NASA Contract Report 181746

Investigation of Structural Behavior of Candidate Space Station Structure

John M. Hedgepeth and R.K. Miller
Astro Aerospace Corporation
Carpinteria, California 93013

Prepared by Astro Aerospace Corporation
for NASA Langley Research Center
under Contract NAS1-17536, Task 8

June 1989

NASA
National Aeronautics and
Space Administration
Langley Research Center
Hampton, Virginia 23665-5225

TABLE OF CONTENTS

TABLE OF CONTENTS, concluded

LIST OF TABLES

LIST OF ILLUSTRATIONS

SECTION 1

INTRODUCTION

Large truss beams composed of 3- to 5-meter cubic cells have been proposed for the Space Station primary structure. Understanding of the behavior of this class of truss beams in configurations varying from a single straight beam to a three-dimensional network is of prime importance in order to guide work on Space Station structural development and to provide interface information to other subsystems and users.

In an effort to provide technical insight into the structural behavior of such truss beams, a research program was initiated to obtain quantitative evaluations of the structural loads, stiffness and deflections due to expected manufacturing, testing, assembly and operational influences. A secondary objective of the investigation was to evaluate the use of personal computers, which have become universally available to analysts, for performing the needed analyses of many-element truss structures. Reported herein are the results of this research program.

The research began with the definition of the geometry and structural properties of a candidate space station. A finite element computer model of this structure was then analyzed for several deterministic loads intended to provide insight into the nature of the structural behavior during some worst case thermal gradients and some deflections and loads during assembly. These analyses were performed by using a finite element analysis program called MSC/pal 2*, which is available for both PC-type and Macintosh personal computers. These results are reported in Sections 2 and 3.

To investigate the influences of manufacturing imperfections, several studies were performed. In section 4, a literature review on random fabrication errors is reported. After examination of the published results on the various

---

*A trademark of the MacNeal-Schwendler Corporation.

sources of random errors and the advantages and disadvantages of various methods of analysis, it was decided to investigate the influences of random member lengths by using Monte Carlo analysis.

Reported in Section 5 are the results of an investigation into the suitability of three popular algorithms for the generation of random numbers. Of particular interest is the ability to generate long sequences of uncorrelated Gaussian deviates, since nonzero autocorrelations could cause cumulative effects in truss structures. The algorithms are compared and ranked based on chi-square goodness-of-fit tests for probability density and extreme value statistics.

The application of MSC/pal 2 to the determination of stresses and deflection for random member lengths is described in Section 6.

The results of analyses of the candidate structure under the influence of random member length imperfections is presented in Section 7. Examples are presented which illustrate the nature and magnitude of the influences of random imperfections on (1) cumulative errors in geometry during assembly, (2) residual stresses in structural members, and (3) misalignment of reference planes at various locations in the structure which may serve as sensor locations for guidance and control.

In order to manage the large amount of computations required for performing the Monte Carlo analyses, programs were written for generating inputs to MSC/pal 2 and storing the outputs in compact form. The compact output files for deflections and stresses can then be examined and analyzed with another computer program. These programs and their use are described in Section 8. The necessary step-by-step procedures are described. Listings of the C source code for the several programs are presented in an appendix to the report.

SECTION 2

GEOMETRY AND MATERIAL PROPERTIES OF A CANDIDATE SPACE STATION TRUSS

Shown in Figures 1 through 4 is the overall geometry of a current candidate for the Space Station primary structure. Only that portion of the structure inboard of the "Alpha" joints is shown and considered. As shown in the figures, the structure is built up from cubic truss cells. For purposes of quantitative analysis, each bay in the truss is assumed to have a dimension of 200 inches (5.08 m). Thus, the central transverse beam, which has 15 bays, has a total length of 3,000 inches (76.2 m). The two longer beams in the closed ring structure each contain a total of 21 bays for a total length of 4,100 inches (106.68 m).

The truss members are assumed to be tubes with an effective EA of 10.97×10^6 lb (48.8×10^6 N). This effective EA may be obtained from many different combinations of tube cross sectional area and modulus of elasticity. For example, the case of a tube with outer diameter of 2 inches (0.0508 meters) and a wall thickness of 0.060 inch (1.52 mm) leads to a cross-section area of $A = 0.36568$ in^2 (2.359×10^{-4} m^2). When used in combination with a material modulus of $E = 30 \times 10^6$ lb/in^2 (206.8×10^9 N/m^2), the cross section just described results in the value of EA reported above.

SECTION 3

STRUCTURAL BEHAVIOR OF CANDIDATE TRUSS MODEL FOR SOME EXAMPLE DETERMINISTIC LOADS

Current plans for the construction of the Space Station primary structure envision on-orbit erection from components. Since the structure as shown in Figures 1 through 4 is statically indeterminant, there may be mismatches during assembly that require large forces to remedy. One illustration of this effect is provided by considering the mismatch deflections which might occur at one corner of the ring structure during final assembly. In particular, inaccuracies in member lengths due, for example, to thermal strains and/or initial member length errors, may accumulate as members are added around the ring so that attempts to close the ring during final assembly may require significant external force to eliminate mismatch deflections in adjoining members. The effective structural stiffness of the complete but open-ring structure is of engineering interest in deciding whether the final assembly operations can be accomplished by the Astronauts alone, or whether some external device and power source may be required. Of similar interest are estimates of the magnitude of worst-case mismatch deflections due to thermal strains. Estimates of these effects are reported in this section.

3.1 OPEN-RING TRUSS MODEL

In order to provide quantitative evaluations of the effects mentioned above, a finite element model of the truss shown in Figures 1 through 4, but including no structural connection at the upper left corner of the ring, was constructed. This model is hereinafter referred to as the open-ring truss model.

The open-ring model was formed from the closed-ring model by creating a duplicate set of nodes at the plane of separation. One set of four nodes is attached to structural members on the left side of the separation plane, and the other set of four nodes to members on the opposite side. Several members lie in the plane of separation. To model these members, a duplicate set of such members was created, and the EA of each set was divided by a factor of 2. One set of those members was attached to the nodes on the left side of the plane of separation, while the other set was attached to nodes on the opposite side. The resulting open-ring structure has identical structural behavior to the closed-ring structure when the duplicate nodes are constrained to move together. The geometry of the open-ring truss model may be easily visualized in Figure 5, which shows the deflections caused by a worst-case thermal strain configuration described later.

3.1.1 Stiffness of the Open-Ring Truss Model

In order to obtain a quantitative estimate of the stiffness of the open-ring truss model as seen by, for example, an Astronaut trying to close the ring during final assembly, a finite element analysis of the open-ring model was performed. The boundary conditions used in the analysis consisted of a set of six statically-determinant displacement constraints applied at three nodes on the plane of separation referred to previously. The deflections resulting from the application of unit loads in the mutually orthogonal coordinate directions were then obtained by computer analysis. The unit loads were applied equally to the four duplicate nodes across the plane of separation from the displacement constrained nodes. The averaged deflections of the four duplicate nodes were recorded.

Based on the analysis just described, a quantitative estimate of the stiffness and flexibility matrices was obtained. In particular, the flexibility matrix was found to be

$$K^{-1} = \begin{bmatrix} 0.088448 \text{ in/lb} & 0.001904 \text{ in/lb} & 0.004513 \text{ in/lb} \\ (0.50505 \text{ mm/N}) & (0.010872 \text{ mm/N}) & (0.025770 \text{ mm/N}) \\[8pt] 0.001904 \text{ in/lb} & 0.019828 \text{ in/lb} & 0.008798 \text{ in/lb} \\ (0.010872 \text{ mm/N}) & (0.11322 \text{ mm/N}) & (0.050238 \text{ mm/N}) \\[8pt] 0.004513 \text{ in/lb} & 0.008798 \text{ in/lb} & 0.016498 \text{ in/lb} \\ (0.025770 \text{ mm/N}) & (0.050238 \text{ mm/N}) & (0.094206 \text{ mm/N}) \end{bmatrix}$$

and the corresponding stiffness matrix is

$$K = \begin{bmatrix} 11.468 \text{ lb/in} & 0.38087 \text{ lb/in} & -3.3402 \text{ lb/in} \\ (2.0084 \text{ N/mm}) & (0.066701 \text{ N/mm}) & (-0.58496 \text{ N/mm}) \\ 0.38087 \text{ lb/in} & 66.079 \text{ lb/in} & -35.343 \text{ lb/in} \\ (0.066701 \text{ N/mm}) & (11.572 \text{ N/mm}) & (-6.1895 \text{ N/mm}) \\ -3.3402 \text{ lb/in} & -35.343 \text{ lb/in} & 80.375 \text{ lb/in} \\ (-0.58496 \text{ N/mm}) & (-6.1895 \text{ N/mm}) & (14.076 \text{ N/mm}) \end{bmatrix}$$

The results indicate that if the total geometric mismatch at the plane of separation does not exceed about one inch (0.0254 m), a force of about 100 pounds (445 N) should be adequate to close the ring during final assembly.

Estimates of the magnitude of the mismatch to be be expected under various conditions are reported later in this report.

3.2 WORST-CASE DEFLECTIONS TO SOME EXAMPLE THERMAL GRADIENTS

3.2.1 Case 1: Uniform Shortening of Outboard Longerons Around the Perimeter of the Open Ring

Temperature variations during assembly and operation of the Space Station will cause changes in member lengths, and consequent thermal deflections and thermal stresses in members. Accurate quantitative analysis of these effects depends on a detailed knowledge of the temperature distribution throughout the truss. The temperature distribution often takes the form of a complex pattern due to shading of portions of members by others, or partial shadowing of the truss by the earth, moon or other satellites.

For purposes of worst-case evaluation of thermal effects on mismatch deflections of the open ring, an analysis of the special case where all outboard longerons undergo a uniform shortening equivalent to a compressive strain of $\epsilon = -1 \times 10^{-5}$ was performed. (While such a pattern of thermal strains is unlikely to occur, it clearly leads to a maximum of mismatch deflections in the open ring.) The longeron strain variation ϵ is the result of a combination of coefficient of thermal expansion and change in temperature. For example, a coefficient of $\alpha = -0.3 \times 10^{-6}/°F$ $(-0.54 \times 10^{-6}/°C)$, which is typical of carefully tailored graphite-epoxy material, together with a change in temperature of ΔT = 33.3°F (18.5°C) results in a total compressive strain of $\epsilon = -1 \times 10^{-5}$.

It should also be noted that the compressive longeron strain could equally well have resulted from manufacturing error rather than thermal strain. A longeron of nominal length of 200 inches (5.08 m) which is fabricated too short by an amount 0.002 inches (0.0508 mm) results in an effective compressive strain of $\epsilon = -1 \times 10^{-5}$.

The deflections resulting from the uniform shortening of outboard longerons are shown (at an exaggerated scale) in Figures 5 and 6. The boundary conditions are the same as those used to determine the stiffness of the open-ring truss model. The average mismatch deflections at the plane of separation were found to be 0.048148 inches (1.223 mm) in the x-direction, 0.29464 inches (7.484 mm) in the y-direction, and 0.22757 inches (5.780 mm) in the z-direction. The vector sum of these deflections has total magnitude of 0.375 inches (9.53 mm).

3.2.2 Case 2: Uniform Shortening of Longerons in One Face Parallel to the Y-Z Plane

The mismatch deflections produced by a uniform shortening of all longerons in a face parallel to the Y-Z plane was also analyzed. The boundary conditions were the same as those used to determine the stiffness of the open-ring truss model. In this case, the deflections are shown (at an exaggerated scale) in Figures 7 and 8. The average mismatch deflections at the plane of separation in this case were found to be 0.0020 inches (0.05 mm) in the x-direction, 0.0004 inches (0.01 mm) in the y-direction, and 0.0021 inches (0.05 mm) in the z-direction. The vector sum of these deflections has total magnitude of 0.003 inches (0.08 mm).

SECTION 4

RANDOM FABRICATION ERRORS IN TRUSS STRUCTURES: A LITERATURE REVIEW

4.1 SCOPE AND OBJECTIVES

A comprehensive review of technical literature relating to the loads and deflections of truss structures caused by random fabrication errors was performed. The Engineering Index and the International Aerospace Abstracts were searched for the years 1980 through986. Several key words were used in the search, including random errors, antennas, reflectors, trusses, structures, and others, both separately and in combination.

The objectives of the search were to identify publications and various contexts in which random fabrication errors have been investigated as they relate to loads and deflections in truss structures, in such fields as antenna structures, space platform structures, and others in aerospace engineering and in ground-based structures in civil engineering. Of major concern was the identification of different methods of analysis of random errors, in order to guide the ongoing investigations of structural behavior of candidate Space Station structures at Astro Aerospace Corporation.

4.2 REVIEW OF INDIVIDUAL TECHNICAL PAPERS

a) "Achievable Flatness in a Large Microwave Power Antenna Study," (General Dynamics, Convair) NASA CR-151831, August 1978.

> Random member length errors in a large flat truss structure are treated by Monte Carlo methods in this publication. Results are presented indicating the effects of random truss member lengths on the achievable flatness of the power antenna support structure.

b) Hedgepeth, J.M., "Effect of Imperfections on Straightness of Space Structures," Astro Research Corporation, Carpinteria, CA, ARC-TN-1068, 27 October 1978.

> Random member length errors in a lattice beam structure of finite length are treated in this publication. Closed form results for mean

square lateral deflections are presented for independent zero-mean normal member length imperfections. Optimal bay-length-to-beam depth ratio necessary to minimize imperfection sensitivity is identified.

c) Hedgepeth, J.M., and Miller, R.R., "Effect of Member Length Imperfections on the Deformations and Loads in an Isogrid-Truss Structure," Astro Research Corporation, Carpinteria, CA, ARC-R-1012, 1 April 1980.

Random member length errors in a flat and a spherically curved equilateral triangular grid structure of infinite dimensions are treated in this publication. In particular, closed form expressions for the mean square loads and deflections caused by independent zero-mean normal member length imperfections are presented with the aid of the Mellin transform. (The results in this publication are the extension to two dimensions of the results presented in the previous publication.)

d) Wang, S.H.; Yao, J.T.P; and Chen, W.F., "Serviceability and Reliability of Antenna Structures - Part 1: Theory," Purdue University, School of Civil Engineering, Report No. CE-STR-81-25, August 1981; and

Wang, S.H.; Yao, J.T.P; and Chen, W.F., "Serviceability and Reliability of Antenna Structures - Part 2: Application," Purdue University, School of Civil Engineering, Report No. CE-STR-82-9, February 1982.

The effects of random member lengths, as well as random wind loads, pointing direction in a gravitational field, material properties, and member cross-sectional dimensions on the surface accuracy of ground-based antenna structures are investigated in these two publications.

In Part 1, a mathematical model for the supporting truss structure is proposed, and definitions are provided for reliability in terms of smallness of RMS departures from the desired surface shape. Three methods of analysis are presented, including (1) first-order, second-moment analytical approach; (2) "advanced" first-order, second-moment analytical approach; and (3) Monte Carlo simulation.

In Part 2, all three methods of analysis outlined in Part 1 are applied to two example antenna structures, and numerical results are presented. It is shown that the most important source of surface error is the random length of truss members. In addition, an optimality approach for determining the best distribution of member-length tolerances throughout the structure is proposed.

e) Hedgepeth, J.M., "Influence of Fabrication Tolerances on the Surface Accuracy of Large Antenna Structures," AIAA Journal, Vol. 20, No. 5, pp. 680-686, 1982.

An analytical method for determining the effect of random member length errors on the RMS surface distortions of a general class of three-dimensional truss structures is presented. Attention is focused on the case where the imperfections are independent, identically distributed random variables with zero mean. In this case, the mean square distortion of the surface is shown to be proportional to the sum of the inverse square of the vibration frequencies of the structure with an appropriately chosen mass distribution. Applications to both flat and dished cellular antenna structures are provided, and conclusions are drawn about the influence of overall dimension, local cell size, shell curvature and depth, and fabrication accuracy on the surface distortion.

f) Wang, S.H., and Ragsdell, K.M., "Optimal Allocation of Antenna Structural Tolerances Based on Reliability," ASME J. Mechanisms, Transmissions, and Automation in Design, Vol. 105, pp. 415-424, September 1983.

The analysis of item (d) in this list of individual technical papers is extended to include the optimal allocation of member length tolerances to meet specified reliability requirements given nominal values of other design variables. A nonlinear programming algorithm is employed, and the results are applied to the two truss examples of item (d). The first-order second-moment analytical approach is used to model the random member length errors.

g) Green, William H., "Effects of Random Member Length Errors on the Accuracy and Internal Loads of Truss Antennas," J. Spacecraft, Vol. 22, No. 5, pp. 554-559, October 1985.

The effects of random member length errors on the surface accuracy, defocus, and residual internal loads of tetrahedral truss antenna reflectors are studied by Monte Carlo simulation methods. Results presented show that the number of members in a tetrahedral truss antenna of a given diameter has a significant effect on surface accuracy, defocus, and internal loads. It is also shown that the member axial stiffness and antenna focal length have a very small effect on reflector surface accuracy.

4.3 SUMMARY AND CONCLUSIONS REGARDING METHODS OF ANALYSIS

There are two primary approaches in use for the analysis of random member length errors in truss structures, analytical methods and Monte Carlo simulation methods. Each approach has its advantages and disadvantages.

Most analytical approaches seek to avoid massive numerical simulation by making use of analytical relations between random input variables and response measures. These relations may involve terms requiring numerical evaluation and, therefore, need not be closed-form in nature.

The most common analytical approach then linearizes the input-output relation about the (deterministic) mean value of the input variables, by means of a truncated Taylor series. At this point, the deviations of the input variables from their mean values are regarded as small random variables, and first and second moment statistics of the response are obtained by manipulation of the linearized Taylor series. (Evaluation of the results of this deterministic method often require numerical estimates of the required derivatives appearing in the series.) The primary advantage of this approach is its computational efficiency. However, it can result in significantly erroneous estimates of response statistics in cases where the linearizations involved are not justified. Further discussion of these limitations is provided in item (d).

Another somewhat novel analytical approach is based on a diagonalization of the strain energy of the truss using the normal modes of a comparison free vibration problem for an appropriately chosen mass distribution. For the case where all random member length errors are independent, the results for the mean square surface deflections may be expressed as the product of a constant and the sum of inverse squares of the natural frequencies of vibration of the truss. Details of the method are presented in item (e). The advantages of the method are that it avoids direct numerical simulation and that it lends insight into the nature of the random errors. The disadvantages arise from the necessity to compute all (or at least a large number) of the natural frequencies of vibration of the truss.

The Monte Carlo method, on the other hand, is very basic, well known and flexible in application. However, it is computationally intensive and often prohibitively expensive. Furthermore, the accuracy of the results depends on the availability of a generator of random numbers which are uncorrelated over long sequences.

SECTION 5

VALIDATION OF PSEUDO RANDOM NUMBER GENERATORS

5.1 INTRODUCTION AND MOTIVATION

Monte Carlo simulation is a well known general method for evaluation of probabilistic effects in engineering. Many algorithms exist for the generation of pseudo random numbers, and these algorithms are frequently employed in Monte Carlo simulation studies.

When Monte Carlo simulation results are used to estimate the statistics of extreme values of system response, the validity of the pseudo random number generator out in the tails of the distribution is of particular importance. For example, if the pseudo random numbers contain a significantly smaller proportion of very large or very small numbers than would be expected based on a truly Gaussian density, then the occurrence of very large or very small numbers in the system response may also be reduced significantly. As a result, it is likely that simulation results may well underestimate the likelihood of extreme values of response. Hence, care must be taken in the selection of appropriate pseudo random generators.

Presented in this section are the results of limited chi-square goodness-of-fit tests applied to three widely used pseudo random number generators. The generators each produce pseudo random variates which are approximately uniformly distributed on the interval from zero to one. Before testing, the pseudo random variates are all processed through the same subprogram which uses both power series and asymptotic expansion representations for the inverse error function to map the pseudo uniformly distributed variates into pseudo Gaussian distributed variates. Tests are performed on sequences of pseudo random variates for accuracy of the density of the variates themselves, as well as the density of the peak value of each sequence. The effects of starting value (seed number) and length of sequence are investigated to a limited extent, and a selection of one generator for use in Monte Carlo simulation studies is made based on quantitative results.

It is noteworthy that previous investigations of the effects of random member imperfections on the response and behavior of large structures have noted a sensitivity to small correlations in long sequences of random imperfection quantities. Such correlations can cause substantial amplification of cumulative deflections and rotations in the overall structure, particularly for beam-like or one-dimensional structures. Concern over correlation-induced effects for the particular class of random number generators considered herein is diminished, however, by the availability of theorems[1]* which provide the least period of the sequence of variates. Thus, it is possible to choose a generator and starting value appropriately such that the sequence generated is guaranteed not to contain any repeated values. (Note that if a value ever recurs, then all succeeding values must also recur.)

For a comprehensive yet readable treatment of the theory of algorithms for the generation of pseudo random numbers, including periodicity effects, the reader is encouraged to explore the book by Knuth[1].

5.2 THREE WIDELY USED PSEUDO RANDOM NUMBER GENERATORS

Of the many algorithms available for the generation of pseudo random numbers, three were selected for detailed evaluation. These three algorithms were selected on the basis of general popularity in applications and convenience of access. For purposes of this report, the algorithms are hereinafter referred to as (1) Microsoft C Library Routine; (2) International Mathematical and Statistical Library (IMSL) Routine; and (3) Peerless Routine.

5.2.1 Microsoft C Library Routine

The Microsoft C 5.1 Optimizing Compiler contains a function "rand()" in the Run-Time Library. This function returns a pseudo random integer in the range 0 to 32,767. A companion function "srand()" may be used to set the starting value or seed used by rand(). No listing of the algorithm employed in this function is provided. However, the algorithm apparently belongs to the class of linear congruential algorithms discussed in the book by Knuth [1], and

---

*Numbers in brackets refer to items in the reference list.

which also contains the other two algorithms discussed below. Instructions for the use of the rand() and srand() functions are contained in the Microsoft C 5.1 Run-Time Library Reference Manual[2].

5.2.2 IMSL Routine

The algorithm GGUBS() supplied in the IMSL subroutine package for generating pseudo random numbers may be summarized as follows[3]:

$$u_{i+1} = 7^5 u_i \text{ modulo } (2^{31} - 1); \; i = 0,1,2,... \qquad (1)$$

This algorithm may be used to generate a sequence of pseudo random integers in the range 0 to $(2^{31} - 1)$. The algorithm is of the multiplicative congruential type.

5.2.3 Peerless Algorithm

Another algorithm for the generation of pseudo random numbers is that provided in the Peerless packages of subroutines in Fortran and C. It may be summarized as follows[4]:

$$u_{i+1} = 317 u_i \text{ modulo}(1); \; i = 0,1,2,... \qquad (2)$$

The numbers generated by equation (2) lie in the range 0 to 1.0.

5.3 CONVERSION OF PSEUDO RANDOM INTEGERS TO PSEUDO GAUSSIAN VARIATES

After a sequence of pseudo random integers has been generated by one of the algorithms just described, it may be converted into a sequence of pseudo Gaussian variates by a two-step process. First, the sequence of integers is converted to sequence of pseudo uniform variates by dividing each integer by an integer one larger than the largest integer which may be generated by the original algorithm. The result is a sequence of floating point numbers approximately uniformly distributed on the interval from zero to one. Note that the Peerless routine yields such results directly.

Next these pseudo uniform variates are converted to pseudo Gaussian variates by mapping in accordance with the following general rule:

Let u be a random variable uniformly distributed in the interval (0,1). Let x be a random variable with probability density f(x) in the interval (a,b). then the mapping between the two obeys the equality,

$$u = \int_a^x f(\xi) d\xi \tag{3}$$

For a Gaussian distribution, then

$$x_i = \mu + \sqrt{2}\sigma \, \mathrm{erf}^{-1} (2u_i - 1); \; i = 0,1,2,... \tag{4}$$

where u_i is the pseudo uniform variate, x_i is the corresponding pseudo Gaussian variate, μ is the mean value and σ is the standard deviation, and

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-z^2} dz \tag{5}$$

is the "error function." Note that the mapping in equation (4) contains the inverse error function. It was found that a combination of power series and asymptotic expansions for the inverse error function is required to obtain an adequate accuracy in the tails of the distribution.

5.3.1 Truncated Gaussian Variates

It is not likely that members with very large imperfections will be used in assembling the Space Station, because of quality control sampling and rejection procedures on the ground. Thus, the actual lengths of members may have a distribution which has the same shape as the Gaussian distribution near the mean, but with the tails cut off beyond a threshold value $\mu \pm x_0$. The resulting "truncated" Gaussian distribution may be described by three parameters: μ, σ, and x_0.

With reference to the foregoing general mapping,

$$f(x) = \frac{1}{A} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{6}$$

$$\begin{aligned} a &= \mu - x_0 \\ b &= \mu + x_0 \end{aligned} \tag{7}$$

and

$$A = \int_{-x_0}^{x_0} e^{-\frac{\xi^2}{2\sigma^2}} d\xi \tag{8}$$

Note that as $(x_0/\sigma)\to\infty$, the resulting distribution tends toward the Gaussian distribution, but as $(x_0/\sigma)\to 0$, it tends toward a uniform distribution on the interval $\mu - x_0$ to $\mu + x_0$. Thus, a wide range of distribution shapes may be obtained from this truncated Gaussian distribution by appropriate selection of the parameters μ, σ, and x_0.

