inequality prevents shell segments of the types 4 and 11 from having zero or negative lengths.

Fig 1 on p. 28

Analysis of the starting design: (see p. 6 of Table 8)

Table 16 lists the input for the "MAINSETUP" processor for a

Table 16 on p. 132

case in which a fixed design is being analyzed (not optimization, that is, ITYPE=2, not ITYPE=1). The intention here is first to see what local and general buckling loads we obtain from the "trusscomp" model of the optimum design obtained by PANDA2 in the "nasatruss" project [9].

Table 17 lists the test.OPM file corresponding to the starting design established in BEGIN (Table 14 with different factors of safety: those listed in Table 27 rather than those listed in Table 14). Tables 18 and 19 are partial listings of the properly annotated versions of the test.ALL files used for general and local buckling, respectively. The not-properly-annotated versions of these files, which contain valid input for the BIGBOSOR4 preprocessor, B4READ, are produced by SUBROUTINE BOSDEC with INDX = 2 (general buckling model) and INDX = 1 (local buckling model).

Table 17 on pp. 133-136
Table 27 on p. 168
Table 14 on pp. 126-129
Tables 18 & 19 on pp. 137-149

Compare the prebuckling loading, FN20, which is the circumferential stress resultant in the "huge torus" trusscomp model, with the axial resultant, Nx(var), in the PANDA2 model listed for the various PANDA2 shell segments at the bottom of p. 1 and in the top half of p. 2 of Table 7. The value of FN20 in Shell Segment 1 (-2193.258 lb/in) is somewhat less than the corresponding value of Nx(var) listed at the bottom of Table 7 (-2373.9 lb/in) because the "noodles", not present in the PANDA2 model, absorb some of the axial load in the "huge torus" trusscomp model.

Table 7 on pp. 47-50
see p. 137
see p. 47

p. 1 of 4

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

Table 20 lists the input for the "MAINSETUP" processor for a case in which optimization is desired, that is, ITYPE = 1 and NPRINT = 0. Some of the strategy input data is rather opaque. Therefore these data are explained in the following paragraphs:

Table 20 on p. 150

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 20:
```

For the response to the prompt in Table 20:

```
      y         $ Do you want default (RATIO=10) for initial move limit jump?
```

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

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

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

       The purposes of the "move limit jump" are: (1) to enable decision
       variables that are near zero to escape this neighborhood,  and
       (2) to permit exploration of an expanded segment of the domain
       of the decision variable in the search for an optimum.

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

 743.1 Provide a value for the "move limit jump" ratio, RATIO
 743.2
       If zero is included in the domain of any decision variable
       it may be best to use the default value, RATIO = 10.

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

       If in doubt, use the default value.
```

In response to the prompt in Table 20:

```
      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 20:

```
      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 20 for cases similar to "test" you will be ok.

Table 20 on p.150

Optimization with all factors of safety = 1.0:

We terminated the SUPEROPT run after about 6 hours of run time in order to get plots of margins, decision variables, and the objective versus design iterations. Table 21 lists the input for CHOOSEPLOT that leads to Figs. 17 - 19. The command, "DIPLOT" in this example leads to three postscript files, test.3.ps (which contains design margins versus design iterations), test.4.ps (which contains some of the design variables versus design iterations), and test.5.ps (which contains the objective versus design iterations). We want to get plots of additional decision variables versus design iterations. Therefore, we execute CHOOSEPLOT again, this time with the data listed in Table 22 as input. Figure 20 is a plot of the new postscript file, test.4.ps, generated by DIPLOT. Table 23 leads to Fig. 21, and Table 24 leads to Fig. 22. We plot the decision variables in separate figures because they have different orders of magnitude and therefore would not show up well if all the decision variables were plotted in the same frame.

See pp. 151 - 160

Analysis of the optimized design with all factors of safety = 1.0:

Table 25 lists the test.OPM file for the optimized design. This optimum design is preserved by executing the processor, "CHANGE", which generates an input file for CHANGE called "test.CHG" (Table 26). The user is urged always to use CHANGE to save optimum designs. By doing so, the user can easily resurrect previously obtained optimum designs in the future by executing CHANGE and using the file, test.CHG, as input rather than having tediously to provide the previously obtained optimum design in an interactive mode.

Table 25 on pp. 161 - 164
Table 26 on p. 165

Next, we wish to use BIGBOSOR4 to obtain plots of the local and general buckling modes of the optimized design. As indicated in Table 8, this is done with use of the files, test.BEHX1 (local buckling model) and test.BEHX2 (general buckling model). The files, test.BEHX1 and test.BEHX2, are generated in SUBROUTINE BEHX1 and SUBROUTINE BEHX2, respectively. For the local buckling model, the file test.BEHX1 is generated from the following FORTRAN code in SUBROUTINE BEHX1:

Table 8 on pp. 52 - 69

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

See p. 97

There is analogous coding in SUBROUTINE BEHX2 to generate the file, test.BEHX2 for the general buckling model.

(see p. 99)

Figure 23 shows the local buckling mode and Fig. 24 shows the

p. 166, 167

general buckling mode predicted by BIGBOSOR4. In the "huge torus" model N is the number of full circumferential waves around the circumference of the huge torus. As is explained in Appendix 1, N = 800 in the "huge torus" model corresponds to 8 axial half-waves over the axial length, FACLEN x LENGTH, of the cylindrical shell. In Figure 24 N = 500 in the "huge torus" model corresponds to 5 axial half-waves over the axial length, LENGTH, of the cylindrical shell.

Appendix 1 on pp 188-190

FACLEN defined in Table 10 on p. 84

Optimization with the use of new factors of safety (f.s. > 1.0):

Initial imperfections cannot be accounted for directly in the "huge torus" model of the cylindrical shell. Therefore, it is bad practice to use factors of safety equal to unity. Doing so leads to unsafe optimum designs. Therefore, we optimize again, this time using factors of safety that are greater than unity.

Table 27 lists the input data in the file, test.BEG, that include the new factors of safety. These new factors of safety might well lead to an optimum design that is safe for a shell that is mildly imperfect.

Table 27 on p. 168

This time SUPEROPT is allowed to continue until it is finished. About 3 days of computer time are required for the approximately 470 design iterations processed in a single SUPEROPT execution.

Figure 25 shows the objective versus design iterations. Each "spike" in the plot corresponds to a new "starting" design, obtained randomly but consistent with upper and lower bounds, equality constraints, and now inequality constraints. (See Item no. 29 in the file,...genopt/doc/genopt.news). Each new "starting" design is obtained with the processor, AUTOCHANGE. By means of the automated repeating pattern of a new "starting" design generated by AUTOCHANGE followed by several sequential executions of OPTIMIZE, it is hoped that a "global" optimum design will be determined, since each new "starting" design represents a different point in design space. A typical repeating pattern taken from the test.OPP file generated by SUPEROPT is as follows:

Fig. 25 on p. 169

ITERATION NUMBER	OBJECTIVE	THE DESIGN IS...	NUMBER OF CRITICAL MARGINS	
				AUTOCHANGE
				OPTIMIZE
144	1.6408E-02	FEASIBLE	1	<--new "starting" design
145	1.4363E-02	FEASIBLE	2	
146	1.3056E-02	FEASIBLE	2	
147	1.2115E-02	FEASIBLE	2	
148	1.1516E-02	FEASIBLE	2	
149	1.1201E-02	FEASIBLE	2	
				OPTIMIZE
150	1.1201E-02	FEASIBLE	2	
151	1.0477E-02	FEASIBLE	2	
152	9.9538E-03	FEASIBLE	3	
153	9.7464E-03	FEASIBLE	4	
154	9.6760E-03	FEASIBLE	4	
155	9.6198E-03	FEASIBLE	4	
				OPTIMIZE
156	9.6198E-03	FEASIBLE	4	
157	9.5526E-03	FEASIBLE	5	
158	9.4745E-03	FEASIBLE	5	
159	9.3899E-03	ALMOST FEASIBLE	5	
160	9.4655E-03	FEASIBLE	5	
161	9.4632E-03	FEASIBLE	5	
				OPTIMIZE
162	9.4632E-03	FEASIBLE	5	
163	9.3167E-03	ALMOST FEASIBLE	5	
164	9.4351E-03	ALMOST FEASIBLE	5	
165	9.4540E-03	ALMOST FEASIBLE	5	
166	9.4030E-03	ALMOST FEASIBLE	5	
167	9.4594E-03	FEASIBLE	5	
				OPTIMIZE
168	9.4594E-03	FEASIBLE	5	
169	9.4680E-03	FEASIBLE	5	
170	9.3639E-03	ALMOST FEASIBLE	5	
171	9.4864E-03	FEASIBLE	5	

```
   172         9.4412E-03     ALMOST FEASIBLE         5
   173         9.3905E-03     ALMOST FEASIBLE         5
 ------------------------------------------------------------
```

In this particular example the user specified in the launch
of the SUPEROPT run that there be 5 execution of OPTIMIZE
following each execution of AUTOCHANGE.

In this particular case an interesting phenomenon occurs.
All new "starting" designs except one lead to a local optimum
design with specific weight from about 0.00939 lb/in^2 to 0.00945 lb/in^2.
In one instance only the "starting" design leads to a slightly
lower specific weight, 0.009245 lb/in^2. This occurs as follows:

```
 ------------------------------------------------------------
 ============================================================
    ITERATION                                    NUMBER OF
  NUMBER      OBJECTIVE     THE DESIGN IS...   CRITICAL MARGINS
 ------------------------------------------------------------
 ------------------------------------------------------AUTOCHANGE
 ------------------------------------------------------OPTIMIZE
   228         1.2238E-02      NOT FEASIBLE           3
   229         1.3235E-02     MOSTLY UNFEASIB         3
   230         1.3167E-02         FEASIBLE            2
   231         1.2201E-02         FEASIBLE            2
   232         1.1575E-02         FEASIBLE            3
   233         1.1199E-02         FEASIBLE            3
 ------------------------------------------------------OPTIMIZE
   234         1.1199E-02         FEASIBLE            3
   235         1.0393E-02         FEASIBLE            4
   236         1.0072E-02         FEASIBLE            3
   237         9.8877E-03         FEASIBLE            4
   238         9.7369E-03         FEASIBLE            4
   239         9.6301E-03         FEASIBLE            4
 ------------------------------------------------------OPTIMIZE
   240         9.6301E-03         FEASIBLE            4
   241         9.4053E-03         FEASIBLE            3
   242         9.2551E-03         FEASIBLE            4
   243         9.2450E-03         FEASIBLE            4
 ------------------------------------------------------OPTIMIZE
   244         9.2450E-03         FEASIBLE            4
 ------------------------------------------------------OPTIMIZE
   245         9.2450E-03         FEASIBLE            4
 ------------------------------------------------------------
```

Consistent with the strategy described in ITEM 17 (dated
November, 2005) of the file, ...genopt/doc/genopt.news,
the "best" design (specific weight = 0.009245 lb/in^2) is revisited
twice more during the SUPEROPT execution, as follows:

```
 ------------------------------------------------------------
 ------------------------------------------------------AUTOCHANGE
 ------------------------------------------------------OPTIMIZE
   297         1.2028E-02     ALMOST FEASIBLE         3
   298         1.1578E-02         FEASIBLE            3
   299         1.1031E-02         FEASIBLE            3
   300         1.0723E-02         FEASIBLE            4
   301         1.0496E-02         FEASIBLE            4
   302         1.0345E-02         FEASIBLE            4
 ------------------------------------------------------OPTIMIZE
   303         9.2450E-03         FEASIBLE            4
 ------------------------------------------------------OPTIMIZE
   304         9.2450E-03         FEASIBLE            4
 ------------------------------------------------------OPTIMIZE
   305         9.2450E-03         FEASIBLE            4
 ------------------------------------------------------OPTIMIZE
   306         9.2450E-03         FEASIBLE            4
 ------------------------------------------------------------
```

and

```
 ------------------------------------------------------------
 ------------------------------------------------------AUTOCHANGE
 ------------------------------------------------------OPTIMIZE
   447         1.7639E-02         FEASIBLE            1
   448         1.5509E-02         FEASIBLE            2
   449         1.4000E-02         FEASIBLE            2
   450         1.2909E-02         FEASIBLE            2
   451         1.2098E-02         FEASIBLE            3
   452         1.1584E-02         FEASIBLE            3
```

453	9.2450E-03	FEASIBLE	4 OPTIMIZE
454	9.2450E-03	FEASIBLE	4 OPTIMIZE
455	9.2450E-03	FEASIBLE	4 OPTIMIZE
456	9.2450E-03	FEASIBLE	4 OPTIMIZE

It is clear from these listings that SUPEROPT cannot find a better design (less weight and FEASIBLE or ALMOST FEASIBLE) in the immediate neighborhood of the design the specific weight of which is 0.009245 lb/in^2.

Analysis of the optimized design with all factors of safety > 1.0:

Table 28 lists the optimum design obtained after a single execution of SUPEROPT. Note that all the thicknesses go to their lower bounds. The reader should perhaps rerun this case using different layups for the truss-core and for the face sheets. A lighter-weight design might well be found if fewer layers were used in these parts of the structure.

Table 28 on pp. 170-173

Note!

The design margins of the optimized structure are listed on page 3 of Table 28. Both the local and general buckling margins are critical. One of the stress margins, tension transverse to fibers, is critical. This stress is produced mostly by the curing operation (non-zero curing temperature, TEMCUR = 240 degrees specified in test.BEG).

See p. 126

Table 29 lists the file by means of which the optimized design is saved. Again, the user is urged ALWAYS to use CHANGE to save optimum designs. The file, test.CHG, can be renamed so that it will not be replaced by a file of the same name generated in some future execution of CHANGE that pertains to a different optimum design. For example, we have stored the following "CHANGE" files in the directory, ...genopt/case/trusscomp:

Table 29 on p. 174

test.fs1.0.CHG (test.CHG file listed in Table 26) ← p. 165
test.fs2.0.superopt1.CHG (test.CHG file listed in Table 29) ← p. 174

Figures 26 and 27 show critical local and general buckling modes obtained from the use of the files, test.BEHX1 and test.BEHX2, respectively, in a manner that is explained in Table 8 and that has been explained previously in this text.

pp. 175, 176

Table 8 on pp 52-69

Convergence of local buckling load with increasing modules:

It is a good idea to perform a convergence study of local buckling load versus the number of modules, NMODULL, used in the local buckling model. The sequence of commands to accomplish this occurs starting on p. 14 of Table 8. Figures 28 - 30, together with Fig. 26, show the results of such a study. It is clear that the one-module local buckling model yields an accurate enough prediction of local buckling load in this particular case.

pp. 177-179

The slightly heavier "local" optimum to which SUPEROPT tends often to converge in Fig. 25:

Fig. 25 on p. 169

Table 30 and Figs. 31 and 32 pertain to this section.

pp. 180-182

Table 30 lists results corresponding to one of the slightly heavier locally optimum designs. Figures 31 and 32 show the local and general buckling modes for that slightly heavier locally optimum design. The truss-core is significantly thicker (that is, HEIGHT is greater) in the slightly heavier design than in the "global" optimum design depicted in Figs. 26 and 27. Also, the truss core webs have less of a slant in the slightly heavier design than they do in the "global" optimum design. Compare Figs. 31 with 26 and 32 with 27.

p. 180
pp. 181-182

Figs. 26 & 27 on pp. 175 & 176

A "bomb" of the SUPEROPT run is possible but now unlikely:

With the 6 inequality constraints described in Table 15 and listed on page 4 of Table 17, a "bomb" of SUPEROPT is now unlikely. Before some of these inequality constraints were introduced into the test.DEC file (input for DECIDE), a configuration such as that shown in Fig. 33 occurred during optimization cycles. This configuration caused SUPEROPT to "bomb" because Shell segment no. 7 (see Fig.1)

Table 15 on pp. 130, 131 (see p. 136)

Fig 33 on p. 183

Fig. 1 on p. 28

had a length very close to zero. The "bomb" occurred in the execution of the BIGBOSOR4 preprocessor, B4READ, in SUBROUTINE BEHX1 because the nodal point numbers specified as call-outs in Segment 7 did not follow certain rules set forth in BIGBOSOR4 regarding call-out points. (see p. 97)

If a "bomb" should occur during a SUPEROPT run, the user should inspect the file called "test.OUT". The end of this test.OUT file should contain an error message. The user can tell if a "bomb" occurred in SUPEROPT by looking at the file, test.OPP, to see if the number of design iterations is greater than or equal to 470. If it is less than 470, then for some reason SUPEROPT terminated before it was finished. Assuming that the electricity did not go off or that the user's computer did not otherwise fail during the long SUPEROPT execution, a logical error occurred. Should this happen to you, you can do the following:

1. Inspect the end of the test.OUT file to see if there is an error message.

2. Use CHANGE to increase the height, HEIGHT, of the truss-core.

3. Re-execute SUPEROPT so that it can finish the approximately 470 design iterations.

COMPARISON OF PANDA2 PREDICTIONS AND GENOPT/BIGBOSOR4 PREDICTIONS FOR LOCAL BUCKLING OF THE COMPOSITE TRUSS-CORE SANDWICH SHELL WITH THE OPTIMUM CONFIGURATION LISTED IN TABLE 28

Table 31 and Figs. 34 - 36 pertain to this section. There is a rather lengthy explanation of what is going on starting at the bottom of page 15 in Table 8 and continuing to the end of Table 8. That explanation will not be repeated here. pp. 184 – 187 Table 8 on pp. 52-69

WARNING!

With the "trusscomp" model it is likely that you will obtain an optimum design in which both the local and general buckling margins are critical. No provision has been made here to accouunt for the increased imperfection sensitivity that occurs whenever local and general buckling occur simultaneously or whenever local buckling occurs at a lower load than general buckling. In the optimization in which the factor of safety for general buckling is set equal to 2.0 and the factor of safety for local buckling is set equal to 1.5, the optimized design undergoes local buckling at a lower load than for general buckling. What probably actually happens in this situation is that the skin buckles locally and the shell continues to accept some additional axial compression. However, the general buckling load is lower than predicted for the "trusscomp" model because the effective axial stiffness of locally buckled skin or webs is about half that of the unbuckled skin or webs. Therefore, the general buckling load really does not have a factor of safety of 2.0, but some factor of safety between 1.5 (at which local buckling occurs) and 2.0 (at which general buckling occurs with the assumption that there is no effect of local buckling on general buckling). (see Table 27 on p. 168)

Also, the local buckling phenomenon may actually be a process rather than a single event. Suppose Segment 6 in Fig. 1 is the first to buckle locally. In doing so, it sheds axial load. The diminished load in Segment 1 must then be carried by the remaining unbuckled segments of the module cross section. The additional axial load on the remaining segments of the cross section will give rise to earlier buckling of these parts than would be predicted by the "trusscomp" theory developed here. Fig. 1 is on p. 28

In the theory developed here the effect of local postbuckling with resulting locally diminished axial stiffness is in no way accounted for.

The reader should bear this warning in mind when establishing what factors of safety to use for both local and general buckling, and for stress as well. Note!

There is no prebuckling bending at all in the "trusscomp" model. The reader should take this into account when establishing factors of safety for stress components.

CONCLUSION

Engineers and researchers at NASA Langley are urged to use the "trusscomp" capability developed here to undertake further investigation of optimum designs of axially compressed composite truss-core sandwich cylindrical shells. For example, it seems probable that the use of fewer layers in the truss-core and fewer layers in the face sheets would lead to lighter-weight optimum designs, since all the thickness decision variables went to their lower bounds (ply thickness = 0.0052 inch) during optimization cycles with use of the configuration provided in the input file for BEGIN (the test.BEG file).

(test.BEG in Table 14 on pp. 126-129)

Before any further "trusscomp" work can be done at NASA Langley, NASA Langley will have to obtain the latest versions of GENOPT and BIGBOSOR4, since changes in these tools have been made in order to accommodate the "trusscomp" geometry. The writer will await a request from NASA Langley before sending the latest versions of GENOPT and BIGBOSOR4, in case he wants to make further changes to these codes in the near future.

