

AD-A112 456

DUKE UNIV DURHAM NC DEPT OF MECHANICAL ENGINEERING A--ETC F/6 6/19
DEVELOPMENT OF A STRAIN RATE DEPENDENT LONG BONE INJURY CRITERI--ETC(U)
JAN 82 T K NIGHT

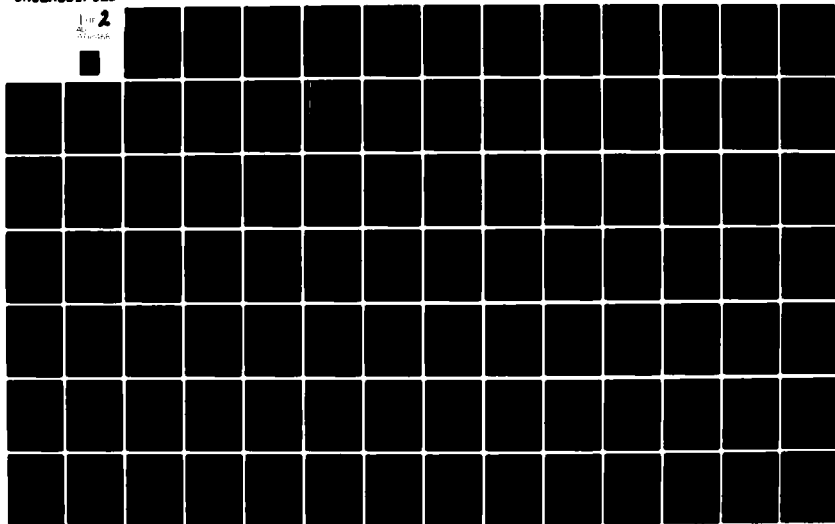
AFOSR-81-0662

UNCLASSIFIED

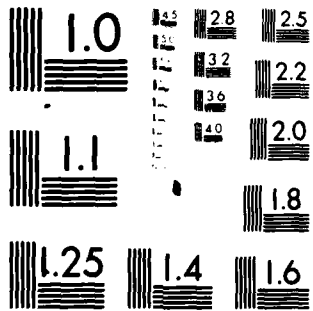
AFOSR-TR-82-0138

NL

1 of 2
2



A/124



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

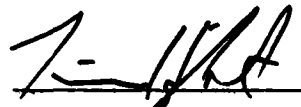
110

ADA 112458

DEVELOPMENT OF A STRAIN RATE DEPENDENT LONG BONE INJURY
CRITERION FOR USE WITH THE ATB MODEL

FINAL REPORT

CONTRACT AFOSR-81-0062



Timothy K. Hight
Assistant Professor
Dept. of Mechanical Engineering
and Materials Science
Duke University
Durham, N.C. 27706

DTIC
EXECTE
1982

DTIC FILE COPY

January 12, 1982

82 03 22

Approved for publ
distribution unl.

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH (AFSC)
NOTICE OF TEMPORAL TO DTIC
This technical report has been reviewed and is
approved for public release under E.O. 13526.
Distribution is unlimited.
MATTHEW S. MURPHY
Chief, Technical Information Division

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM	
AFOSR-TR- 82 - 0188		GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER <i>AD-A112 458</i>
4. TITLE (and Subtitle) Development of a Strain Rate Dependent Long Bone Injury Criterion for Use with the ATB Model.		5. TYPE OF REPORT & PERIOD COVERED Final Report	
7. AUTHOR(s) Timothy K. Hight		6. PERFORMING ORG. REPORT NUMBER	
9. PERFORMING ORGANIZATION NAME AND ADDRESS Dept. of Mechanical Engineering & Materials Sci. Duke University Durham, N.C. 27706		8. CONTRACT OR GRANT NUMBER(s) AFOSR-81-0062	
11. CONTROLLING OFFICE NAME AND ADDRESS AFOSR/NL Bolling Air Force Base Washington, D.C. 20332		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS <i>61102F</i> 231A/D9	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE January 12, 1982	
		13. NUMBER OF PAGES 102	
		15. SECURITY CLASS. (of this report) Unclassified	
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE	
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.			
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)			
18. SUPPLEMENTARY NOTES			
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)			
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) See Over			

JAN 20 1982

Unclassified

ABSTRACT

An improved version of a previously reported long bone injury criterion has been developed. This new criterion was based on fitting the Ramberg-Osgood equation, through optimization techniques, to existing stress-strain-strain rate curves for compact bone and using this equation to solve for the strain at a time step given the current stress and the strain from the previous time. Refinements have also been made to the geometric modeling of the bones where adequate data existed. When coupled with the (ATB) model and relations between ultimate stress and strain rate, this criterion has provided an effective means of judging long bone injury potential.

Accession No.	HTS 10121	BY	DISPATCH
	DTIC 203		AVAILABILITY STATE
	UNCLASSIFIED		
	DECLASSIFIED		

A

I. INTRODUCTION

The Articulated Total Body (ATB) Model has been developed as a tool to predict and analyze the response of the human body to potential injury causing environments. This model has been primarily used for the analysis of vehicular crashes, and has been extremely useful in predicting gross motion of the body. The model also predicts the loads on each of the articulated joints. The USAF has been involved in the development and testing of this computer model and has applied it to the analysis of the response of pilots to ejection from jet aircraft. During these events the body is subjected to high accelerations from the ejection itself, to impacts of the limbs with each other and with the cockpit, and to "wind flail" from the high velocity wind stream which impacts the body after clearing the aircraft. It is hoped that a better understanding of these events will lead to safer designs and lower injury rates.

Despite the excellent results obtained from the ATB model, the current version does not provide the information necessary to predict whether the event will or will not cause an injury. It is therefore necessary to augment the current model by the addition of some sort of injury criterion in order for the full potential of the ATB to be realized. With this addition, the model can be used to compare different acceleration profiles, restraint systems and other variables as to their injury preventing potential. Currently these assessments must be made, in a very qualitative way, on the basis of motion of the limbs and loads in the joints.

Since the majority of serious injuries resulting from ejections are bone fractures, it is of particular interest to estimate the likelihood of long bone fracture. (It should be noted that a separate computer model, the Head-Spine Model,

is specifically designed to study the question of head and spine injuries, and we have consciously ignored this aspect of injury).

The research carried out under this minigrant was begun by the principal investigator during a Summer Faculty Research Program stay at WPAFB. The major objectives of that research program were to establish a simplified injury criterion for bone and to implement that criterion in a computer program (BREAK) compatible with the existing ATB. Due to the constraints of such a short time period (10 weeks), the project was necessarily restricted to a rather narrow scope. Within these constraints the objectives were accomplished. A detailed report of this project can be found in the Final Report submitted to SCEE, and published as a technical report by AFAMRL (1), but the significant developments will be summarized here.

Injury criteria reported in the literature have focused on two distinct approaches - "experimental" and "analytical". The experimental approach comes largely from the automobile industry and involves the experimental determination of loads which, under simulated crash tests, cause injury to cadavers. For long bones this is restricted to the Femur Injury Criterion which is a measure of the axial load applied to the shaft of the femur which causes fracture. The current Federal Standard is 1700 lb force without any loading rate dependence. A number of authors have suggested different criteria with loading rate sensitivity (2,3). Even so, the basic fault of these criteria is their specificity - they say nothing about other long bones or other loading conditions. This type of criterion was therefore inappropriate to a study of ejection seat injuries.

The alternate type of injury criterion is based on the analytic approach using the material and geometric properties of the bone and the calculated stresses. This approach has the advantage of being applicable to any loading situation if the properties of the bone are known. This was the approach taken.

The material properties of bone are highly variable, depend on a large number of parameters and are time dependent (4,5,6). In order to accurately model the behavior of bone, consistent data including complete strain rate dependent stress strain curves are needed for human bone. The data which most nearly fulfilled these criteria was the work of McElhanev (7) which was for embalmed human bone in compression. His results indicated the following relation between ultimate stress (σ) and strain rate ($\dot{\epsilon}$)

$$\sigma = 4200 \text{ Log } \dot{\epsilon} + 33000. \quad (1)$$

This equation, after modifications, became the basis for the injury criteria developed. Two major changes to this equation were made - the strain rate dependency was replaced by a stress rate dependence, and the constant term was modified to bring the results in line with published static results for fresh human compact bone. The rate modification was accomplished by first finding a relation between elastic modulus and strain rate, and then differentiating the elastic stress strain relation with respect to time for a constant strain rate. The second modification was based on $\dot{\epsilon} = .001$ as a static strain rate and was straightforward. The resulting equation for compression (and in psi units) is

$$\sigma = 3936 \log \dot{\epsilon} + 12000 \quad (2)$$

Similar expressions were generated for tension and shear based on the assumption that the material behavior (rate dependence) would be the same in the other modes and that only the constant offset would change. These equations formed the basis of the injury criterion models developed that summer.

An alternative to these criterion was generated using stress pulse duration, rather than stress rate, as the time variable. This alternate approach was examined because most published criteria from the auto industry are based on pulse duration.

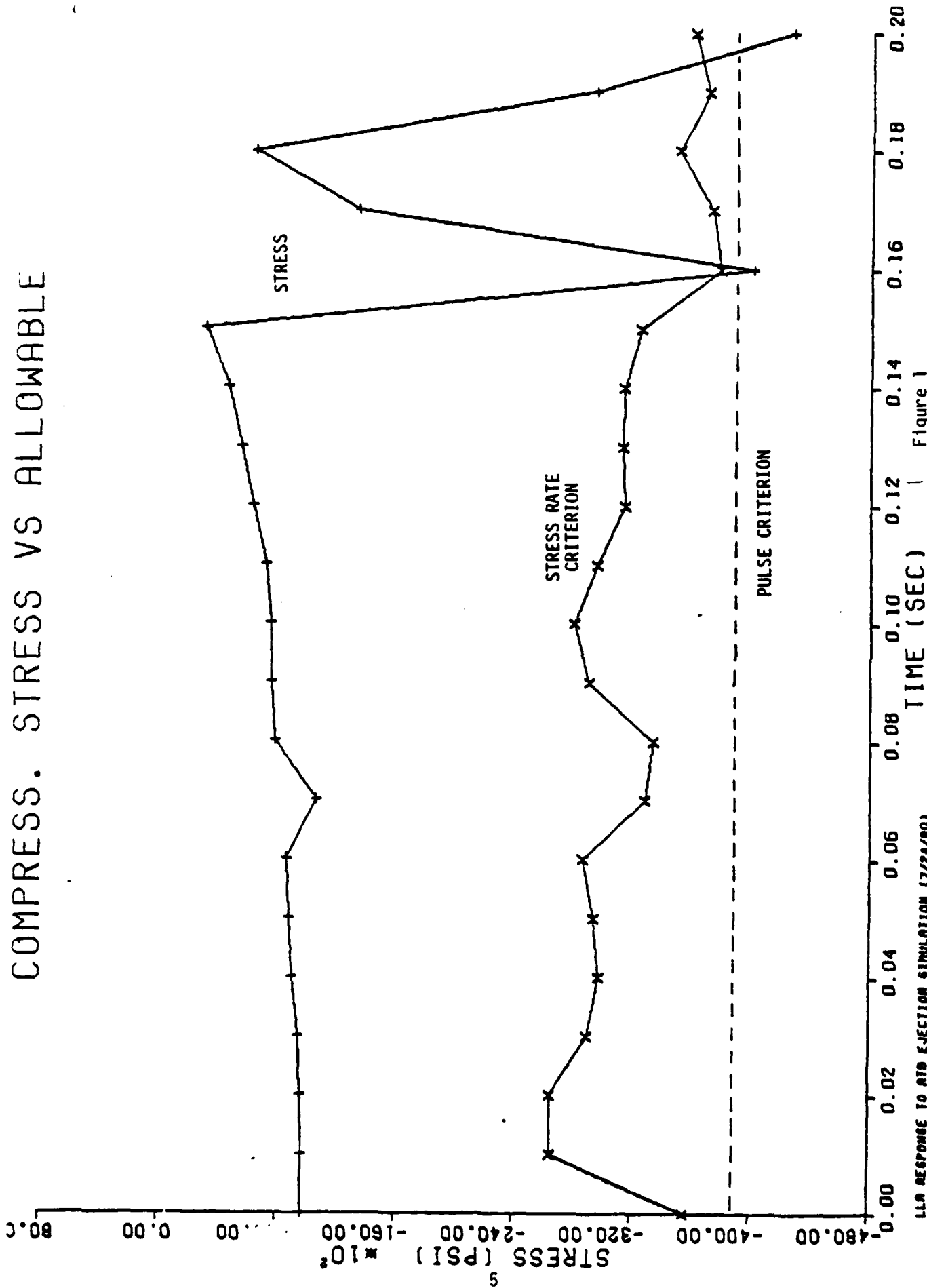
The pulse duration was approximated by assuming a sinusoidal pulse for the stress. The "pulse" criterion tended to be slightly less conservative than the stress rate criteria.

The stress calculations were based on a very simplified geometry, namely straight, uniform, isotropic hollow tubes for all the long bones. Principal stresses were then calculated at some predetermined number of sections equally spaced along the bone axis and the peak tension, compression and shear stresses were monitored. The peak stresses as a function of time for a particular limb were therefore not necessarily stresses at the same point, the position along the axis of the bone could vary with time also.

A test case was run based on an ejection simulation to demonstrate the program. Results are presented in the Technical Report mentioned above and a sample figure is reproduced here. Figure 1 shows the maximum compressive stress as a function of time for the left lower arm. There was contact at .07 secs and some high level, higher frequency loading at the elbow near the end of the data used (only the first 200 msec of the event were examined). The stress rate dependence of the allowable stress is evident as is the slightly less conservative nature of the pulse criterion. Note that the pulse criterion has only been calculated for the maximum stress pulse and hence appears as a constant.

This summarizes the state of this project at the start of the minigrant.

COMPRESS. STRESS VS ALLOWABLE



LLA RESPONSE TO ATB EJECTION SIMULATION (7/24/80)

Figure 1

II. OBJECTIVES

As outlined in the grant proposal, there were five main objectives of this investigation, and they were:

1. Improve the efficiency of the developed program, BREAK.
2. Search out and reexamine the available data base on bone material properties.
3. Develop more precise constitutive equations for bone.
4. Provide a statistical analysis of the available data.
5. Redefine the injury criteria based on the new information.

III. RESULTS

In this section each of the objectives described in the previous section will be addressed in turn, and the outcome of each investigation will be detailed.

A. The greatest problem with BREAK, as discussed in the proposal, has been the lack of compatibility between the output of the ATB Model and the input needs of BREAK. Basically the problems are ones of transformation of coordinate systems and reconstruction of contact forces from resultants. As an example, suppose that a limb contacts a panel resulting in normal and friction forces. The output of the ATB Model will include the linear and angular position, velocity and acceleration of the center of mass of the segment along with the forces and moments resulting at the joints of the segment. In addition, the resultant contact force and the point of contact will be given. For BREAK, since it calculates stress levels at a number of points along the axis of the bone, all forces must be known in local coordinates and the contact forces must be known in a vector sense, rather than in magnitude only. The reconstruction of the contact forces is a laborious task which repeats the work already done by the ATB program. In fact, all necessary information exists within the ATB program at some point during the calculations and, as proposed earlier, a file could be set up containing the appropriate information. (See program CONTACT in Appendix). Unfortunately, this has not yet been carried out. The major difficulty has been the lack of available memory on the CDC-CYBER computer system used by AFAMRL. All "non essential" data are currently eliminated during processing of the program in order to save space, and that loss currently includes the data which is needed for BREAK. Therefore the current version of BREAK (see CONTACT in Appendix) includes all of the manipulations which must be performed to transform the given data into the needed format.