The mapping for the truncated Gaussian variate is:

$$x = \mu + \sqrt{2}\sigma \operatorname{erf}^{-1} \left[\operatorname{erf} \frac{x_0}{\sqrt{2}\sigma} (2u - 1)\right] \tag{9}$$

Of interest also is the resultant actual standard deviation for the truncated distribution. It can be found to be:

$$\sigma_{act} = \sigma \quad 1 - \frac{2}{\sqrt{\pi}} G \frac{x_0}{\sqrt{2}\sigma} \tag{10}$$

where

$$G(z) = \frac{ze^{-z^2}}{\operatorname{erf}(z)} \tag{11}$$

5.4 CHI-SQUARE GOODNESS-OF-FIT TESTS ON PSEUDO GAUSSIAN SEQUENCES

The three random number generators previously described were used to generate sequences of pseudo Gaussian random variates. These sequences were used to perform chi-square goodness-of-fit tests on the distribution of the variates, and on the distribution of the peak or extreme values obtained from sequences of a fixed length. Limited investigations of the effects of different starting values (or seed number) and length of sequence were also performed. Based on the quantitative results so obtained, the Microsoft C Library Routine was selected for use in performing Monte Carlo simulation studies reported later.

5.4.1 Summary of Results for Distribution of Variates

A series of chi-square goodness-of-fit tests were performed on the pseudo random numbers generated by the algorithms described previously. The first series of tests was aimed at ranking the three candidate algorithms based on the "Gaussianness" of the distribution of the variates which they produce. For this purpose, the chi-square statistic was calculated for tests in which the real number line (which constitutes the sample space for the generated variates) was broken into the following 14 intervals, each of which has essentially equal probability when the variates are truly standard normal in distribution:

interval 1 : [-∞,-1.466]
interval 2 : [-1.466,-1.067]
interval 3 : [-1.067,-0.792]
interval 4 : [-0.792,-0.566]
interval 5 : [-0.566,-0.366]
interval 6 : [-0.366,-0.180]
interval 7 : [-0.180,0.0]
interval 8 : [0.0,0.180]
interval 9 : [0.180,0.366]
interval 10: [0.366,0.566]
interval 11: [0.566,0.792]
interval 12: [0.792,1.067]
interval 13: [1.067,1.466]
interval 14: [1.466,∞]

Starting with a given initial value or seed, the candidate algorithm was used to generate a sequence of length n, where n = 100, or n = 1,000, or n = 5,000. Note that the expected number of occurrences of variates in each interval is then 7.14, or 71.4, or 357, respectively. Thus, even in the case when n = 100, the expected number of occurrences in every interval exceeds the often quoted minimum value of 5 (see page 42 of Knuth[1]).

The effect of varying the initial value of seed number used to start the algorithm was explored to a limited extent. Sequences of length n = 100 were generated for six different seed values with each of the algorithms.

The results of this series of tests are presented in Table 1, which has the form of a matrix whose entries are values of the chi-square statistic (except for the left column, which lists the sample size.) With the exception of one result (Peerless algorithm, n = 100, seed = 320), all the values of the chi-square statistic lie between the 5 and 95 percentile values for 13 degrees of freedom. Hence, all three algorithms produced results of acceptable quality in this sense. However, the Microsoft C routine produced significantly lower values of chi-square than the two other algorithms for sample sizes of n = 100 and n = 1,000. Hence the Microsoft C routines performed best at producing short to moderately long sequences of appropriately random numbers. It is precisely this ability to produce sequences with good "locally random" properties which is of greatest importance in the intended application. Of course, the testing reported here is not extensive enough to be conclusive, but the results are assumed to be representative of the general quality of random numbers produced by the algorithms.

5.4.2 Summary of Results for Distribution of Peaks in Sequences of Variates

In addition to the tests on the distribution of the variates, Chi-square goodness-of-fit tests on the peak or extreme values observed in sequences of length n = 100 were also performed. Since extreme value statistics are of interest in the application, the ability of the algorithm to produce peak values with the appropriate distribution is also of interest.

Consider a set $\{x_i | i=1,2,3,...,n\}$ of independent random variables with the same cumulative distribution function $F_X(x)$. Then, if Z is defined as the maximum value of the set of X_i, the cumulative distribution $F_Z(z)$ of Z is given by

$$F_Z(z) = [F_X(z)]^n \qquad (5)$$

Thus, for a set of n standard normal variates, the cumulative distribution of the peak value of the set is the nth power of the standard normal cumulative distribution function.

Using the results provided by Equation (5), the chi-square statistic was calculated for tests in which the positive real number line was broken into the following 10 intervals, each of which has essentially equal probability when the variates are truly standard normal in distribution, and when n = 100:

interval 1: [0,1.998]

interval 2: [1.998,2.145]
interval 3: [2.145,2.257]
interval 4: [2.257,2.360]
interval 5: [2.360,2.460]
interval 6: [2.460,2.570]
interval 7: [2.570,2.690]
interval 8: [2.690,2.850]
interval 9: [2.850,3.060]
interval 10: [3.060,∞]

Starting with a given initial value or seed, the candidate algorithm was used to generate an ensemble of 50 different sequences consisting of 100 variates each. Each sequence was then scanned and the maximum and minimum values were extracted. A single sequence of 100 extreme values was obtained by taking the absolute value of the minimum values and treating the resulting positive values as additional maxima. The result is a sequence of 100 positive maxima obtained from the ensemble of 50 sequences of variates. The sequence of 100 maxima was then used together with the intervals above to generate a chi-square statistic. Note that the expected number of outcomes in each interval is 10 in this case. The results are presented in Table 2, which lists the chi-square values corresponding to different seeds or starting values, for each of the three algorithms tested. With the exception of one result (Peerless algorithm, seed = 325), all values of the Chi-square statistic lie between the 5 and 95 percentile values for nine degrees of freedom. Hence, again it is concluded that all three algorithms produced results of acceptable quality. Again it must be emphasized that the tests performed were limited in scope, and should not be regarded as conclusive. It is possible that more extensive tests carried out with much larger samples of variates could arrive at different conclusions.

However, based on the available test results in Tables 1 and 2, the Microsoft C routines rand()and srand() were chosen for the purposes of the Monte Carlo simulation studies reported later. The selection was based equally on the implications of the test results and on the convenience of the application in the C code written for the simulation studies. As previously noted, the Peerless algorithm was alone in producing a test result whose randomness may be regarded as suspect in each test. Eliminating the Peerless algorithm, the Microsoft C routines appear to perform consistently better than the IMSL routine for the distribution of the variates (Table 1), and only slightly worse in tests of the distribution of the peaks (Table 2).

SECTION 6

APPLICATION OF THE MSC/pal 2 FINITE ELEMENT ANALYSIS OF TRUSS RESPONSE TO RANDOM LOADS

6.1 INTRODUCTION

Numerous commercial finite element programs exist which are capable of efficient load and deflection analysis of three-dimensional truss structures. No systematic comparison of software was performed, however, after discussions with technical representatives at NASA Langley Research Center who expressed a strong preference for the MSC/pal 2 software for reasons of compatibility and availability. It is noteworthy that for purposes of Monte Carlo simulation studies of truss response to random member imperfections, the MSC/pal 2 software should not necessarily be considered an optimal selection.

6.2 OUTLINE OF THE MSC/pal 2 STATIC ANALYSIS CAPABILITIES

For a complete description of the MSC/pal 2 software capabilities and usage, the reader is directed to Reference 5, the several manuals provided by the vendor, MacNeal-Schwendler Corporation. What is presented here is a brief summary and paraphrase of information excerpted from the manuals or obtained from the authors personal experience with MSC/pal 2.

The MSC/pal 2 software is a collection of programs for stress and vibration analysis of mechanical systems, components, and structures on a personal microcomputer. It uses the finite element displacement method to solve for deflections, forces and stresses of two- and three-dimensional systems. Static, normal modes, transient response, and frequency response analysis capabilities are provided. In addition to tabular output, graphical output is available for model geometry and structural deformations, displacements as functions of time and frequency, animated deformation plots, and stress contour plots.

The software consists of several programs, including the PAL2, STAT2, DYNA2, VIEW2, XYPLOT2, ADCAP2, and PALPREP2 programs, most of which generate data for use by subsequent programs. The organization of programs is shown in Figure 9. Each program is described below.

PAL2 - Generates the mathematical model (geometry, stiffness and mass distribution, etc.) of the unrestrained structural system.

STAT2 - Generates boundary restraints and loads, then performs static analysis, including nodal displacement and element force and stress computations.

DYNA2 - Performs dynamic analysis: natural frequencies and mode shapes; transient response to time varying loads, displacements, and accelerations; and response to frequency-dependent forces, displacements, and accelerations.

VIEW2 - Creates undeformed and deformed structural plots, animation, and contour plots from static and dynamic analysis results.

XYPLOT2 - Creates X-Y screen plots of dynamic response as well as static stress distribution displays.

ADCAP2 - Performs conversion to MSC/NASTRAN format, model data set expansion, and printing system equations.

PALPREP2 - Functions as an interactive preprocessor for simple models.

Although versions of MSC/pal 2 for other machines are available, the one used in the present study was developed specifically for an IBM Personal Computer (PC) or compatible, with at least a 10 megabyte hard disk and at least 640K RAM (random access memory). An IBM PC XT, AT, or 80386 machine is adequate. At least one floppy disk drive is required in order to transfer the programs from the diskettes on which they are supplied. A color monitor and graphics card are required for graphical output. A numeric coprocessor chip (8087 for the XT, 80287 for the AT, or 80387 for 80386 machines) is also required. A printer (for hard copy output) is optional; graphics hard copy is supported on the IBM printer.

6.3 BASIC APPROACH FOR RANDOM IMPERFECTION ANALYSIS

The steps followed in analyzing a single case in the Monte Carlo study are outlined in the following paragraphs. The analysis begins with the creation of a mathematical model of the geometry and structural properties of the truss. This is accomplished by executing the PAL2 subprogram with the input text file such as SPASTAT2.TXT, a listing of which is included in Appendix 1. After running PAL2, a model of the open-ring truss structure is available for static analysis. This model contains 897 elements and 276 nodes, with a total of 828 degrees of freedom.

Next, a random number generator is used to create a set of 897 pseudo random member strains. By setting the coefficient of thermal expansion to one for all members in the truss and setting all member reference initial temperatures to zero, (see SPASTAT2.TXT), these random member strains may be regarded as random member temperature changes using the thermal analysis capabilities of MSC/pal 2, version 3 (November 1987). An ASCII text file containing the necessary displacement boundary conditions (using the DISPLACEMENTS APPLIED 1 command) followed by a blank line and the 897 random member temperature changes (using the ELEMENT TEMPERATURE 1 command) is created. The random element temperature data is terminated by a blank line, and the SOLVE then QUIT commands are invoked. An example is shown in the sample listing RANLOAD.TXT file, also listed in Appendix 1. (The preparation of this load file has been automated with special-purpose software developed for this purpose and described later in this report.)

After the load input text file has been created, the next step is to run the static deflection analysis program STAT2. This is done by using the load file containing the random member temperatures as input to the STAT2 program. STAT2 then achieves a decomposition of the stiffness matrix and calculates the displacements in the truss, storing the results compactly in binary form on the hard disk.

To examine the results of the analysis, it is necessary to exercise the DATA RECOVERY option within STAT2. This option allows the creation of an ASCII disk file containing nodal deflections and/or member stresses which, if requested, are computed from the nodal deflections previously stored. (Graphical examination of the deformations in the truss are possible without running the DATA RECOVERY option, by exiting STAT2 and running VIEW2 instead.) After the STAT2 output results are written to a storage file in ASCII form, they may be scanned to search for peak, rms, or other measures of nodal deflections and member stresses. This may be done manually using an available text editor, or it may be accomplished using the special purpose post-processing software developed for this purpose and described later in this report.

The approach described above was used to obtain the results presented in the next section. However, it is possible to perform the analysis without use of the thermal analysis capabilities of MSC/pal 2, version 3. This may be accomplished by first multiplying the set of random member strains by the set of member properties EA to obtain the set of member fixed-end forces. These forces are those which would develop in the members if the nodes of the truss were somehow locked against all motion by appropriate external constraints, and then the member strains were imposed. Such fixed-end member forces would then result in a set of necessary external nodal constraint forces. These nodal constraint forces may be obtained from the fixed-end member forces together with the geometry of the truss. The displacements of the nodes of the truss with random member strains may then be obtained by solving an auxiliary problem in which nodal forces are applied to an otherwise unstrained and unloaded truss. The nodal forces to be applied are the exact reverse of the nodal constraint forces previously described. The displacements obtained in this way are identical to the displacements obtained by using the thermal analysis capability of MSC/pal 2, version 3.

The member forces obtained from the auxiliary problem must be subtracted from the fixed-end member forces to obtain the correct member forces which result when random member strains are imposed on the structure. This additional post-processing step (which requires additional storage for the fixed-end member forces) is a disadvantage of the direct (non-thermal analysis) approach outlined here. However, this disadvantage is mitigated somewhat in MSC/pal 2, version 3, because the direct approach allows multiple load cases to be submitted, thereby eliminating the stiffness matrix decomposition phase of solution for multiple load cases. The thermal analysis capabilities of MSC/pal 2 do not include multiple load case capabilities as of this writing.

SECTION 7

STRUCTURAL BEHAVIOR OF CANDIDATE TRUSS MODEL SUBJECTED TO SOME RANDOM LOADS

One of the important considerations in the design of the Space Station primary structure is the effect of fabrication imperfections and tolerances on assembly loads, geometric precision, and residual stresses in the structure. In order to provide some quantitative evaluation of such effects, a limited Monte Carlo simulation study of the open-ring truss model was performed.

The Monte Carlo simulation study was performed using the Microsoft C pseudo random number generation routines described in Section 5, and the MSC-PAL2 finite element analysis software described in Section 6. The analysis was carried out on various IBM PC XT or AT machines (or compatibles), as well as Compaq 386 machines. To investigate the nature of the structural response to random member-length errors, an ensemble of 100 sets of pseudo random member lengths were generated. The nodal deflections and member stresses which resulted from each set of member length imperfections were determined by finite element analysis. The ensemble of results so obtained was then used to examine the statistical behavior of the structural response. In particular, the maximum and rms values of the ring-closing mismatch errors in the open-ring truss model were determined, as well as the peak and rms member residual streses, and the relative misalignment between two reference planes attached to the truss. The statistics of these results are presented later in this section.

A graphic illustration of the typical nature of the structural response to these random member-length imperfections is provided in Figures 10 through 13. Note in the figures, which greatly exaggerate the magnitude of the deflections for clarity, that the cumulative effect of the random member lengths may result in significant ring-closing mismatch errors, as expected. However, the deflected shape shown in the figures also reveals significant distortions and lack of geometric precision distributed broadly throughout the structure. The magnitude of these deflections was found to be relatively small for the parameters used in the simulation.

As previously mentioned, the worst-case results and extreme value statistics for both deflections and stresses are of particular engineering interest. Furthermore, the shape of the distribution of member length imperfections is generally expected to have a significant effect on these extreme value statistics. To explore these effects, the entire simulation study was performed twice, once using a set of Gaussian member length imperfections and again using a uniform distribution of member length imperfections. The generating distributions each used a mean value of zero. The Gaussian data was generated with a nominal standard deviation of effective fixed-end member strains of 1.0×10^{-5}, so that the actual standard deviation for the uniform distribution was only 5.77×10^{-6}, which is 42.3 percent smaller than that used for the Gaussian data. As a result, the deflections and stresses obtained from the simulation results for the uniformly distributed case were consistently smaller than those for the Gaussian case. To facilitate evaluation of the effects of the shape of distribution of random imperfections on the results, normalized data in which the structural deflections and stresses are divided by the standard deviation of input random member strains are provided, together with the raw results.

7.1 RING-CLOSING MISMATCH DEFLECTIONS DUE TO PSEUDO RANDOM MEMBER LENGTH IMPERFECTIONS

Presented in Table 3 are the results obtained for ring-closing mismatch deflections in the open ring truss model for the case where pseudo random member lengths were generated from a Gaussian distribution. The generating distribution had a mean value of effective fixed-end member strains of zero, and a standard deviation of 1×10^{-5}. An ensemble of 100 cases was used to construct the statistical results shown in the table.

The table provides results for the x, y and z components of displacement at nodes 273, 274, 275 and 276. As shown in Figure 14, these nodes are located at the corners of the plane of separation of the open-ring model, directly adjacent to the constrained duplicate nodes at which the six displacement boundary conditions are applied. Thus, the average deflection of these nodes provides a measure of the ring-closing mismatch deflections.

As seen in the table, the mean value of all deflections is small, as expected. In the limit of infinite ensemble size, the mean of all deflections should approach zero. The standard deviation of the deflections in the x-direction is more than twice that in either the y- or z-directions, and the extreme values obtained in the x-direction are approximately twice those obtained in the other directions. An examination of the case numbers at which extreme values occurred for each node shows that the displacements at all four nodes are highly correlated, as expected.

Although the standard deviation of effective fixed-end member strains of 1×10^{-5} was selected somewhat arbitrarily, this value may be representative of the level of accuracy of fabrication which may be achieved with reasonable effort using current technology. The results then indicate that the ring-closing mismatch deflections are highly likely to be smaller than 0.5 inches (12.7 mm) in any direction. Furthermore, if the results in Table 3 are used to model the average x, y and z mismatch deflections as zero-mean random variables with standard deviations of approximately

$$\sigma_x = 0.169 \text{ in } (4.29 \text{ mm})$$

$$\sigma_y = 0.072 \text{ in } (1.83 \text{ mm})$$

$$\sigma_z = 0.065 \text{ in } (1.65 \text{ mm})$$

the stiffness matrix presented in Section 3 may be used to estimate the statistical properties of the loads required during assembly to close the ring. If correlations between the x, y and z deflections are ignored, then the ring-closing forces are found to be zero-mean random variables with standard deviations of approximately

$$\sigma_{f_x} = 1.95 \text{ lb } (8.68 \text{ N})$$

$$\sigma_{f_y} = 5.28 \text{ lb } (23.5 \text{ N})$$

$$\sigma_{f_z} = 5.85 \text{ lb } (26.0 \text{ N})$$

Thus, it is very unlikely that a ring-closing load larger than about 25 lbs (111 N) would be required during assembly. This estimate is not far off from the value of 33.1 lb (147 N) which results when the worst case extreme value deflections in each direction are used without regard to the fact that they occur in different cases, and without regard to algebraic sign.

Furthermore, if the ring-closing loads are regarded as Gaussian random variables, then the standard deviations provided above may be used to obtain rough estimates of the probability of occurrence of various events. For example, the maximum likelihood extreme value for an ensemble of size 100 is 2.375 σ for any zero-mean Gaussian process. Thus, the maximum values of ring-closing forces in an ensemble of size 100 might be expected to be

$$f_{x_{max}} \cong 4.63 \text{ lb } (20.6 \text{ N})$$

$$f_{y_{max}} \cong 12.5 \text{ lb } (55.6 \text{ N})$$

$$f_{z_{max}} \cong 13.9 \text{ lb } (61.9 \text{ N})$$

Of course, these values all depend on the assumption of zero-mean Gaussian member-length imperfections with a standard deviation of fixed-end strain of 1×10^{-5}. However, since the analysis is entirely linear, results for any other size of standard deviation of member strain may be obtained by simply multiplying the results presented here by the appropriate ratio of standard deviations.

Presented in Table 4 are similar results from another Monte Carlo simulation which used pseudo uniformly distributed member length imperfections instead of Gaussian imperfections. The mean value used was again zero, but the range of effective fixed-end member strains used was $\pm 1 \times 10^{-5}$, resulting in a standard deviation of 5.77×10^{-6}, which is somewhat smaller than that corresponding to Table 3. The results obtained in this way display all the same general features previously described with regard to Table 3, except the values of the deflections are proportionately smaller, as expected.

In order to examine the effects of the shape of distribution used to generate member length imperfections, the results presented in Tables 3 and 4 were condensed and normalized, as shown in Table 5. The mean value, standard deviaensemble statistics for the mean and standard deviation of the largest stress which occurs among the 897 members in the truss in each case. Also presented is the ensemble maximum value of the largest stress in the 100 cases.

For the assumed standard deviation of effective fixed-end member strain of 1×10^{-5}, the global maximum member stress of approximately 500 psi (3.4 MN/m^2) is not especially large. Since the analysis is entirely linear, results for different values of the standard deviation of initial member strain may be obtained by simply multiplying the tabulated results by the appropriate ratio of standard deviations.

Note that the results for the mean and standard deviation of largest member stress may be used to provide estimates of the probability of occurrence of various events, if an appropriate assumption is made for the probability distribution associated with the largest member stress. For example, if it is assumed that the largest member stress is Gaussian in distribution, then the maximum likelihood value of the largest value in an ensemble of size 100 is ($\mu + 2.375\sigma$), or, in this case, 499.43 psi (3.44 MN/m^2), which compares surprisingly well with the reported simulation results. Similarly, the maximum likelihood value for a sample of size 1,000 is ($\mu + 3.114\sigma$), or 534.11 psi (3.75 MN/m^2). Since no simulation results are available for an ensemble of this size, the estimate cannot be verified as yet.

Also shown in Table 6 is a sample of ten results from the ensemble of size 100. For each of the ten cases shown, the mean and standard deviation of the 897 member stresses are given. Note that the mean stress is nearly zero, as expected. Approximate statistics for member loads are provided next. These results were obtained by multiplying the member stress results by the value of cross-sectional area of $A = 0.36568\ in^2$. The results are therefore approximate, since ten of the 897 members (all of which lie in the plane of separation) actually have a cross-sectional area which is half this value. Note that the mean member load is nearly zero, and the standard deviation is in the neighborhood of 30 to 35 lbs (130 to 150 N). In the right-hand column are the stress, load and element identification numbers corresponding to the most highly stressed member in the truss. Note that element number 431 was the most highly stressed member in case 25 and also in case 35. The value of maximum member stress is often more than four times the standard deviation of the

PRECEDING PAGE BLANK NOT FILMED

PAGE 30 INTENTIONALLY BLANK

member stresses, even though the truss has fewer than 1,000 members. Thus, it appears that the member streses may not be well modeled by the Gaussian distribution.

A final observation from Table 6 is that member 439, which is the most highly stressed member in case 45, lies in the plane of separation. Thus, it was modeled with a cross-sectional area one-half that used for other members, as explained in Section 3.1. As a result, the member load of 54.73 lbs (243.45 N) is approximately one-half the other values in the table, all of which correspond to members which are not in the plane of separation.

Presented in Table 7 are results similar to those in Table 6, but for the case of uniformly distributed initial member length imperfections. In this case, the mean value of the effective fixed-end member strain is zero, the range is $\pm 1 \times 10^{-5}$, and the standard deviation is 5.77×10^{-6}. Because the standard deviation of initial imperfections is smaller in Table 7 than in Table 6, the most obvious difference between the two tables is that the stresses and loads in Table 7 are substantially smaller. Other trends and details are similar to those previously discussed with regard to Table 6.

Of particular interest in comparing results in Tables 6 and 7 is the effect of the shape of distribution of initial member length imperfections on the results for the most highly stressed member in the truss. In order to address this issue, the ensemble statistics of largest member stress at the top of Tables 6 and 7 were normalized by dividing them by the corresponding value of the standard deviation σ_ϵ of effective fixed-end member length imperfections. The results are shown in Table 8. It is clear from these normalized results that the mean value of the maximum stress is very nearly the same for each distribution. It therefore appears that the shape of distribution has a very small effect on the mean value of the maximum member stress. Similar results are obtained for the largest maximum stress in the ensemble of 100 cases. However, the results for the normalized standard deviation of maximum member stress seem to indicate that the Gaussian distribution may produce significantly larger results.

The normalized standard deviation result for the Gaussian distribution is 15 to 20 percent larger than the result for the uniform distribution. While more statistical testing is necessary to obtain more confidence in the significance of this result, it appears that the shape of distribution of initial imperfection may indeed exert a significant influence over the shape of the distribution of maximum member stress, at least for problems and ensembles of this size.

In addition to providing information on the magnitude of the maximum member stress, the simulation also provided information on the type of member and spatial location within the truss of the most highly stressed member in each case. These results are presented in Table 9. The columns in the table correspond to different member types. Longerons are members which are parallel to the longitudinal axis of beam-like segments of the truss. Battens are members which are orthogonal to longerons. (The distinction between longerons and battens is blurred in bays which lie at corners or intersections of beam-like segments.) Face diagonals are diagonal members which lie in an exterior face of a bay, and core diagonals are diagonal members which pierce through the interior of a bay.

The rows in Table 9 correspond to the general spatial location of segments of the truss. These segments are defined in Figure 15. The numbers occurring at each location in the matrix of Table 9 represent the number of times in 100 cases that the most highly stressed member in the truss happened to be of the type and location shown. The numbers in parentheses represent the expected number of occurrences in a truly random distribution in which each truss member has an equal probability of attracting the highest stress in the truss.

As noted from the results, several remarkable patterns emerge from Table 9. First, focusing on the right-most column in the table, it is found that the most highly stressed member occurred in a bay which was either a corner or intersection in 43 percent of the cases, whereas a truly random selection would predict only 12 percent. Thus, it is inferred that residual stresses due to random member-length imperfections are likely to be largest at corner and intersection bays of the truss, almost half the time.

Another noteworthy feature of Table 9 is revealed in the last row. Here it is found that a core diagonal was the most highly stressed member in the truss in 55 percent of the cases, whereas a truly random selection would predict only about 8 percent. Furthermore, battens and face diagonals would appear to be only moderately stressed unless they are located in a corner or intersection bay of the truss.