!! ← Note!

The writer hopes that engineers and researchers at NASA Langley and elsewhere will learn how to use GENOPT to obtain optimum designs of objects for which they already have analysis capability. In particular, it would seem beneficial for NASA Langley to have someone local who knows how to use the GENOPT/BIGBOSOR4 system to optimize shells of revolution.

← Note!

Table 1

REFERENCES

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

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

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

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

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

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

[7] Bushnell, David, "Optimization of an axially compressed ring and stringer-stiffened cylindrical shell with a general buckling modal imperfection", AIAA Paper 2007-2216, 48th AIAA Structures Meeting, Honolulu, Hawaii, April, 2007

[8] Bushnell, David, "Stress, buckling and vibration of prismatic shells", AIAA Journal, Vol. 9, No. 10, pp 2004-2013, October, 1971

[9] Bushnell, David, "Optimum designs from PANDA2 of a uniformly axially compressed cylindrical shell with a composite truss-core sandwich wall and verification of the design by BIGBOSOR4, unpublished report sent to NASA Langley Research Center, March 16, 2009

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

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

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

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

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

Table 2 (2 pages)

This table is taken from Ref. [4], the "weldland" project.

```
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

This table is taken from Ref. [4], the "weld land" project.

```
********** 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".

In the TRUSS-core sandwich case the generic case name is "trusscomp" 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. 3-6)

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

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

Typically, you type a command,

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

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

or, more simply, type the following:

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

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

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

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

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

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

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

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

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

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

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

for guidance in this important part of your effort.

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

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

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

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

****

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

Table 3 (p. 4-16)

********* 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, mainsestup.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

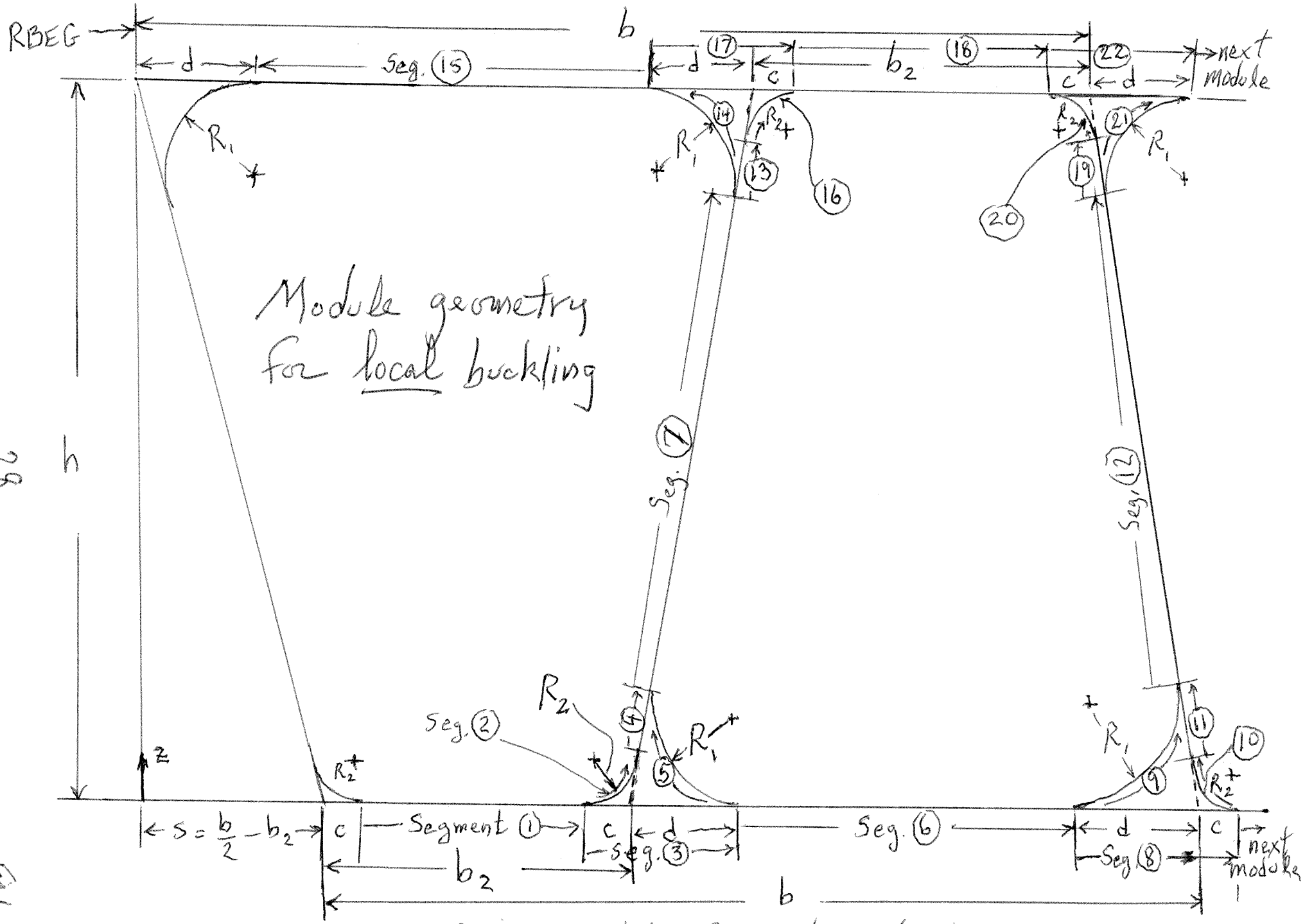
In order to re-run this case,
these three files must be copied to the correct locations.
If we are already in the directory, /home/progs/genoptcase,
we type the following:

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)
================================================================================
```



Single Module Segment numbering. Fig. 1

Output from BIGBOSOR4
from the input file, test. BEHX1

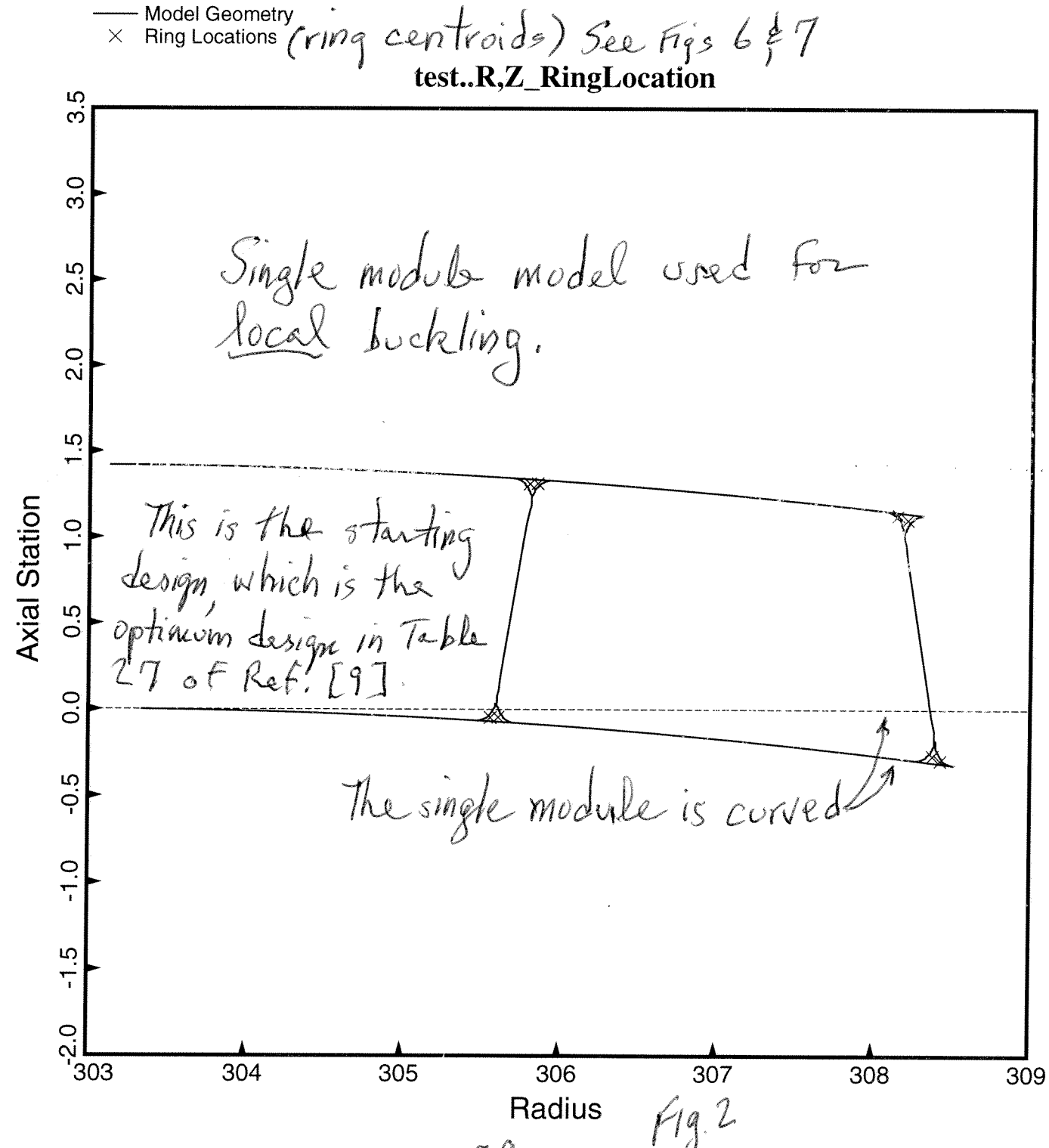
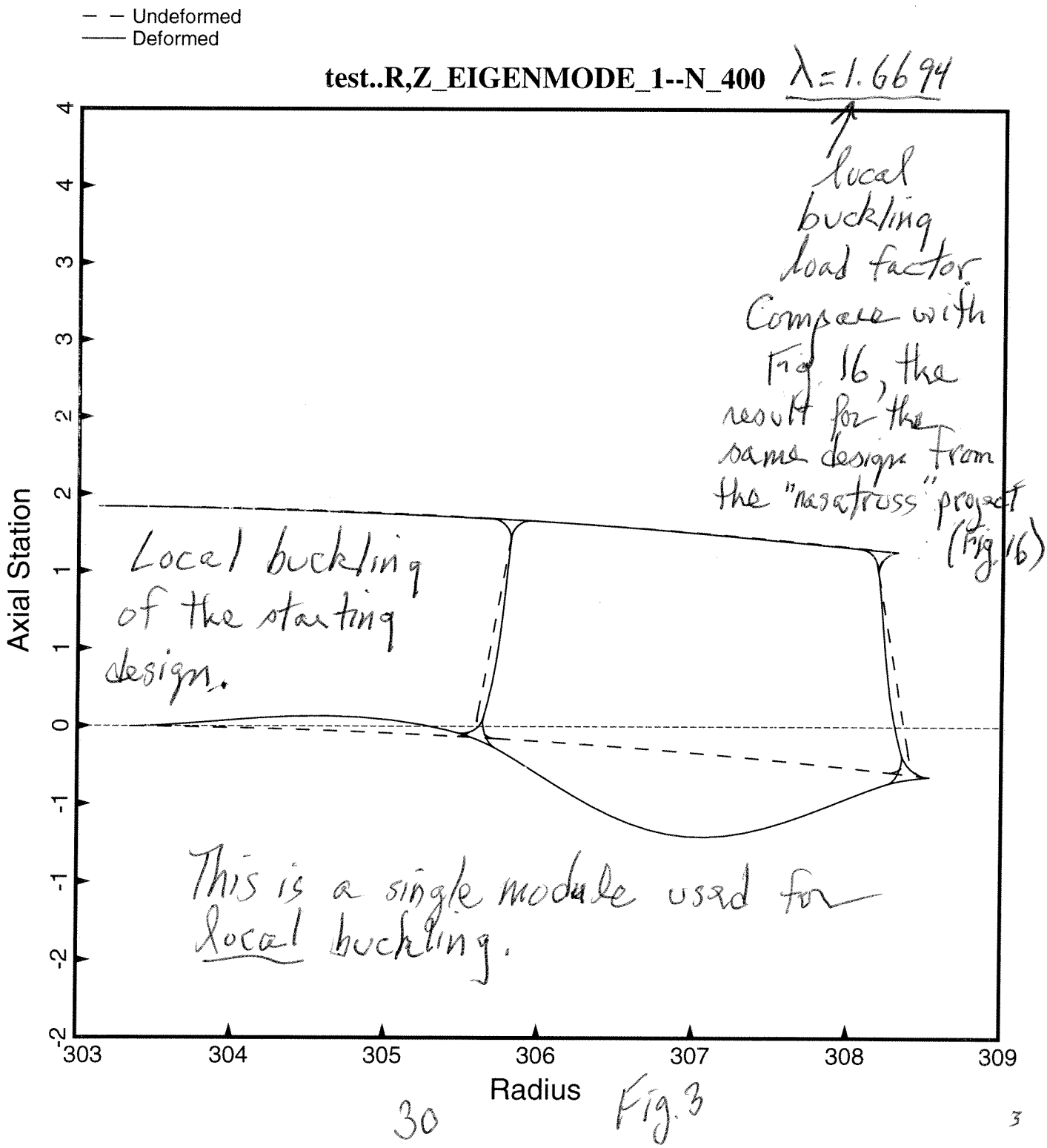


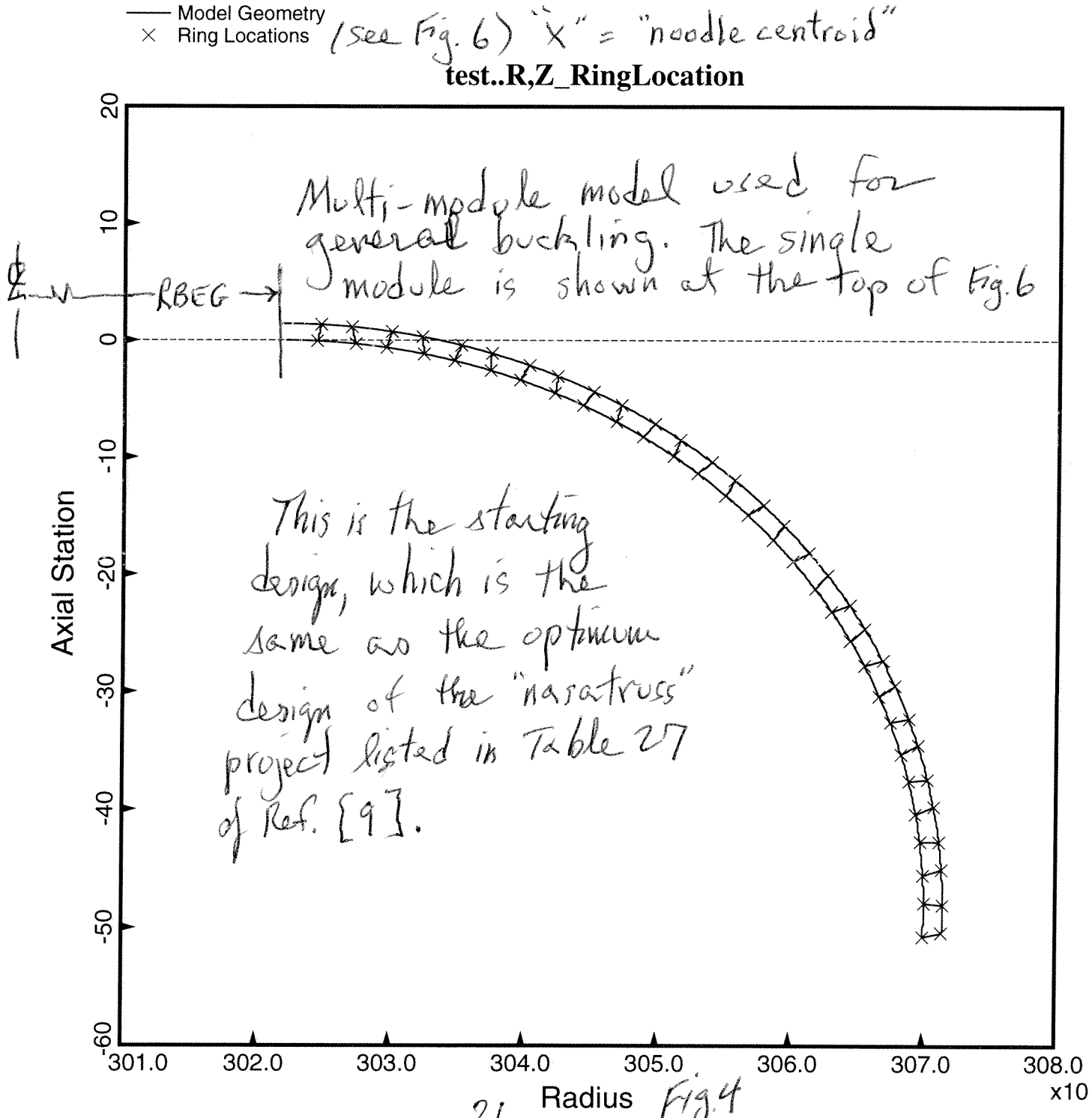
Fig. 2

Output from BIGBOSOR4
from the input file, test. BEHX1



30 Fig. 3

Output from BIGBOSOR4
from the input file, test. BEHX2



31

Fig. 4

Output from BIGBOSOR4
from the input file, test. BEHX2

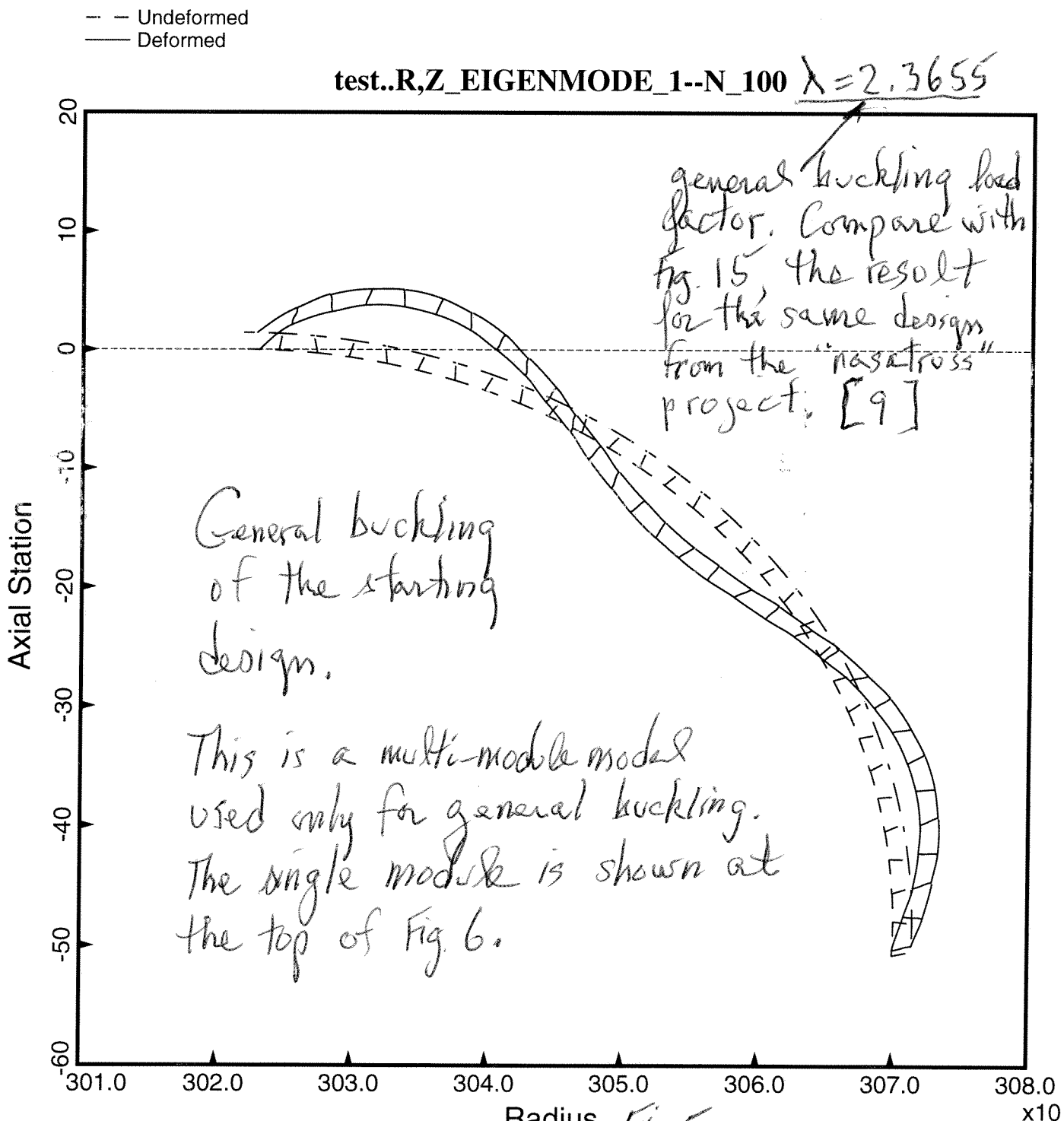
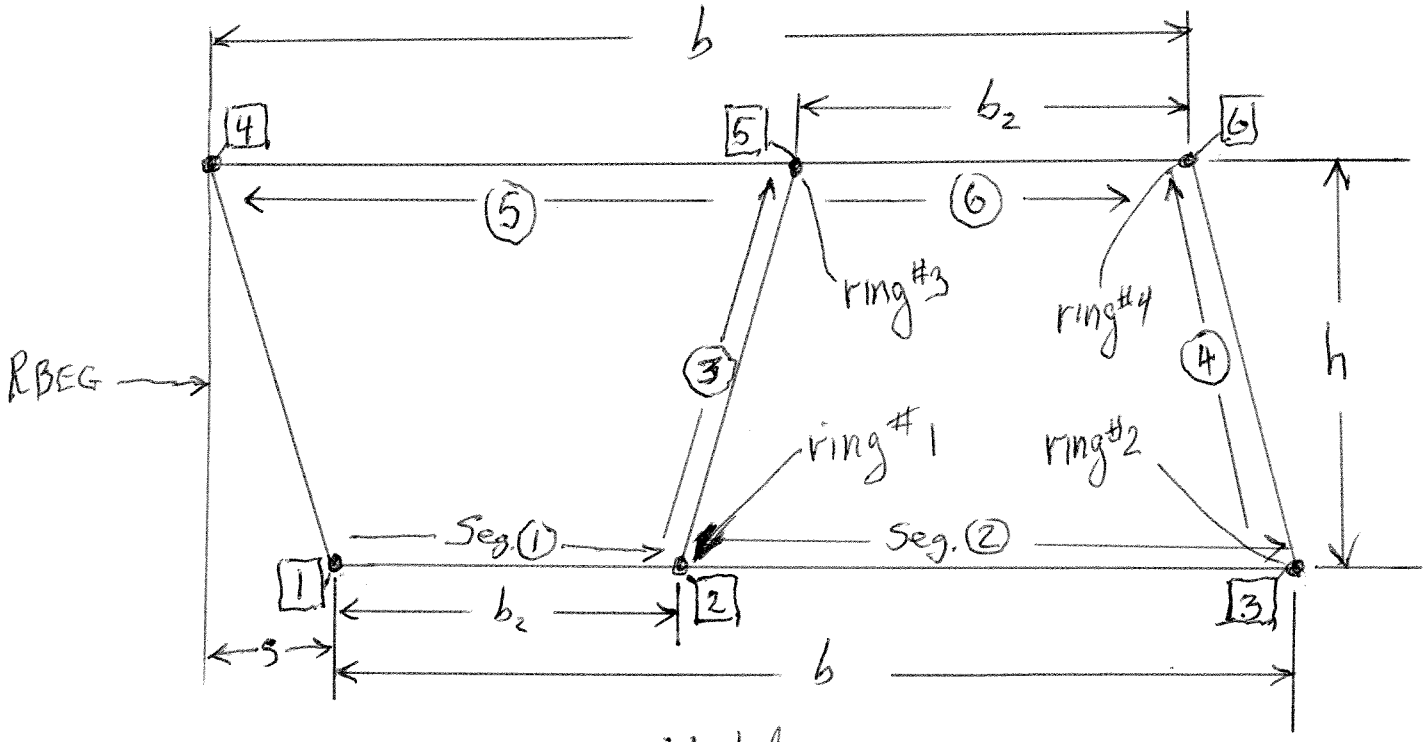
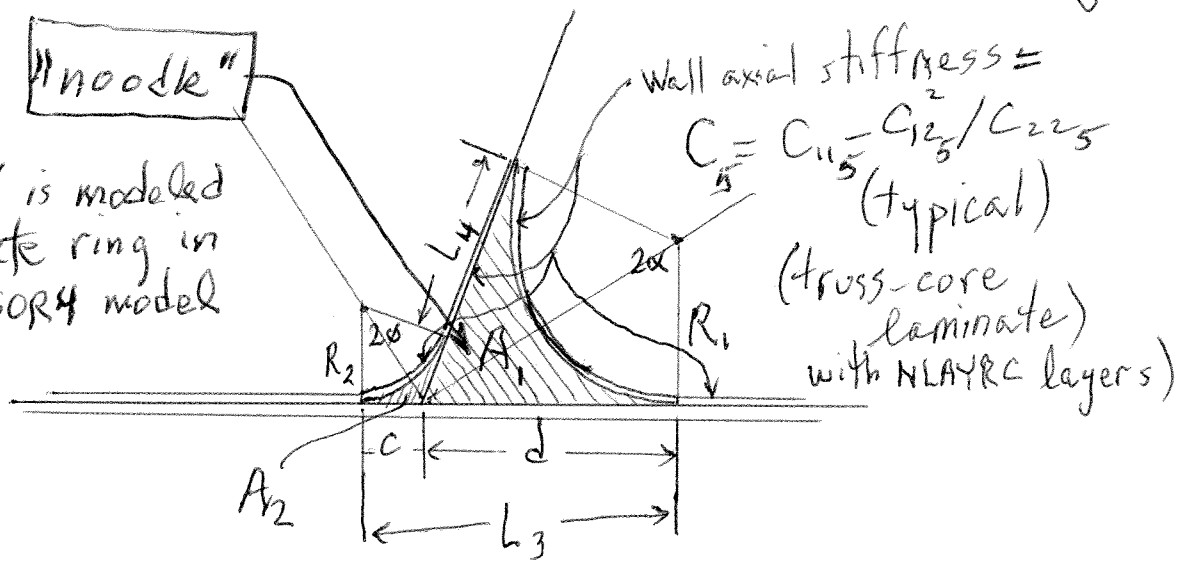


Fig. 5



Module

Simplified Model for General Buckling



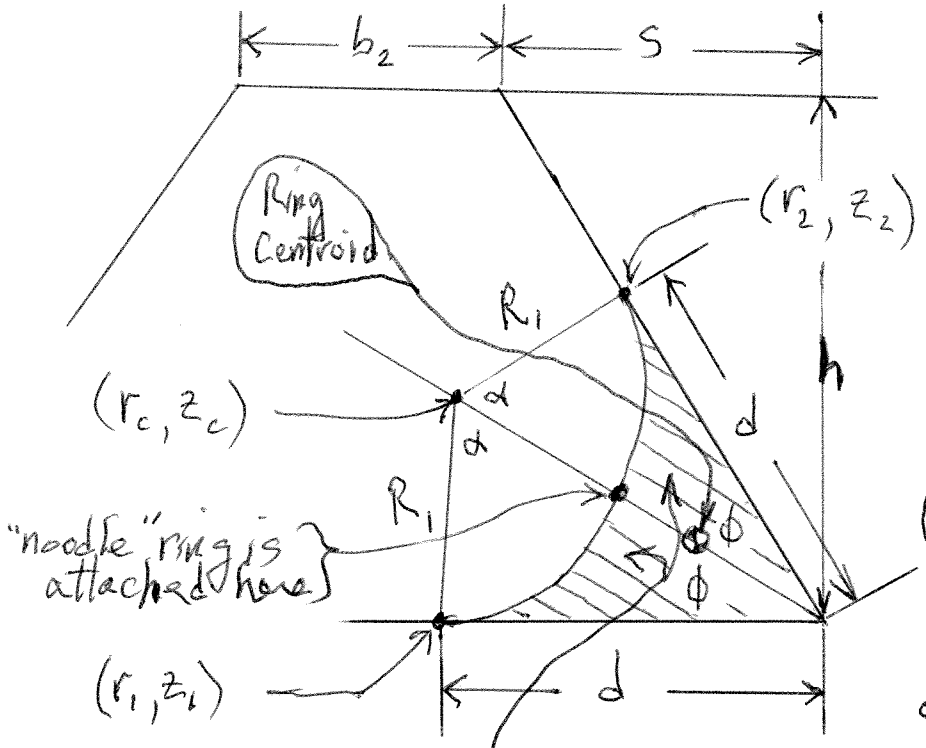
The "noodle" is modeled as a discrete ring in the BIGBOSOR4 model

Typical Ring

Ring axial stiffness = $ENOODL(A_1 + A_2) + C_5(2\phi R_2 + 2\alpha R_1 + L_4 - L_3)\sqrt{-2d}$

with $L_4 = d - c$; $L_3 = \frac{d}{33} + c$

Fig. 6



$$s = \frac{b}{2} - b_2$$

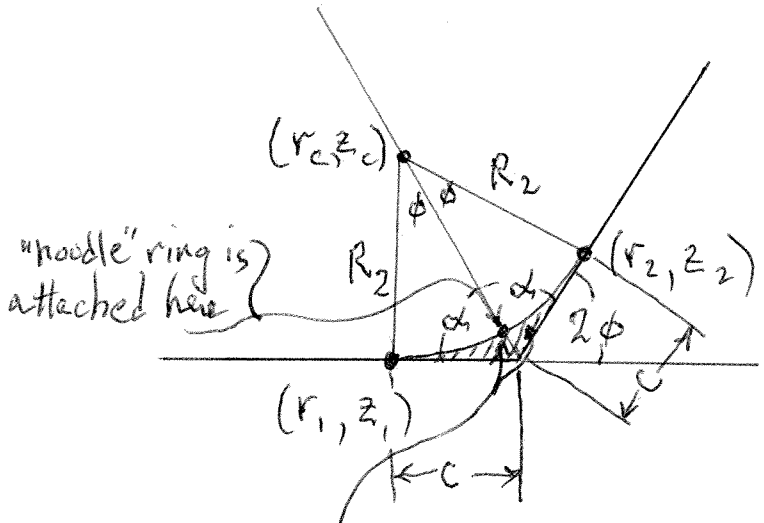
$$2\phi = \tan^{-1}(h/s)$$

$$(r_c, z_c) = (r_1, z_1 + R_1)$$

$$(r_2, z_2) = \left[r_1 + d(1-\cos 2\phi),\ z_1 + d\sin 2\phi\right]$$

$$d = \frac{R_1}{\tan\phi}$$

Shaded $AREA1 = R_1 d - \pi R_1^2 \frac{2\alpha}{2\pi}$



$$\alpha = 90 - \phi$$

$$c = \frac{R_2}{\tan\alpha}$$

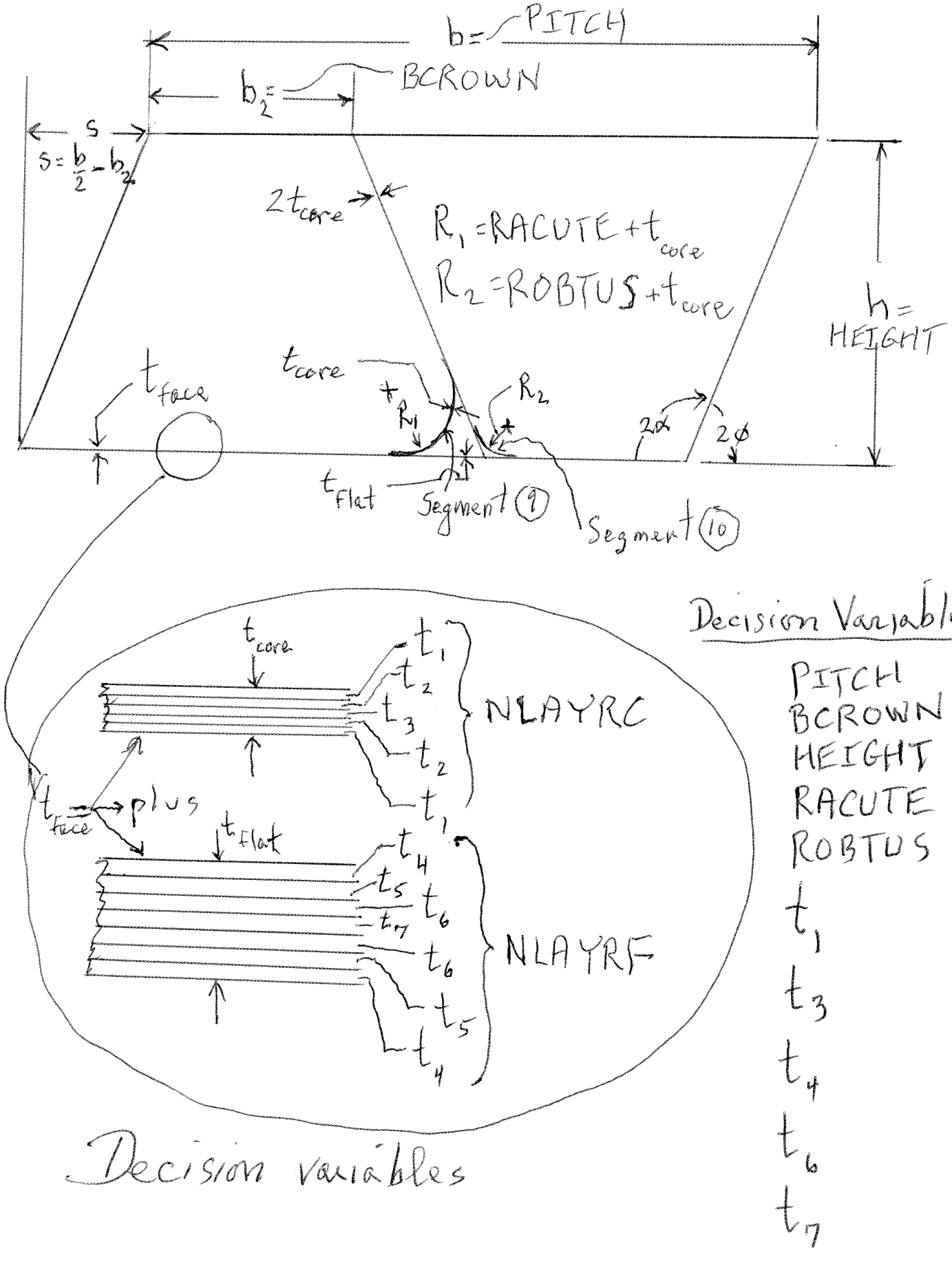
$$(r_c, z_c) = (r_1, z_1 + R_2)$$

$$(r_2, z_2) = \left[r_1 + c(1+\cos 2\phi),\ z_1 + c\sin 2\phi\right]$$

Shaded $AREA2 = R_2 c - \pi R_2^2 \frac{2\phi}{2\pi}$

Discrete ring ("noodle") cross section areas.

34 Fig. 7

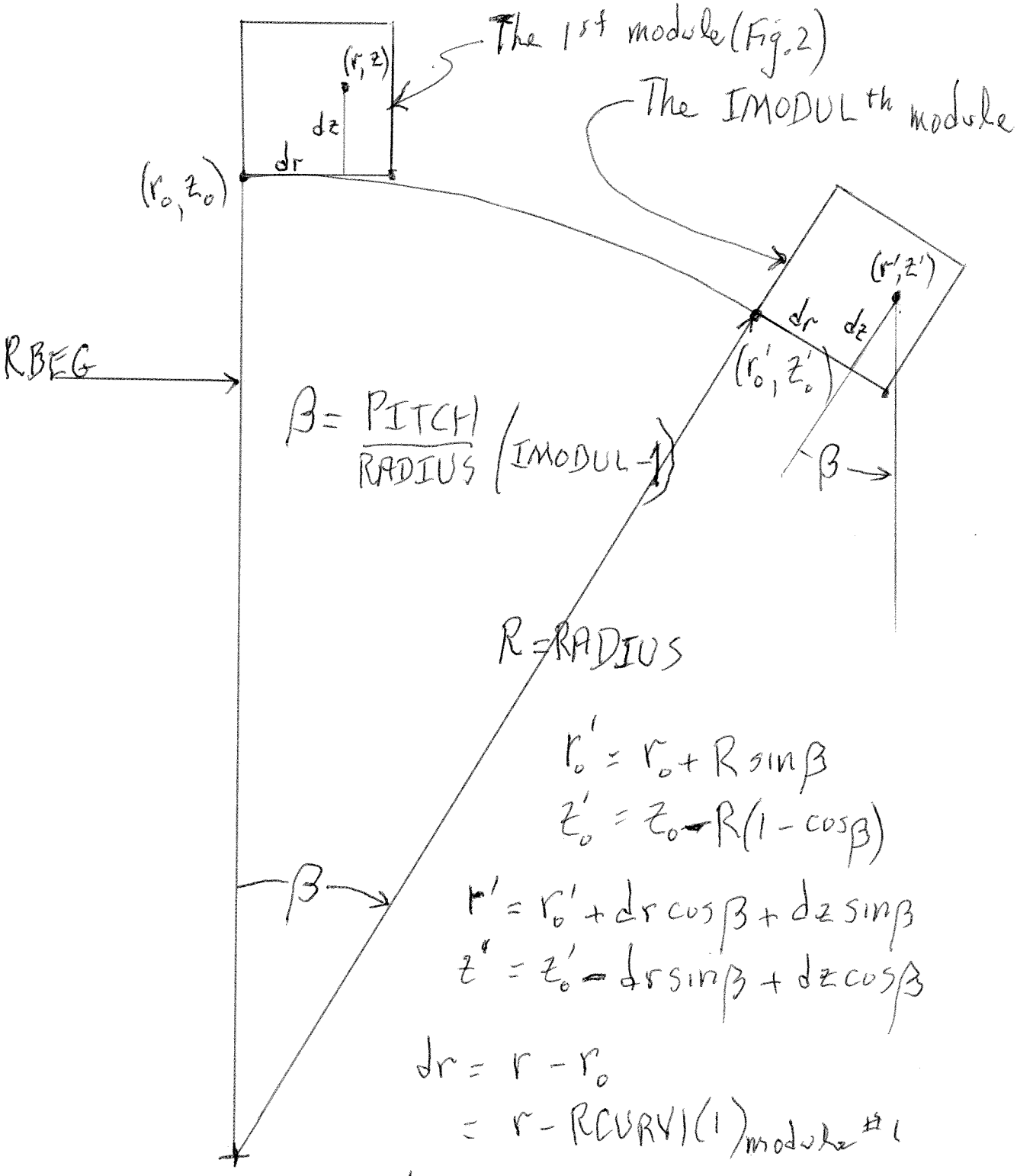


Decision variables

Decision Variables

PITCH
BCROWN
HEIGHT
RACUTE
ROBTUS
t_1
t_3
t_4
t_6
t_7

Fig. 8



$$r_0' = r_0 + R\sin\beta$$

$$z_0' = z_0 - R(1-\cos\beta)$$

$$r' = r_0' + dr\cos\beta + dz\sin\beta$$

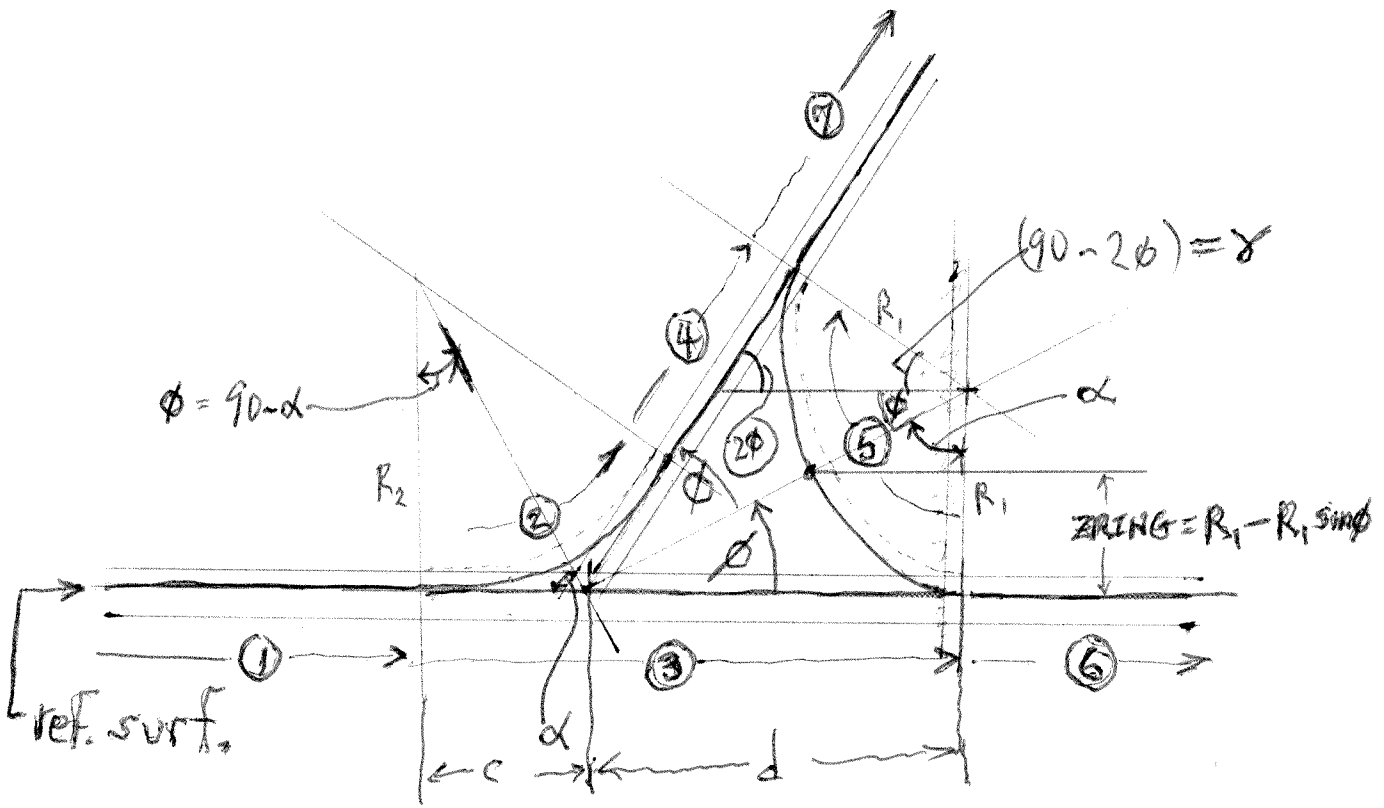
$$z' = z_0' - dr\sin\beta + dz\cos\beta$$

$$dr = r - r_0$$
$$= r - RCURV1(1)_{module\ \#1}$$

$$dz = z - z_0$$
$$= z - ZCURV1(1)_{module\ \#1}$$

Fig. 9

Fig. 10



Segment

① $(r_1 = RBEG + s + c\ ;\ z_1 = 0)\ ;\ (r_2 = r_1 + b_2 - c\ ;\ z_2 = 0)$

② $(r_1 = r_2(\text{seg}①),\ z_1 = z_2(\text{seg}.①))\ ;\ (r_2 = r_1 + c(1+\cos 2\phi)\ ;\ z_2 = z_1 + c\sin 2\phi)\ ;$

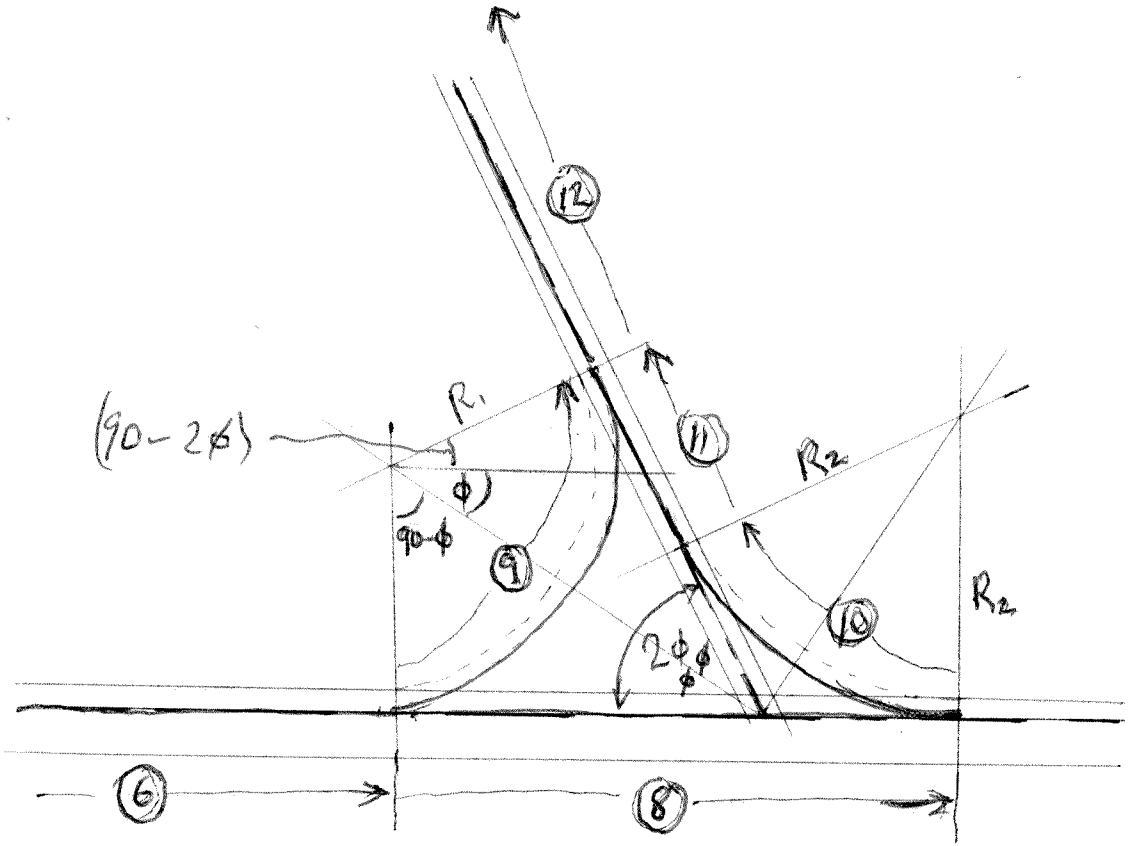
③ $(r_1 = r_2(\text{seg}.①)\ ;\ z_1 = z_2(\text{seg}.①))\ ;\ (r_2 = r_1 + d\text{+g}\ \ z_2 = 0)$ $(r_c = r_2(\text{seg}①)\ ;\ z_c = R_2)$