Another important change to the program dealt with bone geometry. Rather than continue using uniform, hollow tubes to represent bones, we have tried to imitate some of the variations in actual bone geometry. Using the limited amount of available bone geometry data (8,9) we have been able to model variations in cross-section properties along the bone axes (for 20 to 80% of distal distance) for the femur and tibia. No data was found for the upper extremity bones. For this model the bone is assumed to be made up of a discontinuous series of short sections of uniform hollow tubes.

It was necessary to scale the available data, which appeared to be from approximately 50th percentile subjects, up to a 95th percentile man so that the bone data would coincide with the other pilot data. This was accomplished by making the assumption that, for different size bodies, the stress levels in the bones would be relatively constant. Using the body weight and limb length, relative ratios of cross-sectional area and area moment of inertia could be generated using axial compressive stress and bending stress equations. The data used, in both original and scaled form, are shown in Figures 2 and 3. See program STRESS for a complete description of the analysis, along with the calculation of stresses at various cross-sections.

B. It has been well established that the elastic and ultimate properties of compact bone are loading rate dependent (e.g. 7,10). Ultimate stress levels typically vary by a factor of two or more over a range of 5 or 6 decades of strain rates. This effect is simply too large to ignore. However, the use of ultimate stress values alone, even if they are well known, is insufficient.

The ATB model represents the body as a series of rigid, though resilient, links connected by springs. As such, for each segment the end loads and motions are well

GEOMETRIC PROPERTIES OF BONE CROSS-SECTION
TIBIA

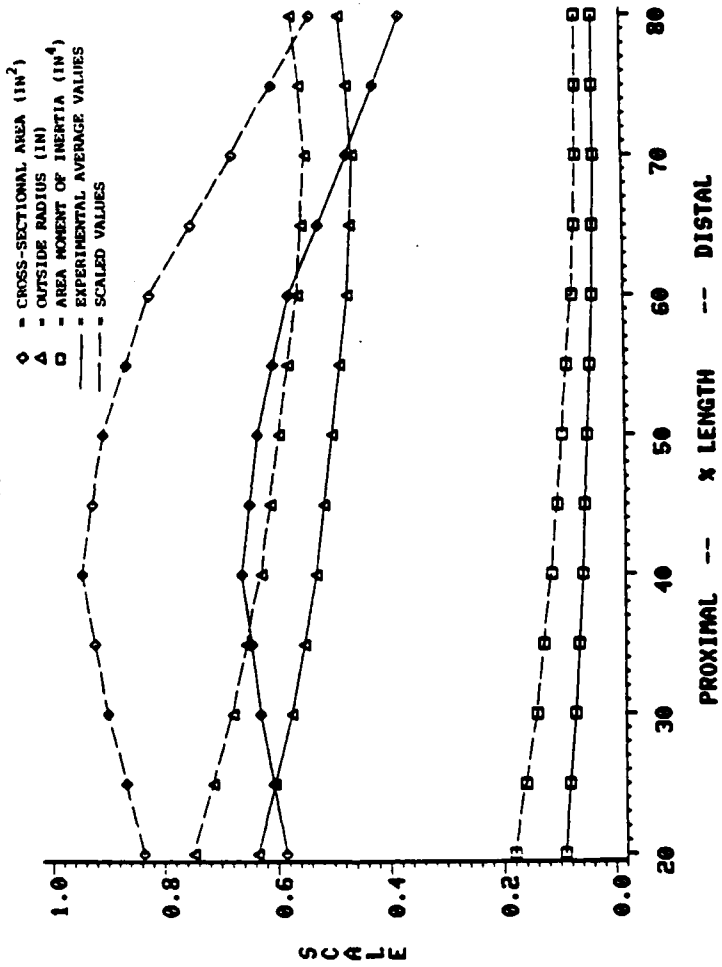


FIGURE 2

**GEOMETRIC PROPERTIES OF BONE CROSS-SECTION
FEMUR**

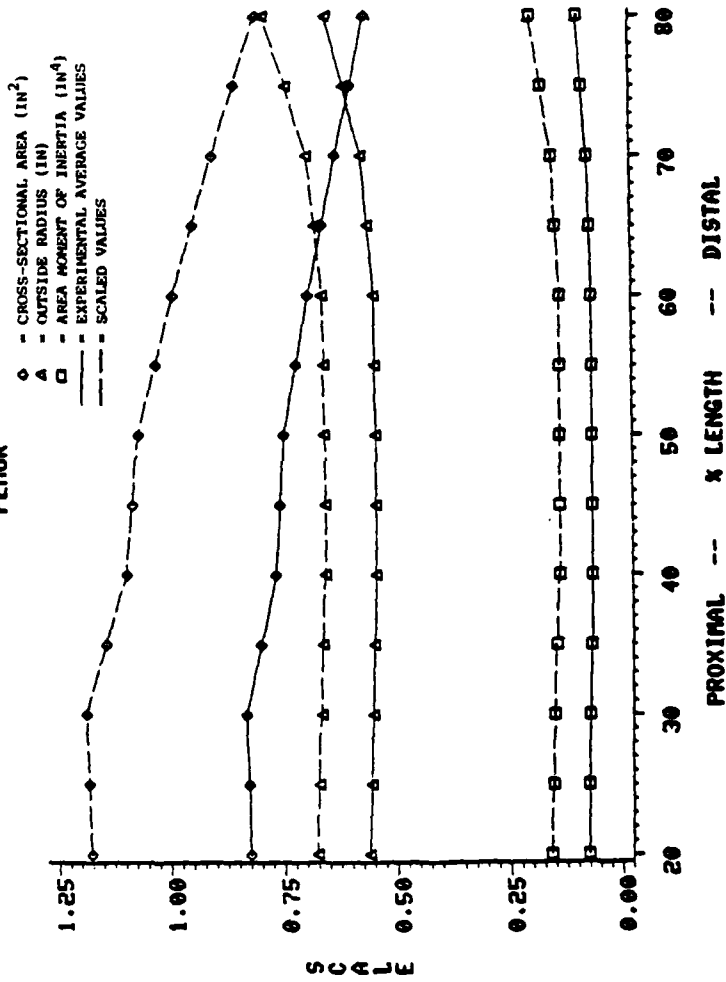


FIGURE 3

defined. These knowns, along with the bone geometries, are sufficient to describe the stress state at any timestep and at any point along the limb. The strains are then related to the stress through the theory of elasticity (for an elastic structure). For bone, the modulus of elasticity is a function of strain rate, and that function is not analytically invertable.

Three sets of stress-strain curves for various strain rates have been uncovered in the literature. McElhaney published the results of both bovine and embalmed human bone in compression using constant strain rates between 10^{-3} and 1500 sec^{-1} (7). More recently, Wood has reported on the tensile properties of fresh cranial bone for strain rates from $.003 \leq \dot{\epsilon} \leq 150 \text{ sec}^{-1}$ (11). This data is of little utility for the ATB model because Wood used very small bone samples and because cranial bone is significantly different in structure from diaphysial compact bone. Crowninshield and Pope (12) have also published results for compact bovine bone in tension over strain rates of $.00167 \leq \dot{\epsilon} \leq 250$. Their results indicate a much larger plastic strain region for longitudinal samples than do the results previously reported. However, Burstein et.al. (13) have also shown considerable plastic strain in bovine bone in tension and very little in compression. In Burstein's tests strain rates were not reported, but loading duration was between 1/10 and 1/2 sec.

The above three papers constitute a very small data set and none of the results are for fresh human long bone. Other researchers have dealt with portions of this problem, however they usually report only ultimate properties and not the full stress-strain curves. Lewis and Goldsmith (14), for example, used a split Hopkinson Bar method to measure the fracture stress of bovine bone in compression. The strain histories were pulses rather than constant strain rate. The fracture strain rates were calculated by dividing the fracture strain by the time and these rates varied

over approximately 6 orders of magnitude. These results are consistently higher than the results of McElhanev. Wright and Hayes (15) reported ultimate tensile strength for fresh bovine bone for $5.3 \times 10^{-4} \leq \dot{\epsilon} \leq 237 \text{ sec}^{-1}$. They also present two characteristic load-displacement curves, but no stress-strain data.

C. Much progress has been made in the area of establishing analytic functions which describe the relationship between stress, strain and strain rate. Using available published data and our optimizing curve fit computer program we have been able to get good fits of the data using a standard Ramberg-Osgood equation,

$$\epsilon = \frac{\sigma}{c\dot{\epsilon}^d} + a\sigma^N \dot{\epsilon}^b \quad (3)$$

This equation is a standard equation for modeling visco elastic-plastic behavior of materials (16). Results for the three available sets of curves are shown numerically in Table 1, and graphically in Figures 4, 5, and 6. The ability of this function to fit such a wide variation of results is encouraging.

Because of the orders of magnitude variations in the strain rate, certain constants in equation (3) were extremely sensitive. For example, the constant multiplier "a" for the plastic term varied by as much as 55 orders of magnitude from author to author. It was therefore necessary to take our optimization approach (see program MYFIT in Appendix) to minimize the function F (the sum of weighted errors of data-point fits), where F is defined by equation (4).

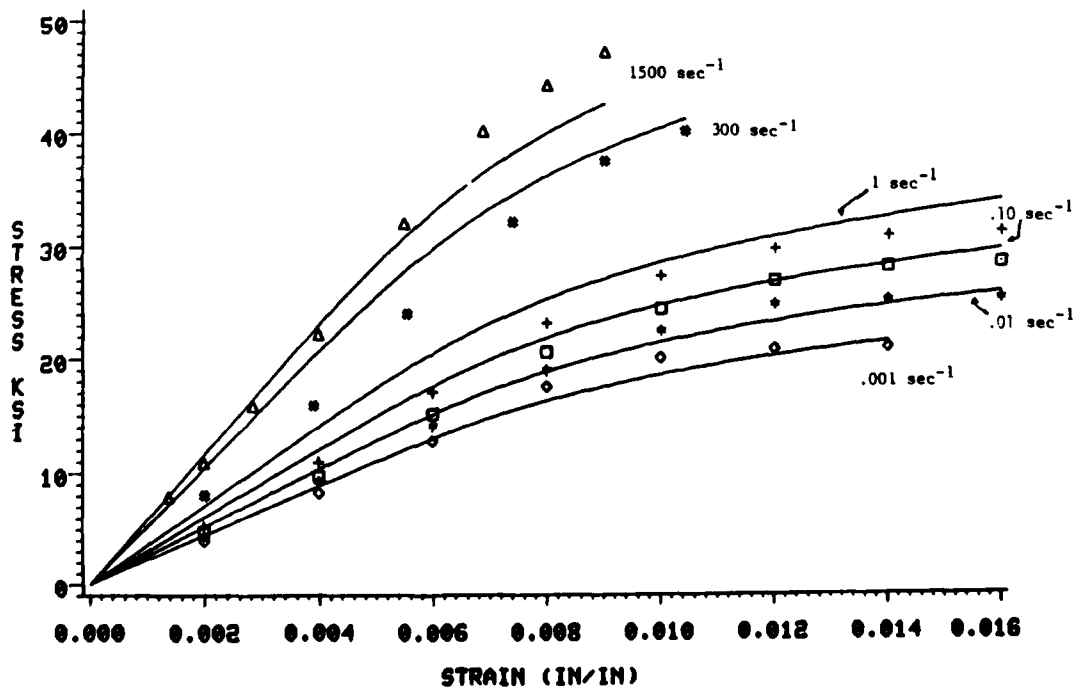
$$F = \sum_{i=1}^M [H(\epsilon_p - \epsilon_0)]^N + r \sum_{i=1}^M \langle gi \rangle^2 \quad (4)$$

BONE TYPE	Test Mode	c	d	a	N	b
emb. human femur	C	3551	.9671	6.12×10^{-13}	6.53	-.3740
bovine (long.)	T	1694	.9180	3.71×10^{-68}	45.3	-2.336
cranial compact	T	2200	.0567	3.68×10^{-12}	7.66	-.4127

Table 1 (for stress in ksi)

STRESS - STRAIN CURVES

EMBALMED HUMAN FEMUR : COMPRESSION (MCELHANEY)



STRAIN (IN/IN)

FIGURE 4

STRESS - STRAIN CURVES

BOVINE : TENSION (CROWNSHIELD & POPE)
LONGITUDINAL

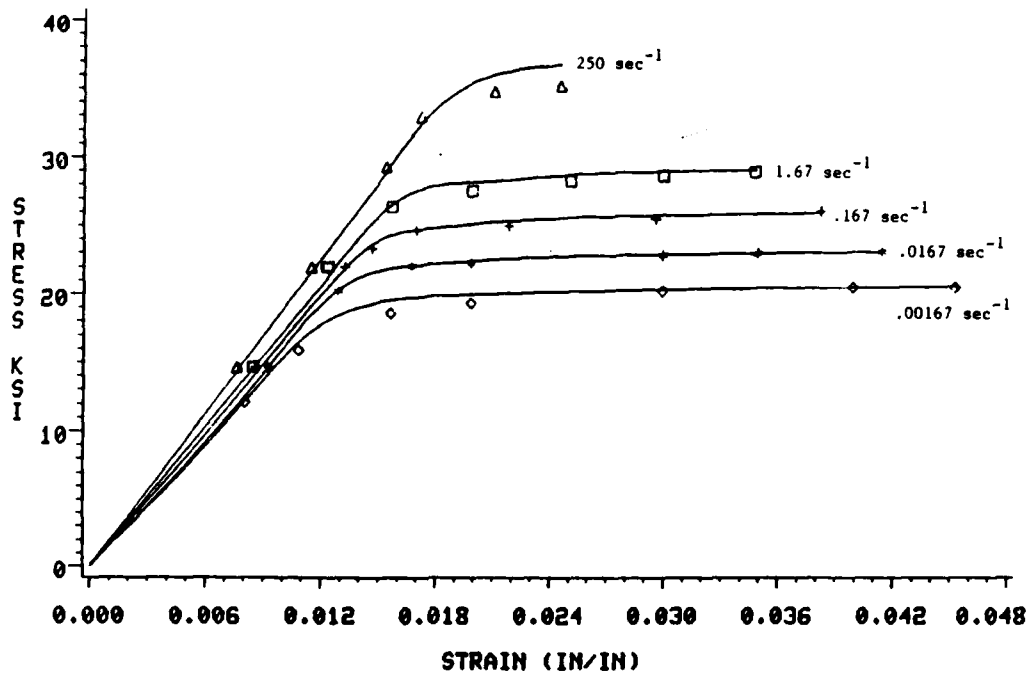


FIGURE 5

STRESS - STRAIN CURVES

CRANIAL COMPACT BONE : TENSION (WOOD)

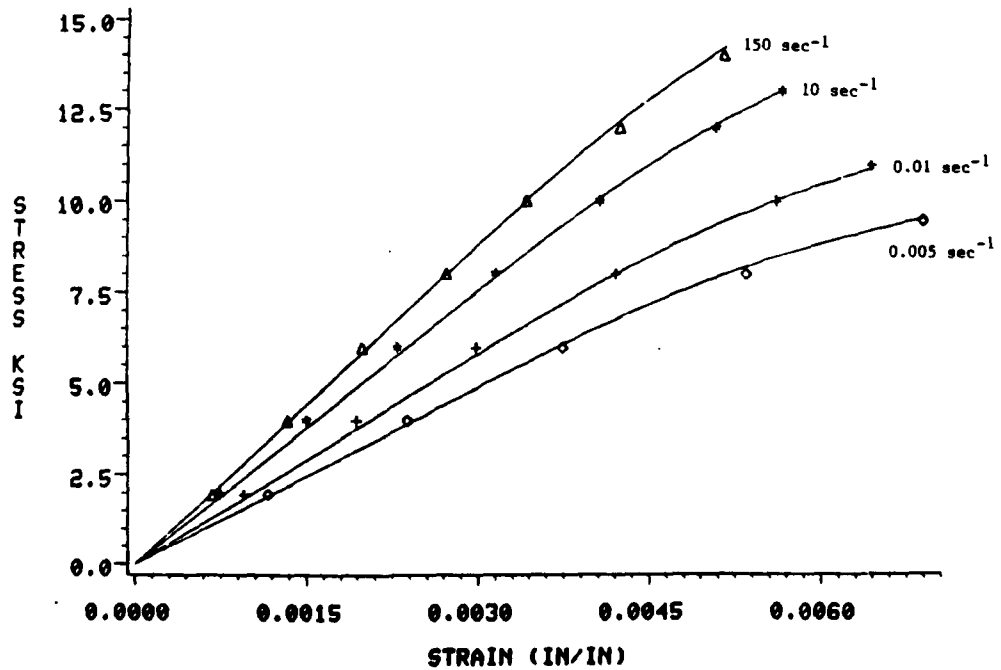


FIGURE 6

and where

- M = the number of experimentally observed points of $(\sigma, \epsilon, \dot{\epsilon})$
- ϵ_p = strain predicted by curve
- ϵ_o = strain observed by experiment
- r, H = constant multipliers for weighting
- N, z = constant powers for weighting
- $\langle gi \rangle$ = exterior penalty function for weighting

The objective function F is minimized in each of the five coordinate directions (a, b, N, d, c) using a combination of golden section search and parabolic interpolation methods (17). The resultant coordinates are the coefficients of the equation that best fits the set of observed data.