While further study would lend deeper insight, the predominance of occurrence of the most highly stressed member in corners and intersections of truss beams and in diagonal members is explained in part by the deterministic results of a compliance study reported in Table 10. Shown in the table are values for the compliance of the open-ring truss model at various locations. To obtain each value shown, an analysis of the truss deflections due to a pair of self-equilibrated unit loads was performed. The unit loads were applied in opposite directions at the ends of, and parallel to, a single truss member, after removing that member from the truss. The truss deflections in the direction of the applied loads was divided by the magnitude of the applied load to obtain the compliance. As shown in the table, a wide range of compliances exist, depending on member type and location within the truss. For example, the results show that the compliance of the truss for battens is higher than that for any other member, except at an intersection where the distinction between battens and longerons is blurred. Thus, a unit member length error in a batten always produces lower stresses than similar length errors in any other member type in the same truss region. This is one explanation for the fact that battens very rarely occur as the most highly stressed member in the truss, as shown in Table 9.

Furthermore, the results in Table 10 also show that in regions away from corners and intersections, core diagonals have the smallest compliance. As a result, it is expected that core diagonals would be the most likely candidates for attracting the highest member stresses in such locations, an expectation which matches with the statistical observations in Table 9.

For members in corners and intersections, the compliance of face diagonals is found to be the smallest of all other members. Furthermore, the results in the fourth column of Table 9 show the expected trend that the population of face diagonals is quite large among the most highly stressed members in corner and intersection regions of the truss, but not elsewhere. Longeron compliances are not notably different in different truss regions, and as a result, the distribution of occurrences of longerons as the most highly stressed member is fairly uniform, as shown in Table 9.

A study of the element identification (ID) numbers used to construct Table 9 reveals further useful observations, which are summarized in Table 11. Of the 100 cases examined, it was found that a total of only 70 unique element ID numbers occurred. Fifty of these 70 members occurred exactly once as the most highly stressed member, and the remaining 20 members occurred two or more times. A summary of the member type and spatial location of the 20 repeating members is shown near the top of Table 11. Note the predominance of occurrences in core diagonals and in corner and intersection bays.

Of the twenty members which repeated at least once in the ensemble of most highly stressed members, seven occurred three or more times. The results for these seven cases are shown next in Table 11, with results which are similar to those just discussed. Finally, near the bottom of Table 10 are the results for the three cases within the seven discussed above, in which a single member occurred four or more times as the most highly stressed member in the truss. Again, the pattern seems to be that corner and intersection bays attract the largest stresses, but this time there are no core diagonals among the observed outcomes. In order for this particular statistical analysis to be sufficiently powerful to make confident decisions, much more simulation is necessary. For example, a chi-square goodness-of-fit test normally requires a sample size sufficiently large that each entry in the matrix has an expected number of occurrences of five or more. The largest expected occurrence at the bottom of Table 11 is only 1.85, so the results here must be interpreted accordingly.

That the list of 100 most highly stressed members is found to contain some repeated members is an event which is not surprising. In fact, assuming a truly random selection among the 897 truss members, the probability that all 100 cases will occur in different members (without any repeats) is only 0.00343. However, the results tend to indicate that members located at corner or intersection bays and members which are core diagonals are at a much higher risk of receiving high residual stresses due to member length imperfections than might be expected. This result is partially explained by the compliance results reported in Table 10.

7.3 RANDOM IMPERFECTION-INDUCED MISALIGNMENT OF REFERENCE PLANES ATTACHED TO THE TRUSS MODEL

One of the concerns of designers of the Space Station is the geometric precision of the primary structure for purposes of attachment of instruments for use in guidance and attitude control. Due to the large size of the structure, it is likely that sensitive instruments will be located at several reference stations throughout the truss and, therefore, structural distortions may cause difficulties for attitude measurement and control. The structural distortions may be caused by member-length imperfections, or by small thermal gradients and random variations in the coefficient of thermal expansion in different members of the truss.

To obtain some quantitative evaluation of such effects, the results of a Monte Carlo simulation study were used to examine the imperfection-induced misalignment of two reference planes attached to the truss. The two reference planes were assumed to be determined by the location of the following two sets of three nodes each: Node numbers 30, 31 and 34 (plane 1) and node numbers 218, 222 and 223 (plane 2). The location of these planes relative to the overall truss is shown in Figure 16. Note that the outer normal vector of each reference plane is parallel to the x-axis, in the absence of structural deformations.

The simulation results consist of an ensemble of 100 cases generated assuming zero-mean Gaussian member-length imperfections. The standard deviation of the effective fixed-end member strain was 1×10^{-5}, and a seed or starting value of 325 was used. The open-ring truss model with statically determinant displacement boundary conditions along the plane of separation was analyzed using the previously described MSC/pal 2 software.

The variations in member lengths may be regarded as the result of a deterministic temperature change and random variations in the coefficient of thermal expansion in different truss members, or it may be regarded as the result of random variations in temperature changes throughout the truss, or some combination of all these effects.

For each of the 100 cases in the ensemble, the deflections at the three nodes which define each reference plane were used to determine the locations of these nodes in the deformed configuration of the truss. The deformed nodal locations were then used to determine the direction cosines of the outer normal vectors of each reference plane. Finally, these direction cosines were used to determine the angle between outer normal vectors or, equiva-lently, the angular misalignment of reference planes.

The statistical results for angular misalignment obtained from this simulation are given in Table 12. The values obtained for misalignment are quite small for the case considered. Since the problem analyzed is entirely linear, the results for different values of standard deviation of initial member strain may be obtained by simply multiplying the results in Table 12 by the appropriate ratio of standard deviations of input member strains.

SECTION 8

STEP-BY-STEP PROCEDURES FOR SIMILAR ANALYSIS OF GENERAL TRUSS STRUCTURES WITH RANDOM IMPERFECTIONS

To provide maximum benefit of this report to readers who intend to perform similar analyses on trusses in configurations other than that of the Space Station, a summary of step-by-step procedures used to perform the Monte Carlo simulation is reported in this section. The primary analysis tool used in the simulation study is the MSC/pal 2 finite element program, and user familiarity with this program is assumed. Since the MSC/pal 2 program generates rather large ASCII output files and overwrites its own internal binary files with each case, the total amount of disk storage required to perform a sizeable Monte Carlo simulation can easily exceed the available disk storage on a microcomputer. In order to mitigate these storage requirement problems and to streamline the file handling needed, three additional programs named RAN2PAL, PAL2BIN, and BIN2STAT were written at Astro Aerospace Corporation specifically for this purpose. A listing of the source code of each of these three programs (written in C language) is provided in Appendix 2 of this report. The programs prompt the user for the required input and are reasonably user-friendly. Detailed explanation of their use is outlined later in this section.

The program RAN2PAL is used to create an ensemble of pseudo random member temperature changes to simulate the input member length imperfections for PAL2 structural analysis. The resulting input files may be stored in binary form to save disk space, and subsequent calls to RAN2PAL may be used to extract a particular case number from the ensemble, expand it into ASCII form, and write it out to disk in a form suitable for direct input as a load file to the STAT2 analysis program within the MSC/pal 2 software package.

After performing a structural analysis of a particular case, the PAL2BIN program may be used to read the verbose PAL2 output ASCII files for displacements and stresses, compress them into binary form, and append them to an archive binary storage file for later processing. When all cases have been run, and the archive binary storage files for displacement and stress are complete, the program BIN2STAT may be used to perform certain statistical analyses of the results. In particular, BIN2STAT can be used to scan the binary displacement results file and provide ensemble statistics (mean, standard deviation, maximum value, minimum value) for the x, y, and z components of displacement at any node in the truss. Furthermore, it may also be used to scan the binary stress results file and provide ensemble statistics for the largest stress which occurs in any member of the truss. In addition, the program may be used to provide statistical averages of stresses across all members of the truss in a given case selected from the ensemble.

For clarity, a typical complete truss analysis scenario is presented next. Step-by-step instructions are provided to guide the potential user through the use of the new software, from model formulation to the interpretation of ensemble statistics for nodal displacements and member stress.

Step 1: Use a text editor to create an ASCII file containing the geometry of the truss and properties of the materials used to make its members using PAL2 model definition commands (e.g., NODAL LOCATIONS, MATERIAL PROPERTIES, BEAM TYPE 1, ZERO, CONNECT, END OF DEFINITION, etc.). Be sure to set the coefficient of thermal expansion to one for all members, and the reference temperature to zero. (An example is provided in Appendix 1 by the listing of SPASTAT2.TXT file.) For illustration, we refer to this new model file as TRUSS.TXT.

Step 2: The MSC/pal 2 program VIEW2 may be used to create three dimensional line drawings of the newly created model TRUSS.TXT in order to verify the geometry of the model before analysis.

Step 3: After the TRUSS.TXT file has been verified and is error free, the MSC/pal 2 program PAL2 can be run in order to "build a new model." This requires selecting the input B from the main menu. In response to the query for data set name, the model text file name TRUSS.TXT should be given. PAL2 then creates several internal binary files containing the truss unrestrained global stiffness matrix, etc.

Step 4: Next, the program RAN2PAL (see Appendix 2) can be run in order to generate an ensemble of pseudo random input load cases. On the first execution of the RAN2PAL program, an "INPUT" file containing summary information about the truss (number of nodes, number of elements, etc.) and the desired statistical properties of the ensemble of member length imperfections (ensemble size, mean and standard deviation of effective fixed-end member strains, etc.) must be created. This may be accomplished by providing the following responses to the first query generated by RAN2PAL. (Only the characters which appear to the left of the equal sign should be typed and followed by a "carriage return." Descriptive comments are provided to the right of the equal sign.)

TRUSS.INP = pathname of desired new input file

The program responds with a message which reads like the following:

```
Opening new input file C:TRUSS.INP
```

and then continues by asking a series of straightforward questions about the truss model and desired ensemble of pseudo random input. The questions require either numerical data as responses or ASCII character strings to form the displacement boundary conditions to be used in the STAT2 analysis. The boundary conditions must be written using the MSC/pal 2 Solution Definition Commands (see Reference 4 and Appendix 2, RANLOAD.TXT file for examples). A blank line terminates this input phase, and causes the program to present a summary of parameters. An opportunity for data corrections is provided, and when all corrections are finished, a zero may be entered to move on to the random number generation phase. At this point, the user is asked to indicate whether the generated load files should be written to disk in expanded ASCII form (A), compressed binary form (B), or both (C). Unless the problem being run is very small, it is recommended that option B be selected:

B = write only a compressed binary file

The program immediately begins generating the pseudo random input files and storing them in a single binary file named (in this case) TRUSS.BIN. Messages are provided on the video monitor to indicate progress on the computation, since a large Monte Carlo simulation may require a significant amount of time on a slow machine to generate these input files.

At the conclusion of the program, a check of the directory will show that two files have been written: TRUSS.INP, which contains summary information about the truss and random loads, and TRUSS. BIN, which contains the complete ensemble of random load cases, stored in binary format.

Step 5: Run RAN2PAL a second time to extract an expanded ASCII version of the first case in the ensemble, case 0. This is done by providing the following responses:

TRUSS.INP = pathname of existing input file

The program responds with the summary of parameters corresponding to the simulation and aks if there are corrections:

0 = no corrections to displayed parameter values

The program responds with the message: A previously generated binary file of random values will be used. Separate PAL2 load files will be written.

Next, the program prompts for the desired case number:

0 = first case number in the binary file

The program responds with a message that case number 0 is being read from the binary file, then it writes an ASCII file named (in this case) TRUSS.00 containing the PAL2 load file for running the first case in the ensemble.

Step 6: Run the MSC/pal 2 program STAT2 to compute the structural response to this first load case. In running STAT2, the user should select the "S" or Static Analysis option, and read an input data set whose name is TRUSS.00.

Step 7: Before exiting STAT2, the "D" or Data Recovery option should be exercised twice to extract ASCII output files for nodal displacements and member stresses. The format of these files is important for the subsequent use of the file management program PAL2BIN, which is discussed later.

To extract the displacement output, the user should first select the "D" or Data Recovery option, then the "D" or Output to Disk option. Next, a temporary file name such as DISP.PRT should be provided as the destination of the output displacements. The following output format options should be selected next:

N = No components of applied forces printed
N = No components of external forces printed
F = all defined components of displacements printed
N = No element stress output printed

The program then creates an ASCII file named DISP.PRT containing (in verbose format) the output nodal displacements from the structural analysis.

To extract the element stress output, the user should again select the "D" or Data Recovery option, followed by the "D" or Output to Disk option. Then, another temporary file name such as STRN.PRT should be provided as the destination of the output stresses. The following output format options should be selected:

N = No components of applied forces printed
N = No components of external forces printed
N = No components of displacements printed
F = All element stresses analyzed
1 = Print level 1, average nodal stresses
-blank- = Print all stresses, no matter how small
-blank- = Print stresses for every element in truss

The program creates an ASCII file named STRN.PRT containing (in verbose format) the output element stresses from the structural analysis.

At this point, the analysis of the first case in the ensemble is complete and the user should select the "Q" or Quit option in STAT2.

Step 8: Next, the file management program PAL2BIN (see Appendix 2) can be run twice to create archive binary storage files for output displacement and stresses.

To create new binary storage files on the first execution of PAL2BIN, the following options should be selected:

i = initialize the binary storage file
d = to deal with nodal displacements
TRUSS.INP = ASCII-from configuration file with extension .INP

The program then creates a new binary storage file names TRUSS.DIS in which to store compressed STAT2 displacement output results.

A new binary storage file for stress output may be created next by another execution of PAL2BIN in which the following options are selected:

i = initialize the binary storage file
s = to deal with strut stresses
TRUSS.INP = ASCII-form configuration file with extension .INP

The program then creates another new binary storage file named TRUSS.STR in which to store compressed STAT2 stress output results.

Step 9: The STAT2 output displacements in DISP.PRT may be compressed and appended to TRUSS.DIS by another execution of PAL2BIN in which the following options are selected:

a = to add output from a PAL2 data file
d = to deal with nodal displacements
DISP.PRT = pathname of PAL2 data file
1 = number of cases in this PAL2 file
TRUSS.DIS = pathname of binary data file with extension .DIS

The program responds by compressing and appending the nodal displacements in DISP.PRT to TRUSS. DIS.

Similarly, the output stresses in STRN.PRT may be compressed and appended to TRUSS.STR by a final execution of PAL2BIN in which the following options are selected:

a = to add output from a PAL2 data file
s = to deal with strut stresses
STRN.PRT = pathname of PAL2 data file
1 = number of cases in this PAL2 file
TRUSS.STR = pathname of binary data file with extension .STR

The program responds by compressing and appending the member stresses in TEMP.STR to TRUSS.STR. At this point there is no further need for the ASCII files TRUSS.00, DISP.PRT, and STRN.PRT, and they should be deleted to provide more disk storage space for the next case in the ensemble.

Step 10: Repeat Steps 5, 7, and 9, each time extracting and processing the next case in the ensemble of random input loads until the ensemble analysis is complete.

Step 11: The statistical analysis postprocessor program BIN2STAT (see Appendix 2) may be used at this point to generate ensemble statistics on the results.

To examine the nodal displacement results, the following options in BIN2STAT should be selected:

d = to examine displacements
TRUSS.DIS = name of binary DISplacement storage file to be read

The program responds with a summary of parameters corresponding to the ensemble of results in the storage file, and prompts the user for a node number for statistical analysis. For each node number selected, the program provides the ensemble statistics (mean, standard deviation, maximum and minimum values with corresponding case numbers) for deflections in the x, y and z directions.

When finished examining the displacements, an "x" should be entered, which returns the user to the main menu, where stress results may be selected for review by specifying the following input:

s = to examine stresses
TRUSS.STR = name of binary STRess storage file to be read

The program responds with a summary of the parameters corresponding to the ensemble of results in the storage file. Entering any key then causes the program to generate ensemble statistics for the most highly stressed member in each case. The results displayed include the average, standard deviation, and global maximum value of the maximum stress in each case.

Next, the program prompts the user for a case number for more detailed analysis. For each case selected, the program then provides statistical results across all members in the truss, including average, standard deviation, and maximum stress (including element ID number for most highly stressed member).

When finished examining stresses, the user must enter "x" to return to the main menu, and "Q" to terminate the program.

Automated Processing: Following the foregoing step-by-step process for a large number of cases would require the attention of the analyst for a prohibitively long time. For this reason, the program RAN2PAL and PAL2BIN are each able to be run from a command line. Thus, the command:

```
RAN2PAL TRUSSNAME M
```

will extract the Mth case of random strains and write the STAT2 input file TRUSSNAME.M.

Similarly, the command lines:

```
PAL2BIN DIS TRUSSNAME.DIS DISP.PRT
PAL2BIN STR TRUSSNAME.STR STRN.PRT
```

will take the PAL2 output files DISP.PRT and STRN.PRT and append them, in binary form, to the storage file TRUSSNAME.DIS and TRUSSNAME.STR, respectively.

These command lines can be used together with batch files to run 100 cases as follows:

1) Prepare by performing steps 1 through 6 and step 8. Make sure that the binary input file, TRUSS.BIN, contains at least 100 cases.

2) Execute the command line:

```
FOR %I IN (0 1 2 3 4 5 6 7 8 9) DO C:COMMAND /C RUNRAN TRUSS %I
```

The batch file RUNRAN.BAT contains the following commands:

```
FOR %%J IN(0 1 2 3 4 5 6 7 8 9) DO RAN2PAL %1 %2%%J
DEL ESPRAN.TXT
REN %1.%2%%J EPSRAN.TXT

STAT2 <CMD1
STAT2 <CMD2

PAL2BIN DIS %1.DIS DISP.PRT
PAL2BIN STR %1.STR STRN.PRT
```

When complete, the data set for 100 cases is found in the output files DISP.PRT and STRN.PRT. Note that the for-do loops generate the case number. The pipes feed STAT2 with the necessary responses. The input files CMD1 and CMD2 are:

CMD1:

```
S
Y
ESPRAN.TXT
```

CMD2:

```
D
D
DISP.PRT
N
N
F
N
```

```
D
D
STRN.PRT
N
N
N
F
1

Q
```

Note that when STAT2 is first run, it looks for more input after solving for ESPRAN.TXT and finds none. It accordingly exits with an error message, but is ready to be run again for extracting the data.

The foregoing process takes about 10 hours for 100 cases on a Compaq 386 computer.

REFERENCES

1. Knuth, D.E., The Art of Computer Programming, Volume 2, Seminumerical Algorithms, 2nd ed., 1981, Reading, MA, Chapter 3.

2. Microsoft C 5.1 Run-Time Library Reference Manual, 1987, Redmond, WA, pp. 479 and 564-5.

3. IMSL Library Reference Manual, Edition 8, June 1980, Houston, TX, pp. GGUBS-1, GGUBS-2.

4. C Language: Scientific Subroutine Library Manual, Peerless Engineering Service, John Wily & Sons, New York 1985.

5. MacNeal-Schwendler Corporation, MSC/pal 2 Advanced Stress and Vibration Analysis, User's Manual; ..., Reference Manual; ..., Application Manual, Los Angeles 1987.

TABLE 1.

CHI-SQUARE VALUES FOR DISTRIBUTION OF PSEUDO STANDARD NORMAL VARIATE USING 14 EQUIPROBABLE INTERVALS COVERING THE REAL NUMBER LINE.

Algorithm	Sample Size n	χ^2 Values						Average χ^2 Value
		Seed Value U_0 =						
		325	320	330	340	350	123123	
Microsoft C Routines, rand () and Srand()	100	7.80	13.69	9.48	6.40	21.25	16.21	12.47
	1000	7.56	10.89	--	--	--	--	9.22
	5000	11.02	11.98	--	--	--	20.15	14.39
IMSL Routine GGUBS()	100	11.45	21.53	21.53	17.05	15.09	19.29	17.65
	1000	11.31	11.17	--	--	--	--	11.24
	5000	16.78	11.70	--	--	--	14.95	14.48
Peerless Algorithm	100	8.36	33.29	9.48	8.36	8.64	11.45	13.27
	1000	10.66	16.58	--	--	--	--	13.62
	5000	10.56	8.81	--	--	--	17.41	12.26

TABLE 2.

CHI-SQUARE VALUES FOR DISTRIBUTION OF PEAKS IN 50 SEQUENCES OF 100 PSEUDO STANDARD NORMAL VARIATES EACH USING 10 EQUIPROBABLE INTERVALS COVERING THE POSITIVE REAL NUMBER LINE.

Algorithm	χ^2 Values			Average χ^2 Value
	Seed Value U_0 =			
	325	320	123123	
Microsoft C Routines rand () and Srand ()	10.00	5.40	12.60	9.33
IMSL Routine GGUBS ()	9.00	10.00	7.60	8.87
Peerless Algorithm	17.60	6.80	14.40	12.93

TABLE 3.

RING-CLOSING MISMATCH DEFLECTIONS DUE TO PSEUDO GAUSSIAN MEMBER LENGTH IMPERFECTIONS WITH MEAN STRAIN = 0, STANDARD DEVIATION = 1 x 10^{-5}, SEED = 325, ENSEMBLE SIZE = 100 CASES

X-DIRECTION DEFLECTIONS

Node	Mean	Standard Deviation	Maximum Value	Case No.	Minimum Value	Case No.
273	0.01047 in (0.266 mm)	0.16461 in (4.181 mm)	0.39961 in (10.150 mm)	(61)	-0.32613 in (-8.284 mm)	(13)
274	0.01036 in (0.263 mm)	0.16445 in (4.177 mm)	0.39849 in (10.122 mm)	(61)	-0.32848 in (-8.343 mm)	(13)
275	0.00815 in (0.207 mm)	0.17414 in (4.423 mm)	0.42101 in (10.694 mm)	(61)	-0.32740 in (-8.316 mm)	(18)
276	0.00818 in (0.208 mm)	0.17406 in (4.421 mm)	0.42236 in (10.728 mm)	(61)	-0.32848 in (-8.343 mm)	(18)

Y-DIRECTION DEFLECTIONS

Node	Mean	Standard Deviation	Maximum Value	Case No.	Minimum Value	Case No.
273	0.00063 in (0.016 mm)	0.06865 in (1.744 mm)	0.17317 in (4.399 mm)	(10)	-0.14461 in (-3.673 mm)	(84)
274	-0.00202 in (-0.051 mm)	0.06487 in (1.648 mm)	0.13372 in (3.396 mm)	(20)	-0.14947 in (-3.797 mm)	(26)
275	0.00191 in (0.049 mm)	0.07886 in (2.003 mm)	0.19987 in (5.077 mm)	(10)	-0.16909 in (-4.295 mm)	(84)
276	-0.00127 in (-0.032 mm)	0.07495 in (1.904 mm)	0.16222 in (4.120 mm)	(20)	-0.16535 in (-4.200 mm)	(87)

Z-DIRECTION DEFLECTIONS

Node	Mean	Standard Deviation	Maximum Value	Case No.	Minimum Value	Case No.
273	-0.00005 in (-0.001 mm)	0.06588 in (1.673 mm)	0.13707 in (3.482 mm)	(28)	-0.17989 in (-4.569 mm)	(41)
274	-0.00248 in (-0.063 mm)	0.06446 in (1.637 mm)	0.13905 in (3.532 mm)	(28)	-0.20840 in (-5.293 mm)	(41)
275	-0.00033 in (-0.008 mm)	0.06592 in (1.674 mm)	0.13706 in (3.481 mm)	(78)	-0.18381 in (-4.669 mm)	(41)
276	-0.00233 in (-0.059 mm)	0.06441 in (1.636 mm)	0.13660 in (3.470 mm)	(28)	-0.20547 in (-5.219 mm)	(41)

TABLE 4.

RING-CLOSING MISMATCH DEFLECTIONS DUE TO PSEUDO UNIFORMLY DISTRIBUTED MEMBER LENGTH IMPERFECTIONS WITH MEAN STRAIN = 0, RANGE = $\pm 10^{-5}$ (STANDARD DEVIATION = 5.77×10^{-6}), SEED = 325, ENSEMBLE SIZE = 100 CASES

X-DIRECTION DEFLECTIONS

Node	Mean	Standard Deviation	Maximum Value	Case No.	Minimum Value	Case No.
273	-0.00687 in (-0.174 mm)	0.09558 in (2.428 mm)	0.20897 in (5.308 mm)	(13)	-0.22883 in (-5.812 mm)	(85)
274	-0.00681 in (-0.173 mm)	0.09552 in (2.426 mm)	0.21028 in (5.341 mm)	(13)	-0.22694 in (-5.764 mm)	(85)
275	-0.00588 in (-0.149 mm)	0.10131 in (2.573 mm)	0.20751 in (5.271 mm)	(13)	-0.24175 in (-6.140 mm)	(74)
276	-0.00593 in (-0.151 mm)	0.10128 in (2.573 mm)	0.20910 in (5.311 mm)	(13)	-0.23992 in (-6.094 mm)	(74)

Y-DIRECTION DEFLECTIONS

Node	Mean	Standard Deviation	Maximum Value	Case No.	Minimum Value	Case No.
273	0.00077 in (0.020 mm)	0.04088 in (1.038 mm)	0.09914 in (2.518 mm)	(84)	-0.09867 in (-2.506 mm)	(10)
274	0.00217 in (0.055 mm)	0.03861 in (0.981 mm)	0.10110 in (2.568 mm)	(26)	-0.07505 in (-1.906 mm)	(53)
275	0.00007 in (0.002 mm)	0.04680 in (1.189 mm)	0.11500 in (2.921 mm)	(84)	-0.11302 in (-2.871 mm)	(10)
276	0.00174 in (0.044 mm)	0.04466 in (1.134 mm)	0.10926 in (2.775 mm)	(26)	-0.08545 in (-2.170 mm)	(53)

Z-DIRECTION DEFLECTIONS

Node	Mean	Standard Deviation	Maximum Value	Case No.	Minimum Value	Case No.
273	0.00059 in (0.015 mm)	0.04065 in (1.033 mm)	0.10999 in (2.794 mm)	(41)	-0.09157 in (-2.326 mm)	(28)
274	0.00136 in (0.035 mm)	0.04004 in (1.017 mm)	0.12498 in (3.174 mm)	(41)	-0.09387 in (-2.384 mm)	(28)
275	0.00051 in (0.013 mm)	0.04066 in (1.033 mm)	0.11189 in (2.842 mm)	(41)	-0.09072 in (-2.304 mm)	(28)
276	0.00130 in (0.033 mm)	0.04002 in (1.017 mm)	0.12309 in (3.126 mm)	(41)	-0.09223 in (-2.343 mm)	(28)

TABLE 5.