④ $(r_1 = r_2(\text{seg}②)\ ;\ z_1 = z_2(\text{seg}.②))\ ;\ (r_2 = r_2(\text{seg}③) - R_1\cos(90-2\phi)\ ;\ z_2 = R_1 + R_1\sin(90-2\phi))$

⑤ $(r_1 = r_2(\text{seg}③)\ ;\ z_1 = z_2(\text{seg}.③))\ ;\ (r_2 = r_2(\text{seg}④)\ ;\ z_2 = z_2(\text{seg}.④))\ ;\ (r_c = r_2(\text{seg}③),\ z_c = R_1)$

⑥ $(r_1 = r_2(\text{seg}.③)\ ;\ z_1 = z_2(\text{seg}.⑤))\ ;\ (r_2 = RBEG + s + b - d\ ;\ z_2 = 0)$

Fig 11



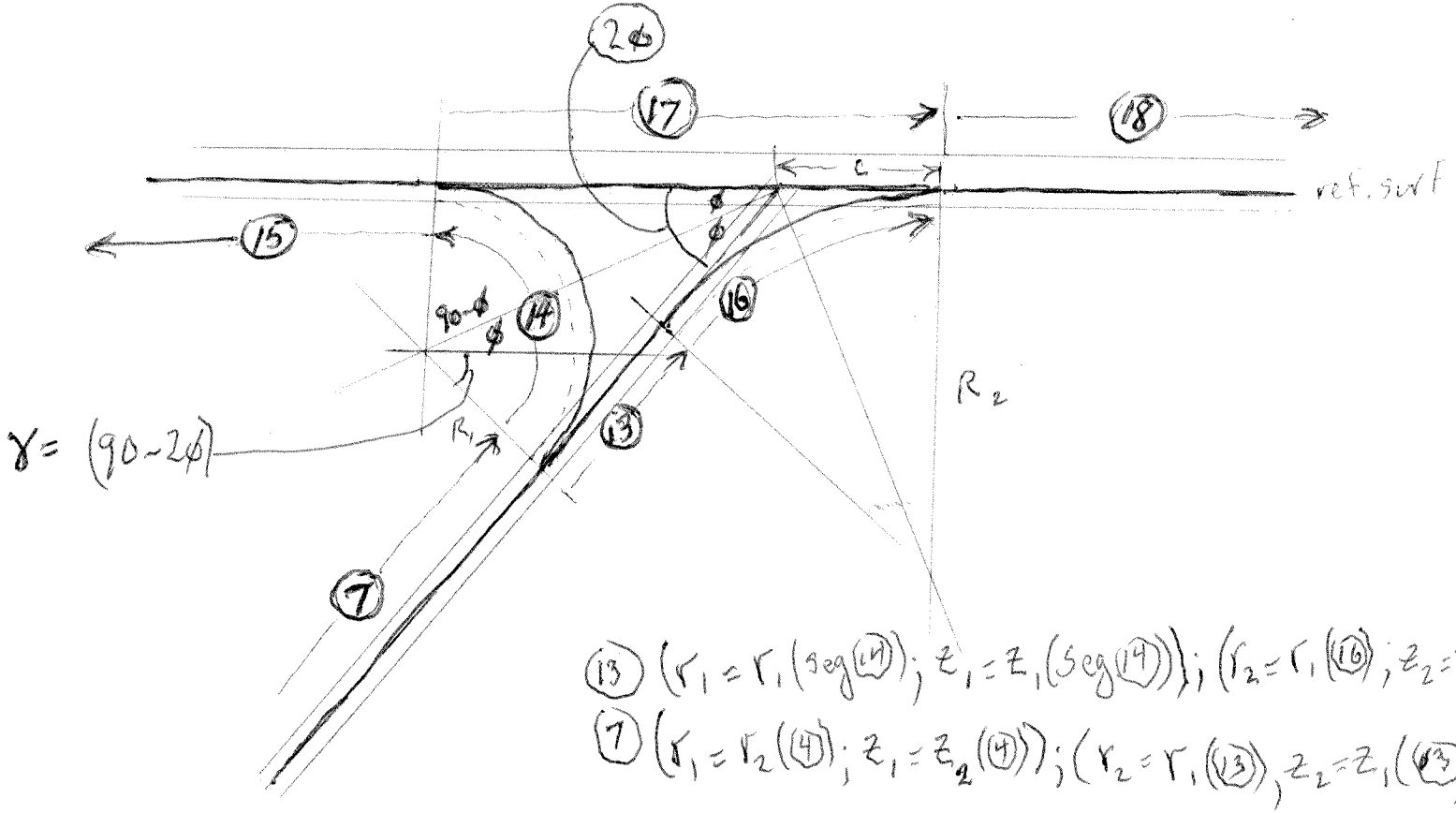
(8) $(r_1 = r_2(\text{seg. }6);\ z_1 = z_2(\text{seg. }6))$; $(r_2 = RBEG + s + b + c;\ z_2 = 0)$

(9) $(r_1 = r_2(\text{seg. }6);\ z_1 = z_2(\text{seg. }6))$; $(r_2 = r_1 + R_1\cos(90-2\phi);\ z_2 = z_2(\text{seg }7))$; $(r_c = r_1,\ z_c = R_1$

(10) $(r_1 = r_2(\text{seg }8));\ z_1 = 0)$; $(r_2 = r_1 - c(1+\cos 2\phi);\ z_2 = z_2(\text{seg. }2))$; $(r_c = r_1;\ z_c = R_2)$

(11) $(r_1 = r_2(\text{seg. }10));\ z_1 = z_2(\text{seg }10))$; $(r_2 = r_2(\text{seg }9);\ z_2 = z_2(\text{seg }9))$

Fig. 12



(13) $(r_1 = r_1(\text{seg}(14)); z_1 = z_1(\text{seg}(14)))$; $(r_2 = r_1(16); z_2 = z_1(16)$

(7) $(r_1 = r_2(14); z_1 = z_2(14))$; $(r_2 = r_1(13), z_2 = z_1(13))$

(15) $(r_1 = RBEG + b - b_2 - d; z_1 = h)$; $(r_2 = RBEG + d; z_2 = h)$

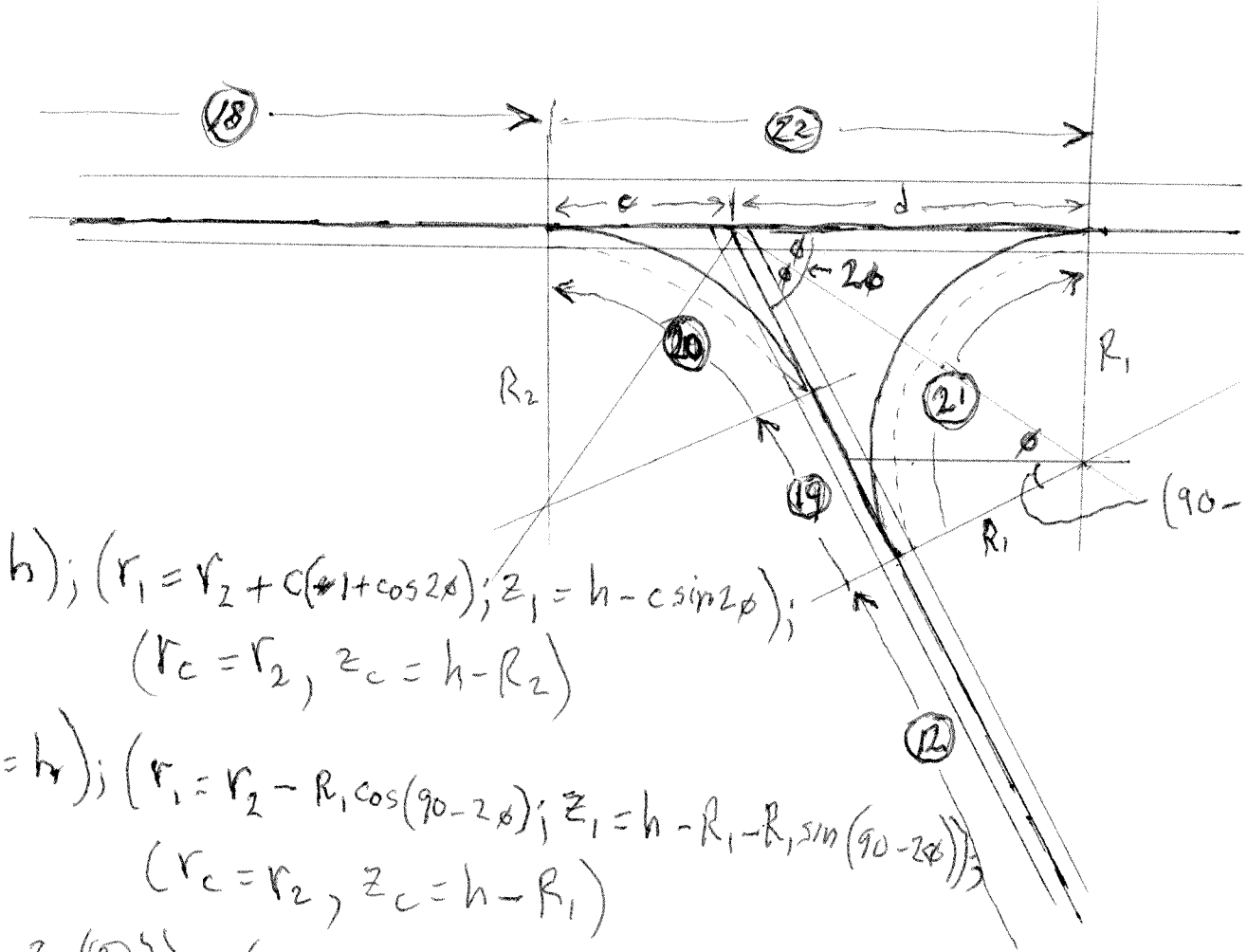
(17) $(r_1 = r_1(\text{seg}(15)); z_1 = h)$; $(r_2 = RBEG + b - b_2 + c; z_2 = h)$

(18) $(r_1 = r_2(\text{seg}(17)); z_1 = h)$; $(r_2 = RBEG + b - c; z_2 = h)$

(22) $(r_1 = r_2(\text{seg}(18)); z_1 = h)$; $(r_2 = r_1 + c + d; z_2 = h)$

(16) $(r_2 = r_1(\text{seg}(18)); z_2 = h)$; $(r_1 = r_2 - c(1 + \cos 2\phi); z_1 = h - c \sin 2\phi)$; $(r_c = r_2, z_c = h - R_1)$

(14) $(r_2 = r_1(\text{seg}(15)); z_2 = h)$; $(r_1 = r_2 + R_1 \cos(90 - 2\phi); z_1 = h - R_1 - R_1 \sin(90 - 2\phi))$; $(r_c = r(\text{seg}(15)) z_c = h - R$



(20) $(r_2 = r_2((18)), z_2 = h)$; $(r_1 = r_2 + c(1+\cos 2\phi); z_1 = h - c\sin 2\phi)$;
$(r_c = r_2, z_c = h - R_2)$

(21) $(r_2 = r_2((22)); z_2 = h)$; $(r_1 = r_2 - R_1\cos(90-2\phi); z_1 = h - R_1 - R_1\sin(90-2\phi))$;
$(r_c = r_2, z_c = h - R_1)$

(19) $(r_1 = r_1((21)); z_1 = z_1((21)))$; $(r_2 = r_1(20); z_2 = z_1(20))$

(12) $(r_1 = r_2((9)); z_1 = z_2((9)))$; $(r_2 = r_1((19)), z_2 = z_1((19)))$

Table 4 THIS IS Table 26 in [9] Output, nasatruss.OPM, from PANDA2 (2 pages)

nasatruss.OPM (abridged, with "legal" layup, that is with T(7)
changed from its optimized value, 0.013491 inch, to a thickness
exactly equal to three plies, 0.0156 inch.)