The results of the optimization process are the curves shown in Figures 4, 5 and 6. It can be seen that there is excellent agreement between most of the predicted and experimental results. The largest variations occur in the very high strain rate tests of McElhane. Forcing the function to try and fit the 1500 sec^{-1} data tends to also skew the rest of the curves. As an experiment, the 1500 sec^{-1} data was removed from the sample population and the curves refit to the remaining data. This trial showed a very good correlation with the remaining data and a high prediction at 1500 sec^{-1} . As this is the only data available at this high a strain rate it is difficult to draw any conclusions about this response.

D. It is obvious from the previous section that no meaningful statistical analysis can be performed on the meager amount of available data. However, it is interesting to compare the various results and this is shown in Figures 7, 8, 9 and 10. These figures show predicted stress-strain curves at a given strain rate based on various authors' results. Be aware first of all that these curves, except

STRESS - STRAIN CURVES

STRAIN RATE = 100 / SEC

LEGEND

- 1 : human femur (embalmed) - compression
McElhaney
- 2 : cranial compact (fresh) - tension
Wood
- 3 : cranial compact (fresh) - tension
Wood VA74
- 4 : bovine - tension (longitudinal)
Crowninshield & Pope

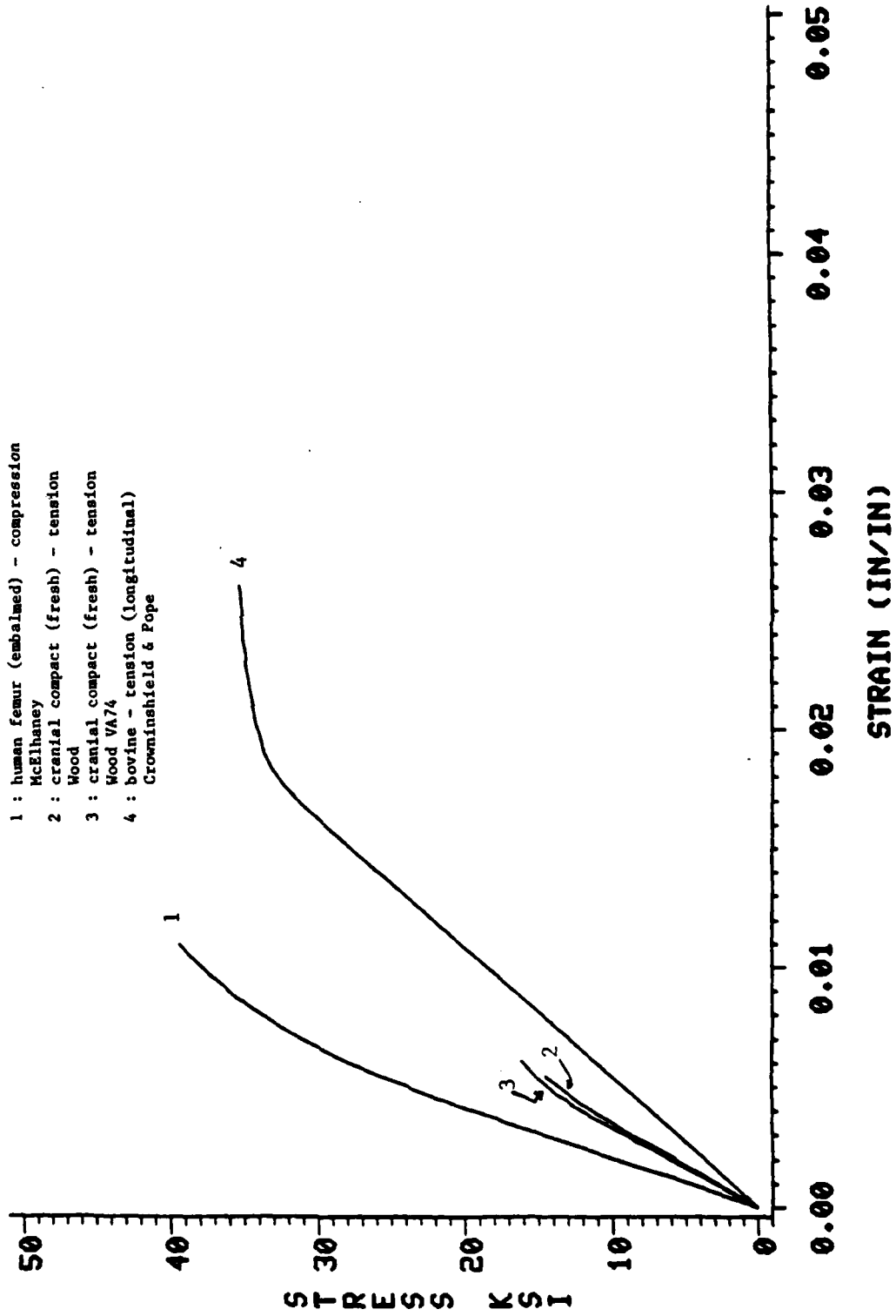


FIGURE 7

STRESS - STRAIN CURVES

STRAIN RATE = 1 / SEC

LEGEND

- 1 : human femur (embalmed) - compression
- McElhaney
- 2 : cranial compact (fresh) - tension
- Wood
- 3 : cranial compact (fresh) - tension
- Wood VA74
- 4 : bovine - tension (longitudinal)
- Crownshield & Pope
- 5 : bovine - tension (transverse)
- Crownshield & Pope

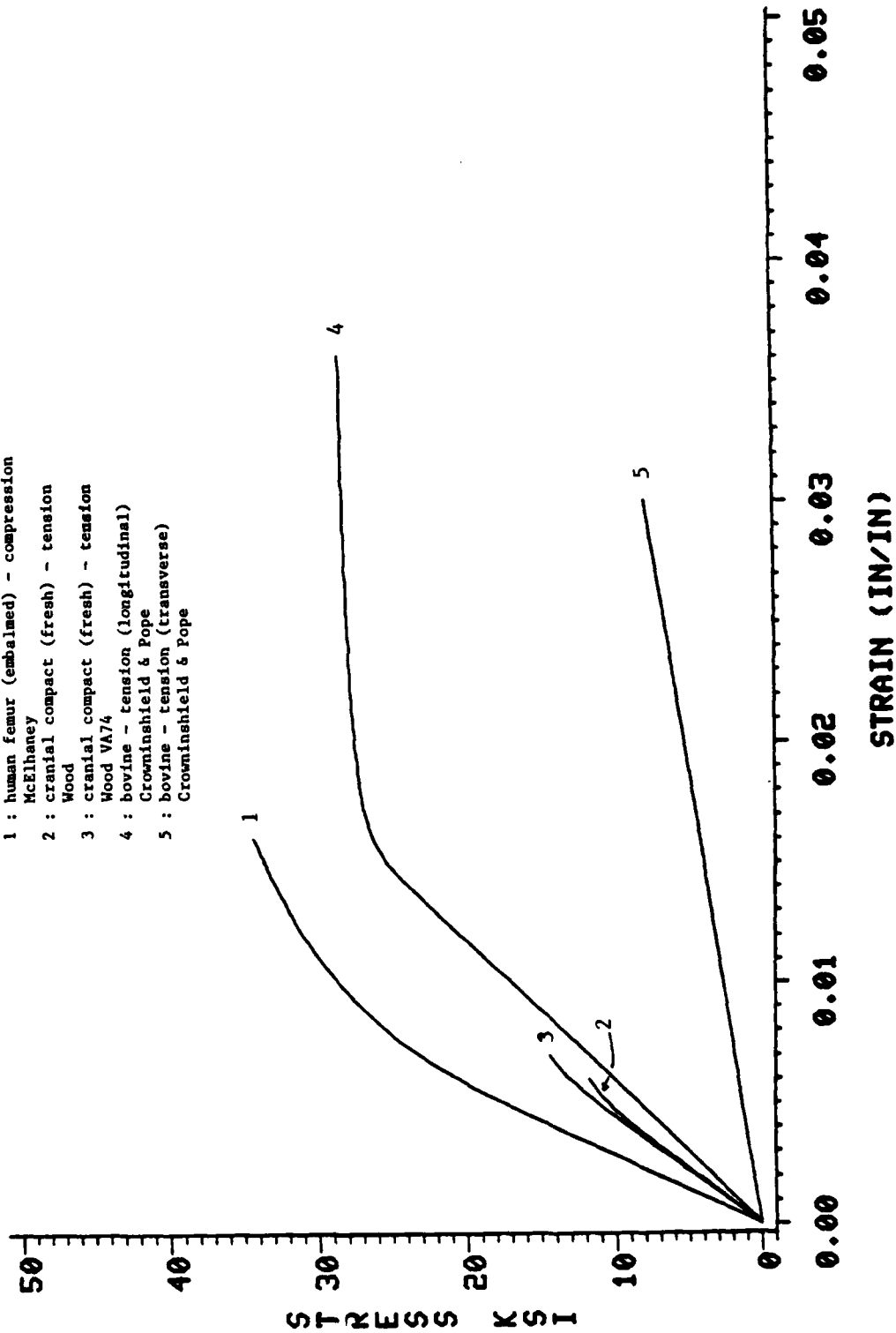


FIGURE 8

STRESS - STRAIN CURVES

STRAIN RATE = .01 / SEC

LEGEND

- 1 : human femur (embalmed) - compression
McElhaney
- 2 : cranial compact (fresh) - tension
Wood
- 3 : cranial compact (fresh) - tension
Wood VA74
- 4 : bovine - tension (longitudinal)
Crowninshield & Pope
- 5 : bovine - tension (transverse)
Crowninshield & Pope

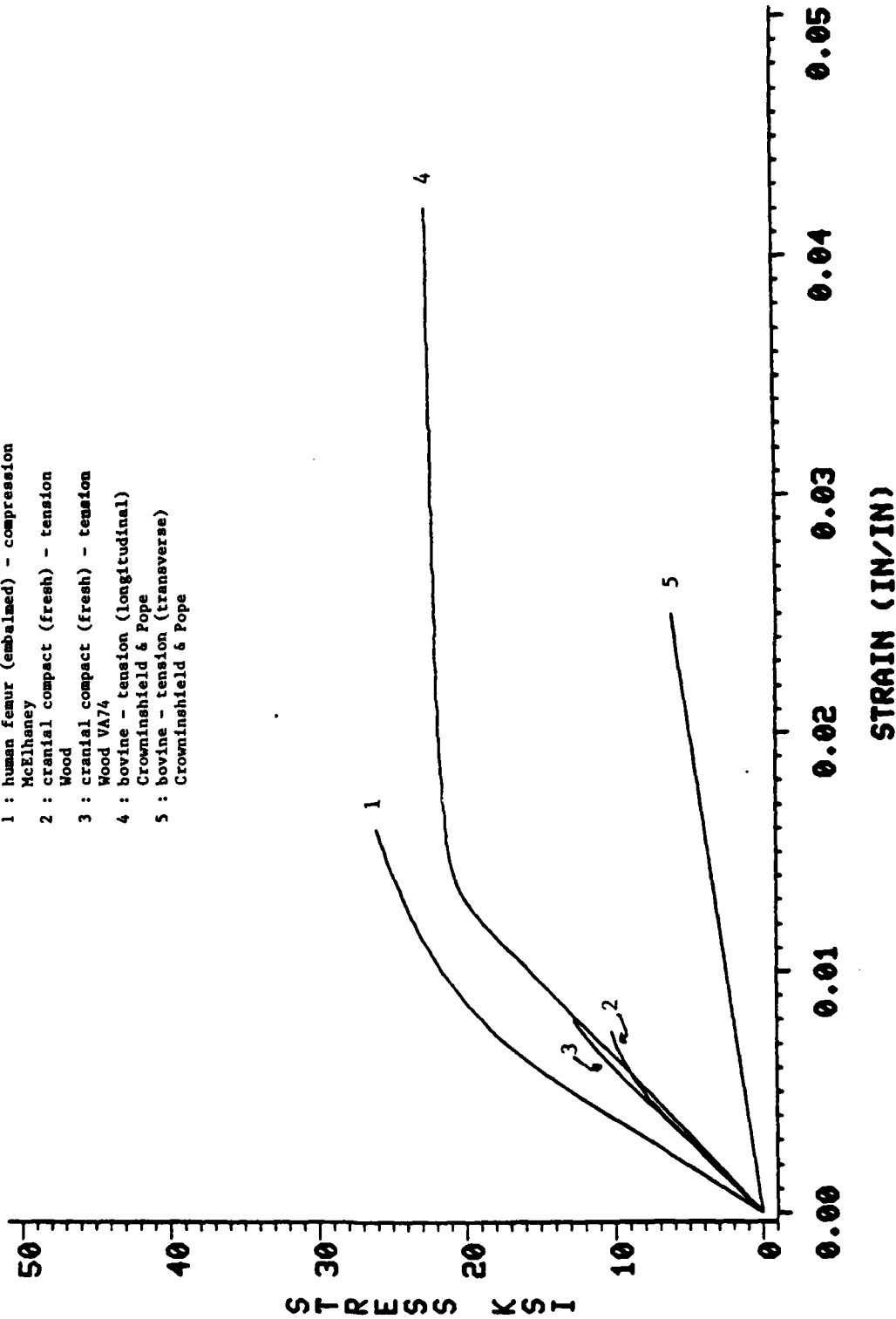


FIGURE 9

STRESS - STRAIN CURVES

STRAIN RATE = .001 / SEC

LEGEND

- 1 : human femur (embalmed) - compression
McElhaney
- 2 : cranial compact (fresh) - tension
Wood
- 3 : bovine - tension (longitudinal)
Crowninshield & Pope
- 4 : bovine - tension (transverse)
Crowninshield & Pope