EFFECTS OF DISTRIBUTION SHAPE ON NORMALIZED RING-CLOSING MISMATCH DEFLECTIONS DUE TO PSEUDO RANDOM MEMBER LENGTH IMPERFECTIONS WITH MEAN STRAIN = 0, ENSEMBLE SIZE = 100 CASES EACH

X-DIRECTION NORMALIZED DEFLECTIONS

Distribution Shape	$\left[\frac{\mu_X}{\sigma_\epsilon}\right]$	$\left[\frac{\sigma_X}{\sigma_\epsilon}\right]$	$\left[\frac{X_{max}}{\sigma_\epsilon}\right]$	$\left[\frac{X_{min}}{\sigma_\epsilon}\right]$
Gaussian	929.0 in (23.60 m)	16,930.0 in (430.0 m)	41,040.0 in (1,042.0 m)	-32,760.0 in (-832.1 m)
Uniform	-1,104.0 in (-28.04 m)	17,047.0 in (433.0 m)	36,190.0 in (919.2 m)	-40,590.0 in (-1,031.0 m)

Y-DIRECTION NORMALIZED DEFLECTIONS

Distribution Shape	$\left[\frac{\mu_Y}{\sigma_\epsilon}\right]$	$\left[\frac{\sigma_Y}{\sigma_\epsilon}\right]$	$\left[\frac{Y_{max}}{\sigma_\epsilon}\right]$	$\left[\frac{Y_{min}}{\sigma_\epsilon}\right]$
Gaussian	-18.75 in (-0.476 m)	7,183.0 in (182.4 m)	16,720.0 in (424.7 m)	-15,713.0 in (-399.1 m)
Uniform	205.7 in (5.225 m)	7,402.0 in (188.0 m)	18,380.0 in (466.9 m)	-16,120.0 in (-409.4 m)

Z-DIRECTION NORMALIZED DEFLECTIONS

Distribution Shape	$\left[\frac{\mu_Z}{\sigma_\epsilon}\right]$	$\left[\frac{\sigma_Z}{\sigma_\epsilon}\right]$	$\left[\frac{Z_{max}}{\sigma_\epsilon}\right]$	$\left[\frac{Z_{min}}{\sigma_\epsilon}\right]$
Gaussian	-129.8 in (-3.297 m)	6,517.0 in (165.5 m)	13,740.0 in (349.0 m)	-19,440.0 in (-493.8 m)
Uniform	162.8 in (4.135 m)	6,988.0 in (177.5 m)	20,350.0 in (516.9 m)	-15,950.0 in (-405.2 m)

TABLE 6.

SIMULATION RESULTS FOR RESIDUAL MEMBER STRESSES IN AN ENSEMBLE OF 100 SETS OF GAUSSIAN INITIAL MEMBER-LENGTH IMPERFECTIONS (MEAN EFFECTIVE FIXED-END MEMBER STRAIN = 0, STANDARD DEVIATION = 1 x 10^{-5}, SEED = 325)

Ensemble Statistics of Largest Member Stress

Mean = 359.03 psi (2.477 MN/m^2)
Standard Deviation = 59.114 psi (0.408 MN/m^2)
Ensemble Maximum of Largest Member Stress = 497.30 psi (3.431 MN/m^2)
(occurring in element 188 for case 56)

Sample of Ten Results from the Ensemble

CASE NO.	STATISTICS OF MEMBER STRESSES		APPROXIMATE STATISTICS OF MEMBER LOADS		MAXIMUM VALUE OF MEMBER		
	Mean psi (N/m^2)	Std. Dev. psi (MN/m^2)	Mean lbs (N)	Std. Dev. lbs (N)	Stress psi (MN/m^2)	Load lbs (N)	ELEMENT ID NO.
5	-0.0125 (-86.2)	82.30 (0.5677)	-0.00457 (-0.02034)	30.10 (133.90)	283.90 (1.958)	103.82 (461.84)	151
15	-0.0722 (-497.7)	96.19 (0.6635)	-0.02640 (-0.11748)	35.18 (156.50)	408.70 (2.819)	149.45 (664.83)	325
25	0.0260 (179.3)	85.32 (0.5885)	0.00951 (0.04232)	31.20 (138.79)	252.50 (1.742)	92.33 (410.73)	431
35	-0.0666 (-459.4)	75.57 (0.5212)	-0.02435 (-0.10836)	27.63 (122.91)	294.30 (2.030)	107.62 (478.75)	431
45	0.1114 (768.4)	81.49 (0.5621)	0.04074 (0.18129)	29.80 (132.57)	299.30 (2.064)	54.73 (243.45)	439
55	-0.0094 (-64.84)	86.87 (0.5992)	-0.00344 (-0.01531)	31.77 (141.33)	425.70 (2.936)	155.67 (692.50)	434
65	0.0236 (162.8)	89.04 (0.6142)	0.00863 (0.03840)	32.56 (144.84)	458.90 (3.165)	167.81 (746.50)	544
75	-0.0477 (-329.0)	87.81 (0.6057)	-0.01744 (-0.07761)	32.11 (142.84)	376.30 (2.596)	137.61 (612.16)	897
85	-0.0589 (-406.3)	89.65 (0.6184)	-0.02154 (-0.09585)	32.78 (145.82)	452.30 (3.120)	165.40 (735.78)	780
95	-0.0062 (-42.8)	110.93 (0.7651)	-0.00227 (-0.01010)	40.56 (180.43)	493.50 (3.404)	180.46 (802.78)	199

TABLE 7.

SIMULATION RESULTS FOR RESIDUAL MEMBER STRESSES IN AN ENSEMBLE OF 100 SETS OF UNIFORMLY DISTRIBUTED INITIAL MEMBER-LENGTH IMPERFECTIONS (MEAN EFFECTIVE FIXED-END MEMBER STRAIN = 0, RANGE = ±1 x 10^{-5}, STANDARD DEVIATION = 5.77 x 10^{-6}, SEED = 325)

Ensemble Statistics of Largest Member Stress

Mean = 201.58 psi (1.391 MN/m^2)
Standard Deviation = 28.519 psi (0.1968 MN/m^2)
Ensemble Maximum of Largest Member Stress = 273.90 psi (1.890 MN/m^2)
(occurring in element 544 for case 65)

Sample of Ten Results from the Ensemble

CASE NO.	STATISTICS OF MEMBER STRESSES		APPROXIMATE STATISTICS OF MEMBER LOADS		MAXIMUM VALUE OF MEMBER		
	Mean psi (N/m^2)	Std. Dev. psi (MN/m^2)	Mean lbs (N)	Std. Dev. lbs (N)	Stress psi (MN/m^2)	Load lbs (N)	Ele-Ment ID No.
5	0.00689 (47.52)	47.614 (0.3284)	0.00252 (0.0121)	17.41 (77.47)	171.90 (1.186)	62.86 (279.73)	207
15	0.03523 (243.00)	54.637 (0.3769)	0.01288 (0.05732)	19.98 (88.91)	223.30 (1.540)	81.66 (363.39)	325
25	-0.00797 (-54.97)	49.748 (0.3431)	-0.00291 (-0.01295)	18.19 (80.95)	157.20 (1.084)	57.48 (255.79)	551
35	0.03328 (229.55)	43.657 (0.3011)	0.01217 (0.05416)	15.96 (71.02)	159.50 (1.100)	58.33 (259.57)	321
45	-0.05616 (-387.36)	46.351 (0.3197)	-0.02054 (-0.09140)	16.95 (75.43)	153.10 (1.056)	55.99 (249.16)	4
55	0.00514 (35.45)	48.192 (0.3324)	0.00188 (0.00837)	17.62 (78.41)	189.30 (1.306)	69.22 (308.03)	679
65	-0.02698 (-186.09)	53.273 (0.3675)	-0.00987 (-0.04392)	19.48 (86.69)	273.90 (1.889)	100.16 (445.71)	544
75	0.01508 (104.01)	50.757 (0.3501)	0.00551 (0.02452)	18.56 (82.59)	200.80 (1.385)	73.43 (326.76)	897
85	0.04176 (288.04)	53.108 (0.3663)	0.01527 (0.06795)	19.42 (86.42)	273.30 (1.885)	99.94 (444.73)	780
95	-0.05732 (-395.37)	63.968 (0.4412)	-0.02096 (-0.09327)	23.39 (104.09)	270.20 (1.864)	98.81 (439.70	199

TABLE 8.

EFFECTS OF DISTRIBUTION SHAPE ON NORMALIZED LARGEST RESIDUAL MEMBER STRESS DUE TO RANDOM MEMBER-LENGTH IMPERFECTIONS, WITH MEAN IMPERFECTION = 0, AND ENSEMBLE SIZE = 100 CASES EACH

DISTRIBUTION SHAPE	$\frac{\text{MEAN OF MAXIMUM STRESS}}{\sigma_\epsilon}$	$\frac{\text{STD. DEV. OF MAX. STRESS}}{\sigma_\epsilon}$	$\frac{\text{LARGEST MAX. STRESS IN ENSEMBLE}}{\sigma_\epsilon}$
Gaussian	35.9 x 10^6 psi (248 GN/m^2)	5.91 x 10^6 psi (40.9 GN/m^2)	49.7 x 10^6 psi (343 GN/m^2)
Uniform	34.9 x 10^6 psi (241 GN/m^2)	4.94 x 10^6 psi (34.1 GN/m^2)	47.5 x 10^6 psi (328 GN/m^2)

TABLE 9.

TYPE AND LOCATION OF MOST HIGHLY STRESSED MEMBER IN EACH OF 100 CASES FOR ZERO-MEAN GAUSSIAN INITIAL MEMBER LENGTH IMPERFECTIONS (SEED = 325). NUMBERS IN PARENTHESES REPRESENT EXPECTED NUMBER OF OCCURRENCES IN A TRULY RANDOM DISTRIBUTION AMONG ALL TRUSS MEMBERS.

REGION OF TRUSS	MEMBER TYPE				TOTAL
	LONGERONS	BATTENS	CORE DIAGONALS	FACE DIAGONALS	
Corners and Intersections	6 (2.68)	4 (5.35)	17 (1.34)	16 (2.68)	43 (12.04)
Central Transverse Beam	8 (5.80)	0 (5.35)	4 (1.34)	0 (5.80)	12 (18.28)
Upper Transverse Beam	0 (3.12)	0 (3.12)	5 (0.78)	0 (3.12)	5 (10.14)
Lower Transverse Beam	5 (3.12)	0 (2.68)	9 (0.67)	0 (3.12)	14 (9.59)
Upper Starboard Vertical Beam	1 (4.01)	0 (3.57)	4 (0.89)	0 (4.01)	5 (12.49)
Lower Starboard Vertical Beam	2 (4.01)	0 (3.57)	6 (0.89)	0 (4.01)	8 (12.49)
Upper Port Vertical Beam	0 (4.01)	0 (3.57)	4 (0.89)	0 (4.01)	4 (12.49)
Lower Port Vertical Beam	2 (4.01)	0 (3.57)	6 (0.89)	1 (4.01)	9 (12.49)
TOTAL	24 (30.77)	4 (30.77)	55 (7.69)	17 (30.77)	100

TABLE 10.

COMPLIANCE OF TRUSS MEMBERS OF VARIOUS TYPE AND LOCATION DETERMINED BY DETERMINISTIC FINITE ELEMENT ANALYSIS.

REGION OF TRUSS	ELEMENT TYPE			
	Longeron in/lb (mm/kN)	Batten in/lb (mm/kN)	Core Diagonal in/lb (mm/kN)	Face Diagonal in/lb (mm/kN)
Mid-Bay of Central Transverse Beam	1.33×10^{-4} (0.759)	5.14×10^{-4} (2.934)	1.04×10^{-4} (0.594)	2.41×10^{-4} (1.376)
Intersection of Central Transverse and Port Vertical Beams	1.27×10^{-4} (0.725)	1.38×10^{-4} (0.788)	0.864×10^{-4} (0.493)	0.606×10^{-4} (0.346)
Corner of Port Vertical and Lower Transverse Beams	1.34×10^{-4} (0.765)	2.61×10^{-4} (1.490)	0.938×10^{-4} (0.535)	0.836×10^{-4} (0.477)

TABLE 11.

TYPE AND LOCATION OF MEMBERS WHOSE STRESS IS LARGEST IN TWO OR MORE OF THE 100 CASES. ZERO-MEAN GAUSSIAN INITIAL MEMBER-LENGTH IMPERFECTIONS (SEED = 325) WERE USED. NUMBERS IN PARENTHESES REPRESENT EXPECTED NUMBER OF OCCURRENCES IN A TRULY RANDOM DISTRIBUTION AMONG ALL TRUSS MEMBERS.

Members Whose Stress is Largest in 2 or More Cases

REGION OF TRUSS	MEMBER TYPE			TOTAL
	LONGERONS & BATTENS	CORE DIAGONALS	FACE DIAGONALS	
Corners and intersections	1 (1.61)	5 (0.27)	4 (0.53)	10 (2.41)
All other locations	2 (10.70)	8 (1.27)	0 (5.62)	10 (17.59)
Total	3 (12.31)	13 (1.54)	4 (6.15)	20

Members Whose Stress is Largest in 3 or More Cases

REGION OF TRUSS	MEMBER TYPE			TOTAL
	LONGERONS & BATTENS	CORE DIAGONALS	FACE DIAGONALS	
Corners and intersections	1 (0.56)	2 (0.09)	2 (0.19)	5 (0.84)
All other locations	0 (3.75)	2 (0.44)	0 (1.97)	2 (6.16)
Total	1 (4.31)	4 (0.53)	2 (2.16)	7

Members Whose Stress is Largest in 4 or More Cases

REGION OF TRUSS	MEMBER TYPE			TOTAL
	LONGERONS & BATTENS	CORE DIAGONALS	FACE DIAGONALS	
Corners and intersections	1 (0.24)	0 (0.04)	2 (0.08)	3 (0.36)
All other locations	0 (1.61)	0 (0.19)	0 (0.84)	0 (2.64)
Total	1 (1.85)	0 (0.23)	2 (0.92)	3

TABLE 12.

ANGULAR MISALIGNMENT BETWEEN REFERENCE PLANES DUE TO PSEUDO RANDOM INITIAL MEMBER LENGTH IMPERFECTIONS. (ZERO-MEAN GAUSSIAN INPUT WITH EFFECTIVE FIXED-END MEMBER STRAIN OF STANDARD DEVIATION 1×10^{-5}, AND SEED = 325, ENSEMBLE SIZE = 100.)

Ensemble Statistics

Mean Angular Misalignment = 0.00340° (5.93×10^{-5} rad)
Standard Deviation of Angular Misalignment =
0.002217° (3.87×10^{-5} rad)

Ensemble Maximum Value = 0.01027° (1.79×10^{-4} rad) at case 10
Ensemble Minimum Value = 0.000107° (1.87×10^{-6} rad) at case 78

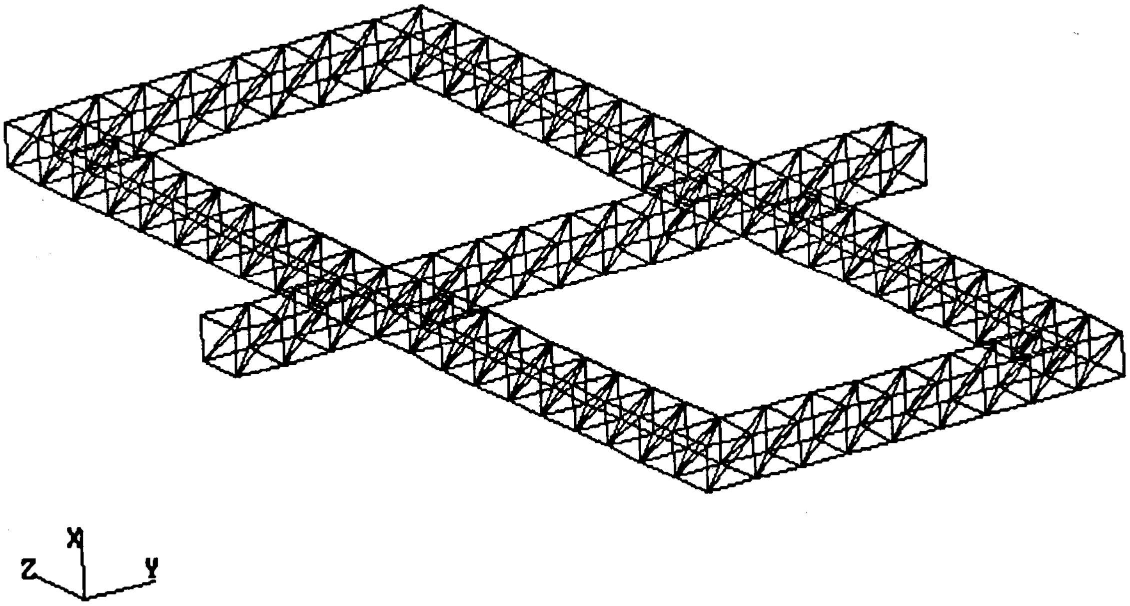


Figure 1. Space Station truss model, overall view.

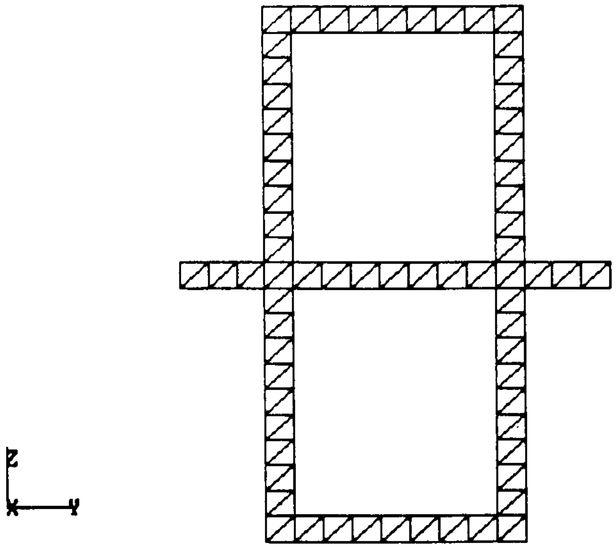


Figure 2. Space Station truss model, Y-Z projection.

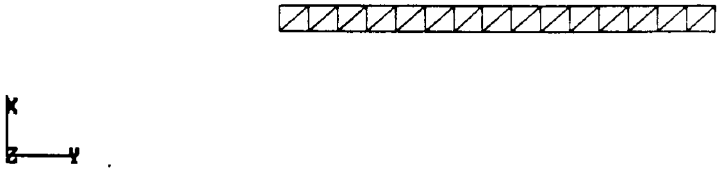


Figure 3. Space Station truss, X-Y projection.

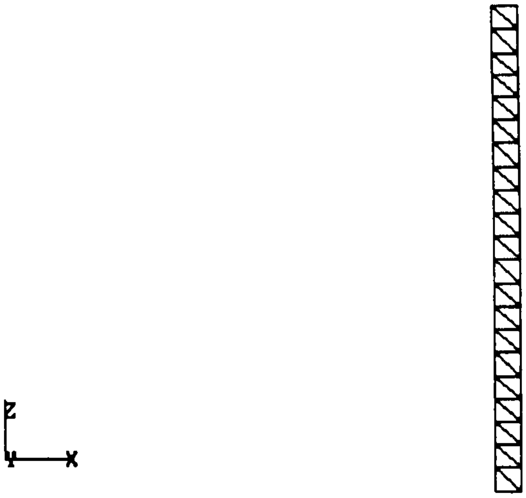


Figure 4. Space Station truss, X-Z projection.

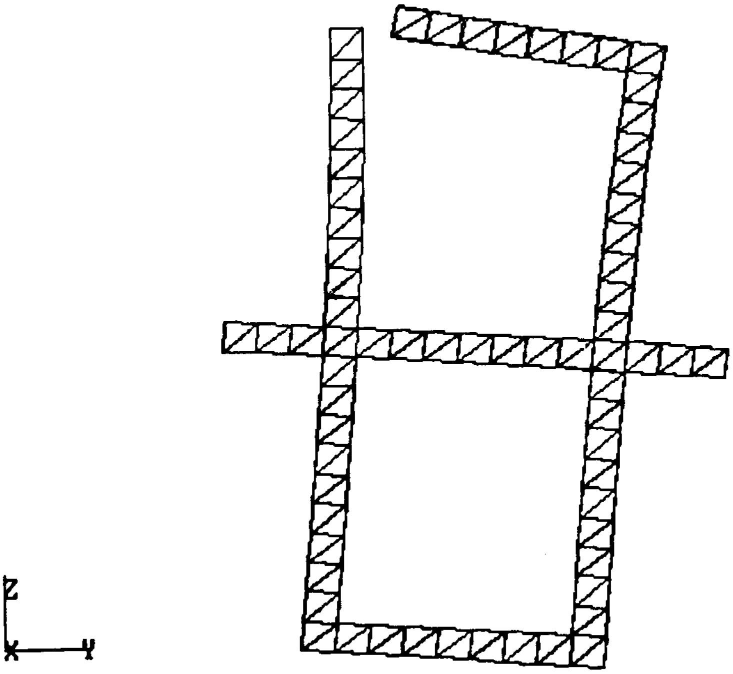


Figure 5. Exaggerated deflections of open-ring truss model due to uniform shortening of outboard longerons around the perimeter of the ring; Y-Z projection.

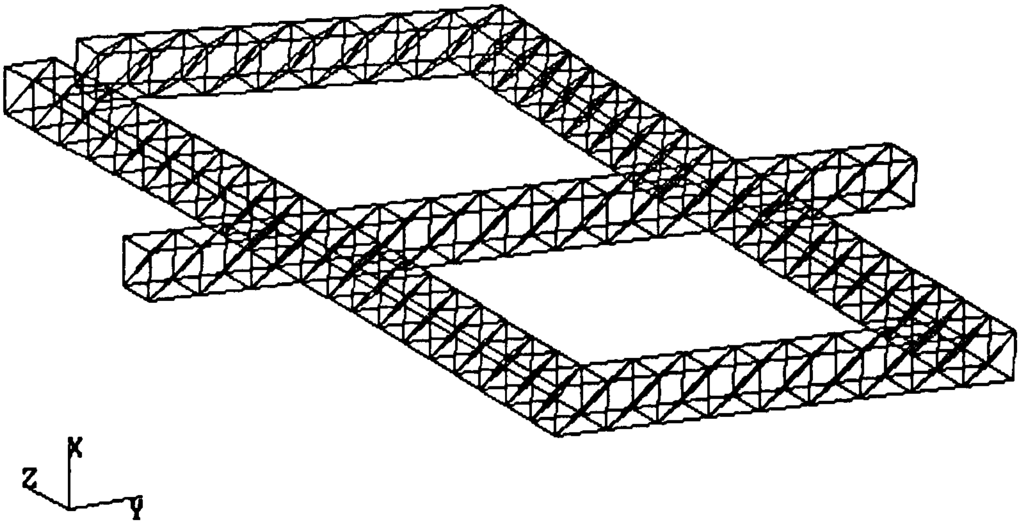


Figure 6. Exaggerated deflections of open-ring truss model due to uniform shortening of outboard longerons around the perimeter of the ring; overall view.

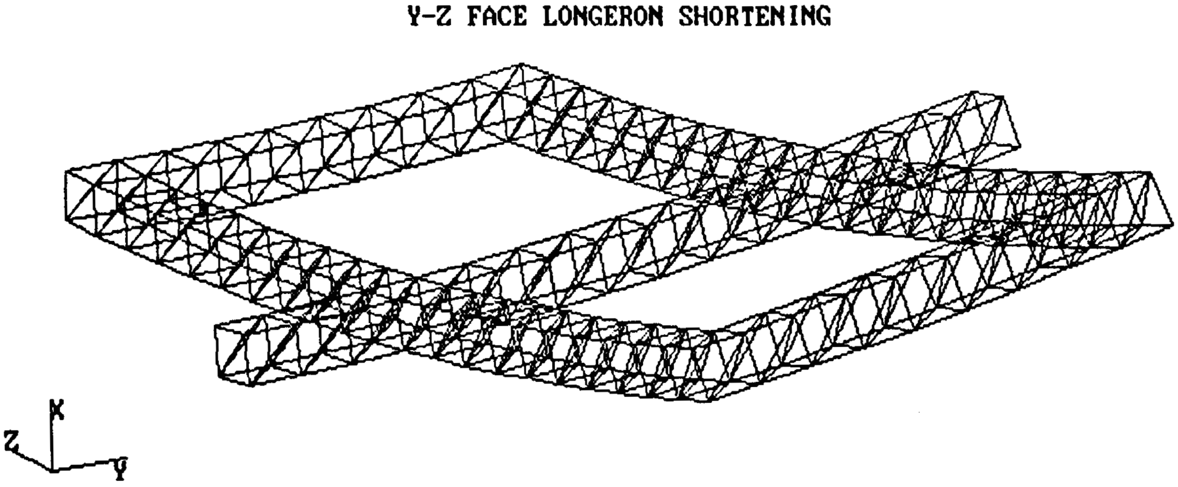


Figure 7. Exaggerated deflections of open-ring truss model due to uniform shortening of all longerons in the upper Y-Z face; overall view.