```
************************************************************
CHAPTER 28 Present design, loading, and margins for the
           current load set and subcase. See Table 6 in
Bushnell, D.
"Optimization of an axially compressed ring and stringer
stiffened cylindrical shell with a general buckling modal
imperfection", AIAA Paper 2007-2216, 48th AIAA SDM Meeting,
Honolulu, Hawaii, April 2007

ANALYSIS: ITYPE=2;  IQUICK=0;  LOAD SET 1;  SUBCASE 1:
LOADING: Nx, Ny, Nxy, Mx, My =  -5.00E+03 -5.00E-03  2.50E+01  0.00E+00  0.00E+00
         Nxo, Nyo, pressure   =   0.00E+00  0.00E+00 -1.04E-04
BUCKLING LOAD FACTORS FOR LOCAL BUCKLING FROM KOITER v. BOSOR4 THEORY:
 Local buckling load factor from KOITER theory =  1.1956E+00 (flat skin)
 Local buckling load factor from BOSOR4 theory =  1.2192E+00 (flat skin)

MARGINS FOR CURRENT DESIGN: LOAD CASE NO. 1, SUBCASE NO. 1
MAR. MARGIN
NO.  VALUE                DEFINITION
 1  2.20E-01 Local buckling from discrete model-1.,M=35  axial halfwaves;FS=0.99
 2  5.22E+00 local wide-column bucking, discrete model(m=1   axial halfwav);FS=0
 3  1.97E-01 Local buckling from Koiter theory,M=36  axial halfwaves;FS=0.999
 4  7.59E+00 fibertensn:matl=1,STR,Dseg=4,node=1,layer=3,z=-0.0156; MID.;FS=1.27
 5  1.36E+00 fibercompr:matl=1,STR,Dseg=3,node=6,layer=3,z=-0.0347; MID.;FS=1.27
 6  4.53E-02 transtensn:matl=1,SKN,Dseg=1,node=7,layer=10,z=0.0347; MID.;FS=1.27
 7  2.57E+00 inplnshear:matl=1,STR,Dseg=4,node=1,layer=1,z=-0.026; MID.;FS=1.27
 8  7.59E+00 fibertensn:matl=1,SKN,Iseg=2,at:TIP,layer=8,z=0.013;-MID.;FS=1.27
 9  1.36E+00 fibercompr:matl=1,SKN,Iseg=1,at:n=1,layer=10,z=0.0347;-MID.;FS=1.27
10 -1.08E-02 transtensn:matl=1,STR,Iseg=3,at:n=1,layer=3,z=-0.0347;-MID.;FS=1.27
11  2.56E+00 inplnshear:matl=1,SKN,Iseg=2,at:TIP,layer=10,z=0.0234;-MID.;FS=1.27
12  2.26E+05 buckling marg.   skin   Iseg.(width-wise wide col.)MID.;FS=1.
13  8.16E+05 buckling marg. stringer Iseg.(width-wise wide col.)MID.;FS=1.
14  2.26E+05 buckling marg.   skin   Iseg.(width-wise wide col.)NOPO;FS=1.
15  8.16E+05 buckling marg. stringer Iseg.(width-wise wide col.)NOPO;FS=1.
16  2.29E-01 buck.(SAND); STRINGERS:  lower skin;  M=33;N=1;slope=0.03;FS=1.
17  2.78E+00 buck.(SAND);simp-support general buck;M=2;N=4;slope=0.;FS=0.999
18  3.72E-02 buck.(SAND): STRINGERS:  web buckling;M=36;N=1;slope=-0.05;FS=1.
19  5.09E-02 buck.(SAND); STRINGERS:  upper skin;  M=30;N=1;slope=0.;FS=1.
20  3.45E+02 (Max.allowable ave.axial strain)/(ave.axial strain) -1; FS=1.
21 -1.42E-03 0.45   *(Stringer spacing, b)/(Stringer base width, b2)-1;FS=1.
22  1.25E+00 (Str. base width, b2)/(0.2    *(Str. spacing, b))-1; FS=1.
23  4.94E-02 1.-V(2)^1+0.5V(1)^1-1


************************************************************
CHAPTER 28 Present design, loading, and margins for the
           current load set and subcase. See Table 6 in
Bushnell, D.
"Optimization of an axially compressed ring and stringer
stiffened cylindrical shell with a general buckling modal
imperfection", AIAA Paper 2007-2216, 48th AIAA SDM Meeting,
Honolulu, Hawaii, April 2007

ANALYSIS: ITYPE=2;  IQUICK=0;  LOAD SET 2;  SUBCASE 1:
LOADING: Nx, Ny, Nxy, Mx, My =  -5.00E+03 -5.00E-03  2.50E+01  0.00E+00  0.00E+00
         Nxo, Nyo, pressure   =   0.00E+00  0.00E+00 -1.04E-04
BUCKLING LOAD FACTORS FOR LOCAL BUCKLING FROM KOITER v. BOSOR4 THEORY:
 Local buckling load factor from KOITER theory =  1.1718E+00 (flat skin)
 Local buckling load factor from BOSOR4 theory =  1.2103E+00 (flat skin)

MARGINS FOR CURRENT DESIGN: LOAD CASE NO. 2, SUBCASE NO. 1
MAR. MARGIN
NO.  VALUE                DEFINITION
1  2.11E-01 Local buckling from discrete model-1.,M=35  axial halfwaves;FS=0.99
2  5.09E+00 local wide-column bucking, discrete model(m=1   axial halfwav);FS=0
3  1.73E-01 Local buckling from Koiter theory,M=35  axial halfwaves;FS=0.999
4  7.59E+00 fibertensn:matl=1,SKN,Dseg=2,node=1,layer=3,z=-0.0156; MID.;FS=1.27
5  1.36E+00 fibercompr:matl=1,SKN,Dseg=1,node=11,layer=10,z=0.0347; MID.;FS=1.2
6  4.66E-02 transtensn:matl=1,STR,Dseg=6,node=11,layer=3,z=-0.0347; MID.;FS=1.2
7  2.57E+00 inplnshear:matl=1,SKN,Dseg=2,node=1,layer=1,z=-0.026; MID.;FS=1.27
8  7.59E+00 fibertensn:matl=1,SKN,Iseg=2,at:TIP,layer=8,z=0.013;-MID.;FS=1.27
9  1.36E+00 fibercompr:matl=1,STR,Iseg=3,at:n=1,layer=3,z=-0.0347;-MID.;FS=1.27
```

(imperfect shell)

OPTIMIZED "nasatruss"

$w_{imp} = +0.125$ inch ⟶

$w_{imp} = \pm 0.125$ inch

$w_{imp} = -0.125$ inch ⟵ w_{imp}

Table 4 (p. 2 of 2) nasatruss.OPM (p. 2 of 2)

```
10 -1.08E-02 transtensn:matl=1,STR,Iseg=3,at:n=1,layer=3,z=-0.0347;-MID.;FS=1.27
11  2.56E+00 inplnshear:matl=1,SKN,Iseg=2,at:TIP,layer=10,z=0.0234;-MID.;FS=1.27
12  4.32E+00 buckling marg.   skin   Iseg.(width-wise wide col.)MID.;FS=1.
13  8.16E+05 buckling marg. stringer Iseg.(width-wise wide col.)MID.;FS=1.
14  2.26E+05 buckling marg.   skin   Iseg.(width-wise wide col.)NOPO;FS=1.
15  8.16E+05 buckling marg. stringer Iseg.(width-wise wide col.)NOPO;FS=1.
16  2.31E+05 buckling marg. stringer Iseg.(width-wise wide col.)NOPO;FS=1.
17  4.91E-02 buck.(SAND); STRINGERS:  lower skin;  M=30;N=1;slope=0.04;FS=1.
18  2.78E+00 buck.(SAND);simp-support general buck;M=2;N=4;slope=0.;FS=0.999
19  3.75E-02 buck.(SAND); STRINGERS:  web buckling;M=36;N=1;slope=-0.05;FS=1.
20  2.88E-01 buck.(SAND); STRINGERS:  upper skin;  M=36;N=1;slope=0.04;FS=1.
21  3.45E+02 (Max.allowable ave.axial strain)/(ave.axial strain) -1; FS=1.
 ***********  ALL 2 LOAD SETS PROCESSED  ***********
 ******************************************************
```

SUMMARY OF INFORMATION FROM OPTIMIZATION ANALYSIS

VAR. NO.	DEC. VAR.	ESCAPE VAR.	LINK. VAR.	LINKED TO	LINKING CONSTANT	LOWER BOUND	CURRENT VALUE	UPPER BOUND	DEFINITION
1	Y	N	N	0	0.00E+00	1.00E+00	5.1924E+00	9.00E+00	B(STR):pitch of truss core, b: seg=NA, layer=NA
2	Y	N	N	0	0.00E+00	1.00E-01	2.3399E+00	7.00E+00	B2(STR):width over which truss core contacts each fa
3	Y	N	N	0	0.00E+00	1.00E-01	1.4172E+00	2.00E+00	H(STR):height of truss, h: WEB seg=2 , layer=NA
4	Y	Y	N	0	0.00E+00	5.20E-03	5.2000E-03	1.56E-02	T(1)(SKN):thickness for layer index no.(1): SKN seg=1
5	N	N	Y	4	1.00E+00	0.00E+00	5.2000E-03	0.00E+00	T(2)(SKN):thickness for layer index no.(2): SKN seg=1
6	Y	Y	N	0	0.00E+00	5.20E-03	5.2000E-03	1.56E-02	T(3)(SKN):thickness for layer index no.(3): SKN seg=1
7	Y	Y	N	0	0.00E+00	5.20E-03	5.2000E-03	1.56E-02	T(4)(SKN):thickness for layer index no.(4): SKN seg=1
8	N	N	Y	7	1.00E+00	0.00E+00	5.2000E-03	0.00E+00	T(5)(SKN):thickness for layer index no.(5): SKN seg=1
9	Y	Y	N	0	0.00E+00	5.20E-03	1.5600E-02	1.56E-02	T(6)(SKN):thickness for layer index no.(6): SKN seg=1
10	Y	Y	N	0	0.00E+00	5.20E-03	1.5600E-02	1.56E-02	T(7)(SKN):thickness for layer index no.(7): SKN seg=1

CURRENT VALUE OF THE OBJECTIVE FUNCTION:

VAR. NO.	STR/ RNG	SEG. NO.	LAYER NO.	CURRENT VALUE	DEFINITION
		0	0	1.783E+01	WEIGHT OF THE ENTIRE PANEL

```
 TOTAL WEIGHT OF SKIN                                =   1.7827E+01
 TOTAL WEIGHT OF SUBSTIFFENERS                       =   0.0000E+00
 TOTAL WEIGHT OF STRINGERS                           =   0.0000E+00
 TOTAL WEIGHT OF RINGS                               =   0.0000E+00
 SPECIFIC WEIGHT (WEIGHT/AREA) OF STIFFENED PANEL=   1.2315E-03
IN ORDER TO AVOID FALSE CONVERGENCE OF THE DESIGN, BE SURE TO
RUN  PANDAOPT  MANY TIMES DURING AN OPTIMIZATION. INSPECT THE
nasatruss.OPP  FILE AFTER EACH OPTIMIZATION RUN. OR BETTER YET,
RUN SUPEROPT.
**********  END OF nasatruss.OPM FILE **********
```

Optimum design of "nasatruss" from PANDA2 obtained for shell with a general buckling modal imperfection, $w_{imp} = \pm 0.125$ inch amplitude.

NOTE: In "nasatruss.BEG" the material density ~~DEN~~ was specified as 0.0057 instead of 0.057.

Output from PANDAOPT (PANDA2) for the optimized shell called "nasatruss"

This is Table 26 in [9]

This is Table 27 in [9]

Table 5 Output, nasatruss.OPM, from PANDA2

```
nasatruss.OPM (abridged, with "legal" layup, that is with T(7)
changed from its optimized value, 0.013491 inch, to a thickness
exactly equal to three plies, 0.0156 inch and with Wimp = 0.0.)

 ****************************************************************
 CHAPTER 28 Present design, loading, and margins for the
            current load set and subcase. See Table 6 in
 Bushnell, D.
 "Optimization of an axially compressed ring and stringer
 stiffened cylindrical shell with a general buckling modal
 imperfection", AIAA Paper 2007-2216, 48th AIAA SDM Meeting,
 Honolulu, Hawaii, April 2007

 ANALYSIS: ITYPE=2;  IQUICK=0;  LOAD SET 1;  SUBCASE 1:
 LOADING: Nx, Ny, Nxy, Mx, My =  -5.00E+03 -5.00E-03  2.50E+01  0.00E+00  0.00E+00
          Nxo, Nyo, pressure  =   0.00E+00  0.00E+00 -1.04E-04
 BUCKLING LOAD FACTORS FOR LOCAL BUCKLING FROM KOITER v. BOSOR4 THEORY:
  Local buckling load factor from KOITER theory =  1.2869E+00 (flat skin)
  Local buckling load factor from BOSOR4 theory =  1.3096E+00 (flat skin)
0
 MARGINS FOR CURRENT DESIGN: LOAD CASE NO. 1, SUBCASE NO. 1
 MAR. MARGIN
 NO.  VALUE                 DEFINITION
 1  3.11E-01 Local buckling from discrete model-1.,M=38  axial halfwaves;FS=0.99
 2  2.61E+01 local wide-column bucking, discrete model(m=1   axial halfwav);FS=0
 3  2.88E-01 Local buckling from Koiter theory,M=36  axial halfwaves;FS=0.999
 4  8.21E+00 fibertensn:matl=1,SKN,Dseg=2,node=1,layer=3,z=-0.0156; MID.;FS=1.27
 5  1.49E+00 fibercompr:matl=1,STR,Dseg=3,node=6,layer=3,z=-0.0347; MID.;FS=1.27
 6  4.84E-02 transtensn:matl=1,SKN,Dseg=1,node=7,layer=10,z=0.0347; MID.;FS=1.27
 7  2.77E+00 inplnshear:matl=1,SKN,Dseg=2,node=1,layer=1,z=-0.026; MID.;FS=1.27
 8  8.22E+00 fibertensn:matl=1,SKN,Iseg=2,at:TIP,layer=8,z=0.013;-MID.;FS=1.27
 9  1.49E+00 fibercompr:matl=1,STR,Iseg=3,at:n=11,layer=5,z=0.0121;-MID.;FS=1.27
10  4.89E-02 transtensn:matl=1,STR,Iseg=3,at:n=11,layer=3,z=-0.0191;-MID.;FS=1.2
11  2.77E+00 inplnshear:matl=1,SKN,Iseg=2,at:TIP,layer=10,z=0.0234;-MID.;FS=1.27
12  2.26E+05 buckling marg.   skin   Iseg.(width-wise wide col.)MID.;FS=1.
13  8.16E+05 buckling marg. stringer Iseg.(width-wise wide col.)MID.;FS=1.
14  2.26E+05 buckling marg.   skin   Iseg.(width-wise wide col.)NOPO;FS=1.
15  8.16E+05 buckling marg. stringer Iseg.(width-wise wide col.)NOPO;FS=1.
16  1.59E-01 buck.(SAND); STRINGERS:  lower skin;  M=33;N=1;slope=0.03;FS=1.
17  3.76E+00 buck.(SAND);simp-support general buck;M=2;N=4;slope=0.;FS=0.999
18  7.20E-02 buck.(SAND); STRINGERS:  web buckling;M=36;N=1;slope=-0.02;FS=1.
19  1.60E-01 buck.(SAND); STRINGERS:  upper skin;  M=33;N=1;slope=0.02;FS=1.
20  3.45E+02 (Max.allowable ave.axial strain)/(ave.axial strain) -1; FS=1.
21 -1.42E-03 0.45   *(Stringer spacing, b)/(Stringer base width, b2)-1;FS=1.
22  1.25E+00 (Str. base width, b2)/(0.2    *(Str. spacing. b))-1; FS=1.
23  4.94E-02 1.-V(2)^1+0.5V(1)^1-1
 ********** ALL 1 LOAD SETS PROCESSED  **********
 ****************************************************
```

(perfect shell)

Wimp = 0.0 inch

not enough "t.s.d." knockdown!

Same optimum design as in the previous table, but with general buckling modal imperfection amplitude, Wimp = 0.0 inch (perfect shell)

see the next table.

This is Table 27 in [9]

From [9]

nasatruss.OPM

Table 6 PANDA2 output from Chapter 26, obtained (with NPRINT=2)

PERFECT SHELL

```
General buckling from PANDA2:

General buckling load factors from PANDA-type model: (closed form soln)
 EIGMNC=  7.99E+00  7.99E+00  7.99E+00  1.08E+01  1.00E+17  7.99E+00  1.86E+01
 SLOPEX=  0.00E+00  0.00E+00  0.00E+00  0.00E+00  0.00E+00  0.00E+00  0.00E+00
 MWAVEX=     2          2          2          4          0          2          3
 NWAVEX=     4          4          4          6          0          4          0