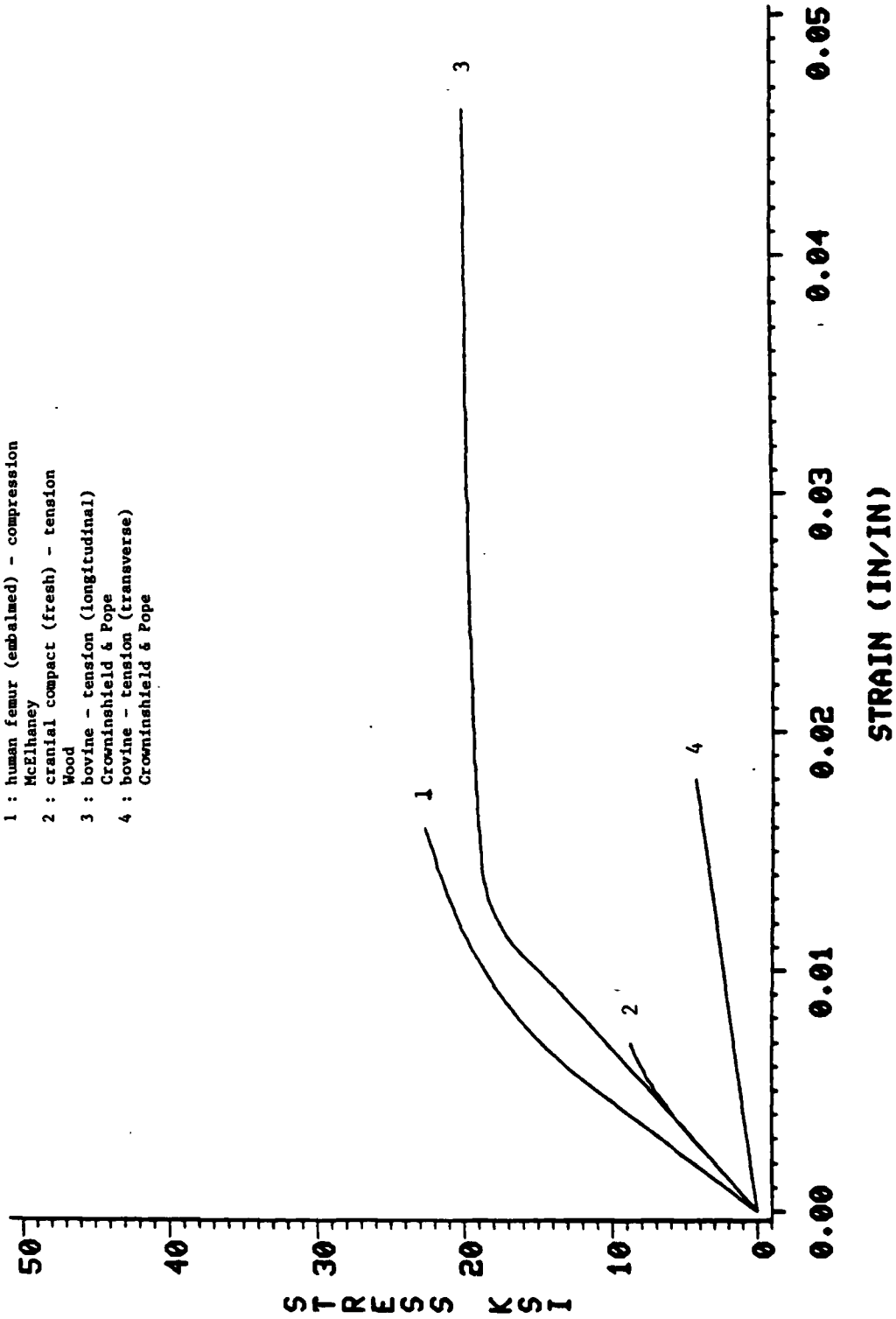


FIGURE 10

for the two by Wood, all represent different tests-embalmed human femur in compression; human cranial compact bone in tension; and bovine femur in tension for longitudinal and transverse directions. In that sense we are comparing apples to oranges. For this reason, these figures are shown only for general comparison. It had been hoped that a more direct comparison of similar data from the literature could have been made.

The cranial bone samples are generally weaker with stiffnesses bracketed by the compression results (stiffest and highest strength) and the tension results (less stiff and most plastic). Results for transverse tension samples are shown to be significantly lower in strength and stiffness. As mentioned earlier, the only significant plastic zone was found by Crowninshield and Pope for bovine bone in longitudinal tension. The results of Wood indicate that there is some consistency among data from tests on similar material and under similar conditions.

A study was carried out during the initial summer's work to assemble a small subset of this data, namely the ultimate strengths for fresh human compact bone from femurs tested statically in the longitudinal direction. The results are reproduced in Table 2. For the four modes of failure, the means and standard deviations are, respectively; tension 99.3/26/3, compression 174.4/34.2, shear 69.0/14.8 and bending 161.4/11.9, all $\times 10^6$ N/m². These figures illustrate the wide discrepancies which exist for even a narrowly defined subset of available data. The wide variations in these values could be due to a number of causes including the age and moisture content of the specimens and the test procedures utilized by the investigators.

E. The injury criteria, as first developed, were based on relationships between the ultimate stress and the strain rate (equation 1). This equation was then modified, based on a constant strain rate approximation, to yield a relation between ultimate

Table 2

Ultimate Strengths of Fresh Human Compact Bone From Femurs, Tested
 Statically and Stressed in the Longitudinal Direction
 ($\times 10^6$ N/m²/ $\times 10^3$ psi)

<u>Tension</u>	<u>Compression</u>	<u>Shear</u>	<u>Bending</u>
122/17.7*	159/23.1*	53.1/7.7*	152/22.0 ^{&}
86.5/12.5*	193/28.0 ^{\$}	82.4/11.9*	153/22.1 [¢]
133/19.3 ^{\$}	134.5/19.5 [¢]	71.6/10.4 ^{\$}	157/22.8*
76.2/11.0 ⁺	210.9/30.6 ⁺		164/23.8*
78.9/11.4 ⁺			181/26.2*

Compiled from various sources reported in (*) Reilly and Burstein (6), (+) Evans (4), (\$) Reilly and Burstein (18), (&) Mather (19) and (¢) Vose and Kubala (20).

stress and stress rate which could be calculated directly from the available stress time histories (equation 2). This solution is rather simplistic and not entirely satisfactory. A more systematic approach based on the Ramberg-Osgood equation (equation 3) has yielded more promising results.

The elemental question is, how to calculate strain when only the stress time history is known and the relation between stress and strain is strain rate dependent? It is clear on inspection that equation (3) does not invert or separate into convenient parts, particularly since the factors b and d are, in general, not integers. The approach taken has been one of a numerical approximation based on the following equation at time step i:

$$\epsilon_i = \frac{\sigma_i}{c} \frac{1}{\left(\frac{\epsilon_i - \epsilon_{i-1}}{\Delta t}\right)^d} + a\sigma_i^N \left(\frac{\epsilon_i - \epsilon_{i-1}}{\Delta t}\right)^b \quad (5)$$

Given the stress at a time step and the strain at the previous time step, equation (5) can be solved numerically for the current strain. This solution procedure is easily initialized by assuming zero stress and strain at time = 0. Solution of this equation has been successfully accomplished and Figure 11 shows an example stress strain response to the sinusoidal stress wave shown in Figure 12. The solution displays the expected hysteresis type response.

There were a number of stumbling blocks which had to be overcome in order to achieve this solution. The second term of equation (5), which accounts for the plastic portion of the response, can have a severe destabilizing effect on the solution procedure, depending on the values of the constants and the strain rate. It was therefore necessary to use a two step method based on the Regula-Falsi technique. An initial solution was generated using only the first term of the equation to be used as a starting point. With this initial guess as a basis the final solution

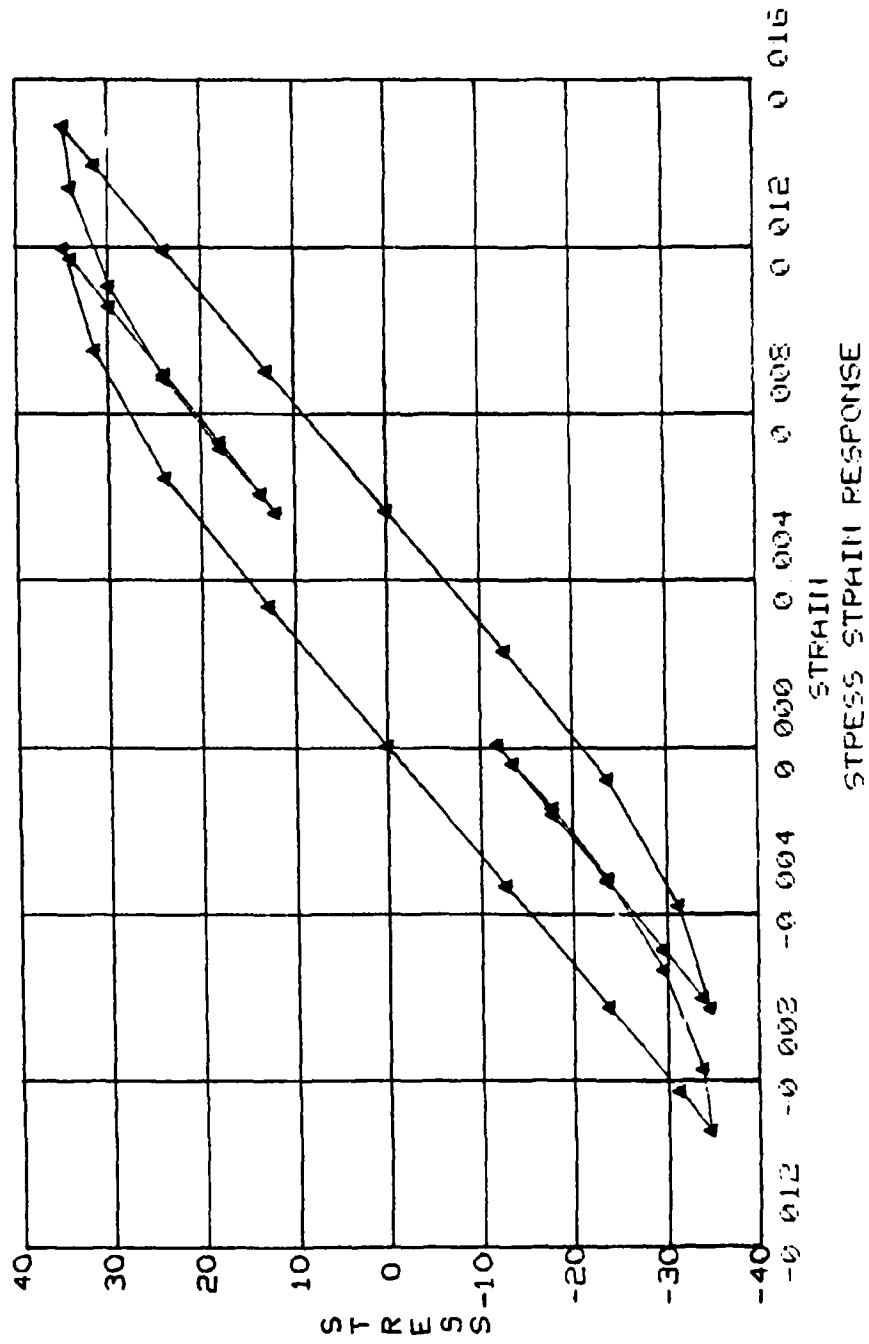
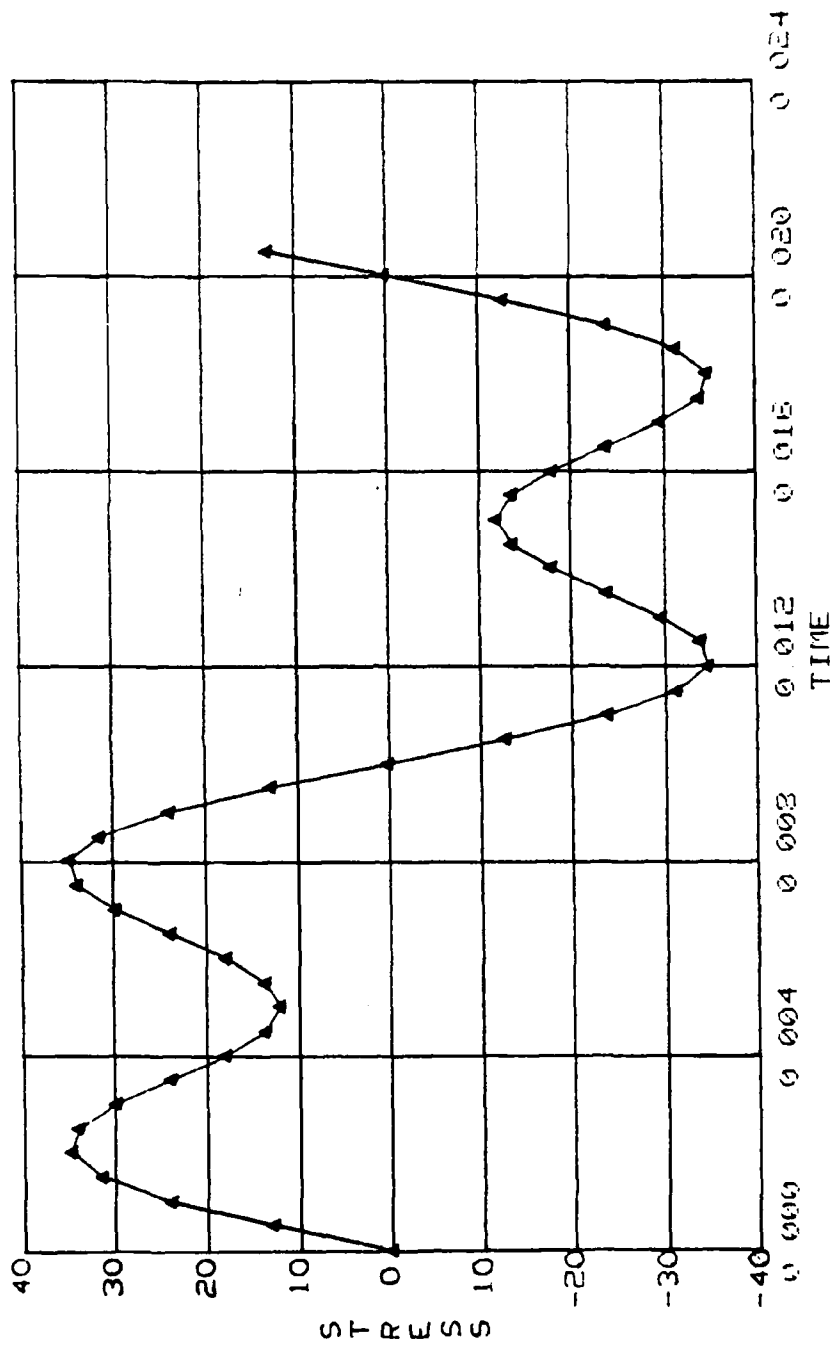


FIGURE 11



STRESS TIME HISTORY

FIGURE 12

was obtained using both terms. This two step procedure greatly reduced the incidence of non convergence, but, even using double precision, did not entirely eliminate these difficulties if the stress levels were too high.

A second problem was encountered due to the relaxation phase of the loading. Since b and d are not integers, a negative strain rate raised to such a power has no meaning. Furthermore, it can be assumed from standard hysteresis loop response that relaxation is essentially an elastic phenomenon. For these reasons, only the first term of the equation, with appropriate absolute values, is used during relaxation.

The final problem arose from the plastic deformation which may occur. Since equation (5) presumes a zero strain at zero stress, an appropriate plastic strain offset must be added to the solution after any relaxation. This plastic effect is clearly shown in Figure 9.

Using the solutions generated by equation (5) it is now possible to follow the strain and strain rate response of the system and to directly use equation (1) to predict ultimate stress. This procedure enables one to realistically use available ultimate stress values as a fracture criterion.

The point may be made that yield rather than ultimate stress would be a more conservative, and more meaningful injury criterion. Yield stress would unquestionably be more conservative, though some questions must first be answered. First, of course, yield point must be defined. This definition is arbitrary, depending on the material. For most engineering metals, yield is defined in terms of that stress which produces a plastic strain of .002. There is no such accepted standard for bone and all figures reported in the literature are for ultimate strengths. If a yield point definition could be agreed upon for bone, then yield strengths could be defined from the solution of equation (5).

A perhaps more interesting question is, what is the biological effect in living bone of exceeding the "yield strength?" No known investigation has been made into this phenomenon and its relation to injury. In fact, the overall level of understanding of bone fracture mechanics is quite low. For example, it is conceivable that reconstruction in living bone would overcome yielding effects without injury.