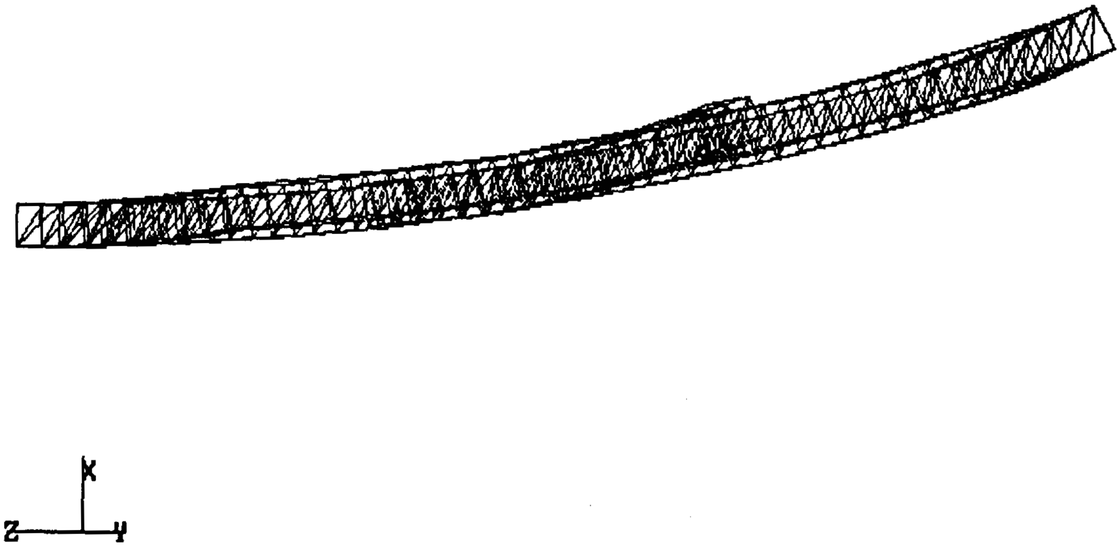


Figure 8. Exaggerated deflections of open-ring truss model due to uniform shortening of all longerons in the upper Y-Z face; view showing surface curvature.

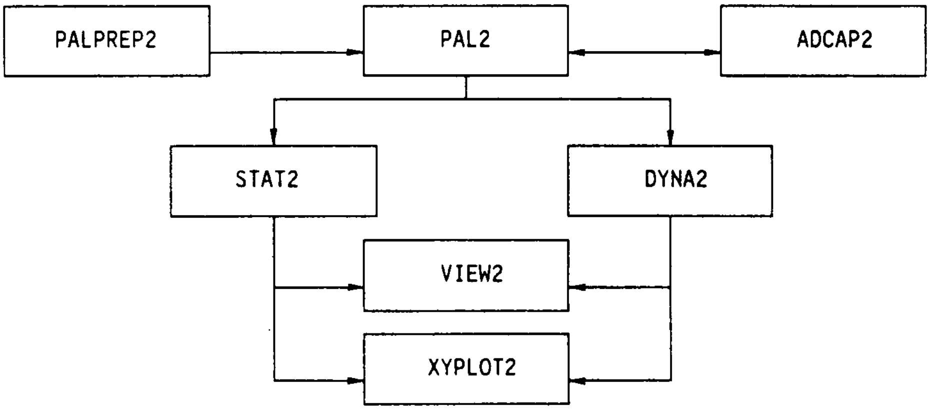


Figure 9. Organization of programs in the MSC/pal 2 software. Arrows indicate the order of solution.

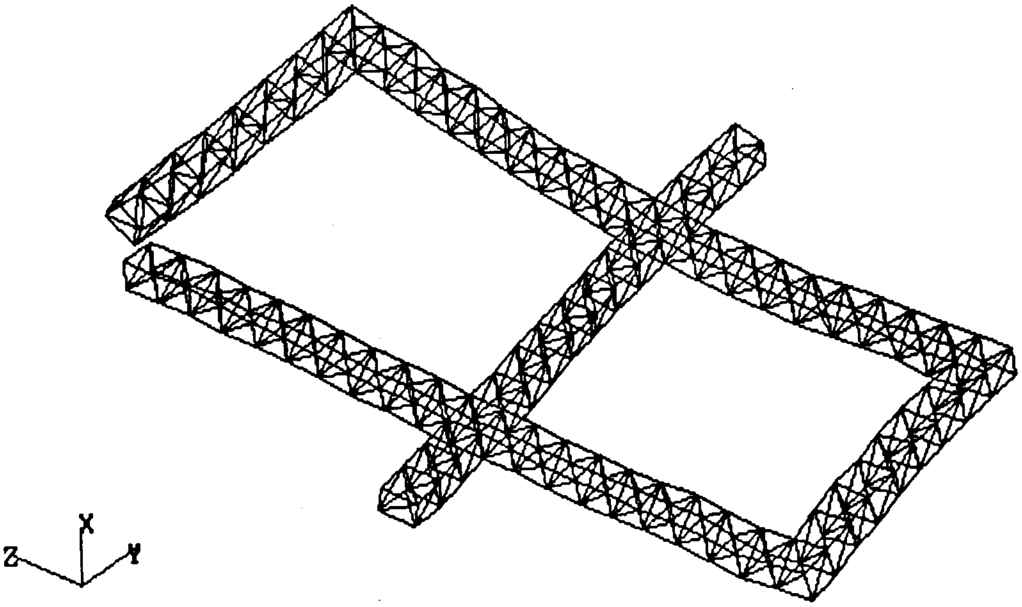


Figure 10. Typical deflections (exaggerated scale) due to random initial member-length imperfections, overall view.

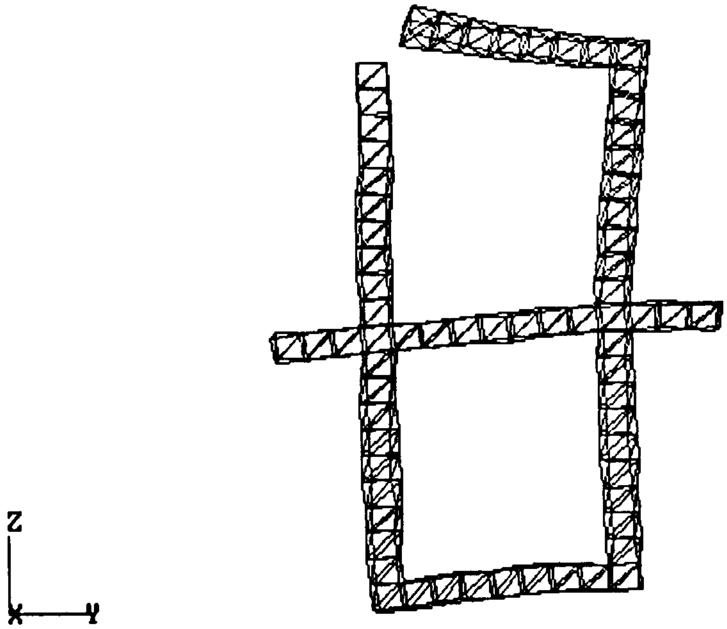


Figure 11. Typical deflections (exaggerated scale) due to random initial member-length imperfections, Y-Z projection..

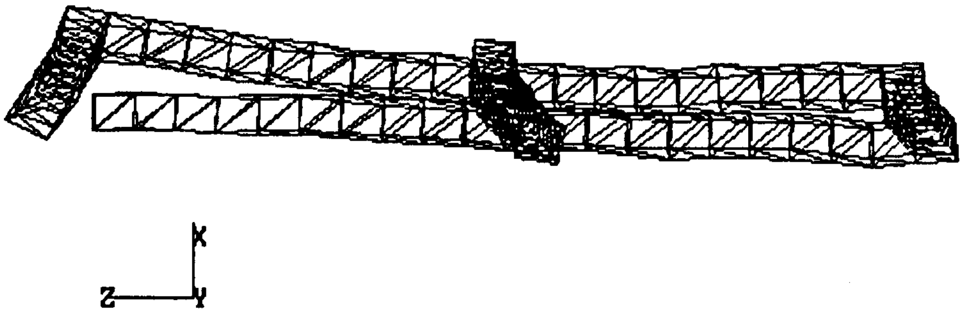


Figure 12. Typical deflections (exaggerated scale) due to random initial member-length imperfections, X-Z projection..

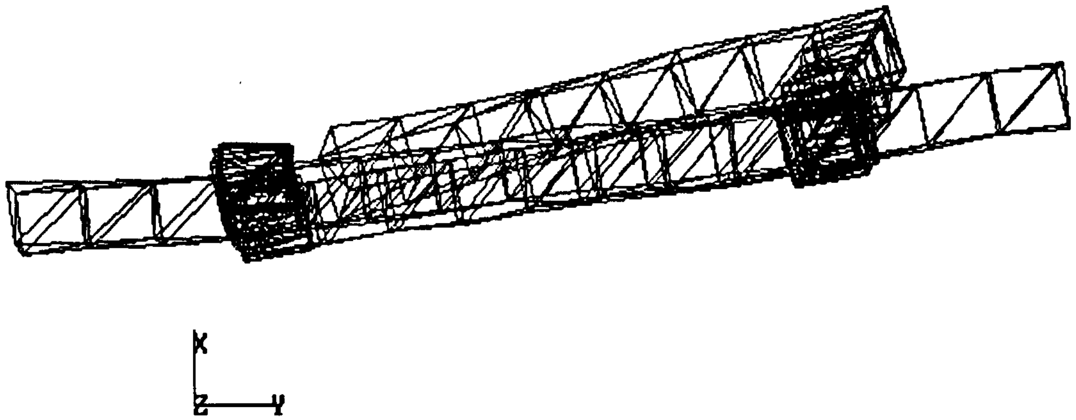
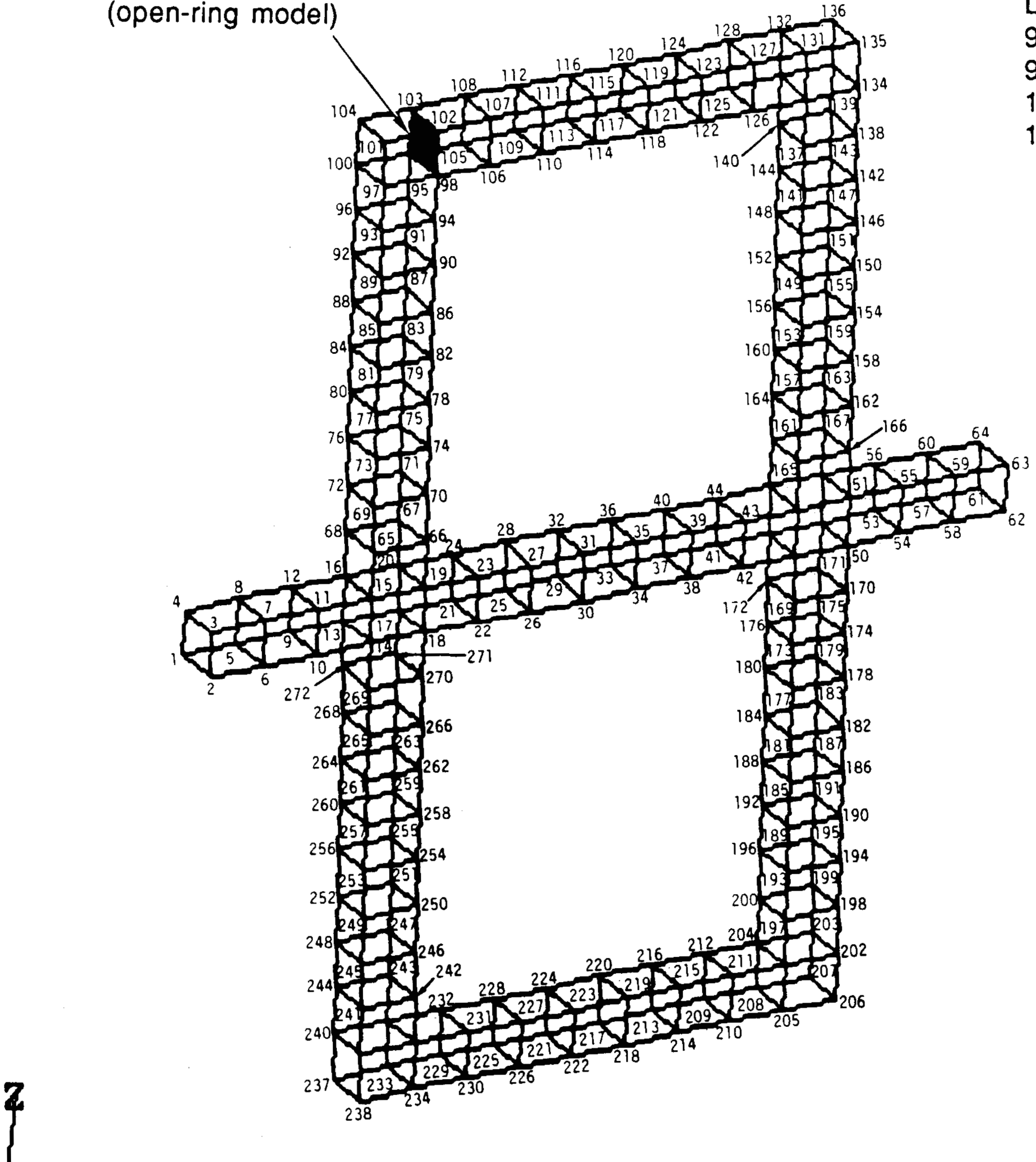


Figure 13. Typical deflections (exaggerated scale) due to random initial member-length imperfections, X-Y projection..

Plane of separation
(open-ring model)



Duplicate nodes:
98 = 273
99 = 274
102 = 275
103 - 276

Figure 14. Node numbering system used for the open-ring truss model.

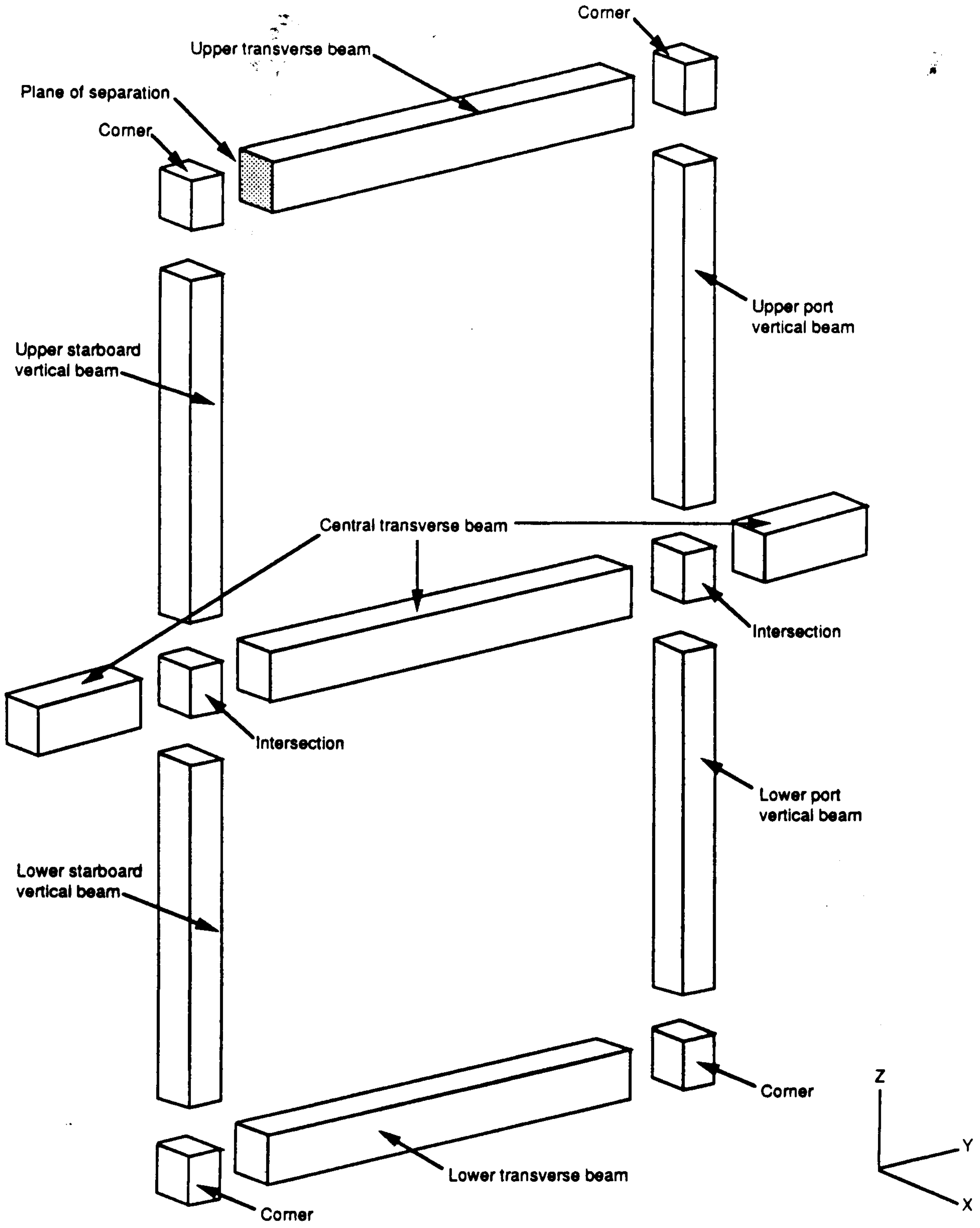


Figure 15. Nomenclature used to describe different regions of the open-ring truss model.

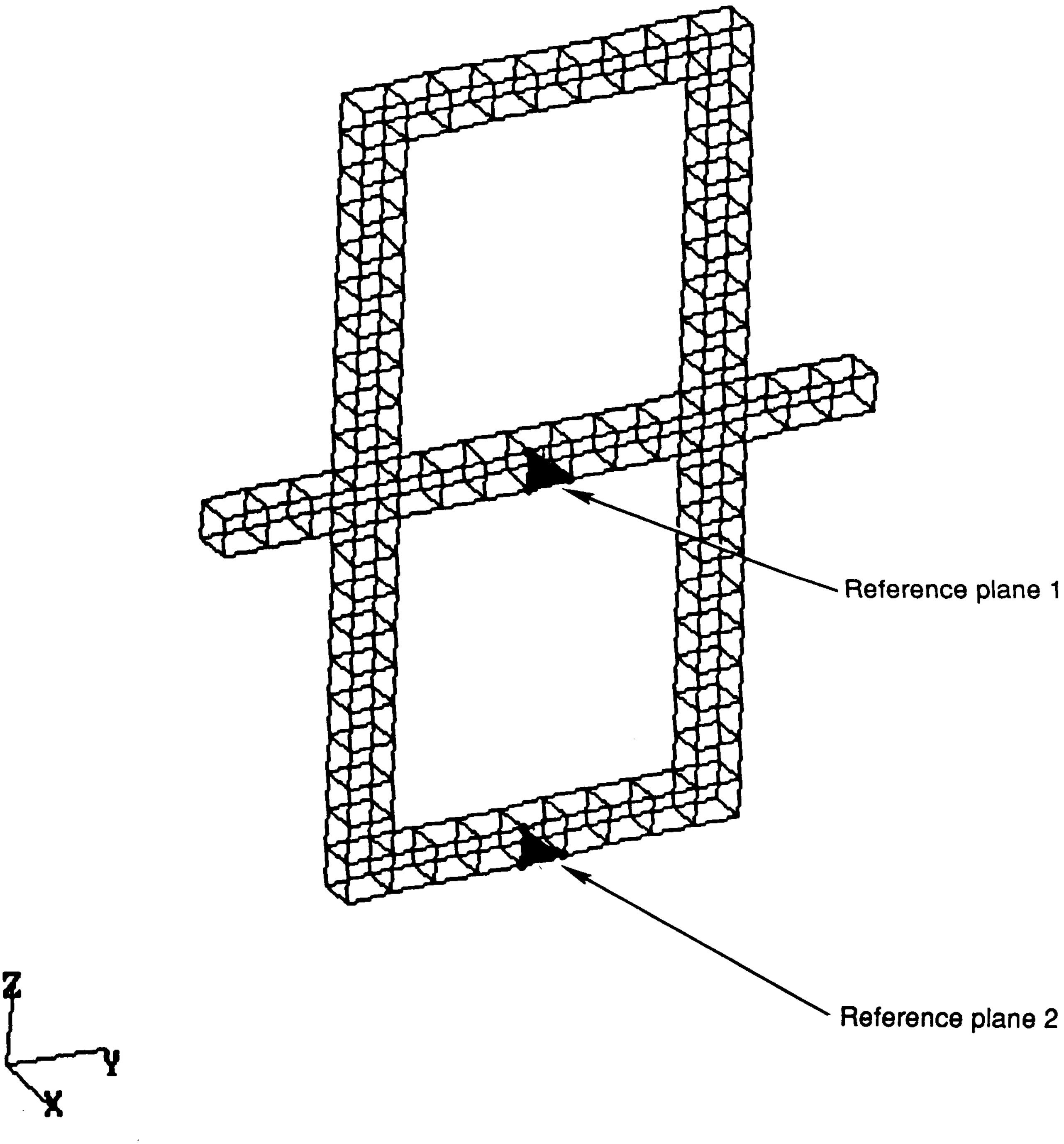


Figure 16. Location of reference planes used for angular misalignment study.

Appendix 1

1. Listing of SPASTAT2.TXT, the PAL2 model generation file used to create the Open-Ring Truss Model discussed in Section 3.1.

2. Listing of RANLOAD.TXT, the STAT2 boundary conditions and random loads file used to create a single case of structural response to random member length imperfections using the thermal analysis capability of MSC/pal 2.