 Buckling load factor before t.s.d.=  7.9938E+00 After t.s.d.=  5.0738E+00
 Buckling load factor BEFORE knockdown for smeared stringers=  5.0738E+00
 Buckling load factor AFTER  knockdown for smeared stringers=  4.7582E+00
  4.75819E+00   buckling load factor simp-support general buck;M=2;N=4;slope=0.
 Margin=
  3.7629E+00  buck.(SAND);simp-support general buck;M=2;N=4;slope=0.;FS=0.999
```

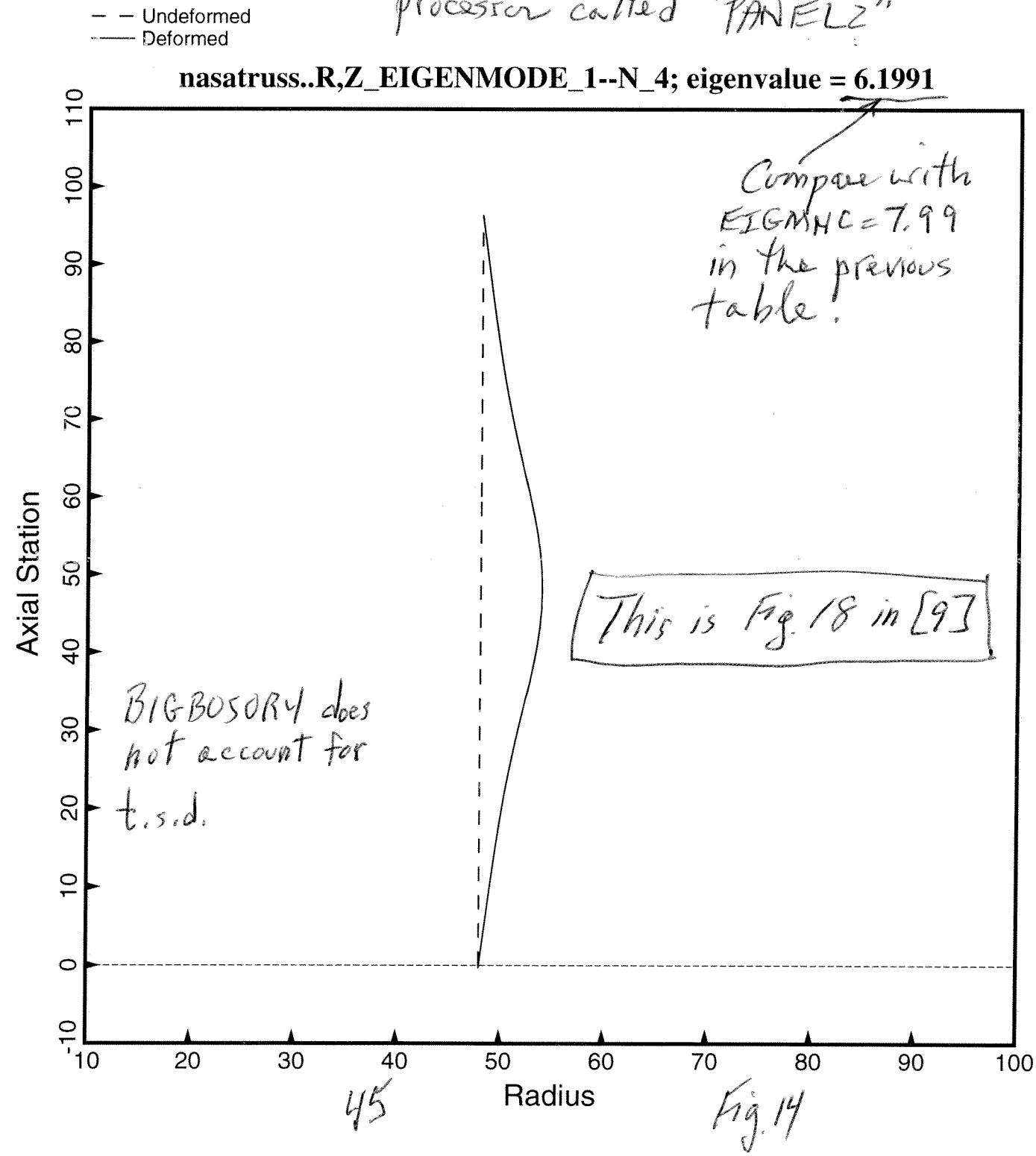
Note: not nearly as much t.s.d. knockdown as is evident from the "huge torus" model.

See Margin No. 17 in the previous table.

Note: the knockdown factor to compensate for the effect of transverse shear deformation (t.s.d.), $\left(\frac{5.0738}{7.9938}\right)$ that is used in PANDA2 is apparently too large, since the "huge torus" model shown in the figure following the next figure (Fig. 15) predicts a much lower general buckling load factor ~~that~~ than 4.7582. ("huge torus" model → 2.00324 in Fig 15)

From [9]

Output from BIGBOSOR4 from input file, nasatruss.ALL, generated by the PANDA2 processor called "PANEL2"



Compare with EIGMHC = 7.99 in the previous table!

This is Fig. 18 in [9]

BIGBOSOR4 does not account for t.s.d.

Fig. 14

From [9]

Output from BIGBOSOR4 from input file, nasatruss.ALL, generated by the PANDA2 Procesor, "PANEL"

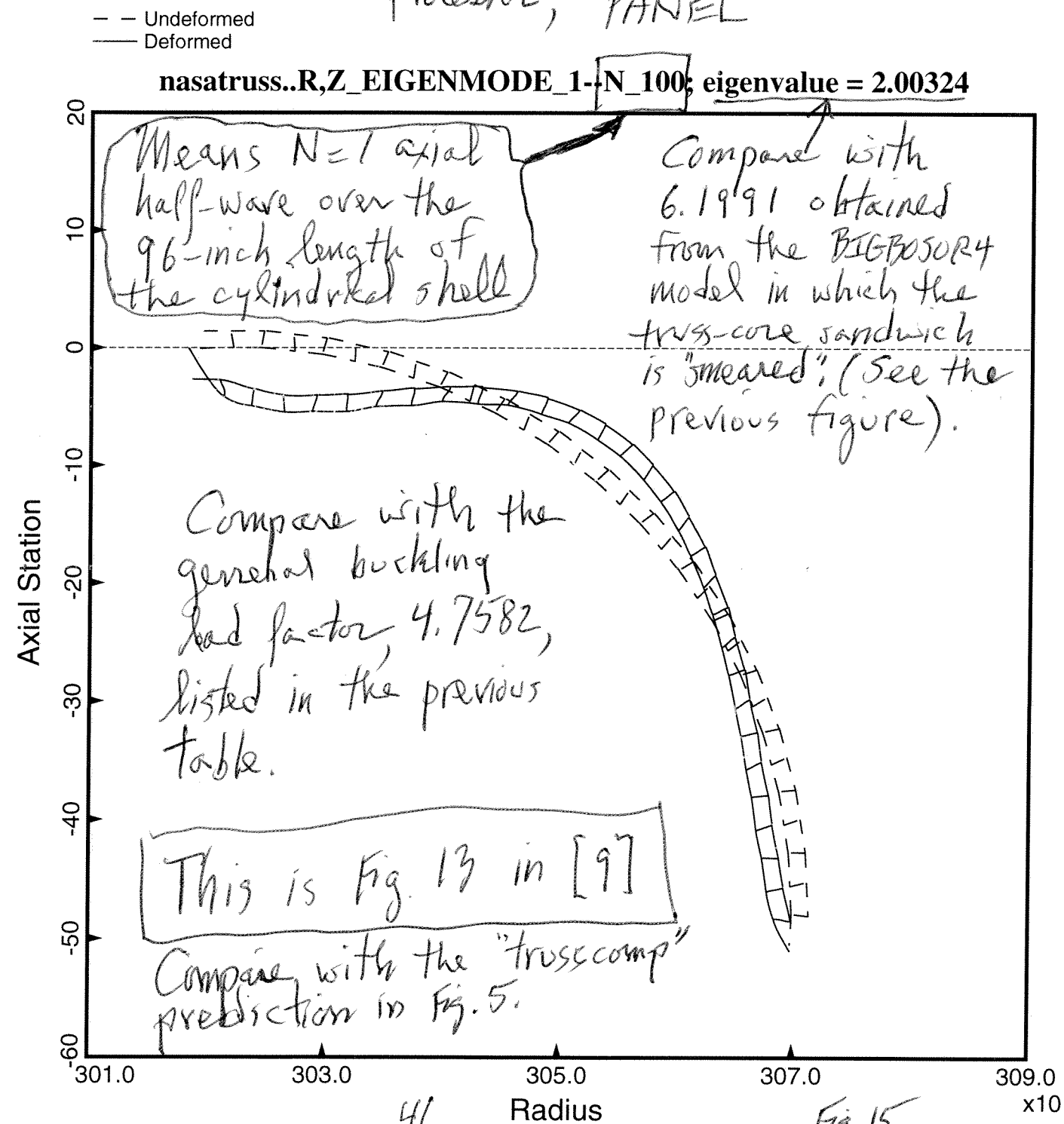


Fig. 15

From [9]

Table 7 Output, nasatruss.OPM, from PANDA2 (4 pages) pertaining to local buckling of the truss-core sandwich shell optimized by PANDA2.

```
**************************************************************
**************************************************************
**************************************************************
************           CHAPTER 13                 ****************
**************************************************************
**************************************************************
**************************************************************
**** CHAPTER 13: DESIGN PERTURBATION INDEX, IMOD=  0 *****

**************************************************************
CHAPTER 13 Get prebuckling stress resultants, Nx, Ny, needed
           for the discretized single-module skin-stringer
           model used for local buckling and bending-torsion
           buckling (BOSOR4-type model: see Figs. 18, 20,
           22, 97, and 98 of [1A], for examples of the
           discretized single skin-stringer BOSOR4-type
           module model.).

Effective circumferential radius of curvature, RADNEW=  4.8000E+01

Truss-core sandwich wall with extra segments (b2):

         < b2 >  ---- Seg. 3 ---->
-------------------------------------------------<-Upper skin
        /-Seg6>\S                 /       \      ^    middle
       /        \e              4/         \     !    surface
      /         . \g.           g/           \    !
     /          !\ \           e/ /           \   h =height from
    /             \ \2        S/ /             \  !  midsurface
   /               \          /!/               \ !  to midsurface
  /  --- Seg. 1----> \<Seg5-/ "                  \V
==========================================================<-Lower skin
                      < b2 >                               midsurface
  <---------- b ------------>

A single module consists of Seg. 1 through Seg. 6.
Seg. 4 has the same wall construction as Seg. 2.
Seg. 5 has wall construction = Seg. 1.
Seg. 6 has wall construction = Seg. 3.

   EXPLODED VIEW, SHOWING LAYERS and (SEGMENT, NODE) NUMBERS

                    Layer No. 1-.
                                .
(Segment, Node)   (6,11)        .              (3,11)
        (6,1)======== (3,1)-------------------------
                                .
                                 ._Layer No. j
                 (2,11)\                          /(4,1)
                        \<------- layer k --------->/
          Layer No. 1 --> \                       /
                           \                     /
                            \                   /
                             \                 /<-- Layer
                              \               /      No. 1
     Layer No. 1 -.            \             /
                  .             \           /
                   .       (2,1)\         /(4,11)
                    .       (1,11)    (5,1)
      (1,1)--------------------   ========
                  .                (5,11)
                   ._ Layer No. m

================================================
BEGIN: PREBUCKLING STRESS RESULTANTS USED IN THE
DISCRETIZED SINGLE MODULE MODEL WITH IQUICK = 0...
--------------------------------------------------------------
PREBUCKLING STRESS RESULTANTS AT THE FIRST NODE
IN THE DISCRETIZED SINGLE MODULE SEGMENT NO. 1 .....
 "Total." loads, Nx(var),Ny(var),Nxy(var)= -2.3739E+03 -2.5572E-03  1.1618E+01
 "Fixed " loads, Nx(fix),Ny(fix),Nxy(fix)= -3.0091E+01  6.1035E-05  1.7980E-05
PREBUCKLING STRESS RESULTANTS AT THE FIRST NODE
IN THE DISCRETIZED SINGLE MODULE SEGMENT NO. 2 .....
```

(Perfect shell: $w_{imp} = 0.0$)

← PANDA2 model of the truss-core sandwich wall for local buckling only.

Table 7 (p. 2 of 4)

nasatruss.OPM (p. 2 of 4)

```
  "Total." loads, Nx(var),Ny(var),Nxy(var)= -4.5447E+02 -5.4965E-04  1.7863E+01
  "Fixed " loads, Nx(fix),Ny(fix),Nxy(fix)=  1.0849E+02 -2.7482E-04  2.6375E-05

 PREBUCKLING AXIAL RESULTANTS IN STRINGER WEB: SEGMENT NO.  2
  "Eigenvalue" axial resultant, Nx(var)=
-4.5447E+02 -4.5447E+02 -4.5447E+02 -4.5447E+02 -4.5447E+02 -4.5447E+02
-4.5447E+02 -4.5447E+02 -4.5447E+02 -4.5447E+02 -4.5447E+02
  "  fixed   " axial resultant, Nx(fix)=
 1.0849E+02  1.0849E+02  1.0849E+02  1.0849E+02  1.0849E+02  1.0849E+02
 1.0849E+02  1.0849E+02  1.0849E+02  1.0849E+02  1.0849E+02

 PREBUCKLING STRESS RESULTANTS AT THE FIRST NODE
 IN THE DISCRETIZED SINGLE MODULE SEGMENT NO. 3
  "Total." loads, Nx(var),Ny(var),Nxy(var)= -2.3739E+03 -2.5044E-03  1.1618E+01
  "Fixed " loads, Nx(fix),Ny(fix),Nxy(fix)= -3.0091E+01  4.6066E-05  3.4756E-05
 PREBUCKLING STRESS RESULTANTS AT THE FIRST NODE
 IN THE DISCRETIZED SINGLE MODULE SEGMENT NO. 4 .....
  "Total." loads, Nx(var),Ny(var),Nxy(var)= -4.5447E+02 -5.4965E-04  1.7863E+01
  "Fixed " loads, Nx(fix),Ny(fix),Nxy(fix)=  1.0849E+02 -2.7482E-04  2.6375E-05

 PREBUCKLING AXIAL RESULTANTS IN STRINGER WEB: SEGMENT NO.  4
  "Eigenvalue" axial resultant, Nx(var)=
-4.5447E+02 -4.5447E+02 -4.5447E+02 -4.5447E+02 -4.5447E+02 -4.5447E+02
-4.5447E+02 -4.5447E+02 -4.5447E+02 -4.5447E+02 -4.5447E+02
  "  fixed   " axial resultant, Nx(fix)=
 1.0849E+02  1.0849E+02  1.0849E+02  1.0849E+02  1.0849E+02  1.0849E+02
 1.0849E+02  1.0849E+02  1.0849E+02  1.0849E+02  1.0849E+02

 PREBUCKLING STRESS RESULTANTS AT THE FIRST NODE
 IN THE DISCRETIZED SINGLE MODULE SEGMENT NO. 5 .....
  "Total." loads, Nx(var),Ny(var),Nxy(var)= -2.3739E+03 -2.5572E-03  1.1618E+01
  "Fixed " loads, Nx(fix),Ny(fix),Nxy(fix)= -3.0091E+01  6.1035E-05  1.7980E-05

 PREBUCKLING STRESS RESULTANTS AT THE FIRST NODE
 IN THE DISCRETIZED SINGLE MODULE SEGMENT NO. 6 .....
  "Total." loads, Nx(var),Ny(var),Nxy(var)= -2.3739E+03 -2.5044E-03  1.1618E+01
  "Fixed " loads, Nx(fix),Ny(fix),Nxy(fix)= -3.0091E+01  4.6066E-05  3.4756E-05
 ------------------------------------------------
 END:  PREBUCKLING STRESS RESULTANTS USED IN THE
 DISCRETIZED SINGLE MODULE MODEL WITH IQUICK = 0...
 ================================================================
 ********** END OF OUTPUT FOR STATE CHANGE ************


 ****************************************************************
 ****************************************************************
 ****************************************************************
 ************           CHAPTER 14                 *****************
 ****************************************************************
 ****************************************************************
 ****************************************************************
 **** CHAPTER 14: DESIGN PERTURBATION INDEX, IMOD=  0 *****

 ************************************************************
 CHAPTER 14 Compute local buckling from BOSOR4-type
            discretized skin-stringer single module model.
            See Section 12.2 (upper table on p. 511) and
            Figs. 46c and 98b in [1A], for examples.

 EXPLANATION OF FOLLOWING CALCULATIONS (LOAD SET NO. 1):
     Eigenvalues corresponding to local buckling of the skin
     obtained from the BOSOR4-discretized panel module model
     are calculated for a range of axial wavenumbers. All four
     edges of the panel are assumed to be simply supported,
     even if you indicated "clamped" in your input, and the in-
     plane loading is assumed to be uniform (N1, N2, N12), even
     if you provided for axial load Nx varying in the L2 (cir-
     cumferential) direction.

**** BEGIN SUBROUTINE LOCAL (INITIAL LOCAL BUCKLING SEARCH) ****

 BUCKLING LOAD FACTORS FROM BOSOR4-TYPE DISCRETIZED MODEL...
          (skin-stringer discretized module of local buckling)
  AXIAL      BUCKLING          KNOCKDOWN FOR     KNOCKDOWN FOR       BUCKLING
  HALF-     LOAD FACTOR      TRANSVERSE SHEAR   IN-PLANE SHEAR     LOAD FACTOR
  WAVES   BEFORE KNOCKDOWN     DEFORMATION      LOADING AND/OR   AFTER KNOCKDOWN
                                                   ANISOTROPY
    M         EIGOLD              KSTAR             KNOCK      EIGOLD*KSTAR*KNOCK
```

Compare with results in the "huge truss" model from "trusscmp". (Table 18)

Table 7 (p. 3 of 4)

nasatruss.OPM (p. 3 of 4)

```
   35      1.37672E+00       1.00000E+00       9.98972E-01      1.37531E+00
   38      1.37425E+00       1.00000E+00       9.98972E-01      1.37283E+00
   41      1.38612E+00       1.00000E+00       9.98972E-01      1.38469E+00
 Buckling load factor before t.s.d.=  1.3728E+00 After t.s.d.=  1.3096E+00
   38      1.37425E+00       9.54355E-01       9.98972E-01      1.30959E+00

 Margin for local buckling of perfect shell:
  3.11E-01 Local buckling from discrete model-1.,M=38  axial halfwaves;FS=0.99

 Number of axial halfwaves between rings, N=   38
 ****  END  SUBROUTINE LOCAL (INITIAL LOCAL BUCKLING SEARCH) ****
  **** END OF LOCAL BUCKLING EIGENVALUE CALC.****

**** BEGIN SUBROUTINE MODE (LOCAL BUCKLING MODE SHAPE) ****

     NORMAL MODAL DISPLACEMENTS IN THE PANEL MODULE SHOWN ABOVE
          SKIN-STRINGER PANEL MODULE HAS 6 SEGMENTS
        NUMBER OF HALF-WAVES IN THE AXIAL DIRECTION, M= 38
 NODE    Z          W          WD         WDD          U          V          WDDD

                    MODAL DISPLACEMENTS FOR SEGMENT NO. 1
  1  0.00E+00 -7.48E-17  7.98E-01  6.28E-01 -6.77E-03  0.00E+00 -2.07E+00
  2  0.00E+00  2.77E-01  9.76E-01  3.58E-02 -8.08E-03  7.57E-04 -2.07E+00
  3  0.00E+00  5.57E-01  9.02E-01 -5.56E-01 -7.65E-03  1.10E-03 -2.07E+00
  4  0.00E+00  7.91E-01  6.82E-01 -9.80E-01 -5.90E-03  1.38E-03 -1.49E+00
  5  0.00E+00  9.46E-01  3.66E-01 -1.24E+00 -3.20E-03  1.57E-03 -9.08E-01
  6  0.00E+00  1.00E+00  0.00E+00 -1.33E+00  0.00E+00  1.65E-03 -3.06E-01
  7  0.00E+00  9.46E-01 -3.66E-01 -1.24E+00  3.20E-03  1.57E-03  3.06E-01
  8  0.00E+00  7.91E-01 -6.82E-01 -9.80E-01  5.90E-03  1.38E-03  9.08E-01
  9  0.00E+00  5.57E-01 -9.02E-01 -5.56E-01  7.65E-03  1.10E-03  1.49E+00
 10  0.00E+00  2.77E-01 -9.76E-01  3.58E-02  8.08E-03  7.57E-04  2.07E+00
 11  0.00E+00 -2.19E-16 -7.98E-01  6.28E-01  6.77E-03  0.00E+00  2.07E+00

                    MODAL DISPLACEMENTS FOR SEGMENT NO. 2
  1  0.00E+00  6.66E-03 -7.98E-01  1.72E+00 -1.20E-03  0.00E+00 -1.85E+00
  2 -1.42E-01 -9.10E-02 -5.73E-01  1.46E+00 -1.15E-03  5.05E-06 -1.85E+00
  3 -2.83E-01 -1.59E-01 -3.83E-01  1.19E+00 -1.11E-03 -4.05E-06 -1.85E+00
  4 -4.25E-01 -2.01E-01 -2.25E-01  1.01E+00 -1.08E-03 -2.46E-05 -1.25E+00
  5 -5.67E-01 -2.23E-01 -8.78E-02  8.91E-01 -1.08E-03 -5.07E-05 -8.21E-01
  6 -7.09E-01 -2.27E-01  3.53E-02  8.18E-01 -1.10E-03 -7.66E-05 -5.09E-01
  7 -8.50E-01 -2.13E-01  1.51E-01  7.82E-01 -1.14E-03 -9.65E-05 -2.51E-01
  8 -9.92E-01 -1.83E-01  2.63E-01  7.82E-01 -1.19E-03 -1.04E-04  2.20E-04
  9 -1.13E+00 -1.37E-01  3.79E-01  8.24E-01 -1.27E-03 -9.42E-05  2.92E-01
 10 -1.28E+00 -7.42E-02  5.04E-01  9.21E-01 -1.35E-03 -5.99E-05  6.74E-01
 11 -1.42E+00  8.03E-03  6.39E-01  1.02E+00 -1.45E-03  0.00E+00  6.74E-01

                    MODAL DISPLACEMENTS FOR SEGMENT NO. 3
  1 -1.42E+00  1.26E-18  6.39E-01  5.30E-01  8.16E-03  0.00E+00 -1.71E+00
  2 -1.42E+00  2.23E-01  7.88E-01  4.22E-02  8.65E-03 -1.01E-03 -1.71E+00
  3 -1.42E+00  4.49E-01  7.30E-01 -4.45E-01  7.74E-03 -1.49E-03 -1.71E+00
  4 -1.42E+00  6.40E-01  5.54E-01 -7.93E-01  5.78E-03 -1.79E-03 -1.22E+00
  5 -1.42E+00  7.65E-01  2.97E-01 -1.01E+00  3.08E-03 -1.97E-03 -7.44E-01
  6 -1.42E+00  8.09E-01  0.00E+00 -1.08E+00  0.00E+00 -2.03E-03 -2.51E-01
  7 -1.42E+00  7.65E-01 -2.97E-01 -1.01E+00 -3.08E-03 -1.97E-03  2.51E-01
  8 -1.42E+00  6.40E-01 -5.54E-01 -7.93E-01 -5.78E-03 -1.79E-03  7.44E-01
  9 -1.42E+00  4.49E-01 -7.30E-01 -4.45E-01 -7.74E-03 -1.49E-03  1.22E+00
 10 -1.42E+00  2.23E-01 -7.88E-01  4.22E-02 -8.65E-03 -1.01E-03  1.71E+00
 11 -1.42E+00 -3.61E-16 -6.39E-01  5.30E-01 -8.16E-03  0.00E+00  1.71E+00