IV. CONCLUSIONS AND RECOMMENDATIONS

Given the available bone fracture data and the constraints of the current ATB model, a reasonable mechanism has been developed for utilizing a long bone fracture injury criterion for the ATB model. Specifically, this mechanism is based on a mathematical representation of the behavior of bone using the Ramberg-Osgood equation and on an approximation of the stress-time history of the bone generated using the ATB joint and contact forces. The specific injury criterion which should be employed is still under consideration. Options include: ultimate strength, which would be non conservative but for which considerable data exist; yield strength, which would be conservative, but for which a definition, and hence data, do not exist; some fraction of either of these as a factor of safety; or some other criterion such as maximum strain or strain energy. At present the ultimate strength, reduced by a factor of safety, is thought to be the best criterion. (The pulse criterion discussed in the first report (1) is felt to have value only as a comparison to existing automobile standards and hence to have no applicability to the pilot ejection problem.) The resulting injury criterion program remains in the "post processor" mode, i.e. it is not integrated into the ATB model, and this is the preferred configuration until the methodology becomes more fixed.

The Ramberg-Osgood equation appears to provide a very good means of mathematically modelling the stress-strain-strain rate behavior of bone. The fact that this equation, through proper optimization techniques, can be used to represent the three very different responses of bone shown here from the literature leads one to suspect that, whatever the "true" response of bone may be, it can be adequately described by equation (3). This flexibility, coupled with the dearth of good data on dynamic bone properties, was a prime motivating factor in taking this approach.

Further areas of inquiry point strongly in the direction of establishing a more comprehensive and consistent data base. The state-of-the-art in modeling has clearly outstripped both the available information on bone material properties and whole bone geometry, and the understanding of the mechanisms of bone fracture. A thorough understanding of these properties and mechanisms is necessary before any strides can be made toward formulating a more effective long bone injury criterion. The present criterion is designed to adapt to new information as it becomes available through modification of the constants in the Ramberg-Osgood equation and represents, in its present form, the most sophisticated injury criterion available.

BIBLIOGRAPHY

1. Hight, T., "Long bone injury criteria for use with the Articulated Total Body Model", AFAMRL-TR-81-3, 1981.
2. King, J.J., W.R.S. Fan and R.J. Vargovick, "Femur load injury criteria - a realistic approach," 17th Stapp Conference, paper #730984, pp. 509-525, 1973.
3. Viano, D.C., "Considerations for a Femur Injury Criterion," 21st Stapp Conference, October 19-21, New Orleans, paper #77095, pp. 443-373, 1977.
4. Evans, F.G., Stress and Strain in Bone, Charles C. Thomas, Springfield, IL. 1957.
5. Curry, J.D., "The mechanical properties of bone," Clin. Orth. and Rel. Res. Vol. 73, Nov-Dec. 1970, pp. 210-231.
6. Reilly, D.T. and A.H. Burstein, "The mechanical properties of cortical bone," JBJS, Vol. 56A No. 5, July 1974, pp. 1001-1022.
7. McElhaney, J.H., "Dynamic response of bone and muscle tissue," J. Appl. Physiology, Vol. 21, No. 4, pp. 1231-1236, 1966.
8. Minns, R.J., Bremble, G.K. and Campbell, J. "The geometrical properties of the human tibia," Tech. Note, J. Biomech. Vo. 8, pp. 253-255, 1975.
9. Piziali, R.L., Hight, T.K. and Nagel, D.A., "Geometric properties of human leg bones," J. Biomech., V. 13, pp. 881-885, 1980.
10. Panjabi, M.M., White, A.A. and Southwick, W.D., "Mechanical properties of bone as a function of rate of deformation," JBJS V. 55A, pp. 322-330, March 1973.
11. Wood, J.L., "Dynamic response of human cranial bone," J. Biomech. V. 4, pp. 1-12, 1971.
12. Crowninshield, R.D. and M.H. Pope, "The response of compact bone in tension at various strain rates," Annal. Biomed. Eng. V. 2, pp. 217-225, 1974.
13. Burstein, A.H., Currey, I.D., Frankel, V.H. and Reilly, D.T., "The ultimate properties of bone tissue: the effects of yielding," J. Biomech., V. 5, pp. 34-44, 1972.
14. Lewis, J.L. and Goldsmith, W., "The dynamic fracture and prefracture response of compact bone by split Hopkinson bar methods," J. Biomech. V. 8, No. 1, pp. 27-40, 1975.
15. Wright, T.M. and Hayes, W.C., "Tensile testing of bone over a wide range of strain rates: effects of strain rate, microstructure and density," Med. Biol. Eng. V. 14, No. 6, pp. 671-680, 1976.

16. McLellan, D.L., "Constitutive equations for mechanical properties of structural materials," AIAAJ, V. 5, No. 3, pp. 446-450, 1967.
17. Forsythe, G.E., Computer Methods for Mathematical Computations, Prentice-Hall, New Jersey, pp. 182-190, 1977.
18. Reilly, D.T. and A.H. Burstein, "The elastic and ultimate properties of compact bone tissue," J. Biomech., V. 8, pp. 393-405, 1975.
19. Mather, B.S., "The effect of variation in specific gravity and ash content on the mechanical properties of human compact bone," J. Biomech., V. 1, pp. 207-210, 1968.
20. Vose, G.P. and A.L. Kubala, "Bone strength - its relationship to x-ray determined ash content," Human Biol., V. 31, pp. 261-270, 1959.

PERSONNEL:

Principal Investigator: Timothy K. Hight
Department of Mechanical Engineering
and Materials Science
Duke University
Durham, N.C. 27705

Graduate Student: John F. Brandeau

(While no support for this graduate student was received from this grant, he did contribute much of the work reported here while working toward his M.S. degree. A thesis is currently in preparation in absentia with an expected completion date of May 1982).

PUBLICATIONS:

in preparation, "Mathematical modeling of the stress-strain-strain rate behavior of bone using the Ramberg-Osgood equation," T.K. Hight, J.F. Brandeau (probable journal - J. Biomech.),

in preparation, "Development of a strain rate dependent injury criterion for long bones," T.K. Hight, J.F. Brandeau (probable journal - J. Biomech.).

INTERACTIONS:

Seminar presented at WPAFB/AFAMRL August 7, 1981 "Improved long bone injury criteria for use with the ATB model."

APPENDIX

```

*****00000010
PROGRAM NAME : MYFIT *00000020
WRITTEN BY : J.F. BRANDEAU *00000030
COMPILER(S) : WATFIV (DOUBLE PRECISION) *00000040
----- *00000050
PURPOSE : FIT A FUNCTION OF N VARIABLES TO A SET OF M OBSERVED DATA *00000060
POINTS BY MINIMIZING THE DIFFERENCES BETWEEN OBSERVED AND PREDICTED *00000070
VALUES. THIS IS DONE USING SUBROUTINE FMIN (A COMBINATION OF GOLDEN *00000080
SEARCH & PARABOLIC INTERPOLATION METHODS) IN EACH OF THE COORDINATE *00000090
DIRECTIONS (X). THIS PROCEDURE STOPS WHEN ONE OR MORE TERMINATION *00000100
CRITERIA ARE MET : *00000110
1) ALL STEPS THROUGH TWO CONSECUTIVE SERIES ARE ABSOLUTELY LESS *00000120
THAN TOL. *00000130
2) FRACTIONAL CHANGE OF FUNCTION VALUE IS ABSOLUTELY LESS THAN *00000140
TOL THROUGH ONE SERIES. *00000150
3) MAXIMUM # OF SERIES IS EXCEEDED. *00000160
----- *00000170
PROGRAM VARIABLES : SUGGESTED VALUES TO START WITH IN ( ) *00000180
NSCALE : CONTROLS LENGTH OF INTERVAL SENT TO FMIN. THE VALUE OF X(I) *00000190
IS MULTIPLIED BY (1+NSCALE(I)) AND (1-NSCALE(I)). IF THE INTERVAL *00000200
INCLUDES ZERO, THIS ALLOWS FOR MAJOR CHANGES OF VARIABLE X(I) IN THE *00000210
ABSOLUTELY SMALLER DIRECTION. NOTE -- IF NSCALE(I) IS 0.0, THE *00000220
VALUE OF X(I) DOES NOT CHANGE IN THE PROGRAM. IN THIS WAY THE VAR- *00000230
IABLE X(I) CAN BE FIXED. (1.0 FOR ALL). *00000240
TOL : CONVERGENCE LIMIT FOR ALL CRITERIA (1.0D-7). *00000250
MAXFN : MAXIMUM # OF FUNCTION EVALUATIONS ALLOWED (25). *00000260
TIMES : MULTIPLIER FOR EACH ABSOLUTE ERROR (VALUE WHICH WILL MAKE THE *00000270
ERROR GREATER THAN 1.0). *00000280
UP : POWER WHICH (TIMES * ABS. ERROR) IS RAISED TO (2.0). *00000290
----- *00000300
PENALTY CONTROLS : PENALIZES FUNCTION IF ABS ERROR AT ANY POINT IS *00000310
GREATER THAN A SPECIFIED AMOUNT. *00000320
ERROR : ALLOWABLE ERROR WITHOUT PENALTY (1.5D-3). *00000330
Z : POWER WHICH (ABS. ERROR * TIMES) IS RAISED TO. THIS IS THE *00000340
PENALTY STEP FUNCTION (IF ABS. ERROR GT. ERROR ; PENALTY = ABS. *00000350
ERROR * TIMES ) ** Z, -- ELSE PENALTY = 0) THIS IS A RUNNING SUM. *00000360
(Z IS USUALLY 2.0). *00000370
----- *00000380
OTHERS VARIABLES : *00000390
X & COEFF : MANTISSA & EXPONENT OF VARIABLES. AFTER EACH SERIES X IS *00000400
NORMALIZED TO BETWEEN 1.0 & 10.0 TO ALLOW LARGE DELTA X'S AND AID *00000410

```

```

: CONVERGENCE CRITERIA ACCURACY. THE PRODUCT X(I) * COEFF(I) IS THE *00000510
: VALUE OF THE VARIABLE COEFFICIENT. *00000520
: *00000530
: STRESS : OBSERVED VALUES OF STRESS (KSI). *00000540
: *00000550
: RATE : OBSERVED VALUES OF STRAIN RATE (1/SEC). *00000560
: *00000570
: Y : OBSERVED VALUES OF STRAIN (IN/IN). *00000580
: *00000590
: YAPROX : PREDICTED VALUES OF STRAIN (IN/IN). *00000600
: ----- *00000610
: CCOMP OPTIONS : FORM C$OPTIONS CCOMP=????? *00000620
: *00000630
: 3 : OUTPUTS FUNCTION AND POINT OF EVALUATION FOR EACH EVALUATION. *00000640
: 4 : OUTPUT VALUE OF X & FMIN INTERVAL (AX & BX), PENALTIES AND *00000650
: FUNCTION VALUES FOR EACH COORDINATE DIRECTION. *00000660
: 5 : ALLOWS READING FROM INPUT LIST AT END OF PROGRAM FOR X & COEFF. *00000670
: PROGRAM ALWAYS READS THESE FROM FILE #5 REGARDLESS. FILE #1 *00000680
: VALUES WILL OVERRIDE THESE. *00000690
: 6 : OUTPUTS PENALTY SUMMATION AT TERMINATION. *00000700
: 7 : OUTPUTS LIST OF OBSERVED AND PREDICTED VALUES AT TERMINATION. *00000710
: ----- *00000720
: I/O REQUIREMENTS : *00000730
: *00000740
: FILE #1 : CONTAINS INPUT VARIABLES AS DESCRIBED ABOVE FOLLOWING $DATA *00000750
: CARD AT END OF PROGRAM. *00000760
: FILE #4 : CONTAINS OBSERVED VALUES. THE FIRST RECORD MUST BE A TITLE *00000770
: FILE #5 : CONTAINS X & COEFF ON TWO RECORDS. UPDATED AT TERMINATION. *00000780
: *00000790
: ***** *00000800
: $OPTIONS CCOMP=0 00000810
: ON ERROR GO TO 70 00000820
: IMPLICIT REAL * 8 (A-H, O-Z) 00000830
: REAL NSCALE(5) 00000840
: DIMENSION X(5), STRESS(60), Y(60), RATE(60), YAPROX(60) 00000850
: DIMENSION A(5), DIFF(60) 00000860
: INTEGER TITLE(20) 00000870
: COMMON STRESS, Y, RATE, YAPROX, M, SSQ, DIFF, TIMES 00000880
: COMMON /WEIGHT/ R, Z, ERROR, NUMPEN, PENAL 00000890
: COMMON /SEARCH/ EPS 00000900
: COMMON /PUN1/ COEFF(5), TOTAL, UP, A, I 00000910
: EXTERNAL FUNCT 00000920
: DATA NSCALE /0.90, 0.90, 1.00, 1.00, 1.00 / 00000930
: *00000940
: SET PROGRAM PARAMETERS 00000950
: *00000960
: Z = 2.000 ; UP = 2.000 ; R = 10.000 00000970
: TOL = 1.0D-7 ; ERROR = 1.0D-4 ; MAXFN = 20 00000980
: TIMES = 1.0D3 00000990
: *00001000