Listing of SPASTAT2.TXT

```
TITLE SPACE STATION TRUSS
NODAL POINT LOCATIONS 1
1 0,0,0 THROUGH 61 0,3000,0 STEP 4
2 200,0,0 THROUGH 62 200,3000,0 STEP 4
3 200,0,200 THROUGH 63 200,3000,200 STEP 4
4 0,0,200 THROUGH 64 0,3000,200 STEP 4
65 200,600,400 THROUGH 101 200,600,2200 STEP 4
66 200,800,400 THROUGH 102 200,800,2200 STEP 4
67 0,800,400 THROUGH 103 0,800,2200 STEP 4
68 0,600,400 THROUGH 104 0,600,2200 STEP 4
105 0,1000,2000 THROUGH 133 0,2400,2000 STEP 4
106 200,1000,2000 THROUGH 134 200,2400,2000 STEP 4
107 200,1000,2200 THROUGH 135 200,2400,2200 STEP 4
108 0,1000,2200 THROUGH 136 0,2400,2200 STEP 4
137 200,2200,1800 THROUGH 165 200,2200,400 STEP 4
138 200,2400,1800 THROUGH 166 200,2400,400 STEP 4
139 0,2400,1800 THROUGH 167 0,2400,400 STEP 4
140 0,2200,1800 THROUGH 168 0,2200,400 STEP 4
169 200,2200,-200 THROUGH 205 200,2200,-2000 STEP 4
170 200,2400,-200 THROUGH 206 200,2400,-2000 STEP 4
171 0,2400,-200 THROUGH 207 0,2400,-2000 STEP 4
172 0,2200,-200 THROUGH 208 0,2200,-2000 STEP 4
209 0,2000,-2000 THROUGH 237 0,600,-2000 STEP 4
210 200,2000,-2000 THROUGH 238 200,600,-2000 STEP 4
211 200,2000,-1800 THROUGH 239 200,600,-1800 STEP 4
212 0,2000,-1800 THROUGH 240 0,600,-1800 STEP 4
241 200,600,-1600 THROUGH 269 200,600,-200 STEP 4
242 200,800,-1600 THROUGH 270 200,800,-200 STEP 4
243 0,800,-1600 THROUGH 271 0,800,-200 STEP 4
244 0,600,-1600 THROUGH 272 0,600,-200 STEP 4
273 200,800,2000
274 0,800,2000
275 200,800,2200
276 0,800,2200

ZERO 1
RA OF ALL

C
C   *********************   CENTRAL TRANSVERSE BEAM ****************************
C
C LONGERONS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
DO CONNECT 1,5 THROUGH 57,61 STEP 4,4
DO CONNECT 2,6 THROUGH 58,62 STEP 4,4
DO CONNECT 3,7 THROUGH 59,63 STEP 4,4
DO CONNECT 4,8 THROUGH 60,64 STEP 4,4

C BATTENS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
DO CONNECT 1,2 THROUGH 61,62 STEP 4,4
DO CONNECT 2,3 THROUGH 62,63 STEP 4,4
DO CONNECT 3,4 THROUGH 63,64 STEP 4,4
DO CONNECT 4,1 THROUGH 64,61 STEP 4,4
```

```
C FACE DIAGONALS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
DO CONNECT 1,6 THROUGH 57,62 STEP 4,4
DO CONNECT 4,7 THROUGH 60,63 STEP 4,4
DO CONNECT 1,8 THROUGH 57,64 STEP 4,4
DO CONNECT 2,7 THROUGH 58,63 STEP 4,4

C CORE DIAGONALS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
DO CONNECT 2,4 THROUGH 62,64 STEP 4,4

C
C   ********************   UPPER STARBOARD VERTICAL BEAM   ********************
C
C LONGERONS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
CONNECT 15 TO 65
CONNECT 19 TO 66
CONNECT 20 TO 67
CONNECT 16 TO 68
DO CONNECT 65,69 THROUGH 97,101 STEP 4,4
DO CONNECT 66,70 THROUGH 94,98 STEP 4,4
DO CONNECT 67,71 THROUGH 95,99 STEP 4,4
DO CONNECT 68,72 THROUGH 100,104 STEP 4,4

C HALF-LONGERONS
MATERIAL PROPERTIES 30.0E6,0,0,0.30,0,1,0
BEAM TYPE 1, 0.18284
CONNECT 98 TO 102
CONNECT 99 TO 103

C BATTENS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
DO CONNECT 65,66 THROUGH 101,102 STEP 4,4
DO CONNECT 66,67 THROUGH 94,95 STEP 4,4
DO CONNECT 67,68 THROUGH 103,104 STEP 4,4
DO CONNECT 68,65 THROUGH 104,101 STEP 4,4

C HALF-BATTENS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.18284
CONNECT 98 TO 99
CONNECT 102 TO 103

C FACE DIAGONALS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
CONNECT 15 TO 66
CONNECT 19 TO 67
CONNECT 16 TO 67
CONNECT 15 TO 68
DO CONNECT 65,70 THROUGH 97,102 STEP 4,4
DO CONNECT 66,71 THROUGH 94,99 STEP 4,4
DO CONNECT 68,71 THROUGH 100,103 STEP 4,4
DO CONNECT 65,72 THROUGH 97,104 STEP 4,4
```

```
C HALF-FACE DIAGONALS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.18284
CONNECT 98 TO 103

C CORE DIAGONALS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
DO CONNECT 66,68 THROUGH 102,104 STEP 4,4
C
C   ******************* UPPER TRANSVERSE BEAM ************************************
C
C LONGERONS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
CONNECT 274 TO 105
CONNECT 273 TO 106
CONNECT 275 TO 107
CONNECT 276 TO 108
DO CONNECT 106,110 THROUGH 130,134 STEP 4,4
DO CONNECT 107,111 THROUGH 131,135 STEP 4,4
DO CONNECT 108,112 THROUGH 132,136 STEP 4,4
DO CONNECT 105,109 THROUGH 129,133 STEP 4,4

C BATTENS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
DO CONNECT 105,106 THROUGH 133,134 STEP 4,4
DO CONNECT 106,107 THROUGH 134,135 STEP 4,4
DO CONNECT 107,108 THROUGH 135,136 STEP 4,4
DO CONNECT 108,105 THROUGH 136,133 STEP 4,4

C HALF-BATTENS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.18284
CONNECT 273 TO 274
CONNECT 273 TO 275
CONNECT 275 TO 276
CONNECT 274 TO 276

C FACE DIAGONALS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
CONNECT 273 TO 107
CONNECT 276 TO 107
CONNECT 274 TO 108
CONNECT 274 TO 106
DO CONNECT 106,111 THROUGH 130,135 STEP 4,4
DO CONNECT 108,111 THROUGH 132,135 STEP 4,4
DO CONNECT 105,112 THROUGH 129,136 STEP 4,4
DO CONNECT 105,110 THROUGH 129,134 STEP 4,4

C CORE DIAGONALS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
DO CONNECT 106,108 THROUGH 134,136 STEP 4,4

C HALF-CORE DIAGONALS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.18284
```

```
CONNECT 273 TO 276

C
C   ********************* UPPER PORT VERTICAL BEAM *****************************
C
C LONGERONS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
CONNECT 130 TO 137
CONNECT 134 TO 138
CONNECT 133 TO 139
CONNECT 129 TO 140
CONNECT 165 TO 47
CONNECT 166 TO 51
CONNECT 167 TO 52
CONNECT 168 TO 48
DO CONNECT 137,141 THROUGH 161,165 STEP 4,4
DO CONNECT 138,142 THROUGH 162,166 STEP 4,4
DO CONNECT 139,143 THROUGH 163,167 STEP 4,4
DO CONNECT 140,144 THROUGH 164,168 STEP 4,4

C BATTENS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
DO CONNECT 137,138 THROUGH 165,166 STEP 4,4
DO CONNECT 138,139 THROUGH 166,167 STEP 4,4
DO CONNECT 139,140 THROUGH 167,168 STEP 4,4
DO CONNECT 140,137 THROUGH 168,165 STEP 4,4

C FACE DIAGONALS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
CONNECT 47 TO 166
CONNECT 133 TO 140
CONNECT 48 TO 167
CONNECT 129 TO 137
CONNECT 47 TO 168
CONNECT 133 TO 138
CONNECT 51 TO 167
DO CONNECT 134,137 THROUGH 162,165 STEP 4,4
DO CONNECT 139,144 THROUGH 163,168 STEP 4,4
DO CONNECT 140,141 THROUGH 164,165 STEP 4,4
DO CONNECT 139,142 THROUGH 163,166 STEP 4,4

C CORE DIAGONALS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
DO CONNECT 138,140 THROUGH 166,168 STEP 4,4

C
C   ************************ LOWER PORT VERTICAL BEAM **************************
C
C LONGERONS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
CONNECT 46 TO 169
CONNECT 50 TO 170
CONNECT 49 TO 171
CONNECT 45 TO 172
DO CONNECT 169,173 THROUGH 201,205 STEP 4,4
```

```
DO CONNECT 170,174 THROUGH 202,206 STEP 4,4
DO CONNECT 171,175 THROUGH 203,207 STEP 4,4
DO CONNECT 172,176 THROUGH 204,208 STEP 4,4

C BATTENS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
DO CONNECT 169,170 THROUGH 205,206 STEP 4,4
DO CONNECT 170,171 THROUGH 206,207 STEP 4,4
DO CONNECT 171,172 THROUGH 207,208 STEP 4,4
DO CONNECT 172,169 THROUGH 208,205 STEP 4,4

C FACE DIAGONALS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
CONNECT 50 TO 169
CONNECT 49 TO 172
CONNECT 45 TO 169
CONNECT 49 TO 170
DO CONNECT 170,173 THROUGH 202,205 STEP 4,4
DO CONNECT 171,176 THROUGH 203,208 STEP 4,4
DO CONNECT 172,173 THROUGH 204,205 STEP 4,4
DO CONNECT 171,174 THROUGH 203,206 STEP 4,4

C CORE DIAGONALS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
DO CONNECT 170,172 THROUGH 206,208 STEP 4,4

C
C   ********************** LOWER TRANSVERSE BEAM *******************************
C
C LONGERONS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
CONNECT 208 TO 209
CONNECT 205 TO 210
CONNECT 201 TO 211
CONNECT 204 TO 212
DO CONNECT 209,213 THROUGH 233,237 STEP 4,4
DO CONNECT 210,214 THROUGH 234,238 STEP 4,4
DO CONNECT 211,215 THROUGH 235,239 STEP 4,4
DO CONNECT 212,216 THROUGH 236,240 STEP 4,4

C BATTENS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
DO CONNECT 209,210 THROUGH 237,238 STEP 4,4
DO CONNECT 210,211 THROUGH 238,239 STEP 4,4
DO CONNECT 211,212 THROUGH 239,240 STEP 4,4
DO CONNECT 212,209 THROUGH 240,237 STEP 4,4

C FACE DIAGONALS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
CONNECT 201 TO 210
CONNECT 204 TO 209
CONNECT 201 TO 212
CONNECT 205 TO 209
DO CONNECT 211,214 THROUGH 235,238 STEP 4,4
```

```
DO CONNECT 212,213 THROUGH 236,237 STEP 4,4
DO CONNECT 211,216 THROUGH 235,240 STEP 4,4
DO CONNECT 210,213 THROUGH 234,237 STEP 4,4

C CORE DIAGONALS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
DO CONNECT 210,212 THROUGH 238,240 STEP 4,4

C
C   *********************** LOWER STARBOARD VERTICAL BEAM **********************
C
C LONGERONS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
CONNECT 239 TO 241
CONNECT 235 TO 242
CONNECT 236 TO 243
CONNECT 240 TO 244
CONNECT 13 TO 272
CONNECT 14 TO 269
CONNECT 18 TO 270
CONNECT 17 TO 271
DO CONNECT 241,245 THROUGH 265,269 STEP 4,4
DO CONNECT 242,246 THROUGH 266,270 STEP 4,4
DO CONNECT 243,247 THROUGH 267,271 STEP 4,4
DO CONNECT 244,248 THROUGH 268,272 STEP 4,4

C BATTENS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
DO CONNECT 241,242 THROUGH 269,270 STEP 4,4
DO CONNECT 242,243 THROUGH 270,271 STEP 4,4
DO CONNECT 243,244 THROUGH 271,272 STEP 4,4
DO CONNECT 244,241 THROUGH 272,269 STEP 4,4

C FACE DIAGONALS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
CONNECT 239 TO 242
CONNECT 240 TO 243
CONNECT 239 TO 244
CONNECT 235 TO 243
CONNECT 18 TO 269
CONNECT 17 TO 272
CONNECT 13 TO 269
CONNECT 17 TO 270
DO CONNECT 241,246 THROUGH 265,270 STEP 4,4
DO CONNECT 244,247 THROUGH 268,271 STEP 4,4
DO CONNECT 241,248 THROUGH 265,272 STEP 4,4
DO CONNECT 242,247 THROUGH 266,271 STEP 4,4

C CORE DIAGONALS
MATERIAL PROPERTIES 30.0E6,0,0,0.3,0,1,0
BEAM TYPE 1, 0.36568
DO CONNECT 242,244 THROUGH 270,272 STEP 4,4

END DEFINITION
```

Listing of RANLOAD.TXT

```
DISPLACEMENTS APPLIED 1
TA 0 103
TY 0 102
TZ 0 102
TY 0 98

ELEMENT TEMPERATURES 1
 1.8312E-005 1
-9.9670E-006 2
-9.4942E-006 3
-6.4162E-006 4
-1.8238E-006 5
 5.0679E-006 6
 1.1492E-005 7
-1.6639E-005 8
 1.3762E-005 9
-1.7099E-005 10
-5.5827E-006 11
 5.4048E-006 12
-2.8700E-006 13
 1.1822E-005 14
-6.2294E-006 15
 1.2887E-006 16
 6.4482E-006 17
-1.1923E-006 18
 6.7795E-006 19
-7.4102E-006 20
-1.0873E-005 21
-4.1320E-007 22
-2.0475E-006 23
 3.5084E-006 24
 2.5676E-006 25
-5.9394E-006 26
 7.5845E-006 27
 5.5871E-007 28
 2.2981E-006 29
 9.6028E-006 30
-9.8894E-006 31
-4.0349E-006 32
 8.5385E-006 33
-3.9827E-006 34
 3.4069E-006 35
-3.0833E-007 36
 1.2269E-005 37
 1.4174E-005 38
-8.2663E-006 39
-1.4710E-006 40
-4.5602E-006 41
 1.0281E-005 42
-2.6864E-006 43
-2.7168E-005 44
-1.1106E-005 45
-6.0953E-006 46
 1.8028E-006 47
-1.0932E-005 48
 5.7029E-006 49
 3.1775E-006 50
-1.6435E-005 51
 1.4594E-006 52
-1.0207E-006 53
```

```
 3.7716E-006 54
-2.2832E-006 55
 3.6691E-006 56
-2.9994E-006 57
-4.2835E-006 58
-2.2983E-005 59
 7.1826E-006 60
 3.2440E-007 61
 4.4915E-006 62
-9.4930E-006 63
 3.3049E-006 64
-7.3410E-006 65
-5.0488E-008 66
 1.9406E-005 67
 6.4906E-006 68
-5.6759E-006 69
-4.9318E-006 70
-9.4141E-006 71
 1.1839E-006 72
 5.3500E-006 73
-2.0842E-006 74
-9.4786E-006 75
-2.9147E-006 76
-4.4594E-006 77
-7.9706E-006 78
 7.7875E-007 79
 9.4427E-006 80
-9.4367E-006 81
 1.8415E-005 82
 2.4119E-005 83
-1.2240E-007 84
-4.0275E-006 85
 1.6885E-005 86
-8.3478E-007 87
 1.7109E-005 88
 5.9330E-006 89
-6.3422E-006 90
 1.9585E-005 91
 8.1314E-006 92
-1.0336E-005 93
-1.0815E-005 94
 2.4516E-005 95
-1.9211E-006 96
-5.1001E-006 97
-8.2609E-006 98
-9.0521E-006 99
 8.5253E-006 100
-2.0003E-005 101
-3.0154E-006 102
 1.8963E-005 103
 1.1885E-006 104
-6.5275E-006 105
 1.2440E-005 106
 1.6147E-005 107
 3.1590E-006 108
-2.4350E-006 109
-1.0100E-005 110
 1.1384E-006 111
-9.4702E-006 112
-7.9790E-006 113
```

```
  1.4490E-005 114
  2.2814E-005 115
  1.3859E-006 116
  2.8665E-005 117
  8.8622E-007 118
  1.6548E-005 119
 -6.9421E-006 120
 -3.1783E-006 121
  1.5932E-006 122
  1.3794E-005 123
  2.8884E-006 124
 -7.2511E-006 125
  8.0677E-006 126
  5.1953E-006 127
 -6.4266E-006 128
 -9.4511E-006 129
  7.3841E-006 130
 -1.3189E-005 131
  1.4056E-005 132
 -1.2453E-005 133
  1.1381E-005 134
 -1.1238E-005 135
  2.3555E-006 136
 -5.8094E-006 137
 -1.1006E-005 138
  1.2232E-005 139
  1.7517E-005 140
 -2.8350E-006 141
  7.4012E-006 142
 -3.6307E-006 143
  4.9145E-006 144
 -1.8777E-005 145
  4.3087E-006 146
 -5.3650E-006 147
  5.8602E-006 148
 -7.6223E-006 149
  6.3806E-006 150
  1.6987E-006 151
 -1.0777E-005 152
 -9.1168E-006 153
 -1.1377E-006 154
  1.4459E-005 155
  5.3623E-006 156
 -1.3558E-005 157
 -2.4705E-006 158
  5.3376E-006 159
  3.6217E-006 160
 -1.4377E-005 161
 -7.7704E-006 162
 -2.9467E-006 163
  8.7969E-006 164
 -1.1082E-005 165
  6.4850E-006 166
  7.1585E-007 167
  8.5782E-006 168
 -9.7359E-006 169
  5.9212E-006 170
 -2.2345E-005 171
 -1.2623E-005 172
  1.2959E-005 173
```

```
  1.5772E-005 174
 -1.5754E-005 175
  1.1006E-005 176
  1.3709E-005 177
 -1.4613E-005 178
 -2.2382E-005 179
 -1.3844E-006 180
 -2.2001E-006 181
 -1.1248E-005 182
 -3.4629E-006 183
 -6.7084E-006 184
 -1.4199E-005 185
  1.1523E-005 186
  1.5515E-005 187
 -2.4335E-005 188
  3.2791E-006 189
  1.9366E-005 190
  7.2980E-006 191
 -1.7657E-005 192
 -3.8646E-006 193
 -5.4340E-006 194
  1.9032E-006 195
  4.1381E-006 196
 -5.3597E-006 197
 -9.7175E-006 198
 -2.9139E-006 199
 -6.1275E-006 200
 -2.6745E-006 201
  1.6816E-005 202
 -9.3831E-006 203
 -7.3069E-006 204
 -9.9181E-006 205
  7.6931E-006 206
 -1.4425E-005 207
 -1.0123E-005 208
 -6.7853E-006 209
  6.5683E-006 210
 -1.3901E-005 211
  3.7831E-006 212
  1.4060E-006 213
 -6.5022E-008 214
  6.2722E-006 215
 -1.0429E-005 216
 -1.3000E-005 217
 -4.4153E-007 218
  6.3188E-006 219
  1.5262E-005 220
 -3.3591E-006 221
  5.8040E-006 222
  2.4871E-005 223
  1.2397E-005 224
 -1.8993E-006 225
 -1.1022E-006 226
 -1.8436E-005 227
 -1.1495E-005 228
 -3.1132E-006 229
  9.9071E-007 230
 -6.2694E-006 231
 -1.6738E-005 232
  1.1754E-006 233
```

```
 1.0090E-005 234
-1.4449E-005 235
 2.5393E-005 236
 1.8986E-005 237
 2.6594E-006 238
-8.9843E-006 239
 2.2163E-005 240
 2.0885E-007 241
-2.5797E-005 242
-5.4873E-006 243
 5.7633E-006 244
-7.2680E-006 245
 1.0115E-005 246
-2.1726E-006 247
-3.2041E-006 248
-3.5125E-006 249
 3.8901E-006 250
 1.5441E-005 251
-3.2420E-006 252
-2.2071E-006 253
-1.4503E-005 254
 9.5110E-006 255
 2.8318E-006 256
 9.2991E-006 257
 1.2110E-005 258
 1.8036E-005 259
-4.4451E-006 260
 6.9958E-006 261
-5.6791E-007 262
-1.4298E-005 263
-9.8695E-006 264
-5.4331E-006 265
-1.1371E-005 266
-6.4935E-006 267
 1.3311E-007 268
 6.3197E-006 269
 1.7932E-005 270
 1.5376E-007 271
 6.7660E-006 272
-6.3207E-006 273
 1.2455E-005 274
-6.0888E-006 275
 3.5884E-007 276
 3.1012E-006 277
-2.6927E-006 278
-4.4536E-007 279
-1.5305E-006 280
-1.3166E-005 281
-5.8093E-007 282
-2.2127E-005 283
 4.3145E-006 284
-1.0022E-006 285
 4.9473E-006 286
-1.5138E-005 287
 1.8533E-006 288
-9.7876E-006 289
-3.6573E-007 290
-1.9398E-006 291
 1.0914E-005 292
-2.3076E-006 293
```

```
 8.8522E-006 294
-1.4744E-005 295
 2.7697E-006 296
-2.5344E-006 297
-6.4256E-006 298
-1.2062E-005 299
-3.8596E-006 300
-1.2124E-006 301
 2.9906E-006 302
 5.5843E-008 303
 8.7721E-006 304
-9.2076E-006 305
-9.5955E-006 306
-1.0863E-007 307
-3.4816E-006 308
-2.6471E-007 309
 2.2440E-006 310
-3.5052E-006 311
 9.2403E-006 312
 2.6103E-006 313
 7.4566E-006 314
-6.1026E-006 315
 3.5222E-006 316
 8.5407E-006 317
-1.5259E-006 318
 9.7605E-006 319
-1.1505E-005 320
-1.0129E-005 321
 1.1222E-005 322
-5.0966E-006 323
 8.0243E-006 324
 1.2619E-005 325
 2.4011E-006 326
 1.0038E-005 327
-3.7856E-006 328
 5.9906E-006 329
-2.1813E-006 330
 1.1447E-005 331
 2.0522E-006 332
 5.4509E-006 333
 1.2108E-006 334
 2.0354E-005 335
-1.2852E-007 336
-2.3026E-005 337
-5.0270E-006 338
-5.7434E-006 339
-1.5376E-005 340
-8.4245E-006 341
-1.3423E-005 342
 9.2920E-006 343
 1.9051E-005 344
-5.3703E-006 345
-1.4835E-005 346
 1.1026E-005 347
 1.8409E-006 348
-4.4189E-006 349
-2.1899E-006 350
 3.0242E-006 351
 2.0451E-006 352
 2.6190E-006 353
```

```
-3.1912E-006 354
-3.8407E-006 355
 3.5182E-006 356
-3.0258E-006 357
-1.3027E-005 358
-4.3515E-006 359
-4.5526E-006 360
-7.6305E-006 361
-5.1964E-007 362
 2.8956E-006 363
 5.8911E-006 364
 4.8430E-006 365
 1.6988E-005 366
-1.3598E-005 367
-1.2512E-005 368
 2.8238E-006 369
 1.6065E-007 370
 8.2297E-006 371
 1.1425E-005 372
 5.5193E-006 373
 7.6008E-006 374
 1.9637E-005 375
-5.1306E-006 376
 4.4459E-006 377
 3.0106E-006 378
 8.1837E-006 379
-7.4042E-006 380
 4.4459E-007 381
 1.4107E-005 382
-7.1559E-006 383
-7.5326E-006 384
-2.3956E-006 385
-1.5722E-005 386
-1.0679E-005 387
-2.3469E-006 388
 2.9538E-006 389
 8.6081E-006 390
 9.2157E-006 391
-1.8642E-006 392
 1.5421E-005 393
-1.4967E-005 394
-6.6263E-006 395
 7.1648E-006 396
-9.7383E-006 397
-6.5048E-006 398
-4.6879E-006 399
 1.3395E-005 400
 1.4410E-005 401
 4.6742E-006 402
-1.7879E-005 403
 7.7568E-007 404
 1.7445E-006 405
 6.6559E-006 406
-6.0502E-006 407
-8.2469E-006 408
-4.5458E-006 409
-2.8039E-006 410
 1.2703E-005 411
 9.6441E-006 412
-1.4385E-006 413
```

```
 -2.5246E-007 414
  1.0630E-005 415
 -4.9162E-006 416
  1.3473E-006 417
  5.0348E-006 418
 -1.2286E-005 419
  1.6506E-006 420
  1.6362E-005 421
 -1.7733E-006 422
  1.5434E-005 423
  6.8211E-007 424
 -2.2244E-005 425
  1.5988E-007 426
  1.4308E-006 427
 -9.7740E-006 428
  3.8870E-007 429
 -1.3241E-005 430
  3.7109E-007 431
 -1.0582E-005 432
  2.3901E-005 433
 -9.6381E-007 434
  1.0745E-005 435
 -7.1737E-006 436
  1.4271E-005 437
 -4.7007E-006 438
  3.5884E-007 439
 -1.9056E-005 440
  1.5711E-005 441
 -1.4683E-005 442
  1.0981E-005 443
 -1.4311E-005 444
  7.4627E-006 445
  1.1146E-005 446
  2.1795E-005 447
 -1.5545E-005 448
 -1.1089E-005 449
  7.4516E-006 450
  5.5641E-007 451
 -1.6572E-005 452
 -4.6461E-006 453
  1.2577E-005 454
  3.9397E-006 455
 -8.4169E-007 456
  4.6947E-006 457
  1.0124E-005 458
 -1.1248E-005 459
  1.0522E-005 460
  5.2953E-006 461
 -4.7700E-006 462
  1.0534E-005 463
 -2.1230E-005 464
  1.5793E-005 465
 -8.2814E-006 466
  1.6778E-005 467
  4.1531E-006 468
  2.3716E-007 469
 -1.9075E-005 470
  1.0059E-005 471
 -1.2950E-005 472
 -5.1192E-006 473
```

```
-1.4789E-005 474
-7.7157E-006 475
-1.1587E-005 476
 7.3059E-006 477
-2.0733E-005 478
-5.4952E-007 479
 4.1631E-006 480
-4.5313E-006 481
 1.8437E-007 482
-1.1245E-007 483
-6.3768E-006 484
-6.2164E-006 485
-3.8811E-006 486
-5.8421E-006 487
-1.6330E-005 488
-6.1562E-006 489
 1.0255E-005 490
-1.8876E-006 491
-7.4294E-006 492
-5.7606E-006 493
-2.6154E-005 494
 1.0895E-005 495
 5.2444E-006 496
-2.5510E-005 497
 2.0060E-005 498
 1.0692E-005 499
 1.5866E-005 500
 4.1197E-006 501
 3.2944E-006 502
 6.5578E-006 503
 5.3729E-006 504
-5.6301E-006 505
 2.7542E-007 506
-4.1339E-006 507
 2.6745E-006 508
-5.7164E-006 509
-1.1313E-005 510
 7.8643E-007 511
-5.9897E-006 512
-2.5013E-006 513
 9.5170E-006 514
 2.2672E-005 515
-1.4145E-005 516
 1.3766E-005 517
-1.0224E-005 518
-6.1082E-007 519
-1.5746E-005 520
-8.4158E-006 521
-3.1583E-005 522
 6.3085E-006 523
-1.0132E-005 524
-2.2981E-006 525
-1.2625E-006 526
-2.3248E-006 527
-4.1881E-006 528
-2.0654E-008 529
 1.8316E-006 530
 1.6645E-005 531
 5.0357E-006 532
-5.2505E-006 533
```

```
-1.3102E-005 534
-1.6657E-005 535
-1.2305E-005 536
 1.0693E-005 537
-7.0702E-006 538
-1.5375E-006 539
 3.1325E-006 540
 7.3921E-006 541
 2.5407E-006 542
 4.8421E-006 543
-1.9228E-005 544
 6.0916E-006 545
-2.8661E-006 546
 8.5264E-006 547
 1.0010E-005 548
 2.3015E-005 549
 1.2344E-005 550
-6.9606E-006 551
-7.3670E-006 552
 2.2835E-005 553
 2.0808E-007 554
-8.3744E-006 555
 1.8601E-005 556
 9.4678E-006 557
-9.7383E-006 558
 5.3809E-006 559
-1.7287E-005 560
 9.8670E-006 561
-5.8137E-008 562
-9.7688E-007 563
 2.1705E-005 564
 1.6315E-005 565
 3.0114E-006 566
 1.4607E-005 567
 1.0368E-006 568
-1.4111E-005 569
 9.1214E-006 570
 1.9718E-006 571
 6.5863E-006 572
-6.1534E-006 573
-1.5994E-005 574
 9.3938E-006 575
-2.1492E-005 576
 7.3390E-006 577
 1.2239E-006 578
 2.1734E-006 579
-3.5296E-006 580
 8.5650E-006 581
-3.4499E-006 582
 1.4585E-005 583
-5.7173E-006 584
-6.9159E-006 585
-5.4598E-006 586
 8.5231E-006 587
-7.1662E-007 588
 1.1694E-005 589
 1.1828E-005 590
-5.0045E-006 591
 1.2460E-005 592
 7.4500E-007 593
```

```
 2.0508E-005 594
-1.1823E-006 595
-3.4199E-006 596
 5.8112E-006 597
 2.7872E-006 598
 1.2906E-005 599
-8.4682E-006 600
-1.2407E-005 601
 7.8029E-007 602
 4.1656E-006 603
-2.3556E-005 604
-2.0842E-006 605
-4.3095E-006 606
-9.1272E-006 607
-5.0870E-006 608
-1.4410E-005 609
-2.5178E-006 610
 9.4308E-006 611
 1.4056E-005 612
-6.1257E-006 613
 2.3854E-006 614
-7.7828E-006 615
 1.2779E-005 616
 4.4071E-006 617
 2.5651E-005 618
-3.3891E-006 619
 1.5248E-005 620
-1.3158E-007 621
 1.2363E-006 622
-1.1793E-005 623
 1.5740E-005 624
-9.3345E-006 625
 1.8158E-005 626
 9.9181E-006 627
 1.0393E-005 628
-4.7067E-006 629
 5.2154E-006 630
 9.5351E-006 631
-6.8142E-006 632
-8.7015E-006 633
-3.1518E-006 634
-1.5917E-006 635
 1.4218E-005 636
-1.3782E-006 637
-1.3081E-007 638
-4.6427E-006 639
 1.4773E-005 640
-5.3606E-006 641
 6.6903E-006 642
 1.0816E-005 643
-6.3001E-006 644
 1.2308E-005 645
 9.0832E-006 646
-1.0458E-005 647
 6.0861E-006 648
-1.5344E-005 649
 1.9664E-005 650
 1.6087E-006 651
-1.2579E-005 652
-8.3159E-006 653
```

```
-4.2157E-006 654
-3.0298E-006 655
-8.1976E-006 656
-7.5344E-007 657
-1.1813E-005 658
-3.1598E-006 659
 1.4163E-005 660
-1.6673E-005 661
 6.2638E-006 662
 1.3885E-005 663
 9.3274E-006 664
 4.5891E-006 665
-1.4442E-005 666
-1.1322E-007 667
-1.0061E-006 668
-9.2944E-006 669
 9.0613E-006 670
-9.7950E-006 671
 1.2407E-005 672
-9.3903E-006 673
-1.1959E-005 674
 1.0930E-005 675
 2.8581E-006 676
 7.1836E-006 677
-1.3999E-007 678
-1.3211E-005 679
-2.7578E-006 680
-4.9145E-006 681
-4.8757E-006 682
 1.3171E-005 683
-8.9728E-006 684
-1.1553E-005 685
 3.3559E-006 686
 4.7161E-006 687
 4.4670E-006 688
-8.3267E-006 689
 9.2979E-006 690
 4.1856E-006 691
 7.7799E-007 692
 8.4180E-006 693
 7.8316E-006 694
 5.6292E-006 695
-1.8390E-005 696
 3.0381E-005 697
-1.5272E-005 698
-6.0392E-006 699
-1.5083E-005 700
-1.9014E-005 701
 8.3581E-006 702
 1.8452E-005 703
-9.7728E-006 704
 1.1107E-006 705
 6.6320E-006 706
 1.0625E-005 707
 3.8728E-006 708
 2.0186E-006 709
-5.2540E-006 710
 7.5509E-006 711
-5.9626E-007 712
-2.4902E-006 713
```

```
 8.6026E-006 714
 6.7795E-006 715
 1.0649E-005 716
 1.9233E-005 717
-1.7545E-005 718
 1.6639E-005 719
-5.3332E-006 720
-6.5721E-006 721
-9.0532E-006 722
-1.0752E-005 723
-1.2852E-005 724
-1.5223E-005 725
-1.1206E-005 726
-1.2297E-005 727
 1.6085E-005 728
 4.9309E-006 729
 9.9445E-009 730
-1.6621E-005 731
 3.9273E-006 732
-2.2101E-005 733
-1.0245E-005 734
-4.2333E-006 735
 4.8206E-006 736
-4.2215E-006 737
 2.5708E-006 738
 5.3835E-006 739
 8.5551E-007 740
 3.1735E-006 741
-1.1307E-005 742
-6.5256E-006 743
 1.0199E-006 744
-2.2581E-006 745
 1.8580E-005 746
-5.2487E-006 747
-6.2294E-006 748
-1.5751E-005 749
-2.8876E-006 750
-8.1098E-007 751
-2.6471E-005 752
 2.6713E-006 753
-2.1333E-005 754
-1.2838E-005 755
 4.0554E-007 756
 1.4136E-005 757
-5.7371E-006 758
-4.0707E-006 759
 2.6832E-006 760
-3.2589E-006 761
-6.3984E-006 762
 1.5904E-005 763
 2.5985E-006 764
 6.3207E-006 765
 8.9717E-006 766
-7.8503E-006 767
-3.4686E-006 768
 1.8604E-006 769
-3.5459E-006 770
-1.1845E-005 771
 1.2074E-005 772
 7.5865E-006 773
```

```
-1.3005E-005 774
 1.6215E-005 775
-1.2230E-005 776
 2.5629E-007 777
-1.7767E-005 778
 2.4563E-006 779
-1.8640E-005 780
-5.8121E-006 781
 5.6867E-006 782
 1.0259E-005 783
 4.8800E-006 784
-4.6222E-006 785
 1.1030E-005 786
 4.3507E-006 787
-1.0822E-005 788
-1.6697E-005 789
 1.7767E-005 790
 2.3605E-005 791
-2.0065E-005 792
 7.2890E-006 793
 1.5556E-005 794
 1.4551E-005 795
 7.2372E-006 796
-2.6396E-006 797
 1.5431E-005 798
 1.2753E-005 799
-1.4085E-005 800
-3.2347E-006 801
-1.8759E-006 802
 3.8053E-006 803
 8.4190E-006 804
 1.6906E-007 805
 8.8699E-007 806
 2.4863E-006 807
 4.7101E-006 808
 1.8210E-005 809
 1.8406E-005 810
 7.7875E-007 811
-7.2342E-006 812
-1.0918E-005 813
-5.5738E-006 814
-1.6060E-005 815
-8.1165E-006 816
 1.7445E-006 817
 2.2181E-005 818
 9.2087E-006 819
 8.2900E-006 820
 5.5890E-006 821
-1.2229E-005 822
 2.0458E-005 823
-9.3369E-006 824
 4.8301E-006 825
 1.0386E-005 826
 8.6948E-006 827
-9.9225E-007 828
 7.8931E-006 829
 1.2528E-005 830
-2.0846E-005 831
 4.5950E-006 832
 1.6007E-005 833
```

```
 1.3519E-005 834
 5.0010E-006 835
 9.8037E-006 836
 1.1903E-005 837
 2.6902E-005 838
-5.1184E-006 839
 5.7047E-006 840
-1.0976E-005 841
-1.7865E-006 842
 9.7040E-006 843
-1.7430E-006 844
 5.7669E-006 845
 2.2212E-006 846
-2.6919E-006 847
-2.5684E-006 848
 2.6571E-006 849
-3.6103E-006 850
-3.1261E-006 851
 7.8067E-006 852
 7.6387E-006 853
 1.2235E-005 854
 1.9063E-006 855
-2.2486E-005 856
-1.8287E-005 857
-1.9717E-005 858
-7.0712E-006 859
-6.1159E-007 860
-5.5291E-006 861
-5.9906E-006 862
 1.3443E-006 863
 1.9146E-005 864
 2.3925E-006 865
 4.0358E-006 866
-3.0374E-007 867
-1.9472E-005 868
 1.6297E-006 869
-6.7363E-006 870
 1.3540E-007 871
 3.7716E-006 872
-6.3611E-007 873
 8.9648E-006 874
-1.3913E-005 875
 3.6291E-006 876
 5.3977E-006 877
-4.8774E-006 878
 4.2545E-007 879
-1.5780E-005 880
 7.7157E-006 881
 4.6197E-006 882
 2.8269E-005 883
 1.0896E-005 884
 5.2777E-006 885
 4.6248E-006 886
-2.7960E-006 887
-1.2494E-006 888
 8.8341E-006 889
-7.9832E-006 890
 5.8611E-006 891
-1.4953E-005 892
 1.2348E-005 893
```

```
 1.2107E-005 894
 9.3713E-006 895
 6.7094E-006 896
 5.8040E-006 897

SOLVE
QUIT
```