                    MODAL DISPLACEMENTS FOR SEGMENT NO. 4
  1 -1.42E+00  8.03E-03 -6.39E-01  1.47E+00  1.45E-03  0.00E+00 -1.72E+00
  2 -1.28E+00 -6.91E-02 -4.47E-01  1.23E+00  1.21E-03 -3.13E-04 -1.72E+00
  3 -1.13E+00 -1.21E-01 -2.89E-01  9.77E-01  9.38E-04 -5.59E-04 -1.72E+00
  4 -9.92E-01 -1.52E-01 -1.61E-01  8.03E-01  6.50E-04 -7.33E-04 -1.21E+00
  5 -8.50E-01 -1.67E-01 -5.39E-02  6.79E-01  3.48E-04 -8.36E-04 -8.62E-01
  6 -7.09E-01 -1.68E-01  3.74E-02  5.90E-01  4.15E-05 -8.70E-04 -6.20E-01
  7 -5.67E-01 -1.56E-01  1.18E-01  5.26E-01 -2.65E-04 -8.34E-04 -4.41E-01
  8 -4.25E-01 -1.34E-01  1.91E-01  4.85E-01 -5.64E-04 -7.30E-04 -2.89E-01
  9 -2.83E-01 -1.01E-01  2.59E-01  4.65E-01 -8.49E-04 -5.56E-04 -1.36E-01
 10 -1.42E-01 -5.94E-02  3.26E-01  4.72E-01 -1.11E-03 -3.10E-04  4.73E-02
 11  0.00E+00 -7.47E-03  3.91E-01  4.79E-01 -1.35E-03  0.00E+00  4.73E-02

                    MODAL DISPLACEMENTS FOR SEGMENT NO. 5
  1  0.00E+00 -1.55E-18  3.91E-01  7.61E-02 -7.59E-03  0.00E+00 -6.72E-01
  2  0.00E+00  9.80E-02  4.09E-01 -8.12E-02 -7.63E-03  1.86E-05 -6.72E-01
  3  0.00E+00  1.92E-01  3.72E-01 -2.38E-01 -7.37E-03 -7.90E-05 -6.72E-01
  4  0.00E+00  2.72E-01  3.01E-01 -3.71E-01 -6.93E-03 -2.62E-04 -5.68E-01
  5  0.00E+00  3.32E-01  2.01E-01 -4.78E-01 -6.40E-03 -5.39E-04 -4.56E-01
```

critical local buckling load factor from PANDAZ.

axial halfwaves over 96-inch length of the shell.

buckling modal normal displacement

Table 7 (p.4 of 4)

nasatruss.OPM (p.4 of 4)

6	0.00E+00	3.66E-01	8.00E-02	-5.60E-01	-5.89E-03	-9.12E-04	-3.49E-01
7	0.00E+00	3.70E-01	-5.84E-02	-6.23E-01	-5.50E-03	-1.35E-03	-2.69E-01
8	0.00E+00	3.39E-01	-2.11E-01	-6.82E-01	-5.34E-03	-1.78E-03	-2.55E-01
9	0.00E+00	2.71E-01	-3.81E-01	-7.68E-01	-5.51E-03	-1.98E-03	-3.64E-01
10	0.00E+00	1.61E-01	-5.79E-01	-9.28E-01	-6.01E-03	-1.55E-03	-6.87E-01
11	0.00E+00	-4.45E-16	-7.98E-01	-1.09E+00	-6.77E-03	0.00E+00	-6.87E-01

MODAL DISPLACEMENTS FOR SEGMENT NO. 6

1	-1.42E+00	7.70E-18	-3.65E-01	1.33E-02	-2.85E-03	0.00E+00	4.99E-01
2	-1.42E+00	-8.91E-02	-3.66E-01	1.30E-01	-2.68E-03	2.91E-04	4.99E-01
3	-1.42E+00	-1.71E-01	-3.22E-01	2.47E-01	-2.13E-03	5.24E-04	4.99E-01
4	-1.42E+00	-2.40E-01	-2.52E-01	3.48E-01	-1.25E-03	7.08E-04	4.34E-01
5	-1.42E+00	-2.89E-01	-1.61E-01	4.29E-01	-1.10E-04	8.13E-04	3.46E-01
6	-1.42E+00	-3.15E-01	-5.37E-02	4.88E-01	1.23E-03	8.15E-04	2.50E-01
7	-1.42E+00	-3.14E-01	6.51E-02	5.27E-01	2.68E-03	6.99E-04	1.68E-01
8	-1.42E+00	-2.85E-01	1.92E-01	5.58E-01	4.15E-03	4.71E-04	1.31E-01
9	-1.42E+00	-2.24E-01	3.28E-01	6.01E-01	5.59E-03	1.75E-04	1.86E-01
10	-1.42E+00	-1.31E-01	4.79E-01	6.95E-01	6.94E-03	-5.09E-05	4.02E-01
11	-1.42E+00	-4.28E-17	6.39E-01	7.90E-01	8.16E-03	0.00E+00	4.02E-01

**** END SUBROUTINE MODE (LOCAL BUCKLING MODE SHAPE) ******

← buckling modal normal displacement, w

End of output, nasatruss.OPM, from PANDAOPT (PANDA2) pertaining to local buckling from the discretized single-module model of the truss-core sandwich wall.

(This output is obtained with NPRINT=2).

From the optimum design listed in Table 27 of [9]

From [9]

Output from BIGBOSOR4 from input file nasatruss.ALL, generated by the PANDA2 processor, "PANEL", for local buckling.

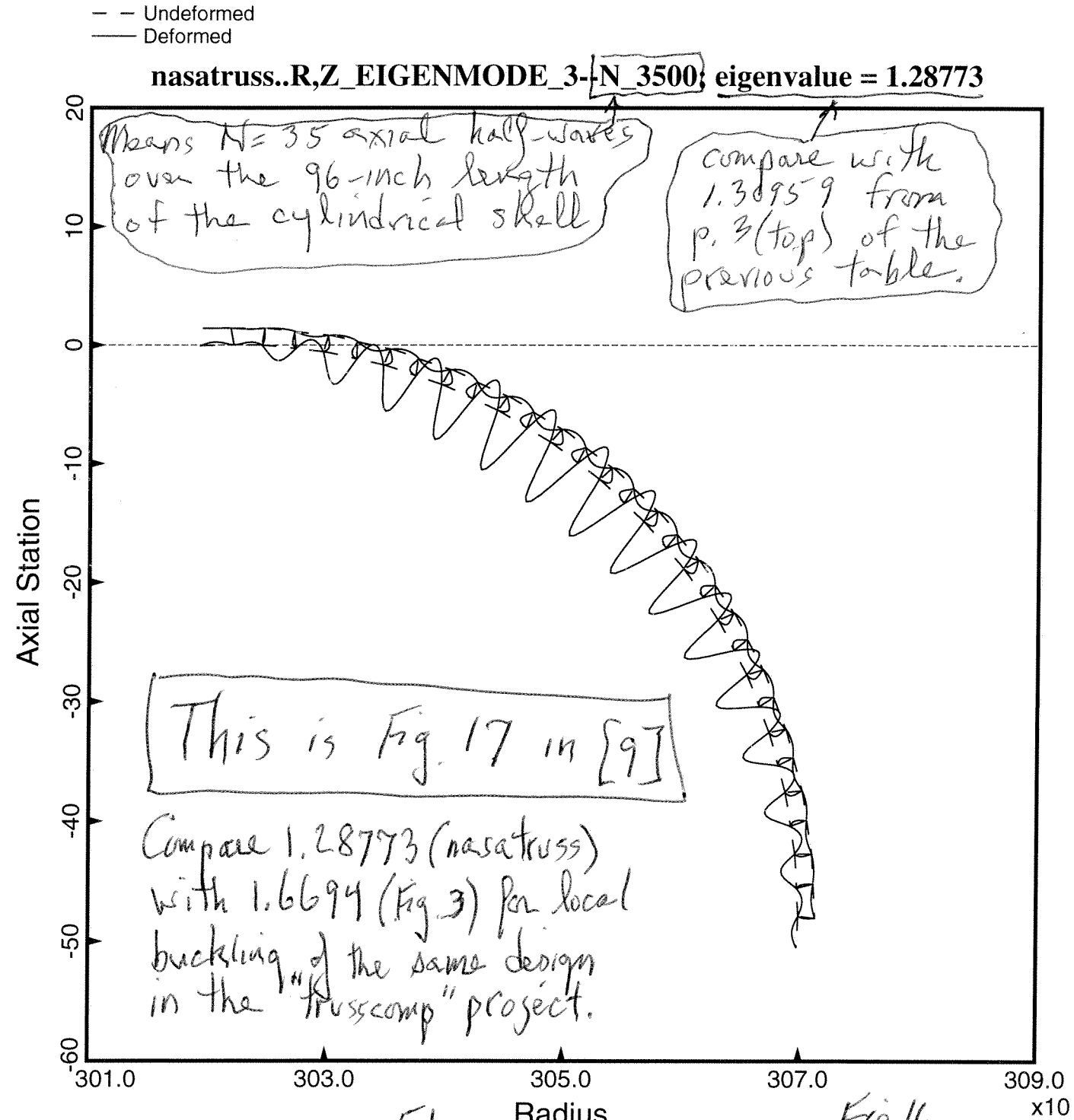


Fig 16

Table 8 RUN STREAM (18 pages)

trusscomp.runstream

```
RUNSTREAM USED TO PRODUCE THE OPTIMUM DESIGN
OF THE TRUSS-CORE SANDWICH CYLINRICAL SHELL
UNDER AXIAL COMPRESSION

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

cd /home/progs/genoptcase

genoptlog       (activate GENOPT command set)

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

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

gentext         [provide generic case name ("trusscomp"),
                 variable names, roles, one-line definitions,
                 "help" paragraphs, etc. The input data from
                 the long GENTEXT interactive session are saved
                 in the file, "trusscomp.INP" (Table 9). Also, see
                 the files produced by GENTEXT called "trusscomp.DEF"
                 (the first part of Table 12, Table 10) and
                  "trusscomp.PRO" (Table 11).]

************ A SMALL DIGRESSION FROM THE RUN STREAM ***************
 After execution of GENTEXT the following "trusscomp" files
 exist in the directory where GENTEXT was executed, that is,
 in the directory, /home/progs/genoptcase :]
-------------------------------------------------------------------
-rw-r--r--  1 bush bush  4529 Jun 12 11:11 trusscomp.CHA
-rw-r--r--  1 bush bush  1378 Jun 12 11:11 trusscomp.COM (common blocks)
-rw-r--r--  1 bush bush  6339 Jun 12 11:11 trusscomp.CON
-rw-r--r--  1 bush bush 58128 Jun 12 11:11 trusscomp.DAT
-rw-r--r--  1 bush bush 32086 Jun 12 11:11 trusscomp.DEF (documentation)
-rw-r--r--  1 bush bush 58161 Jun 12 09:57 trusscomp.INP (input for GENTEXT)
-rw-r--r--  1 bush bush 29517 Jun 12 11:11 trusscomp.NEW
-rw-r--r--  1 bush bush 21560 Jun 12 11:11 trusscomp.PRO (prompting file)
-rw-r--r--  1 bush bush  1882 Jun 12 11:11 trusscomp.REA
-rw-r--r--  1 bush bush 45038 Jun  4 10:26 trusscomp.SAV
-rw-r--r--  1 bush bush   383 Jun 12 11:11 trusscomp.SET
-rw-r--r--  1 bush bush 11357 Jun 12 11:11 trusscomp.SUB
-rw-r--r--  1 bush bush  1882 Jun 12 11:11 trusscomp.WRI
-------------------------------------------------------------

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

READER: Use this table as a guide for "trusscomp" cases or for other cases in which you want to optimize any shell of revolution with use of GENOPT/BIGBOSOR4.

Table 8, p. 2 of 18

Next, create software that computes the design constraints and the objective. In this example create bosdec.trusscomp (Table 13), which generates valid input files for the BIGBOSOR4 preprocessor, B4READ, and "flesh out" the skeletal behavior.new file and the struct.new file that are automatically created by GENOPT. That is, create behavior.trusscomp (Table 12) from behavior.new and struct.trusscomp from struct.new. The "fleshing out" of struct.new is very simple in this "trusscomp" application: only THREE lines are added to the version of struct.new created automatically by GENOPT. These three added lines are indicated below:

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

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

and

The following is part of the struct.new file:

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

The three added statements, CALL OPNGEN, CALL RWDGEN, and CALL CLSGEN, open, rewind, and close various files used by BIGBOSOR4. If you plan to optimize some other shell using GENOPT/BIGBOSOR4 you can "flesh out" struct.new in exactly the same way. To find the places in the "skeletal" version of struct.new that is automatically produced by GENTEXT, search for the string, "YOU MAY WANT" in order to find where you should insert the two lines, CALL OPNGEN and CALL RWDGEN. Search for the string, "NCONSX", in order to find where you should insert the line, CALL CLSGEN. If you want to see what struct.new looks like, see Table 8 of [4] (the case called "weldland"). In this case called "trusscomp" struct.new is different in detail from that listed in [4] (e.g. different variable names and definitions and four BEHXi subroutines here instead of three in [4]), but the overall structure of struct.new is the same. Sometimes, in your other applications of GENOPT, you may want to add more coding to the "skeletal" version of struct.new, as was done in [2] and [3], especially in [3], where all of the computations are done in struct.new and the "skeletal" version of behavior.new produced automatically by GENOPT remains unchanged.

Most of the work in this project was the creation by the GENOPT user (the writer of this report) of the file, bosdec.trusscomp. Some effort was also required to "flesh out" the skeletal "behavior.new" file automatically created by GENOPT. The "fleshed out" version is called "behavior.trusscomp" (Table 12). There are four "behavior" subroutines: BEHX1, BEHX2, BEHX3, BEHX4. The GENOPT user must also "flesh out" the subroutine that computes the objective, SUBROUTINE OBJECT. The "BEHXi", i = 1,2,3,4, subroutines compute the following:

```
SUBROUTINE BEHX1 computes the local   buckling load.
SUBROUTINE BEHX2 computes the general buckling load.
SUBROUTINE BEHX3 computes 5 stress components for
                 material type 1.
SUBROUTINE BEHX4 computes 5 stress components for
                 material type 2.
SUBROUTINE OBJECT computes the objective, which in this case
                  is the weight/area of the shell wall.
```

Table 8, p. 3 of 18

The 5 stress components referred to in connection with SUBROUTINE BEHX3 and SUBROUTINE BEHX4 are those components used for generating stress constraints typical for a composite laminate:

```
 Maximum stress components from BEHX3 and BEHX4:
 0 deg. tension  (tension in the fiber direction of a layer)
 0 deg. comp.    (compression in the fiber direction of a layer)
90 deg. tension  (tension normal to the fibers of a layer)
90 deg. comp.    (compression normal to the fibers of a layer)
 in-plane shear  (shear in the plane of the layer)
```

These 5 stress constraints for each material type are used as stress constraints and margins PANDA2 [9], for example.

```
********************** NOTE ***********************************
If you plan to use GENOPT in combination with BIGBOSOR4 for
optimizing other shells of revolution which have buckling
behavior, you can use the list of SUBROUTINE BEHX1 or of
SUBROUTINE BEHX2 here as a guide. This is what the writer did
in the present "trusscomp" case. The writer simply took the
coding he had added to the "skeletal" versions of SUBROUTINE BEHXi,
i = 1 and 2, listed in Table 10 of [4] (the "weldland" case) and
inserted that coding where it says "INSERT SUBROUTINE STATEMENTS HERE"
in the "skeletal" BEHX1 and BEHX2 subroutines in the "trusscomp" case.
Then he edited that "weldland" coding in order to make it applicable
to the buckling behavior in the "trusscomp" case.
********************* END NOTE ********************************]

NOTE: make it a habit to develop the "behavior", "struct", and
"bosdec" files using different suffices than ".new" or ".src".
By this practice you will not lose work should you execute
GENTEXT again after you have already added FORTRAN coding to behavior.new
and to struct.new, which are over-written by GENTEXT. In this
"trusscomp" case the writer initially copied behavior.new
and struct.new (the "skeletal" versions created automatically by
GENOPT after completion of the GENTEXT interactive session) to
behavior.trusscomp and struct.trusscomp, then "fleshed out"
the behavior.trusscomp and struct.trusscomp files. Also, the
writer developed his version of "bosdec" in a file called
"bosdec.trusscomp". Then, just before execution of "genprograms",
the writer did the following:
-------------------------------------------------------------------------------
cp behavior.trusscomp behavior.new  (behavior.trusscomp and struct.trusscomp are
cp struct.trusscomp struct.new       developed in the directory: /home/progs/genoptcase)
cd /home/progs/bosdec/sources       (NOTE: "bosdec" is stored in a different directory)
cp bosdec.trusscomp bosdec.src
cd /home/progs/genoptcase           (return to the "genoptcase" directory before
                                     executing "genprograms")
-------------------------------------------------------------------------------
************* END OF A SMALL DIGRESSION FROM THE RUN STREAM **********


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


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

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

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

Table 8, p. 4 of 18

NOTE: You may see the lines above even when you still have errors in your newly "fleshed out" and created FORTRAN coding. You will doubtless discover additional errors when you first execute "OPTIMIZE". Make your corrections to the behavior.trusscomp, struct.trusscomp, and bosdec.trusscomp files, then again copy them to behavior.new, struct.new, and bosdec.src as specified above, and then give the command, "genprograms" again. Keep doing this until you are satisfied that there are no more errors in your FORTRAN coding. Go through this "error elimination loop before you try to do any optimization. That is, specify ITYPE = 2 and NPRINT = 2 in the *.OPT file (input for MAINSETUP, listed below) while you are in this error elimination phase of your work. Only when you are satisfied that behavior.trusscomp, struct.truscomp, and bosdec.trusscomp are correct should you attempt to do any optimization (ITYPE = 1 and NPRINT = 0 in the *.OPT file).

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

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

Concerning "behavior" and "struct":

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

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

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

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

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

Concerning "bosdec":

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

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

Table 8, p. 5 of 18

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

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

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

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

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

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

to

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

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

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

```
begin           {provide starting design, material, loading,
                 allowables and factors of safety for the
                 following "behaviors":
                  1. local   buckling (computed in SUBROUTINE BEHX1)
                  2. general buckling (computed in SUBROUTINE BEHX2)
                  3. five stress constraints for material type 1
                     (computed in SUBROUTINE BEHX3)
                  4. five stress constraints for material type 2
                     (computed in SUBROUTINE BEHX4
                 As a starting design, use the optimum design
                 found for the case called "nasatruss" [9]. BEGIN
                 saves your interactive input in the file, test.BEG (Table 14).
                 In this first run all the factors of safety are set
                 equal to unity, which is not a good practice because
                 the cylindrical shell in the GENOPT/BIGBOSOR4 model
                 is assumed to be perfect. The GENOPT/BIGBOSOR4 model
                 does not handle initial imperfections directly.
                 Therefore, factors of safety greater than unity
                 should be used to compensate for this lack of
                 capability. This is done later in this runstream.}
```

See p. 9

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

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

 ENTER THE GENERIC CASE NAME: trusscomp

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: test


**********    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 test.BEG, in which test is a name that you
have chosen for the specific case. (The name, test, 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 test.BEG as input.  Rather than answer all the questions
```

Table 8, p.6 of 18

interactively, you can use test.BEG or an edited version of test.BEG as input to BEGIN. BEGIN also generates an output file called test.OPB. OPB lists a summary of the case, and if you choose the tutorial option, the questions, helps, and your answers for each input datum.

****

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

decide (provide decision variables, bounds, equality constraints (linking expressions), and inequality constraints. DECIDE saves your interactive input in the file, test.DEC (Table 15)).