```


C		00001010
C	CALCULATE SQUARE ROOT OF MACHINE EPSILON FOR SUBROUTINE FMIN	00001020
C		00001030
	EPS = 1.0D00	00001040
10	EPS = EPS / 2.0D0	00001050
	TOL1 = 1.0D0 + EPS	00001060
	IF (TOL1 .GT. 1.0D00) GO TO 10	00001070
	EPS = DSQRT(EPS)	00001080
	KOUNT = NUNPEN = 0	00001090
	READ (1,*) N	00001100
	READ (5,*) (X(I), I = 1, N)	00001110
	READ (5,*) (COEFF(I), I = 1, N)	00001120
CS	READ (1,*) (X(I), I = 1, N)	00001130
CS	READ (1,*) (COEFF(I), I = 1, N)	00001140
	READ (4,95) TITLE	00001150
	M = 1	00001160
20	READ (4,*,END=25) RATE(M), Y(M), STRESS(M)	00001170
	M = M + 1	00001180
	GO TO 20	00001190
25	M = M - 1	00001200
	I = 1 ; POINT = X(I)	00001210
	WRITE (3,98) TITLE	00001220
	WRITE (3,105) TOL, UP, R, Z, ERROR	00001230
	WRITE (3,106) TIMES	00001240
	WRITE (3,108) (NSCALE(J), J = 1, N)	00001250
	WRITE (3,107) M	00001260
	DO 30 J = 1, N	00001270
	WRITE (3,110) J, X(J), COEFF(J)	00001280
30	A(J) = X(J) * COEFF(J)	00001290
	HOLD = FUNCT (POINT)	00001300
	WRITE (3,100) HOLD	00001310
	IF (NUNPEN .GT. 0) WRITE (3,125) NUNPEN, PENAL	00001320
	DO 60 K = 1, MAXPN	00001330
	IOUT = KOUNT2 = 0	00001340
	DO 50 I = 1, N	00001350
	DO 40 J = 1, N	00001360
40	A(J) = X(J) * COEFF(J)	00001370
C4	WRITE (3,110) I, X(I), COEFF(I)	00001380
	IF (X(I) .EQ. 0.0D0 .OR. NSCALE(I) .EQ. 0.0) THEN DO	00001390
	G = X(I)	00001400
	KOUNT2 = KOUNT2 + 1	00001410
	GO TO 50	00001420
	ENDIF	00001430
	IF (X(I) .GT. 0.0D0) THEN DO	00001440
	GLOW = X(I) * (1.0 - NSCALE(I))	00001450
	GHIGH = X(I) * (1.0 + NSCALE(I))	00001460
	IF (GLOW .LT. 1.0D-3 .AND. GLOW .GT. 0.0D0) GLOW = -GLOW	00001470
	ELSE DO	00001480
	GLOW = X(I) * (1.0 + NSCALE(I))	00001490
	GHIGH = X(I) * (1.0 - NSCALE(I))	00001500

```

        IF (GHIGH .GT. -1.0D-3 .AND. GHIGH .LT. 0.0D0) GHIGH = -GHIGH00001510
        ENDIF 00001520
C4 WRITE (3,115) GLOW, GHIGH 00001530
    G = FMIN (GLOW, GHIGH, FUNCT, TOL) 00001540
    TOTAL = FUNCT (G) 00001550
C4 WRITE (3,120) G, SSQ 00001560
C4 IF (NUMPEN .GT. 0) WRITE (3,125) NUMPEN, PENAL 00001570
C4 IF (TOTAL .GT. HOLD) THEN DO 00001580
C4 PRINT, 'BAD STEP?' 00001590
C4 ENDIF 00001600
C4 WRITE (3,160) TOTAL 00001610
45 IF (DABS(X(I)-G) .GT. TOL) IOUT = 1 00001620
    IF (DABS((TOTAL-HOLD)/HOLD) .LT. TOL) KOUNT2 = KOUNT2 + 1 00001630
    HOLD = TOTAL 00001640
50 X(I) = G 00001650
C 00001660
C NORMALIZE X'S TO BETWEEN 1.0 & 10.0. CORRECT CHANGE IN VALUE OF 00001670
C COEFF SO PRODUCT IS SAME. 00001680
C 00001690
    DO 55 I = 1, N 00001700
    IF (X(I) .EQ. 0.0D0) GO TO 55 00001710
    WHILE (DABS(X(I)) .LT. 1.0D0) DO 00001720
        X(I) = X(I) * 10.0D0 00001730
        COEFF(I) = COEFF(I) / 10.0D0 00001740
    END WHILE 00001750
55 CONTINUE 00001760
C 00001770
C CHECK AND UPDATE CONVERGENCE CRITERIA - TERMINATE IF MET 00001780
C 00001790
    IF (IOUT .EQ. 0) KOUNT = KOUNT + 1 00001800
    IF (KOUNT .EQ. 2) THEN DO 00001810
        WRITE (3,135) TOL 00001820
        WRITE (3,145) K 00001830
        GO TO 70 00001840
    ENDIF 00001850
    IF(KOUNT2 .EQ. N) THEN DO 00001860
        WRITE (3,140) TOL 00001870
        WRITE (3,145) K 00001880
        GO TO 70 00001890
    ENDIF 00001900
C 00001910
60 CONTINUE 00001920
    WRITE (3,147) MAXFN 00001930
70 DO 80 J = 1, N 00001940
80 WRITE (3,110) J, X(J), COEFF(J) 00001950
    REWIND 5 00001960
    WRITE (5,130) (X(I), I = 1, N) 00001970
    WRITE (5,130) (COEFF(I), I = 1, N) 00001980
    WRITE (3,150) TOTAL 00001990
    IF (NUMPEN .GT. 0) WRITE (3,155) NUMPEN 00002000

```

```

:6   B3 = PENLTY (1)                                00002010
:7   WRITE (3,165)                                  00002020
:7   DO 85 I = 1, M                                 00002030
:7   WRITE (3,170) I, STRESS(I), RATE(I), Y(I), YAPROX(I) 00002040
:7 85 CONTINUE                                     00002050
      STOP                                          00002060
      95 FORMAT (20A4)                              00002070
      98 FORMAT (1H1,'TITLE : ',20A4/)             00002080
      100 FORMAT (1H , ' AT START, FUNCT =', 1PD21.13//) 00002090
      105 FORMAT (1H-,'PROGRAM PARAMETERS:',/,T24,' TOL =',1PD9.2,/,T4, 00002100
      * 'ABS ERROR RAISED TO POWER:',1PD9.2,/,/, ' PENALTY CONTROLS:',/,T15, 00002110
      * 'MULTIPLIER =',1PD12.5,/,T20,'POWER =',1PD9.2,/,T14, 00002120
      * 'ERROR LEVEL =',1PD12.5//) 00002130
      106 FORMAT (1H , 'TIMES =',1PD10.3) 00002140
      107 FORMAT (1H ,I3,' POINTS WERE OBSERVED'//) 00002150
      108 FORMAT (' SCALE VECTOR IS: ',6F6.2/) 00002160
      110 FORMAT (' ----- X(',I2,') =',1PD21.13,' * ',1PD8.1) 00002170
      115 FORMAT (' SEARCH RANGE ',1PD12.5,' TC ',1PD12.5) 00002180
      120 FORMAT (1H , 'MINIMUM POINT @ G = ',1PD21.13,/,T15, 00002190
      * ' SSQ =', 1PD21.13) 00002200
      125 FORMAT (I4,' CONSTRAINTS VIOLATED -- PENALTY =',1PD13.5) 00002210
      130 FORMAT (5D21.13) 00002220
      135 FORMAT (' CONVERGENCE OF ALL X'S TO WITHIN',1PD12.3) 00002230
      140 FORMAT (' DELTA FUNCT HAS BEEN LESS THAN',1PD12.5,' FOR 5 STEPS') 00002240
      145 FORMAT (1H-,'FINAL RESULT REACHED AFTER ',I3,' SERIES') 00002250
      147 FORMAT (1H-,'NO CONVERGENCE AFTER ',I3,' SERIES') 00002260
      150 FORMAT (1H-,'AT FINISH, FUNCT =',1PD21.13) 00002270
      155 FORMAT ('OAT FINISH, ',I3,' CONSTRAINTS VIOLATED:',/, 00002280
      * 1H ,T6,'#',T10,'STRESS(KSI)',T28,'RATE', 00002290
      * T38,'PRED. STRAIN',T52,'OBS. STRAIN',T69,'DIFF'//) 00002300
      160 FORMAT ('-FUNCT =',1PD21.13/) 00002310
      165 FORMAT (1H0,/,T4,'#',T15,'STRESS(KSI)',T34,'RATE',T45,'OBS. STRAIN' 00002320
      * ,T60,'PRED. STRAIN'//) 00002330
      170 FORMAT (1H ,T3,I2,T10,4(1PD15.3)) 00002340
      END 00002350
      DOUBLE PRECISION FUNCTION FUNCT(POINT) 00002360
      IMPLICIT REAL * 8 (A-H, O-Z) 00002370
      DIMENSION X(5),STRESS(60),Y(60),RATE(60),YAPROX(60),DIFF(60) 00002380
      COMMON STRESS, Y, RATE, YAPROX, M, SSQ, DIFF, TIMES 00002390
      COMMON /FUN1/ COEFF(5), TOTAL, UP, X, II 00002400
      SSQ = 0.0D0 00002410
      X(II) = POINT * COEFF(II) 00002420
      00002430
      CALCULATE PREDICTED STRAINS AND SUM OF ERRORS. 00002440
      00002450
      DO 10 I = 1, M 00002460
      IF (STRESS(I) .EQ. 0.0D0) THEN DO 00002470
      YAPROX(I) = DIFF(I) = 0.0DC 00002480
      GO TO 10 00002490
      ENDIF 00002500

```

```

TEMP2 = STRESS(I) / X(5) / RATE(I) ** X(4)
TEMP3 = RATE(I) ** X(2) * STRESS(I) ** X(3)
YAPROX(I) = TEMP2 + X(1) * TEMP3
:
:
: APPLY WEIGHTING FACTORS TO RESIDUAL
:
: DIF = DABS(YAPROX(I) - Y(I))
:
: SSQ = SSQ + (DIF * TIMES) ** UP
DIFF(I) = DIF
10 CONTINUE
TOTAL = SSQ + PENLTY (0)
FUNCT = TOTAL
33 WRITE (3,100) POINT, FUNCT
RETURN
100 FORMAT (1H , 'AT FUNCT, POINT =', 1PD21.13, ' FUNCT =', 1PD21.13)
END
DOUBLE PRECISION FUNCTION PENLTY (IDUM)
IMPLICIT REAL * 8 (A-H, O-Z)
DIMENSION STRESS(60), Y(60), RATE(60), YAPROX(60), DIFF(60)
COMMON STRESS, Y, RATE, YAPROX, M, SSQ, DIFF, TIMES
COMMON /WEIGHT/ R, Z, ERROR, KOUNT, HOLD
HOLD = 0.0D0
KOUNT = 0
:
:
: CALCULATE PENALTY IF ABS. ERROR GREATER THAN SPECIFIED MAX.
:
: DO 10 I = 1, M
B3 = DIFF(I)
IF (B3 .GT. ERROR) THEN DO
HOLD = HOLD + (B3 * TIMES) ** 2
KOUNT = KOUNT + 1
IF (IDUM .EQ. 1), THEN DO
WRITE (3,100) KOUNT, STRESS(I), RATE(I), YAPROX(I), Y(I), B3
ENDIF
ENDIF
10 CONTINUE
PENLTY = HOLD = HOLD * R
100 FORMAT (1H , I5, 5(1PD14.5))
RETURN ; END
:SOPTIONS NOLIST
DOUBLE PRECISION FUNCTION PHIN(AX, BX, F, TOL)
DOUBLE PRECISION AX, BX, F, TOL
DOUBLE PRECISION A, B, C, D, E, EPS, XM, P, Q, R, TOL1, TOL2, U, V, W
DOUBLE PRECISION PU, PV, FW, FX, X
DOUBLE PRECISION DABS, DSQRT, DSIGN
COMMON /SEARCH/ EPS
C = 0.5D0*(3. - DSQRT(5.0D0))
A = AX
B = BX
00002510
00002520
00002530
00002540
00002550
00002560
00002570
00002580
00002590
00002600
00002610
00002620
00002630
00002640
00002650
00002660
00002670
00002680
00002690
00002700
00002710
00002720
00002730
00002740
00002750
00002760
00002770
00002780
00002790
00002800
00002810
00002820
00002830
00002840
00002850
00002860
00002870
00002880
00002890
00002900
00002910
00002920
00002930
00002940
00002950
00002960
00002970
00002980
00002990
00003000

```

V = A + C*(B - A)	00003010
W = V	00003020
X = V	00003030
E = 0.000	00003040
FX = F(X)	00003050
FV = FX	00003060
FW = FX	00003070
20 XM = 0.500*(A + B)	00003080
TOL1 = EPS*DABS(X) + TOL/3.000	00003090
TOL2 = 2.000*TOL1	00003100
IF (DABS(X - XM) .LE. (TOL2 - 0.500*(B - A))) GO TO 90	00003110
IF (DABS(E) .LE. TOL1) GO TO 40	00003120
R = (X - W)*(FX - FV)	00003130
Q = (X - V)*(FX - FW)	00003140
P = (X - V)*Q - (X - W)*R	00003150
Q = 2.0000*(Q - R)	00003160
IF (Q .GT. 0.000) P = -P	00003170
Q = DABS(Q)	00003180
R = E	00003190
E = D	00003200
30 IF (DABS(P) .GE. DABS(0.500*Q*R)) GO TO 40	00003210
IF (P .LE. Q*(A - X)) GO TO 40	00003220
IF (P .GE. Q*(B - X)) GO TO 40	00003230
D = P/Q	00003240
U = X + D	00003250
IF ((U - A) .LT. TOL2) D = DSIGN(TOL1, XM - X)	00003260
IF ((B - U) .LT. TOL2) D = DSIGN(TOL1, XM - X)	00003270
GO TO 50	00003280
40 IF (X .GE. XM) E = A - X	00003290
IF (X .LT. XM) E = B - X	00003300
D = C*E	00003310
50 IF (DABS(D) .GE. TOL1) U = X + D	00003320
IF (DABS(D) .LT. TOL1) U = X + DSIGN(TOL1, D)	00003330
FU = P(U)	00003340
IF (FU .GT. FX) GO TO 60	00003350
IF (U .GE. X) A = X	00003360
IF (U .LT. X) B = X	00003370
V = W	00003380
FV = FW	00003390
W = X	00003400
PW = FX	00003410
X = U	00003420
FX = FU	00003430
GO TO 20	00003440
60 IF (U .LT. X) A = U	00003450
IF (U .GE. X) B = U	00003460
IF (FU .LE. FW) GO TO 70	00003470
IF (W .EQ. X) GO TO 70	00003480
IF (FU .LE. FV) GO TO 80	00003490
IF (V .EQ. X) GO TO 80	00003500

```
IF (V .EQ. W) GO TO 80
GO TO 20
70 V = W
   FV = FW
   W = U
   FW = FU
   GO TO 20
80 V = U
   FV = FU
   GO TO 20
90 PMIN = X
   RETURN
END
```

00003510
00003520
00003530
00003540
00003550
00003560
00003570
00003580
00003590
00003600
00003610
00003620
00003630
00003640
00003650
00003660
00003670

DATA
5

3.85D0	-2.35D0	4.52D0	2.106D0	1.694D0
1.0D-68	1.0D0	1.0D1	1.0D-2	1.0D3

```

*****00000010
C          *00000020
C PROGRAM NAME : SOLVALL *00000030
C WRITTEN BY : J.F. BRANDEAU *00000040
C COMPILER(S) : WATFIV (DOUBLE PRECISION) *00000050
C          *00000060
C -----
C PURPOSE : TO CONVERT EQUATION STRAIN = F(RATE, STRESS) TO EQUATION *00000070
C STRESS = F(RATE, STRAIN) USING SUBROUTINE ZEROIN TO SOLVE FOR ROOT OF *00000080
C THE EQUATION. CAN HANDLE UP TO SIX SETS OF COEFFICIENTS X TO PRODUCE *00000090
C UP TO SIX DATA PAIRS FOR SAS TO GRAPH ON A SINGLE GRAPH. *00000100
C          *00000110
C -----
C VARIABLES : *00000120
C          *00000130
C KGRAPS : NUMBER OF SETS OF COEFFICIENTS TO BE TAKEN FROM DATA. *00000140
C          *00000150
C RATE : STRAIN RATE TO BE USED. *00000160
C          *00000170
C KEPS : NUMBER OF POINTS TO BE GENERATED FOR EACH UNIQUE VECTOR X. *00000180
C          *00000190
C TOL : CONVERGENCE CRITERION FOR ZEROIN. *00000200
C          *00000210
C A, B, N, D, C (I) - SET OF COEFFICIENTS FOR EACH CURVE. THE I'TH *00000220
C ELEMENT OF EACH CONSTITUTES ONE SET OF COEFFICIENTS X. *00000230
C          *00000240
C SIGMAX : MAXIMUM VALUE OF STRESS (KSI) EXPECTED FOR EACH SET OF *00000250
C COEFFICIENTS X. THIS IS THE HIGH LIMIT SENT TO ZEROIN. THESE MAY *00000260
C BE ADJUSTED DOWNWARD BY THE PROGRAM IF NEEDED TO PREVENT UNDER/OVER *00000270
C FLOWS. IF THE VALUE OF SIGMAX(I) IS NOT HIGH ENOUGH FOR COEFFICIENT *00000280
C X(I), THE CURVE WILL BE FLATTENED AT THE HIGHER STRESSES. *00000290
C          *00000300
C EPSMAX : MAXIMUM VALUE OF STRAIN FOR WHICH EACH CURVE IS TO BE *00000310
C EVALUATED. *00000320
C          *00000330
C -----
C I/O REQUIREMENTS : *00000340
C          *00000350
C FILE #1 : ALL INPUT FROM ABOVE, FOLLOWING $DATA CARD. *00000360
C FILE #6 : OUTPUT OF STRESS-STRAIN PAIRS FOR USE BY SAS (LRECL=130). *00000370
C          *00000380
C          *00000390
C -----
C CCOMP OPTIONS (FORM CSOPTIONS CCOMP=?????) : *00000400
C          *00000410
C 4 : OUTPUT OF RETURNED STRESS VALUES. *00000420
C          *00000430
C *****00000440
C CSOPTIONS CCOMP=0 00000450
C   IMPLICIT REAL * 8 (A-H, N, O-Z) 00000460
C   EXTERNAL PUNCT 00000470
C   DIMENSION A(6), B(6), C(6), D(6), N(6), SIGMAX(6), EPSMAX(6) 00000480
C   DIMENSION EPS(50,6), STRESS(50,6) 00000490
C   COMMON A, B, C, D, N, TLOG, RATE, KOUNT, J, EPS, RATEB, RATED 00000500