Appendix 2

Source Code Listings (in C Language) for:

1. RAN2PAL.C
2. PAL2BIN.C
3. BIN2STAT.C

Source Code Listing for RAN2PAL.C

```
/* RAN2PAL.C - This program gets random strains, either generated internally
 *         or read from a previously generated and stored binary file, and
 *         writes either a sequence of ASCII-form pal2 input files for
 *         STAT2 for each case, or a single binary file for all cases, or
 *         both.
 *
 *      Modified boundary-condition handling                        6/1/88
 *      Changed to truncated-Gaussian approach                      6/1/88
 *
 */



#include <scitech.h>

double stdrnd();
double grand();
void initrand(),abortscan(),genran(),getnewcase();
char *skipblank();
int outform(int),getchange(),getcase(int),getparams(FILE *);

float *elstrain;          /*      Random member strains
                                        (These are regarded as random
                                        member temperature increases in
                                        PAL2 thermal analysis by setting
                                        the coefficient of thermal
                                        expansion equal to one.)*/

int nfiles,nvar,Nnodes;
long seed;
double mean,sdev,tolerance,sigma;
char binname[80];
char bc[200],line[81];

main(argc,argv)
int argc;
char *argv[];

{
        long offset = 0L;
        int i,j,output,ncase,newflag;
        static char infile[80],outfile[80];
        static char filename[80],string1[20],string2[20];
        static char header[256];
        char *ptr;
        FILE *datafile1,*binfile;

    printf("\n\nASTRO AEROSPACE CORPORATION\nSpace Truss");
    printf(" Random Loads Analysis\n");
    printf("(Richard K. Miller, April 7, 1988)\n");
    printf("(John M. Hedgepeth, April 27, 1988)\n");
    printf("MSC/pal 2 Version 3 (C) Preprocessor (Thermal Analysis Version)\n");

/* Get name of input file
 */
    if(argc < 2) {
        printf("\nEnter pathname of existing or desired new input file.  ");
        printf("Default is TRUSS.INP.\n?  ");
        fflush(stdin);
```

```
        gets(line);
        if(!strlen(line))
            strcpy(line,"TRUSS.INP");
    }
    else
        strcpy(line,argv[1]);

    if(argc > 2)
        ncase = atoi(argv[2]);
    else
        ncase = -1;

    if(strchr(line,':') == NULL) {
        getcwd(infile,40);
        strcat(infile,"\\");
        strcat(infile,line);
    }
    else
        strcpy(infile,line);

    strupr(infile);
    if( (ptr = strchr(infile,'.')) == NULL)
        strcat(infile,".INP");
    else {
        if(strcmp(ptr,".BIN") == 0) {
            printf("\007Cannot use .BIN for the extension to an input file."
                                 "  Abort.\n");
            exit(1);
        }
    }

    if ( (datafile1 = fopen(infile,"r+")) )
        newflag = 0;

    else {
        if ( (datafile1 = fopen(infile,"w")) == NULL ) {
            printf("\007Cannot open %s for write.  Abort.\n",infile);
            exit(1);
        }
        else
            printf("\nOpening new input file %s.\n\n",infile);

        newflag = 1;
    }

    strcpy(binname,"NONE");

    strcpy(outfile,infile);
    ptr = strchr(outfile,'.');
    *ptr = '\0';

/*  Read in number of cases, mean, and std dev of member strains, as
        well as beginning seed number for random number generation. */

    if(newflag) {
        getnewcase();
    }
    else {
        newflag = getparams(datafile1);
    }
```

```
/*  List the current values of the parameters and solicit corrections  */

    if(ncase < 0)
        while (getchange() > 0)
            newflag++;

/*  Open binary file for reading if not new truss example.  Then skip over the
 *  header
 */
    if(newflag == 0) {
        if((binfile = fopen(binname,"rb")) == NULL) {
            printf("\007Cannot open %s for read.  Do you want to continue? ",
                                                                    binname);
            fflush(stdin);
            i = getchar();
            if(toupper(i) != 'Y')
                exit(1);
            else {
              newflag = 1;
            }
        }
        else
            fread(header,1,256,binfile);
    }

 /* Determine what kind of output is desired
 */
    if(ncase >= 0)
        output = -1;
    else
        if((output = outform(newflag)) >= 0) {
            strcpy(binname,outfile);
            strcat(binname,".BIN");
            if(newflag) {
                if((binfile = fopen(binname,"wb")) == NULL) {
                    printf("\007Cannot open binary file for writing.  Abort.\n")
                    exit(1);
                }
            }
        }

    if(output <= 0 && ncase < 0)
        ncase = getcase(nfiles);

    if(newflag && output < 0 && strcmpi(binname,"NONE")) {
        remove(binname);
        strcpy(binname,"NONE");
    }

 /*  Write current parameter set to buffer
 */
    memset(header,'\0',256);
    ptr = header;
    ptr += sprintf(ptr,"%d\n",Nnodes);
    ptr += sprintf(ptr,"%d\n",nvar);
    ptr += sprintf(ptr,"%3.d\n",nfiles);
    ptr += sprintf(ptr,"%1E\n",mean);
    ptr += sprintf(ptr,"%1E\n",tolerance);
    ptr += sprintf(ptr,"%1E\n",sdev);
    ptr += sprintf(ptr,"%1d\n",seed);
```

```
    ptr += sprintf(ptr,"%sBC END\n%s\n",bc,binname);

 * Write buffer out to parameter file
 */
    fseek(datafile1,offset,SEEK_SET);
    fwrite(header,1,256,datafile1);

    fclose(datafile1);

/* Write header to binary file if needed
 */
    if(output >= 0 && newflag)
        fwrite(header,1,256,binfile);

/*      Allocate space for member strains
 */
    if((elstrain = (float *)calloc(nvar,sizeof(float))) == NULL) {
        printf("\007Cannot allocate space for strain buffer.  Abort.\n");
        exit(1);
    }

/*      Set pointer in random number generator if we are going to generate a
 *      new case.
 */
    if(newflag)
        initrand(seed);

/*      Begin loop to generate cases   */

    for ( i = 0; i < nfiles; i++) {
/*      Get random member strains
 */
        if(newflag) {
            printf("\nGenerating case number %d ",i);
            genran(elstrain,mean,tolerance,sdev,nvar);
        }
        else {
            fread(elstrain,sizeof(float),nvar,binfile);
        }

/*  If load files are desired open file and write displacement boundary
 *  conditions
 */
        if(output <= 0 && (i==ncase || ncase < 0)) {
            if(!newflag)
                printf("\nReading case number %d from binary file  ",i);

            strcpy(filename,outfile);
            strcat(filename,".");
            if(i <= 100)
                sprintf(string2,"%2.2d",i);
            else
                sprintf(string2,"%3.3d",i);
            strcat(filename,string2);

            if((datafile1 = fopen(filename,"w")) == NULL) {
                printf("\007\n\nCannot open %s.  Abort\n",filename);
                exit(1);
            }
```

```
            fprintf(datafile1,"DISPLACEMENTS APPLIED 1\n");
            fprintf(datafile1,"%s\n",bc);

/*      Write pal2 header before temperature increases for each member  */

            fprintf(datafile1,"ELEMENT TEMPERATURES 1\n");

/*      Write member temperature increases in pal2 load file    */

            for( j=0; j<nvar; j++ ) {
                fprintf(datafile1,"%12.4E %d\n",*(elstrain + j),j+1);
            }

            fprintf(datafile1,"\nSOLVE\nQUIT\n");

            fclose (datafile1);
        }

/*  Write binary values in binary file if desired.
 */
        if(output >= 0 && newflag)
            fwrite(elstrain,sizeof(float),nvar,binfile);

    }
    if(output >= 0 && newflag)
        fclose(binfile);
}

/***********************************************************
 */

char *skipblank(char *buffer,int num,FILE *stream)
{
    char *ptr;
    int flag;
    for(flag = 1; flag;) {
        if(fgets(buffer,num,stream) == NULL) {
            printf("\007\nEnd of file encountered.");
            exit(1);
        }
        for(ptr = buffer; *ptr; ptr++)
           if(!isspace(*ptr)) {
                flag = 0;
                break;
            }
    }
    return ptr;
}

/*************************************************************
 */
void getnewcase()
{
        int i;

        fflush(stdin);

        printf("\n\tNumber of nodes in truss = ");
        scanf(" %d",&Nnodes);
```

```
        printf("\n\tNumber of members in truss = ");
        scanf("%d",&nvar);

        printf("\n\tNumber of files to generate = ");
        scanf("%d",&nfiles);

        printf("\n\tMean strain = ");
        scanf("%lf",&mean);

        printf("\tStandard deviation of untruncated distribution = ");
        scanf("%lf",&sdev);

        printf("\n\tMaximum strain deviation from mean = ");
        scanf("%lf",&tolerance);

        printf("\n\tBeginning number (seed) for random number generation = ");
        scanf("%ld",&seed);

        printf("\n\n\tEnter displacement boundary conditions to use "
                       "in STAT2.\n\tEnd with blank line.\n");
        *bc = '\0';
        fflush(stdin);

        while(1) {
            gets(line);
            if(!strlen(line))
                break;
            strcat(line,"\n");
            strcat(bc,line);
        }
}

/*************************************************************
 */

int getparams(FILE *file)
{
    char *ptr;
    int i;

        ptr = skipblank(line,81,file);
        if(sscanf(ptr,"%d",&Nnodes)!= 1)
            abortscan("Number of nodes");
        ptr = skipblank(line,81,file);
        if(sscanf(ptr,"%d",&nvar)!= 1)
            abortscan("Number of members");
        ptr = skipblank(line,81,file);
        if(sscanf(ptr,"%d",&nfiles) != 1)
            abortscan("Number of files");
        ptr = skipblank(line,81,file);
        if(sscanf(ptr,"%lf",&mean) != 1)
            abortscan("Mean strain");
        ptr = skipblank(line,81,file);
        if(sscanf(ptr,"%lf",&tolerance)!= 1)
            abortscan("Distribution type");
        ptr = skipblank(line,81,file);
        if(sscanf(ptr,"%lf",&sdev)!= 1)
            abortscan("Strain deviation");
        ptr = skipblank(line,81,file);
        if(sscanf(ptr,"%ld",&seed)!= 1)
```

```
            abortscan("Seed number");

        *bc = '\0';
        while(1) {
            ptr = skipblank(line,81,file);
            if(!strcmpi(ptr,"BC END\n"))
                break;
            strcat(bc,line);
        }

        ptr = skipblank(line,80,file);
        if(sscanf(ptr,"%s",binname) != 1)
            abortscan("Binary file name");

        if(strcmpi(binname,"NONE") )
            return 0;
        else
            return 1;
    }

/************************************************************
 */
int getchange()
{
        int i,j;
        char chr,*ptr;
        double factor;

        if((factor = tolerance/(sqrt(2.)*sdev)) > 10)
            sigma = 1.;
        else if(factor > .01)
            sigma = 1. -  2.*factor*exp(-factor*factor)/(sqrt(PI)*erf(factor));
        else
            sigma = 2.*factor*factor/3.;

        sigma = sdev*sqrt(sigma);

        printf("\n\tParameter values to be used are as follows:\n");

        printf("\n\t 1. Number of files to generate = %d\n",nfiles);
        printf("\t 2. Mean member strain = %le\n",mean);
        printf("\t 3. Standard deviation of truncated distribution = %le\n",
                                                                     sdev);
        printf("\t 4. Maximum deviation of strain from mean = %le\n",tolerance);
        printf("\t\t(Actual standard deviation = %le)\n",sigma);
        printf("\t 5. Random number seed = %ld\n",seed);
        printf("\t 6. Number of nodes in truss = %d\n",Nnodes);
        printf("\t 7. Number of members in truss = %d\n",nvar);
        printf("\t 8. Displacement boundary conditions are:\n\n");

        for(ptr = bc, i = 1; *ptr; ptr++) {
            if(i) {
                printf("\t    ");
                i = 0;
            }
            putchar((int)*ptr);
            if(*ptr == '\n')
                i = 1;
        }
```

```
    printf("\n\tEnter item number for corrections to data above\n");
    printf("\t(Zero for no further corrections)  ? ");

    fflush(stdin);
    while(1) {
        gets(line);
        if(strlen(line) == 0) {
            i = 0;
            break;
        }
        if(sscanf(line,"%d",&i) == 1)
            if(i < 9 && i >= 0)
                break;
        putchar('\007');
    }
    putchar('\n');

switch(i) {
    case 1:
        printf("\tnfiles = %d\n\tnew value = ",nfiles);
        fflush(stdin);
        scanf("%d",&nfiles);
        while(nfiles > 1000 || nfiles < 1) {
            printf("\007\nValue must be in range (1, 1000).");
            printf("  Enter again.  ");
            scanf("%d",&nfiles);
        }
        break;

    case 2:
        printf("\tmean = %lf\n\tnew value = ",mean);
        fflush(stdin);
        scanf("%le",&mean);
        break;

    case 3:
        printf("\tsdev = %lf\n\tnew value = ",sdev);
        fflush(stdin);
        scanf("%le",&sdev);
        break;

    case 4:
        printf("\ttolerance = %lf\n\tnew value = ",tolerance);
        fflush(stdin);
        scanf("%le",&tolerance);
        break;

    case 5:
        printf("\tseed = %ld\n\tnew value = ",seed);
        fflush(stdin);
        scanf("%ld",&seed);
        break;

    case 6:
        printf("\tNnodes = %d\n\tnew value = ",Nnodes);
        fflush(stdin);
        scanf("%d",&Nnodes);
        break;

    case 7:
```

```
            printf("\tnvar = %d\n\tnew value = ",nvar);
            fflush(stdin);
            scanf("%d",&nvar);
            break;

        case 8:
            printf("\n\n\tEnter displacement boundary conditions to use "
                        "in STAT2.\n\tEnd with blank line.\n");
            *bc = '\0';
            fflush(stdin);

            while(1) {
                gets(line);
                if(!strlen(line))
                    break;
                strcat(line,"\n");
                strcat(bc,line);
            }
            break;

        default:
            break;
    }

    return i;
 }

/**********************************************************
 */

void abortscan(char *string)
{
    printf("\007\nFailed input scan at %s.\n",string);
    exit(1);
}

/************************************************************
 */
int outform(int flag)
{
    int chr;

    if(!flag) {
        printf("\n\nA previously generated binary file of random values "
                    "will be used.\n");
        printf("Separate pal2 load files will be written.\n\n");
        return (-1);
    }
    else {
        printf("\n\nThis is a new case.  "
                            "New random values will be generated.\n\n");
        printf("If you want to write separate pal2 load files,      enter A.\n")
        printf("If you want to write only a compressed binary file, "
                                                    "enter B.\n");
        printf("If you want to write both types of files,           "
                                                    "enter C.\n");
    }
    printf("    ? ");

    fflush(stdin);
```

```
    for(chr=0; ;putchar('\007')) {
        chr = toupper(getchar());
        if(chr >= 'A' && chr <= 'C')
            break;
        }
    if(chr == 'A')
        return (-1);
    else if(chr == 'B')
        return (1);
    else
        return (0);
}

/****************************************************************
 */
 int getcase(int N)
 {
    int ncase;
    char buffer[10];

    printf("\nWhat is the desired case number (0,1,...,%d)?\n",N-1);
    printf("Enter a negative number if all cases are desired.  ?");
    fflush(stdin);
    while(1) {
        gets(buffer);
        if(sscanf(buffer," %d",&ncase) == 1)
            break;
        putchar('\007');
    }
    return ncase;
}

/****************************************************************
 */
void genran(array,mean,tolerance,sdev,nvar)

/*      This subroutine generates an array of nvar independent
 *      random member strains with a truncated normal distribution.  The
 *      mean value "mean", standard deviation "sdev", and maximum
 *      deviation of strain from mean "tolerance".
 *      This is accomplished by calls to the C-Language subroutines
 *      grand(), and initrand() which are listed below.
 */

float array[];
double mean,sdev,tolerance;
int nvar;

{

    int i,prtcase;
    double p,r,factor,sqrt2,ratio;

    prtcase = 25;
    sqrt2 = sqrt(2.);
    ratio = tolerance/(sqrt2*sdev);
    factor = erf(ratio);

    printf("                Completed element     1");
```

```
    for ( i=0; i < nvar; i++ )  {
        r = 1.0 - 2.0 * grand();
        if(ratio < 0.01)
            p = r;
        else
            p = sqrt2*inv_erf(factor*r);

        array[i] = (float)(mean + sdev * p);

        if ( i % prtcase == (prtcase - 1) || i == ( nvar - 1 ) )
                   printf("\b\b\b\b\b%5.d",i+1);
    }

    printf("\t\t\t\t\t\t\t\t\r");

}

/***********************************************************************
 */


/*      GRAND1.C        Random number generator based on the msc library
 *                      functions "rand" and "srand".
 */

#include <scitech.h>

double grand();
int rand();
void srand();

/**********************************************************************/

void initrand(seed)
long seed;

{

        srand(seed);

}

/**********************************************************************/

double grand()

{

        double x;
        int n;
        double divisor = 32768.0;

        n = rand();

        while ( n < 1 || n > 32767 ) {
            n = rand();
        }

        x = n;
```

```
        return(x/divisor);

}

/**************************************************************************/
```