See p.9

mainsetup {provide strategy, analysis type, etc.. MAINSETUP saves your interactive input in the file, test.OPT (Table 16). In this first execution of MAINSETUP do an analysis of a fixed design (ITYPE = 2) so that you can obtain results for the same optimum design obtained in [9].}

See p.9

optimize (Start the "batch" run that computes results for the fixed design, the optimum design determined in the project reported in [9]. These results are listed in the file, test.OPM (Table 17). "OPTIMIZE", when executed in the ITYPE = 2 mode, produces two files that contain valid input for BIGBOSOR4:
test.BEHX1 = BIGBOSOR4 input file for LOCAL buckling
test.BEHX2 = BIGBOSOR4 input file for GENERAL buckling.
Just below in this runstream test.BEHX1 and test.BEHX2 are used, as you will see next.)

See p.9 & 10

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

```
cp test.BEHX1 /home/progs/bigbosor4/workspace/.     (LOCAL   buckling input)
cp test.BEHX2 /home/progs/bigbosor4/workspace/.     (GENERAL buckling input)
```

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

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

```
bigbosor4log    (activate BIGBOSOR4 commands)
```

(The command, "bigbosor4log", presents the following to your screen:)

The BIGBOSOR4 commands, in the general order in which you would probably use them (except in GENOPT applications), are:

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

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

```
cp test.BEHX1 test.ALL
```

```
bigbosorall     (Start "batch" run for LOCAL buckling. The output
                 file that you want to inspect is called test.OUT.
                 This will be a long file, so search specifically
                 for the string, "EIGENVALUE(", typed with the
                 "(" at the end of the string. You will find the
```

following output there:

Table 8, p. 7 of 18

```
 **** CRITICAL EIGENVALUE AND WAVENUMBER ****
 EIGCRT=  1.6694E+00; NO. OF CIRC. WAVES, NWVCRT=  400
 ***************************************************

 ***** EIGENVALUES AND MODE SHAPES *****
  EIGENVALUE(CIRC. WAVES)
 ========================================
    6.0133E+00( 100)
    2.5222E+00( 200)
    1.8161E+00( 300)
    1.6694E+00( 400)
    1.7336E+00( 500)
    1.9141E+00( 600)
    2.1764E+00( 700)
    2.5054E+00( 800)
    2.8934E+00( 900)
 ========================================
```

(The critical buckling mode has 400 circumferential waves around the entire circumference of the "huge torus" model of the cylindrical shell, and the critical LOCAL buckling load factor computed from the elaborate 22-segment single module model displayed in Figs. 1 and 2 is 1.6694. The LOCAL buckling mode is shown in Fig. 3. The critical number of circumferential waves, 400, corresponds to 4 half-waves along the part of the axial length of the cylindrical shell that is used for local buckling, axial length of cylindrical shell used for local buckling = LENGTH x FACLEN, in which LENGTH is the actual length of the cylindrical shell, 96 inches in this case, and FACLEN is the fraction of the actual length, LENGTH, used in the local buckling model. Please read Appendix 1 to obtain a better understanding of the "huge torus" BIGBOSOR4 model.)

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

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

```
Please enter the BIGBOSOR4 case name: test

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

One, maybe Two moments please...

Text file(s) have been created containing plot data.  The names of the
files explain to a greater or lesser extent what the data represent.
Some plot files contain data for more than one plot.
1)      test..R,Z_EIGENMODE_1--N_100
2)      test..R,Z_EIGENMODE_1--N_200
3)      test..R,Z_EIGENMODE_1--N_300
4)      test..R,Z_EIGENMODE_1--N_400
5)      test..R,Z_EIGENMODE_1--N_500
6)      test..R,Z_EIGENMODE_1--N_600
7)      test..R,Z_EIGENMODE_1--N_700
8)      test..R,Z_EIGENMODE_1--N_800
9)      test..R,Z_EIGENMODE_1--N_900
10)     test..R,Z_RingLocation
CR)     to QUIT
Please choose the number of the file you wish to plot: 4
Plotting: Undeformed & Deformed Axial Station as a function of Radius

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

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

         Example:  mv metafile.ps plot1.ps

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

Table 8, p. 8 of 18

```
      printer available on your system.

         Example:  lpr -PApplelaser metafile.ps

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

                  lpr metafile.ps

      This assumes that your default printer is a PostScript
      printer.

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

Text file(s) have been created containing plot data.  The names of the
files explain to a greater or lesser extent what the data represent.
Some plot files contain data for more than one plot.
1)      test..R,Z_EIGENMODE_1--N_100
2)      test..R,Z_EIGENMODE_1--N_200
3)      test..R,Z_EIGENMODE_1--N_300
4)      test..R,Z_EIGENMODE_1--N_400
5)      test..R,Z_EIGENMODE_1--N_500
6)      test..R,Z_EIGENMODE_1--N_600
7)      test..R,Z_EIGENMODE_1--N_700
8)      test..R,Z_EIGENMODE_1--N_800
9)      test..R,Z_EIGENMODE_1--N_900
10)     test..R,Z_RingLocation
CR)     to QUIT
Please choose the number of the file you wish to plot: <CR>
-----------------------------------------------------------------------

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

gv metafile.ps

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

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

cleanup          (deletes unneeded BIGBOSOR4 files and
                  generates a properly annotated test.ALL
                  file and a properly annotated test.DOC file.
                  (Table 19 is abridged version))


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

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

cp test.BEHX2 test.ALL

bigbosorall      (Start "batch" run for GENERAL buckling. The output
                  file that you want to inspect is called test.OUT.
                  This will be a very, very long file, so search specifically
                  for the string, "EIGENVALUE(", typed with the
                  "(" at the end of the string. You will find the
                  following output there:
-----------------------------------------------------------------
 **** CRITICAL EIGENVALUE AND WAVENUMBER ****
 EIGCRT=  2.3655E+00; NO. OF CIRC. WAVES, NWVCRT=  100
 ***************************************************

 ***** EIGENVALUES AND MODE SHAPES *****
  EIGENVALUE(CIRC. WAVES)
 =======================================
    2.3655E+00( 100)
    2.8234E+00( 200)
    3.5544E+00( 300)
    4.4168E+00( 400)
    5.2736E+00( 500)
```

Table 8, p. 9 of 18

```
====================================
--------------------------------------------------------------------

(The critical buckling mode has 100 circumferential waves around
 the entire circumference of the "huge torus" model of the
 cylindrical shell, and the critical GENERAL buckling load factor
 computed from the multiple-6-segment-module model displayed
 in Figs. 4, 5 and 6 is 2.3655. The GENERAL buckling mode is
 shown in Fig. 5. The critical number of circumferential waves,
 100, corresponds to one half-wave along the complete axial
 length of the cylindrical shell that is used for GENERAL buckling,
 axial length of cylindrical shell used for GENERAL buckling = LENGTH,
 in which LENGTH is the actual length of the cylindrical shell,
 96 inches in this case. Please read Appendix 1 to obtain a better
 understanding of the "huge torus" BIGBOSOR4 model.)


bosorplot       (obtain a plot of the critical GENERAL buckling mode.
                 The postscript file is called "metafile.ps"(Fig. 5).)

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

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

One, maybe Two moments please...

Text file(s) have been created containing plot data.  The names of the
files explain to a greater or lesser extent what the data represent.
Some plot files contain data for more than one plot.
1)      test..R,Z_EIGENMODE_1--N_100
2)      test..R,Z_EIGENMODE_1--N_200
3)      test..R,Z_EIGENMODE_1--N_300
4)      test..R,Z_EIGENMODE_1--N_400
5)      test..R,Z_EIGENMODE_1--N_500
6)      test..R,Z_RingLocation
CR)     to QUIT
Please choose the number of the file you wish to plot: 1
Plotting: Undeformed & Deformed Axial Station as a function of Radius

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

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

cleanup         (deletes unneeded BIGBOSOR4 files and
                 generates  properly annotated test.ALL
                 and test.DOC files (Table 18 is abridged version).)


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

cd /home/progs/genoptcase

mainsetup       (provide input for an optimization. The
                 interactive input for MAINSETUP is saved
                 in the file, test.OPT (Table 20).)

superopt        (obtain a "global" optimum design. Use
                 5 "OPTIMIZEs" per AUTOCHANGE. This computer
                 run takes about 3 days on my LINUX machine
                 if the run is allowed to continue until it
                 is complete. In this case we allow SUPEROPT
                 to run for about 6 hours, then terminate it
                 because we want to obtain some plots for a
                 partial execution of SUPEROPT. In this run
                 all the factors of safety are equal to 1.0.)

(Inspect the test.OPP file)

chooseplot      (choose what to plot vs design iterations. The
                 interactive CHOOSEPLOT session is saved in the
                 file, test.CPL (Table 21).)

diplot          [obtain postscript files, test.3.ps (design
```

See p. 10

See p. 13

```
                  margins vs design iterations), test.4.ps
                  (decision variables vs design iterations),
                  and test.5.ps (objective vs design iterations)
                  for the case with all factors of safety = 1.0.]

xprw test.3.ps   (get a hard copy of the plot of margins v iterations, Fig.17)
xprw test.4.ps   (get a hard copy of the plot of variables v iterations,Fig.18)
xprw test.5.ps   (get a hard copy of the plot of objective v iterations,Fig. 19)

chooseplot       (choose what to plot vs design iterations. The
                  interactive CHOOSEPLOT session is saved in the
                  file, test.CPL (Table 22).)

diplot           [obtain the postscript file, test.4.ps (more decision
                  variables vs design iterations) for the case with
                  all factors of safety = 1.0.]

xprw test.4.ps   (get a hard copy of the plot of variables v iterations,Fig.20)

chooseplot       (choose what to plot vs design iterations. The
                  interactive CHOOSEPLOT session is saved in the
                  file, test.CPL (Table 23).)

diplot           [obtain the postscript file, test.4.ps (more decision
                  variables vs design iterations) for the case with
                  all factors of safety = 1.0.]

xprw test.4.ps   (get a hard copy of the plot of variables v iterations,Fig.21)

chooseplot       (choose what to plot vs design iterations. The
                  interactive CHOOSEPLOT session is saved in the
                  file, test.CPL (Table 24).)

diplot           [obtain the postscript file, test.4.ps (more decision
                  variables vs design iterations) for the case with
                  all factors of safety equal to 1.0.]

xprw test.4.ps   (get a hard copy of the plot of variables v iterations.Fig. 22)


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

mainsetup        (set up a run for the fixed, previously optimized, design,
                  see the top part of Table 25)

optimize         (obtain the test.OPM file (Table 25) corresponding to the
                  optimized design with all factors of safety = 1.0.)

change           (Use the processor, CHANGE, to save the optimum design.
                  The interactive CHANGE session is saved in the file,
                  test.CHG (Table 26).)

(Next, we want to execute BIGBOSOR4 to obtain plots corresponding to
 LOCAL buckling and GENERAL buckling for the case with all factors
 of safety equal to 1.0. First, copy the two files,
 test.BEHX1 and test.BEHX2, into a directory from which you want to
 execute BIGBOSOR4:)

cp test.BEHX1 /home/progs/bigbosor4/workspace/.    (LOCAL   buckling input)
cp test.BEHX2 /home/progs/bigbosor4/workspace/.    (GENERAL buckling input)

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

cd /home/progs/bigbosor4/workspace

bigbosor4log    (activate BIGBOSOR4 commands)

(First, we wish to obtain a plot of the critical LOCAL
 buckling mode from execution of BIGBOSOR4 and BOSORPLOT
 for the case with all factors of safety equal to 1.0:)

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

cp test.BEHX1 test.ALL
```

Table 8, p. 11 of 18

```
bigbosorall      (Start "batch" run for LOCAL buckling. The output
                  file that you want to inspect is called test.OUT.
                  This will be a long file, so search specifically
                  for the string, "EIGENVALUE(", typed with the
                  "(" at the end of the string. You will find the
                  following output there:
-------------------------------------------------------------------
 **** CRITICAL EIGENVALUE AND WAVENUMBER ****
 EIGCRT=  1.5085E+00; NO. OF CIRC. WAVES, NWVCRT=  800
 ***************************************************

 ***** EIGENVALUES AND MODE SHAPES *****
  EIGENVALUE(CIRC. WAVES)
 ======================================
    4.3598E+00( 100)
    3.2431E+00( 200)
    2.7198E+00( 300)
    2.0904E+00( 400)
    1.7450E+00( 500)
    1.5788E+00( 600)
    1.5129E+00( 700)
    1.5085E+00( 800) <--critical local buckling
    1.5457E+00( 900)
 ======================================
-------------------------------------------------------------------

bosorplot        (obtain a plot of the critical local buckling mode,
                  which corresponds to N = 800 circumferential waves
                  around the huge torus. N = 800 corresponds to
                  8 half waves over the axial length of the
                  cylindrical shell equal to FACLEN x LENGTH.
                  The postscript file is called "metafile.ps"(Fig. 23).)

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


(Next, we wish to obtain a plot of the critical GENERAL
 buckling mode from execution of BIGBOSOR4 and BOSORPLOT
 for the case with all factors of safety equal to 1.0:)

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

cp test.BEHX2 test.ALL

bigbosorall      (Start "batch" run for GENERAL buckling. The output
                  file that you want to inspect is called test.OUT.
                  This will be a very, very long file, so search specifically
                  for the string, "EIGENVALUE(", typed with the
                  "(" at the end of the string. You will find the
                  following output there:
-------------------------------------------------------------------------
 **** CRITICAL EIGENVALUE AND WAVENUMBER ****
 EIGCRT=  9.9806E-01; NO. OF CIRC. WAVES, NWVCRT=  500
 ***************************************************

 ***** EIGENVALUES AND MODE SHAPES *****
  EIGENVALUE(CIRC. WAVES)
 ======================================
    1.2862E+00( 100)
    1.1139E+00( 200)
    1.0491E+00( 300)
    1.0120E+00( 400)
    9.9806E-01( 500) <--critical GENERAL buckling mode
 ======================================
-------------------------------------------------------------------------

bosorplot        (obtain a plot of the critical GENERAL buckling mode,
                  which corresponds to N = 500 circumferential waves
                  around the huge torus. N = 500 corresponds to
                  5 half waves over the axial length of the
                  cylindrical shell equal to LENGTH.
                  The postscript file is called "metafile.ps"(Fig. 24).)

cleanup          (deletes unneeded BIGBOSOR4 files and
```

generates properly annotated test.ALL
and test.DOC files.)

Table 8, p. 12 of 18

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

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

(First, clean up the files "test" associated with the specific case:)

```
cleanspec       (clean up SPECIFIC case files, "test".)
```

(Next, change the file, test.BEG. Reset the factors of safety to values greater than 1.0 that reflect the fact that in our GENOPT/BIGBOSOR4 model we cannot account directly for initial imperfections but must compensate for them by using appropriate factors of safety for general buckling (f.s. = 2.0), local buckling (f.s. = 1.5), and stress (f.s. different for different stress constraints). The new factors of safety are included in the part of the new test.BEG file as listed here:

```
   1.000000     $ allowable for local buckling load factor: LOCBUKA( 1)
   1.500000     $ factor of safety for local buckling: LOCBUKF( 1)
   1.000000     $ allowable for general buckling load factor: GENBUKA( 1)
   2.000000     $ general buckling factor of safety: GENBUKF( 1)
         5      $ Number JSTRM1  of columns in the array, STRM1: JSTRM1
    200798.0    $ allowable stress in material 1: STRM1A( 1, 1)
     185925     $ allowable stress in material 1: STRM1A( 1, 2)
    8350.000    $ allowable stress in material 1: STRM1A( 1, 3)
    16400.00    $ allowable stress in material 1: STRM1A( 1, 4)
    17357.00    $ allowable stress in material 1: STRM1A( 1, 5)
   1.500000     $ factor of safety for stress in material 1: STRM1F( 1, 1)
   1.500000     $ factor of safety for stress in material 1: STRM1F( 1, 2)
   1.100000     $ factor of safety for stress in material 1: STRM1F( 1, 3)
   1.500000     $ factor of safety for stress in material 1: STRM1F( 1, 4)
   1.500000     $ factor of safety for stress in material 1: STRM1F( 1, 5)
```

```
begin           (provide starting design, material, loading,
                 allowables and factors of safety. The interactive
                 BEGIN session is saved in the file, test.BEG (Table 14
                 with the changes in factors of safety listed above and
                 in Table 27))
```

see p. 14

```
decide          (provide decision variables, bounds, equality
                 constraints (linking expressions), and inequality
                 constraints. DECIDE saves your interactive input
                 in the file, test.DEC (Table 15)).

mainsetup       (provide input for an optimization. The
                 interactive input for MAINSETUP is saved
                 in the file, test.OPT (Table 20.)

superopt        (obtain a "global" optimum design. Use
                 5 "OPTIMIZEs" per AUTOCHANGE. This computer
                 run takes about 3 days on my LINUX machine.
                 In this case the factors of safety greater
                 than unity compensate for initial imperfections,
                 which cannot be modeled directly in the
                 GENOPT/BIGBOSOR4 "huge torus" model.)
```

see pp. 14-16

(About 3 days later, when the SUPEROPT run is finally finished, inspect the test.OPP file)

```
chooseplot      (choose what to plot vs design iterations. The
                 interactive CHOOSEPLOT session is saved in the
                 file, test.CPL, which is as follows:)
```

```
      n         $ Do you want a tutorial session and tutorial output?
      n         $ Any design variables to be plotted v. iterations (Y or N)?
      n         $ Any design margins to be plotted v. iterations (Y or N)?
      n         $ Do you want to get more plots before your next "SUPEROPT"?
```

Table 8, p. 13 of 18

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

diplot          [obtain the postscript file, test.5.ps (design
                 objective vs design iterations) for the case
                 with all factors of safety greater than 1.0.]

xprw test.5.ps  (obtain a hard copy of the postscript plot, test.5.ps (Fig.25 ).

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

mainsetup       (set up a run for the fixed, previously optimized, design.
                 see the top part of Table 28)

optimize        (obtain the test.OPM file (Table 28) corresponding to the
                 optimized design with all factors of safety greater than 1.0
                 in order to compensate for initial imperfections.)

change          (Use the processor, CHANGE, to save the optimum design.
                 The interactive CHANGE session is saved in the file,
                 test.CHG (Table 29).)

(Next, we want to execute BIGBOSOR4 to obtain plots corresponding to
 LOCAL buckling and GENERAL buckling for the optimized case with all
 factors of safety greater than 1.0. First, copy the two files,
 test.BEHX1 and test.BEHX2, into a directory from which you want to
 execute BIGBOSOR4:)

cp test.BEHX1 /home/progs/bigbosor4/workspace/.    (LOCAL   buckling input)
cp test.BEHX2 /home/progs/bigbosor4/workspace/.    (GENERAL buckling input)

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

cd /home/progs/bigbosor4/workspace

bigbosor4log    (activate BIGBOSOR4 commands)

(First, we wish to obtain a plot of the critical LOCAL
 buckling mode from execution of BIGBOSOR4 and BOSORPLOT
 for the case with all factors of safety greater than 1.0:)

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

cp test.BEHX1 test.ALL

bigbosorall     (Start "batch" run for LOCAL buckling. The output
                 file that you want to inspect is called test.OUT.
                 This will be a long file, so search specifically
                 for the string, "EIGENVALUE(", typed with the
                 "(" at the end of the string. You will find the
                 following output there:
-------------------------------------------------------------------
 **** CRITICAL EIGENVALUE AND WAVENUMBER ****
 EIGCRT=  1.5094E+00; NO. OF CIRC. WAVES, NWVCRT=  700
 ************************************************

 ***** EIGENVALUES AND MODE SHAPES *****
  EIGENVALUE(CIRC. WAVES)
 ======================================
    9.3595E+00( 100)
    4.4118E+00( 200)
    2.6284E+00( 300)
    1.9540E+00( 400)
    1.6638E+00( 500)
    1.5419E+00( 600)
    1.5094E+00( 700)
    1.5322E+00( 800)
    1.5936E+00( 900)
 ======================================
-------------------------------------------------------------------

bosorplot       (obtain a plot of the critical LOCAL buckling mode.
```