```

COMMON /SUB2/ GEPS	00000510
1	00000520
2 CALCULATE MACHINE EPSILON	00000530
3	00000540
4 GEPS = 1.0D0	00000550
5 4 GEPS = GEPS/2.0D0	00000560
6 TOL1 = 1.0D0 + GEPS	00000570
7 IF (TOL1 .GT. 1.0D0) GO TO 4	00000580
8	00000590
9 READ (1,*) KGRAFS, RATE, KEPS, TOL	00000600
10 TEPS = 0.0D0	00000610
11 DO 10 I = 1, KGRAFS	00000620
12 READ (1,*) A(I), B(I), N(I), D(I), C(I), SIGMAX(I), EPSMAX(I)	00000630
13	00000640
14 CHECK FOR POSSIBLE OVERFLOW FOR HIGH N'S	00000650
15 ALTER SIGMAX DOWNWARD IF NECESSARY	00000660
16	00000670
17 RATEB = DLOG10(RATE ** B(I))	00000680
18 6 AX = N(I) * DLOG10(SIGMAX(I)) + RATEB	00000690
19 IF (AX .GT. 75.0D0) THEN DO	00000700
20 SIGMAX(I) = SIGMAX(I) - 0.5D0	00000710
21 GO TO 6	00000720
22 ENDIF	00000730
23 10 CONTINUE	00000740
24	00000750
25 BEGIN MAIN LOOP	00000760
26	00000770
27 DO 30 KOUNT = 1, KGRAFS	00000780
28 STRESS(1,KOUNT) = EPS(1,KOUNT) = 0.0D0	00000790
29 BX = SIGMAX(KOUNT)	00000800
30 DEPS = EPSMAX(KOUNT) / DFLOAT(KEPS-1)	00000810
31 RATED = RATE ** D(KOUNT)	00000820
32 RATEB = RATE ** B(KOUNT)	00000830
33	00000840
34 WATCHING FOR VALUES OF N THAT WILL CAUSE OVERFLOW OR UNDERFLOW	00000850
35	00000860
36 IF (A(KOUNT) .EQ. 0.0D0) THEN DO	00000870
37 TLOG = -80.0D0	00000880
38 ELSE DO	00000890
39 TLOG = DLOG10(A(KOUNT))	00000900
40 ENDIF	00000910
41	00000920
42 INNER LOOP FOR EACH SOLUTION	00000930
43	00000940
44 DO 20 J = 2, KEPS	00000950
45 EPS(J,KOUNT) = DFLOAT(J-1) * DEPS	00000960
46 AX = STRESS(J-1,KOUNT)	00000970
47 IF (J .EQ. 2) AX = (C(KOUNT) * EPS(J,KOUNT) * RATED) / 1.3D0	00000980
48 STRESS(J,KOUNT) = ZEROIN(AX, BX, FUNCT, TOL)	00000990
49 PRINT, 'FROM ZEROIN, STRESS', J, ' = ', STRESS(J,KOUNT)	00001000

20	CONTINUE	00001010
30	CONTINUE	00001020
	DO 40 I = 1, KEPS	00001030
40	WRITE (6,100) (EPS(I,J), STRESS(I,J), J = 1, KGRAPS)	00001040
50	STOP	00001050
100	FORMAT (10(1PD13.5))	00001060
	END	00001070
	DOUBLE PRECISION FUNCTION FUNCT(STRESS)	00001080
	IMPLICIT REAL * 8 (A-H, N, O-Z)	00001090
	DIMENSION A(6), B(6), C(6), D(6), N(6), EPS(50,6)	00001100
	COMMON A, B, C, D, N, TLOG, RATE, KOUNT, J, EPS, RATEB, RATED	00001110
	IF (STRESS.EQ. 0.0D0) THEN DO	00001120
	FUNCT = -EPS(J,KOUNT)	00001130
	RETURN	00001140
	ENDIF	00001150
	TEMP1 = STRESS / (C(KOUNT) * RATED)	00001160
	IF (STRESS .GT. 1.0D-1) GO TO 10	00001170
	IF ((N(KOUNT) * DLOG10(STRESS)) .LT. -60.0D0) THEN DO	00001180
	HOLD = -25.0D0	00001190
	GO TO 15	00001200
	ENDIF	00001210
10	TEMP2 = STRESS ** N(KOUNT) * RATEB	00001220
	HOLD = DLOG10(TEMP2)	00001230
15	IF ((HOLD + TLOG) .LT. -17.0D0) THEN DO	00001240
	FUNCT = TEMP1 - EPS(J,KOUNT)	00001250
	ELSE DO	00001260
	FUNCT = (TEMP1 + A(KOUNT) * TEMP2) - EPS(J,KOUNT)	00001270
	ENDIF	00001280
25	RETURN : END	00001290
!\$OPTIONS NOLIST		00001300
	DOUBLE PRECISION FUNCTION ZEROIN(AX,BX,F,TOL)	00001310
	DOUBLE PRECISION AX,BX,F,TCL	00001320
	DOUBLE PRECISION A,B,C,D,E,EPS,FA,FB,FC,TOL1,XM,P,Q,R,S	00001330
	DOUBLE PRECISION DABS,DSIGN	00001340
	COMMON /SUB2/ EPS	00001350
	A = AX	00001360
	B = BX	00001370
	FA = F(A)	00001380
	FB = F(B)	00001390
20	C = A	00001400
	FC = FA	00001410
	D = B - A	00001420
	E = D	00001430
30	IF (DABS(FC) .GE. DABS(FB)) GO TO 40	00001440
	A = B	00001450
	B = C	00001460
	C = A	00001470
	FA = FB	00001480
	FB = FC	00001490
	FC = FA	00001500

40	TOL1 = 2.0D0*EPS*DABS(B) + 0.5D0*TOL	00001510
	XN = .5*(C - B)	00001520
	IF (DABS(XN) .LE. TOL1) GO TO 90	00001530
	IF (FB .EQ. 0.0D0) GO TO 90	00001540
	IF (DABS(E) .LT. TOL1) GO TO 70	00001550
	IF (DABS(FA) .LE. DABS(FB)) GO TO 70	00001560
	IF (A .NE. C) GO TO 50	00001570
	S = FB/FA	00001580
	P = 2.0D0*XN*S	00001590
	Q = 1.0D0 - S	00001600
	GO TO 60	00001610
50	Q = FA/PC	00001620
	R = FB/PC	00001630
	S = FB/FA	00001640
	P = S*(2.0D0*XN*Q*(Q - R) - (B - A)*(R - 1.0D0))	00001650
	Q = (Q - 1.0D0)*(R - 1.0D0)*(S - 1.0D0)	00001660
60	IF (P .GT. 0.0D0) Q = -Q	00001670
	P = DABS(P)	00001680
	IF ((2.0D0*P) .GE. (3.0D0*XN*Q - DABS(TOL1*Q))) GO TO 70	00001690
	IF (P .GE. DABS(0.5D0*E*Q)) GO TO 70	00001700
	E = D	00001710
	D = P/Q	00001720
	GO TO 80	00001730
70	D = XN	00001740
	E = D	00001750
80	A = B	00001760
	FA = FB	00001770
	IF (DABS(D) .GT. TOL1) B = B + D	00001780
	IF (DABS(D) .LE. TOL1) B = B + DSIGN(TOL1, XN)	00001790
	FB = F(B)	00001800
	IF ((FB*(FC/DABS(FC))) .GT. 0.0D0) GO TO 20	00001810
	GO TO 30	00001820
90	ZEROIN = B	00001830
	RETURN	00001840
	END	00001850
		00001860
		00001870
SDATA		00001880
5	1.0D0 25 1.0D-7	00001890
	5.1183D-13 -3.74D-1 6.5342D0 6.707D-2 3.5514D3 50.0D0 .016D0	00001900
	36.777D-13 -4.1267D-1 7.6617D0 5.67034D-2 2.20498D3 25.0D0 .0060D0	00001910
	3.052D-13 -1.426D-1 7.632D0 6.287D-2 2.284D3 20.0 .0070D0	00001920
	3.709D-68 -2.3357D0 45.2472D0 1.7977D-2 1.694D3 45.0D0 .036D0	
	1.0D0 1.0D0 1.0D0 1.34946D-2 2.6354D2 15.0D0 .030D0	

```

*****00000010
:                                *00000020
: PROGRAM NAME : BAKSOLV          *00000030
: WRITTEN BY : J.F. BRANDEAU     *00000040
: COMPILER(S) : WATFIV (DOUBLE PRECISION) *00000050
:                                *00000060
:-----
: PURPOSE : CONVERT EQUATION STRAIN = F(RATE, STRESS) TO EQUATION *00000070
: STRESS = F(RATE, STRAIN) BY USING SUBROUTINE ZEROIN TO SOLVE FOR ROOT *00000080
: OF EQUATION. PROGRAM CHECKS RATE TO DETERMINE EACH CHANGE AND *00000090
: PREVENT UNDER / OVER FLOW. 6 UNIQUE RATES ARE ALLOWED IN THE INPUT *00000100
: LIST FROM FILE #4.            *00000110
:                                *00000120
:-----
: VARIABLES :                    *00000130
:                                *00000140
: SIGMAX : GREATER THAN MAXIMUM VALUE OF STRESS EXPECTED FOR EACH *00000150
: OBSERVED RATE. THIS IS THE UPPER LIMIT FOR ROOT SEARCH, AND IS *00000160
: ADJUSTED TO PREVENT OVER /UNDEF FLOW. IF THIS IS TOO LOW THE CURVE *00000170
: FOR THAT STRAIN RATE WILL BE FLATTENED AT THE TOP. *00000180
:                                *00000190
: TOL : CONVERGENCE CRITERION FOR ZEROIN. *00000200
:                                *00000210
: X & COEFF : MANTISSA AND EXPONENT OF COEFFICIENT VECTOR. PROGRAM *00000220
: COMBINES BOTH INTO X. *00000230
:                                *00000240
:-----
: I / O REQUIREMENTS : *00000250
:                                *00000260
: FILE #4 : OBSERVED VALUES OF DATA AS USED FOR OTHER PROGRAMS. *00000270
: TITLE MUST BE ON FIRST RECORD, FOLLOWED BY ONE OBSERVATION *00000280
: PER RECORD; STRAIN RATE, STRAIN AND STRESS (IN ORDER). *00000290
: FILE #5 : X & COEFF. X IS THE MANTISSA AND COEFF THE EXPONENT OF *00000300
: THE VARIABLE COEFFICIENTS. PRODUCT X * COEFF SHOULD EQUAL *00000310
: THE COEFFICIENT. *00000320
: FILE #6 : OUTPUT OF POINTS FOR USE BY SAS. *00000330
:                                *00000340
:-----
:*****00000350
: IMPLICIT REAL * 8 (A-H, O-Z)    00000360
: EXTERNAL FUNCT                  00000370
: DIMENSION X(5), SIGMAX(6), COEFF(5) 00000380
: COMMON X, RATE, EPS, RATEB, RATED 00000390
: INTEGER TITLE(20)              00000400
: DATA SIGMAX / 50.0, 50.0, 50.0, 50.0, 50.0, 50.0/ 00000410
: TOL = 1.0D-7                   00000420
: READ (5,*) X                    00000430
: READ (5,*) COEFF                00000440
: READ (4,200) TITLE              00000450
: WRITE (3,300) TITLE             00000460
: DO 5 J = 1, 5                   00000470
: X(J) = X(J) * COEFF(J)         00000480
: 5 WRITE (3,100) X(J)           00000490
: RATE1 = 0.0                    00000500

```

KP = 0	00000510
10 READ (4,*,END=50) RATE, EPS, STRESS	00000520
HAS STRAIN RATE CHANGED IN INPUT LIST?	00000530
IF (RATE .NE. RATE1) THEN DO	00000540
RATE1 = RATE	00000550
KP = KP + 1	00000560
KOUNT = 0	00000570
	00000580
	00000590
	00000600
CORRECT SIGMAX TO PREVENT OVER / UNDER FLOW.	00000610
	00000620
RATEB = DLOG10 (RATE ** X(2))	00000630
15 AX = X(3) * DLOG10 (SIGMAX(KP)) + RATEB	00000640
IF (AX .GT. 75.0D0) THEN DO	00000650
SIGMAX(KP) = SIGMAX(KP) - 0.5D0	00000660
GO TO 15	00000670
ENDIF	00000680
	00000690
RATEB = RATE ** X(2)	00000700
RATED = RATE ** X(4)	00000710
ENDIF	00000720
KOUNT = KOUNT + 1	00000730
IF (EPS .EQ. 0.0D0) THEN DO	00000740
SIG1 = 0.0D0	00000750
ELSE DO	00000760
AX = SIG1	00000770
BX = SIGMAX(KP)	00000780
IF (KOUNT .EQ. 2) AX = STRESS / 1.3	00000790
SIG1 = ZEROIN (AX, BX, FUNCT, TOL)	00000800
ENDIF	00000810
WRITE (6,400) RATE, STRESS, EPS, SIG1	00000820
GO TO 10	00000830
50 STOP	00000840
100 FORMAT (1H ,1PD21.13)	00000850
200 FORMAT (20A4)	00000860
300 FORMAT (1H , 'TITLE : ',20A4)	00000870
400 FORMAT (1H ,5(1PD13.5))	00000880
END	00000890
DOUBLE PRECISION FUNCTION FUNCT(STRESS)	00000900
IMPLICIT REAL * 8 (A-H, O-Z)	00000910
DIMENSION X(5)	00000920
COMMON X, RATE, EPS, RATEB, RATED	00000930
TEMP1 = STRESS / X(5) / RATED	00000940
FUNCT = TEMP1 + X(1) * RATEB * STRESS ** X(3) - EPS	00000950
RETURN	00000960
END	00000970
:\$OPTIONS NOLIST	00000980
DOUBLE PRECISION FUNCTION ZEROIN(AX,BX,P,TOL)	00000990
DOUBLE PRECISION AX,BX,P,TOL	00001000