Source Code Listing for PAL2BIN.C

```
/* PAL2BIN.C - Reads pal2 displacement and stress output files and writes
 *          pertinent quantities for trusses in binary files.  Each binary
 *          file has a 256-byte header in ASCII form which contains vital
 *          data.  The displacement or stress data follow as streams of
 *          four-byte float quantities.  The first group in the case of
 *          displacements contains the coordinates of the truss nodes.
 *
 *                                  J. M. Hedgepeth     4/29/88
 *
 */

#include <scitech.h>

int datatype;
char *headbuff;
float *datarray;

int gettask();
void initbin(),extract(),fillarray();
char *skipcomment();

main(argc,argv)
int argc;
char *argv[];
(

/*      Declaration of variables */

    int i,ncases,Nnodes,Nstrut,Ncase,Ncomp,option;
    long offset;
    static char binname[80],pal2name[80];
    FILE *binfile,*pal2file;

    headbuff = (char *)malloc(256*sizeof(char));
    Ncase = 0;
    ncases = 0;

/*      Print header, then obtain information on file to be compressed
 *      and rewritten.
 */

    printf("\nASTRO AEROSPACE CORPORATION\n");
    printf("Pal2 Data Output Compressor and Expander for Trusses\n");
    printf("(Richard K. Miller, April 21, 1988)\n");
    printf("(John M. Hedgepeth, April 29, 1988)\n");

    if(argc == 4) (
        ncases = 1;
        option = 'A';
        strcpy(binname,argv[2]);
        strupr(binname);

        strcpy(pal2name,argv[3]);
        strupr(pal2name);
        if((datatype = toupper((int)*argv[1])) != 'D' && datatype != 'S') (
            printf("\007\nUnacceptable command line.  Do you want to continue "
                                                  "with dialog input?  ");
            fflush(stdin);
            if(toupper(getche()) != 'Y')
                exit(1);
```

```
                else {
                    ncases = 0;
                    putchar('\n');
                }
            }
        }
        if(!ncases) {
            if((option = gettask(binname)) == 'I') {
                initbin(binname);
                exit(0);
            }

            if(option == 'A') {
                printf("\nPathname of pal2 data file?  ");
                fflush(stdin);
                gets(pal2name);
                strupr(pal2name);
                printf("\tNumber of cases in this file?  ");
                while(scanf(" %d",&ncases) != 1);
            }
        }

        if((binfile = fopen(binname,"rb+")) == NULL) {
            printf("\007\nCannot open %s for read.  Abort.\n",binname);
            exit(1);
        }

        fread(headbuff,sizeof(char),256,binfile);
        sscanf(headbuff," %d %d %d",&Nnodes,&Nstrut,&Ncase);

        if(datatype == 'S')
            Ncomp = Nstrut;
        else
            Ncomp = 3*Nnodes;

        if((datarray = (float *)calloc(Ncomp,sizeof(float))) == NULL) {
            printf("\007\nCannot allocate space for data array.  Abort.\n");
            exit(1);
        }

        if(option == 'E') {
            extract(binfile);
            exit(0);
        }

        if((pal2file = fopen(pal2name,"r")) == NULL) {
            printf("\007\nCannot open file %s for read.  Abort.\n",pal2name);
            exit(1);
        }

        fseek(binfile,0L,SEEK_END);


    /*      Read input displacements from PAL-generated file
     */

        for ( i = 0; i < ncases; i++ ) {
            fillarray(pal2file);
            fwrite(datarray,sizeof(float),Ncomp,binfile);
        }
```

```
/*      Rewrite header with new Ncase
 */

    Ncase += ncases;

    sprintf(headbuff,"%d\n%d\n%3.d",Nnodes,Nstrut,Ncase);
    i = strlen(headbuff);
    *(headbuff + i) = '\n';
    fseek(binfile,0L,SEEK_SET);
    fwrite(headbuff,sizeof(char),256,binfile);

    fclose(binfile);
    fclose(pal2file);

}

/*********************************************************************
 */
int gettask(char *name)
{
    int chr;

    fflush(stdin);

    printf("\n\nSelect one of each of the following options:\n\n");
    printf("Enter i:  to initialize the binary storage file.\n");
    printf("      a:  to add output from a pal2 data file.\n");
    printf("      e:  to extract an ASCII file  for a particular case.\n");
    printf("?  ");

    while((chr = toupper(fgetc(stdin))) != 'I' && chr != 'A' && chr != 'E')
        putchar('\007');

    printf("\n\nEnter d:  to deal with nodal displacements.\n");
    printf("      s:  to deal with strut stresses.\n");
    printf("?  ");

    fflush(stdin);
    while((datatype = toupper(fgetc(stdin))) != 'D' && datatype != 'S')
        putchar('\007');

    fflush(stdin);
    if(chr == 'I') {
        printf("\n\nWhat is the ASCII-form configuration file with extension "
                                             ".INP?\n");
    }
    else {
        printf("\n\nWhat is the binary data file with extension ");
        if(datatype == 'D')
            printf(".DIS?\n");
        else
            printf(".STR?\n");
    }
    printf("Pathname =  ");
    gets(name);

    return chr;
}
```

```
/***************************************************
 */
void initbin(char *name)
{
    int i,j,Ncase;
    char *buffer,*ptr;
    long length;
    FILE *nfile;

    if( (nfile = fopen(name,"r")) == NULL) {
        printf("\007\nCannot open input file %s for read.  Abort.\n",name);
        exit(1);
    }
    length = filelength(fileno(nfile));
    if((buffer = (char *)malloc(length)) == NULL) {
        printf("\007\nCannot allocate space for input parameter buffer.\n");
        exit(1);
    }
    fread(buffer,sizeof(char),length,nfile);
    fclose(nfile);

    memmove(headbuff,buffer,256);

    sscanf(buffer," %d %d %d",&i,&j,&Ncase);

    Ncase = 0;
    sprintf(buffer,"%d\n%d\n%3.d\n",i,j,Ncase);
    i = strlen(buffer);
    memmove(headbuff,buffer,i);

    if((ptr = strchr(name,'.')) != NULL)
        *ptr = '\0';
    if(datatype == 'D')
        strcat(name,".DIS");
    else
        strcat(name,".STR");

    if((nfile = fopen(name,"r")) != NULL) {
        printf("\007\nFile %s already exists.  Do you want to write over it?  "
                                                                   ,name);
        fflush(stdin);
        fclose(nfile);
        if(toupper(getche()) != 'Y')
            exit(1);
    }
    if((nfile = fopen(name,"wb")) == NULL) {
        printf("\007\nCannot open %s for write.  Abort.\n",name);
        exit(1);
    }
    fwrite(headbuff,sizeof(char),256,nfile);
    fclose(nfile);

    exit(0);
}

/*************************************************************
 */
void extract(FILE *binfile)
{
    int Nnodes,Nstrut,Ncase,n,Ncomp,i;
```

```
long offset;
float x,y,z,*ptr;
static char line[80];
FILE *outfile;

sscanf(headbuff," %d %d %d",&Nnodes,&Nstrut,&Ncase);

printf("\nThere are %d cases available.  "
        "What case is to be extracted? (0,1,...,%d) ",Ncase,Ncase - 1);
fflush(stdin);
while(1) {
    gets(line);
    if(sscanf(line," %d",&n) != 1)
        n = -1;
    if(n < 0 || n >= Ncase)
        putchar('\007');
    else
        break;
}

printf("\n\nEnter pathname of output file.  ?");
gets(line);
strupr(line);

if((outfile = fopen(line,"w")) == NULL) {
    printf("\007\nCannot open %s for write.  Abort.\n",line);
    exit(1);
}

offset = 256L;
if(datatype == 'D')
    Ncomp = 3*Nnodes;
else
    Ncomp = Nstrut;

offset += (long)n*Ncomp*sizeof(float);

fseek(binfile,offset,SEEK_SET);
if(fread(datarray,sizeof(float),Ncomp,binfile) != Ncomp)
    printf("\007\nWARNING: Binary file read incorrect.\n\n");
fclose(binfile);

ptr = datarray;

if(datatype == 'D') {
    fprintf(outfile,"         TRUSS DISPLACEMENT COMPONENTS FOR CASE %d\n",
                                                                  n);
    fprintf(outfile,"\n\n Node      x           y           z\n\n");

    for(i=1; i <= Nnodes; i++) {
        x = *ptr++;
        y = *ptr++;
        z = *ptr++;
        fprintf(outfile,"%4.d   %11.3E %11.3E %11.3E\n",i,x,y,z);
    }
}
else {
    fprintf(outfile,"         TRUSS STRUT STRESSES FOR CASE %d\n",n);
    fprintf(outfile,"\n\n Strut       Stress\n\n");
```

```
        for(i=1; i <= Nstrut; i++) {
            x = *ptr++;
            fprintf(outfile,"%4.d    %11.3E\n",i,x);
        }
    }
    fclose(outfile);
}

/****************************************************************
 */
void fillarray(FILE *pal2file)
{
    int i,n,Nnodes,Nstrut,flag;
    char *pchar,*pline,line[81];
    float x,y,z,*ptr;

    sscanf(headbuff," %d %d",&Nnodes,&Nstrut);
    ptr = datarray;

    if(datatype == 'D') {
        memset((void *)datarray,'\0',3*Nnodes*sizeof(float));

        n = 1;
        while(n <= Nnodes) {
            pline = skipcomment(line,pal2file);
            if(strlen(pline) < 10);
            else {
                if(strchr(pline,'a') != NULL);
                else {
                    sscanf(pline," %d %f %f %f",&i,&x,&y,&z);
                    if(i < n) {
                        printf("\007\npal2 file has unusual node order at node"
                                                        " %d.  Abort.\n",n);
                        exit(1);
                    }
                    while(i > n) {
                        printf("pal2 file missing data for node %d.\n",n);
                        ptr += 3;
                        n++;
                    }
                    *ptr++ = x;
                    *ptr++ = y;
                    *ptr++ = z;
                    n++;
                }
            }
        }
    }
    else {
        n = 1;
        while(n <= Nstrut) {
            pline = skipcomment(line,pal2file);
            if(strlen(pline) < 10);
            else {
                if(strchr(pline,'a') != NULL);
                else {
                    sscanf(pline," %d %f %f",&i,&x,&y);
                    if(fabs(y) > fabs(x))
                        x = y;
                    if(i < n) {
```

```
                printf("\007\npal2 file has unusual number order at "
                                                "strut %d.  Abort.\n",
                exit(1);
            }
            while(i > n) {
                printf("pal2 file missing data for strut %d.\n",n);
                *ptr++ = 0;
                n++;
            }
            *ptr++ = x;
            n++;
        }
      }
    }
  }
}

/*********************************************************
 */
char *skipcomment(char *line,FILE *infile)
{
    int chr,old;
    char *ptr;

    while(fgets(line,80,infile) != NULL) {
        ptr = line;
        while(isspace((chr = (int)*ptr)))
            ptr++;
        if(isdigit(chr) || chr == '.' || chr == '+' || chr == '-')
            return ptr;
    }
    if(feof(infile)) {
        printf("\007\nEnd of file reached.\n");
        return line;
    }
    else {
        printf("\007\nError in reading pal2 file.  Abort.\n");
        exit(1);
    }
}

/*********************************************************
 */
```

Source Code Listing for BINSTAT.C

```
/*  This program reads a BINary PAL output displacement or stress file and
 *  computes various statistics on the results for multiple load
 *  cases.
 *  (Richard K. Miller, May 2, 1988)
 *  John M. Hedgepeth       5/12/88
 *                          7/16/88
 */

#include<scitech.h>

int display();
void exam_disp(int,int, FILE *),exam_stress(int,int,FILE *);

main()
{
    static char header[256];
    unsigned seed;
    int Nnodes,Nelements,Ncases,flag;
    double mean,sdev,cutoff;
    FILE *binfile;

    while((flag = display(&binfile)) != 'Q') {

/*  Read header out of binary file
 */
        fread(header,sizeof(char),256,binfile);

        if(sscanf(header," %d %d %d %lE %lE %lE %u",&Nnodes,&Nelements,
                                  &Ncases,&mean,&cutoff,&sdev,&seed) != 7) {
            printf("\007\n\tCouldn't read 7 fields from header buffer. Abort.\n"
            exit(1);
        }

/*  Print out data read from header
 */
        printf("\n\nParameter values pertinent to this set of ");
        if(flag == 'D')
            printf("displacement ");
        else
            printf("stress ");
        printf("data:\n\n");

        printf("\tNnodes  = %d\n",Nnodes);
        printf("\tNstruts = %d\n",Nelements);
        printf("\tNcases  = %d\n\n",Ncases);
        printf("\tMean member strain                          = %lE\n",mean);
        printf("\tStandard deviation of Gaussian distribution = %lE\n",sdev);
        printf("\tCutoff deviation of strain from mean        = %lE\n",cutoff);
        printf("\n\tSeed = %u\n",seed);

        if(flag == 'D')
            exam_disp(Nnodes,Ncases,binfile);
        else
            exam_stress(Nelements,Ncases,binfile);
        fclose(binfile);
```

```
    }
}

/************************************************
 */
int display(FILE **pbinfile)
{
    int flag;
    char name[30];

    printf("\n\n\tStatistical Analysis Postprocessor\n");
    printf("\t    ASTRO AEROSPACE CORPORATION\n");
    printf("\t           July 18,1988\n\n");

    while(1) {
        printf("\nEnter  'd'  to examine displacements.\n");
        printf("       's'  to examine stresses.\n");
        printf("       'q'  to exit examination.\n?  ");

        fflush(stdin);
        flag = toupper(getchar());
        while(flag == 'D' || flag == 'S') {

/*  Open BINary file for reading
 */
            printf("\n\nEnter name of binary ");
            if(flag == 'D')
                printf("DISplacement ");
            else
                printf("STRess ");
            printf("storage file to be read.\n?  ");
            fflush(stdin);
            gets(name);
            strupr(name);

            if ((*pbinfile = fopen(name,"rb")) == NULL )
                printf("\007\nCannot open %s for binary read.  Try again.\n",nam
            else
                break;
        }
        if(flag == 'Q' || flag == 'D' || flag == 'S')
            break;
        else
            putchar('\007');
    }
    return flag;
}

/*******************************************************************
 */

void exam_disp(int Nnodes,int Ncases,FILE *binfile)
{
    long offset;
    int i,n,j,nxmax,nxmin,nymax,nymin,nzmax,nzmin,ytest;
    float x,y,z,*fptr,*datarray;
    float xmax,xmin,ymax,ymin,zmax,zmin,xsum,ysum,zsum,xsq,ysq,zsq;
    char line[20];

/* Allocate array space
```

```
*/
    if((datarray = (float *)calloc(3*Nnodes,sizeof(float))) == NULL) {
        printf("\007\nCannot allocate space for data array.  Abort.\n");
        exit(1);
    }

    ytest = 'y';

    while( ytest == 'y' ) {

        while(1) {
            printf("\nEnter node number for displacements to be scanned.\n");
            printf("Enter 'x' to exit.\n? ");
            gets(line);
            if(tolower((int)*line) == 'x') {
                ytest = 'n';
                break;
            }
            if(sscanf(line," %d",&n) != 1)
                n = 0;
            if(n > 0 && n <= Nnodes)
                break;
            else
                putchar('\007');
        }
        if(ytest == 'n')
            break;

        xmax = 0;
        xmin = 0;
        ymax = 0;
        ymin = 0;
        zmax = 0;
        zmin = 0;

        nxmax = 0;
        nxmin = 0;
        nymax = 0;
        nymin = 0;
        nzmax = 0;
        nzmin = 0;

        xsum = 0;
        ysum = 0;
        zsum = 0;
        xsq = 0;
        ysq = 0;
        zsq = 0;

        fseek(binfile, 256L, SEEK_SET);

        for ( i = 1; i <= Ncases; i++ ) {
            fread(datarray,sizeof(float),3*Nnodes,binfile);

            fptr = datarray + 3*(n - 1);
            x = *fptr++;
            y = *fptr++;
            z = *fptr;

            xsum += x;
```

```
            ysum += y;
            zsum += z;

            xsq += x*x;
            ysq += y*y;
            zsq += z*z;

            if( x > xmax ) {
                xmax = x;
                nxmax = i;
            }

            if( x < xmin ) {
                xmin = x;
                nxmin = i;
            }

            if( y > ymax ) {
                ymax = y;
                nymax = i;
            }

            if( y < ymin ) {
                ymin = y;
                nymin = i;
            }

            if( z > zmax ) {
                zmax = z;
                nzmax = i;
            }

            if( z < zmin ) {
                zmin = z;
                nzmin = i;
            }

        }

        xsum /= Ncases;
        ysum /= Ncases;
        zsum /= Ncases;

        xsq = sqrt((xsq - Ncases*xsum*xsum)/(Ncases - 1));
        ysq = sqrt((ysq - Ncases*ysum*ysum)/(Ncases - 1));
        zsq = sqrt((zsq - Ncases*zsum*zsum)/(Ncases - 1));

/*  Print out results of scan
 */

        printf("\n\tFor node number %d, a scan of %d cases reveals -\n",
                                                                   n,Ncases);
        printf("\n\t\tX-direction results:\n");
        printf("\t\t\tmean  = %12.4e\n\t\t\tsigma = %12.4e\n",xsum,xsq);
        printf("\t\t\tmax,min = %12.4e, %12.4e (at case = %d, %d)\n",
                                                        xmax,xmin,nxmax,nxmin);
        printf("\n\t\tY-direction results:\n");
        printf("\t\t\tmean  = %12.4e\n\t\t\tsigma = %12.4e\n",ysum,ysq);
        printf("\t\t\tmax,min = %12.4e, %12.4e (at case = %d, %d)\n",
                                                        ymax,ymin,nymax,nymin);
```

```
        printf("\n\t\tZ-direction results:\n");
        printf("\t\t\tmean  = %12.4e\n\t\t\tsigma = %12.4e\n",zsum,zsq);
        printf("\t\t\tmax,min = %12.4e, %12.4e (at case = %d, %d)\n\n\n",
                                                        zmax,zmin,nzmax,nzmin);
    }
    free(datarray);
}

/**********************************************************************
 */

void exam_stress(int Nstrut,int Ncases,FILE *binfile)
{
    long offset;
    int i,n,j,nmax,*nmaxarray,nmin,ytest;
    float stress,*datarray,*maxarray,*meanarray,*rmsarray,*fptr;
    float smax,smin,ssum,ssq;
    char line[20];
    ytest = 'y';

    printf("\nEnter any key to scan data file for strut stresses... ");
    getch();
    printf("\nWorking... ");

    nmaxarray = (int *)calloc(Ncases,sizeof(int));
    maxarray = (float *)calloc(Ncases,sizeof(float));
    meanarray = (float *)calloc(Ncases,sizeof(float));
    rmsarray = (float *)calloc(Ncases,sizeof(float));
    if((datarray = (float *)calloc(Nstrut,sizeof(float))) == NULL) {
        printf("\007\nCannot allocate space for data array.  Abort.\n");
        exit(1);
    }

    for(n=0; n<Ncases; n++) {
        fread(datarray,sizeof(float),Nstrut,binfile);
        fptr = datarray;
        smax = 0;
        smin = 0;
        ssum = 0;
        ssq = 0;
        for(i=0; i<Nstrut; i++) {
            stress = *fptr++;
            if(stress > smax) {
                smax = stress;
                nmax = i+1;
            }
            if(stress < smin) {
                smin = stress;
                nmin = i+1;
            }
            ssum += stress;
            ssq += stress*stress;
        }
        ssq -= ssum*ssum/Nstrut;
        *(rmsarray + n) = sqrt(ssq/(Nstrut - 1));
        *(meanarray + n) = ssum/Nstrut;
        if(smax > -smin) {
            *(maxarray + n) = fabs(smax);
            *(nmaxarray + n) = nmax;
        }
```

```
        else {
            *(maxarray + n) = fabs(smin);
            *(nmaxarray + n) = nmin;
        }
    }
    smax = 0;
    ssum = 0;
    ssq = 0;
    for(n=0; n<Ncases; n++) {
        stress = *(maxarray + n);
        ssum += stress;
        ssq += stress*stress;
        if(stress > smax) {
            smax = stress;
            nmax = n;
            i = *(nmaxarray + n);
        }
    }
    ssq -= ssum*ssum/Ncases;
    ssum /= Ncases;
    ssq = sqrt(ssq/(Ncases - 1));

    printf("\nA scan over all %d cases shows that:\n",Ncases);
    printf("\tThe average of the maximum stresses for each case = %12.4lE.\n",
                                                              ssum);
    printf("\tThe standard deviation of these maximum stresses  = %12.4lE.\n",
                                                              ssq);
    printf("\tThe global maximum value of the maximum stresses  = %12.4lE.\n",
                                                              smax);
    printf("\t    and occurs at strut %d for case %d.\n\n",i,nmax + 1);

    while(ytest == 'y') {
        printf("\nEnter case number for more information. (1,2,... %d) \n",
                                                           Ncases);
        printf("or enter 'x' to exit.\n? ");
        gets(line);
        if(tolower((int)*line) == 'x')
            break;

        if(sscanf(line," %d",&n) == 1) {
            if(n > 0 && n <= Ncases) {
                printf("\n\nFor case %d:\n",n);
                printf("\tAverage stress     = %12.4lE.\n",*(meanarray + n - 1))
                printf("\tStandard deviation = %12.4lE.\n",*(rmsarray + n - 1));
                printf("\tMaximum stress     = %12.4lE.\n",*(maxarray + n - 1));
                printf("\t    at strut number %d.\n\n",*(nmaxarray + n - 1));
            }
            else
                putchar('\007');
        }
        else
            putchar('\007');
    }
    free(datarray);
    free(meanarray);
    free(maxarray);
    free(nmaxarray);
    free(rmsarray);
}
```

```
/**************************************************************
 */


/*  Read in nodal coordinates
 */

/*  offset = 256 * sizeof(char);
    fseek(binfile,offset,SEEK_SET);

    for( i = 0; i < Nnodes; i++ ) {

        fread(&xtemp,sizeof(float),1,binfile);
        fread(&ytemp,sizeof(float),1,binfile);
        fread(&ztemp,sizeof(float),1,binfile);

        xc[i] = xtemp;
        yc[i] = ytemp;
        zc[i] = ztemp;

    }

    Print nodal coordinate values


    printf("\n\tNODAL COORDINATES\n");

    for ( i = 0; i < Nnodes; i++ ) {

        printf("\t %d %f %f %f\n",i+1,xc[i],yc[i],zc[i]);

    }

 */
```

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No. NASA CR-181746	2. Government Accession No.	3. Recipient's Catalog No.

4. Title and Subtitle Investigation of Structural Behavior of Candidate Space Station Structure	5. Report Date June 1989
	6. Performing Organization Code
7. Author(s) John M. Hedgepeth Richard K. Miller	8. Performing Organization Report No. AAC-TN-1152
	10. Work Unit No. 585-02-31-01
9. Performing Organization Name and Address Astro Aerospace Corporation 6384 Via Real Carpinteria, CA 93013-2993	11. Contract or Grant No. NAS1-17536, Task 8
	13. Type of Report and Period Covered Contractor Report
12. Sponsoring Agency Name and Address NASA Langley Research Center Hampton, VA 23665-5225	14. Sponsoring Agency Code

15. Supplementary Notes

Langley Technical Monitor: Wilbur B. Fichter

16. Abstract

The report contains quantitative evaluations of the structural loads, stiffness and deflections of an example Space Station truss due to a variety of influences, including manufacturing tolerances, assembly operations, and operational loading. The example truss is a dual-keel design composed of 5-meter-cube modules. The truss is 21 modules high and 9 modules wide, with a transverse beam 15 modules long.

One problem of concern is the amount of mismatch which will be expected when the truss is being erected on orbit. Worst-case thermal loading results in less than 0.5 inch of mismatch. The stiffness of the interface is shown to be less than 100 pounds per inch. Thus, only moderate loads will be required to overcome the mismatch.

The problem of manufacturing imperfections is analyzed by the Monte Carlo approach. Deformations and internal loads are obtained for ensembles of 100 example trusses.

All analyses are performed on a personal computer. The necessary routines required to supplement commercially available programs are described.

17. Key Words (Suggested by Author(s)) Space station Truss Fabrication errors Residual forces	18. Distribution Statement Unclassified - Unlimited Subject Category 39

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of pages 128	22. Price A07

NASA FORM 1626 OCT 86