DOUBLE PRECISION	A, B, C, D, E, EPS, FA, FB, FC, TOL1, XM, P, Q, R, S	00001010
DOUBLE PRECISION	DABS, DSIGN	00001020
	EPS = 1.000	00001030
10	EPS = EPS/2.000	00001040
	TOL1 = 1.000 + EPS	00001050
	IF (TOL1 .GT. 1.000) GO TO 10	00001060
	A = AX	00001070
	B = BX	00001080
	FA = F(A)	00001090
	FB = F(B)	00001100
20	C = A	00001110
	FC = FA	00001120
	D = B - A	00001130
	E = D	00001140
30	IF (DABS(FC) .GE. DABS(FB)) GO TO 40	00001150
	A = B	00001160
	B = C	00001170
	C = A	00001180
	FA = FB	00001190
	FB = FC	00001200
	FC = FA	00001210
40	TOL1 = 2.000*EPS*DABS(B) + 0.500*TOL	00001220
	XM = .5*(C - B)	00001230
	IF (DABS(XM) .LE. TOL1) GO TO 90	00001240
	IF (FB .EQ. 0.000) GO TO 90	00001250
	IF (DABS(E) .LT. TOL1) GO TO 70	00001260
	IF (DABS(FA) .LE. DABS(FB)) GO TO 70	00001270
	IF (A .NE. C) GO TO 50	00001280
	S = FB/FA	00001290
	P = 2.000*XM*S	00001300
	Q = 1.000 - S	00001310
	GO TO 60	00001320
50	Q = FA/FC	00001330
	R = FB/FC	00001340
	S = FB/FA	00001350
	P = S*(2.000*XM*Q*(Q - R) - (B - A)*(R - 1.000))	00001360
	Q = (Q - 1.000)*(R - 1.000)*(S - 1.000)	00001370
60	IF (P .GT. 0.000) Q = -Q	00001380
	P = DABS(P)	00001390
	IF ((2.000*P) .GE. (3.000*XM*Q - DABS(TOL1*Q))) GO TO 70	00001400
	IF (P .GE. DABS(0.500*E*Q)) GO TO 70	00001410
	E = D	00001420
	D = P/Q	00001430
	GO TO 80	00001440
70	D = XM	00001450
	E = D	00001460
80	A = B	00001470
	FA = FB	00001480
	IF (DABS(D) .GT. TOL1) B = B + D	00001490
	IF (DABS(D) .LE. TOL1) B = B + DSIGN(TOL1, XM)	00001500

```
FB = F(B)
IF ((FB*(FC/DABS(FC))) .GT. 0.0D0) GO TO 20
GO TO 30
90 ZEROIN = B
RETURN
END
```

```
00001510
00001520
00001530
00001540
00001550
00001560
```

```

*****00000010
PROGRAM NAME : MAP2 *00000020
WRITTEN BY : J.F. BRANDEAU *00000030
COMPILER(S) : WATFIV (DOUBLE PRECISION) *00000040
----- *00000050
PURPOSE : USE A SET OF OBSERVED POINTS AND COEFFICIENT VECTOR X TO *00000060
MATCH A PREDICTED VALUE OF STRAIN = F(STRESS, STRAIN RATE) TO EACH *00000070
OBSERVED POINT. READS FROM EXTERNAL FILE FOR COEFFICIENTS AND OBS- *00000080
ERVED VALUES, AND CAN READ COEFFICIENTS FROM TRAILING LIST (THESE *00000090
OVERRIDE EXTERNAL VALUES). SENDS OBSERVED POINTS AND PREDICTED *00000100
POINTS TO A FILE TO BE USED BY SAS. CAN ALSO DO SENSITIVITY ANALYSIS *00000110
WITHOUT CHANGING EXTERNAL VALUES OF COEFFICIENTS. *00000120
----- *00000130
VARIABLES : *00000140
X & COEFF : MANTISSAS AND EXPONENTS OF COEFFICIENT VECTOR. THESE ARE *00000150
COMBINED INTO X IN THE PROGRAM. *00000160
DELTA : FRACTIONAL CHANGE IN VARIABLE K FOR SENSITIVITY ANALYSIS. *00000170
SET THIS EQUAL TO ZERO TO GET TRUE COEFFICIENTS. *00000180
K : VARIABLE THAT WILL BE ALTERED BY AMOUNT (DELTA * X(K)). MUST BE *00000190
BETWEEN 1 AND 5 ALWAYS. *00000200
STRESS : OBSERVED VALUES OF STRESS (KSI). *00000210
RATE : OBSERVED VALUES OF STRAIN RATE (1/SEC). *00000220
EPS : OBSERVED VALUES OF STRAIN (IN/IN). *00000230
EPS1 : PREDICTED VALUE OF STRAIN RATE (IN/IN). *00000240
----- *00000250
I/O REQUIREMENTS : *00000260
FILE #1 : (OPTIONAL) X & COEFF VECTORS ON TWO RECORDS. *00000270
FILE #4 : OBSERVED VALUES OF RATE, EPS, STRESS IN THIS ORDER. THE *00000280
FIRST CARD MUST BE A TITLE. THE REMAINING CARDS CONTAIN *00000290
ONE OBSERVATION EACH. *00000300
FILE #5 : X & COEFF VECTORS ON TWO RECORDS. WILL ALWAYS BE READ. *00000310
----- *00000320
OPTIONS (FORM C$OPTIONS CCOMP=??????) : *00000330
3 : READ FROM DATA CARDS AT END OF PROGRAM LIST, FOLLOWING $DATA CARD *00000340
----- *00000350
$OPTIONS CCOMP=0 *00000360
IMPLICIT REAL * 8 (A-H, N, O-Z) *00000370
DIMENSION X(5), COEFF(5) *00000380
INTEGER TITLE(20) *00000390

```

	READ (5,*) X	00000510
	READ (5,*) COEFF	00000520
13	READ (1,*) X	00000530
13	READ (1,*) COEFF	00000540
	DELTA = 0.000	00000550
	K = 1	00000560
	X(K) = X(K) * (1.000 + DELTA)	00000570
	DO 5 J = 1, 5	00000580
	X(J) = X(J) * COEFF(J)	00000590
	5 WRITE (3,75) X(J)	00000600
	READ (4,200) TITLE	00000610
	WRITE (3,300) TITLE	00000620
	10 READ (4,*,END=50) RATE, EPS, STRESS	00000630
	...	00000640
	...	00000650
	...	00000660
	TEMP1 = STRESS / X(5) / RATE ** X(4)	00000670
	TEMP2 = STRESS ** X(3) * RATE ** X(2)	00000680
	EPS1 = TEMP1 + X(1) * TEMP2	00000690
	...	00000700
	WRITE (6,100) RATE, STRESS, EPS, EPS1	00000710
	GO TO 10	00000720
	50 STOP	00000730
	75 FORMAT (1H ,1PD20.12)	00000740
	100 FORMAT (4D25.13)	00000750
	200 FORMAT (20A4)	00000760
	300 FORMAT (1H ,20A4)	00000770
	END	00000780
	5DATA	00000790
	5.63 -0.589 6.19 3.20 3.10	00000800
	1.0D-13 1.000 1.000 1.0D-2 1.0D3	00000810


```

//FORCE1 JOB DU.D08.AQ0221,BRANDEAU,T=(,10),M=(2,0) 00000010
// EXEC WATFIV 00000020
//GO.FT09F001 DD DSN=DU.D08.AQ0221.BRANDEAU.DATA.ONE,DISP=SHR 00000030
//GO.FT08F001 DD DSN=DU.D08.AQ0221.BRANDEAU.ATBDATA.ONE,DISP=SHR 00000040
//GO.SYSIN DD * 00000050
JOB 00000060
$OPTIONS CCOMP=2,NOEXT,NOCHECK,DEC 00000070
***** 00000080
SUBROUTINE FORCE IN MAIN PROGRAM FORM 00000090
THIS PROGRAM ANALYZES THE ATB OUTPUT DATA FOR DATA NEEDED IN THE 00000100
OPERATION OF THE BREAK PROGRAM. 00000110
1) THE TOP OF EACH TIME INCREMENT DATA SET IS FOUND 00000120
2) ROTATIONS AND ANGULAR VELOCITIES FOR TIME STEP ARE FOUND 00000130
3) DISPLACEMENTS AND LINEAR VELOCITIES ARE FOUND 00000140
4) JOINT FORCES AND TORQUES ARE FOUND 00000150
5) DATA IS WRITTEN TO PRINTER AND/OR DISK 00000160
A. OUTPUT TO DISK IS IN UNFORMATTED FORM 00000170
B. FOR EACH TIME INCREMENT, THE TIME (MS) AND DATA FOR EACH 00000180
LIMB IS OUTPUT. EACH LIMB 00000190
DATA SET FOR THAT TIME INCREMENT IS ON A RECORD, PRECEDED 00000200
BY THE IDENTIFYING NUMBER FOR THAT LIMB (1 - 8). 00000210
***** 00000220
KCON IS THE NUMBER OF "OTHER CONSTRAINT FORCES" 00000230
-1 = NONE 00000240
>0 = NUMBER OF ROWS OF DATA TO BE FOUND FOR EACH TIME STEP 00000250
: 00000260
POSIT = POINT OF ATTACHMENT RELATIVE TO C.G. OF SEGMENT IN SEGMENT 00000270
LOCAL Z-AXIAL COORDINATES. 00000280
CONSTRN(I,J) = FORCES IN INERTIAL COORDINATES (J = 1, 3) FOR SEGMENT I 00000290
(I = 1, KCON) 00000300
: 00000310
CHARACTER *4 IDUM,IPLAG 00000320
REAL D(8,30), VEH(18), POSIT(24), CNSTRN(8,3) 00000330
REAL * 8 DD(3), DA(3) 00000340
INTEGER KT3(8) 00000350
DATA KT3 / 8 * 1 /, CNSTRN /24 * 0.0/ 00000360
NT = 31 ; DLT = 0.01 00000370
IW = 3 ; IR = 1 ; IDISK1 = 9 ; IDISK2 = 8 00000380
WRITE (IW,100) 00000390
JP = 1 00000400
READ (1,*) KCON 00000410
KCON4 = KCON * 4 00000420
IF (KCON4 .EQ. 0) KCON4 = 1 00000430
POSIT(1) = 0 00000440
DO 2 I = 1, KCON4, 4 00000450
READ (1,*) POSIT(I), POSIT(I+1), POSIT(I+2), POSIT(I+3) 00000460
2 KT3(POSIT(I)) = 3 00000470
:2 WRITE (IDISK2) NT, DLT, KCON4, (POSIT(I), I = 1, KCON4) 00000480
DO 4 J = 1, 200 00000490
READ (IDISK1,400) IDUM 00000500

```

	IF (IDUM .EQ. ' 5 H') GO TO 6	00000510
	4 CONTINUE	00000520
	6 DO 8 J = 1, 10	00000530
	READ (IDISK1,420) I, W, X, Y, Z, A, B, C	00000540
	IF (J .EQ. 3 .OR. J .EQ. 6) GO TO 8	00000550
13	WRITE (IW, 120) I, W, X, Y, Z, A, B, C	00000560
		00000570
	OUTPUT TO DISK IN X-AXIAL COORDINATES AND CONSECUTIVE SEGMENT NUMBERS	00000580
		00000590
12	WRITE (IDISK2) JP, W, Z, X, Y, C, A, B	00000600
	JP = JP + 1	00000610
	8 CONTINUE	00000620
	WRITE (IW, 130) (POSIT(KK), KK = 1, KCON4)	00000630
	IP = 0	00000640
	DO 80 I = 1, 1000	00000650
	ON ERROR GOTO 75	00000660
	IF (IP .GE. NT) THEN DO	00000670
	PRINT,IP,' TIME STEPS FOUND'	00000680
	STOP ; ENDIF	00000690
		00000700
	FIND TOP OF DATA SET FOR EACH TIME INCREMENT	00000710
		00000720
	13 READ (IDISK1 ,900, END = 90) IFLAG	00000730
	IF (IFLAG .NE. 'MAIN') GO TO 13	00000740
	BACKSPACE IDISK1	00000750
	READ (IDISK1, 905) TIME	00000760
	TIME = TIME / 1000.	00000770
		00000780
	FIND ROTATIONS , ANGULAR VEL. AND ANGULAR ACC. FOR THIS TIME STEP	00000790
		00000800
	IP = IP + 1	00000810
	DO 20 II = 1, 15	00000820
	READ (IDISK1, 1000) IDUM	00000830
	IF (IDUM .NE. 'H ') GO TO 20	00000840
	JP = 1	00000850
	DO 15 J = 1, 10	00000860
	READ (IDISK1 ,1100) (D(JP,KK),KK = 7,9), (DD(KK),KK = 1,3),	00000870
	* (DA(KK), KK = 1,3)	00000880
	IF (J .EQ. 3 .OR. J .EQ. 6) GO TO 15	00000890
	DO 14 K4 = 1, 3	00000900
	D(JP,K4+18) = SNGL (DA(K4))	00000910
14	D(JP,K4+9) = SNGL (DD(K4))	00000920
	JP = JP + 1	00000930
	15 CONTINUE	00000940
	READ (IDISK1,800) (VEH(KK), KK = 7, 12)	00000950
	GO TO 22	00000960
	20 CONTINUE	00000970
		00000980
	FIND LINEAR POSITION, VELOCITY AND ACCELERATION FOR THIS TIME STEP	00000990
		00001000

22	DO 40 II = 1, 15	00001010
	READ (IDISK1,1000) IDUM	00001020
	IF (IDUM.NE.'H ') GO TO 40	00001030
	JP = 1	00001040
	DO 25 J = 1, 10	00001050
	READ (IDISK1,2000) (D(JP, KK), KK = 1, 6), (D(JP, KK), KK = 22, 24)	00001060
	IF (J .EQ. 3 .OR. J .EQ. 6) GO TO 25	00001070
	JP = JP + 1	00001080
25	CONTINUE	00001090
	READ (IDISK1,1000) IDUM	00001100
	READ (IDISK1,2000) (VEH(KK), KK = 1, 6)	00001110
	GO TO 50	00001120
40	CONTINUE	00001130
		00001140
	FIND U1 & U2 ARRAYS	00001150
		00001160
50	DO 42 II = 1, 25	00001170
	READ (IDISK1,400) IDUM	00001180
42	IF (IDUM.EQ.'5 H') GO TO 44	00001190
44	JP = 1	00001200
	DO 46 JJ = 1, 10	00001210
	READ (IDISK1,2000) (D(JP, KK), KK = 25, 30)	00001220
	IF (JJ.EQ.3 .OR. JJ.EQ.6) GO TO 46	00001230
	JP = JP + 1	00001240
46	CONTINUE	00001250
		00001260
	FIND FORCES AND TORQUES FOR THIS TIME INCREMENT	00001270
		00001280
	DO 70 IJ = 1, 50	00001290
	READ (IDISK1,1000) IDUM	00001300
	IF (IDUM.NE.'HP ') GO TO 70	00001310
	JP = 1	00001320
	DO 60 JJ = 1, 10	00001330
	READ (IDISK1,4000) (D(JP, KK), KK = 13, 18)	00001340
	IF (JJ.EQ.3 .OR. JJ.EQ.6) GO TO 60	00001350
	JP = JP + 1	00001360
60	CONTINUE	00001370
	READ (IDISK1,4000) (VEH(KK), KK = 13, 18)	00001380
	GO TO 71	00001390
70	CONTINUE	00001400
71	DO 74 II = 1, 20	00001410
	READ (IDISK1,400) IDUM	00001420
	IF (IDUM.NE.'NO.') GO TO 74	00001430
	READ (IDISK1,400) IDUM	00001440
	DO 72 IJ = 1, KCON	00001450
72	READ (IDISK1, 500) (CNSTRN (POSIT (4*IJ-3), IN), IN = 1, 3)	00001460
	GO TO 75	00001470
74	CONTINUE	00001480
75	CONTINUE	00001490
		00001500

```

: OUTPUT RESULTS                                00001510
: C2 = DISK OUTPUT (UNIT = IDISK2)             00001520
: C3 = LINE PRINTER (UNIT = IW)                00001530
C2 WRITE (IDISK2) TIME, VEH                    00001540
C3 PRINT, 'TIME =', TIME, ' VEHICLE =', VEH    00001550
C2 PRINT, 'TIME =', TIME, ' SEC'              00001560
      DO 85 J = 1, 8                            00001570
      K3 = KT3(J)                              00001580
C2 WRITE (IDISK2) J, K3, (D(J, KK), KK=1, 30), (CNSTRN(J, KK), KK=1, K3) 00001590
C3 WRITE (IW, 808) J, K3, (D(J, KK), KK=1, 30), (CNSTRN(J, KK), KK=1, K3) 00001600
      85 CONTINUE                              00001610
:                                              00001620
      80 CONTINUE                              00001630
      90 PRINT, 'END OF FILE REACHED ON UNIT', IDISK1 00001640
      PRINT, IP, ' TIME STEPS FOUND'          00001650
      STOP                                     00001660
:                                              00001670
: INPUT FORMATS                                00001680
C 130 FORMAT (' EXTRA FORCE LOCATIONS :', F5.0, 3F12.3) 00001690
C 400 FORMAT (1X, A4)                          00001700
C 500 FORMAT (25X, 3F15.5)                    00001710
C 420 FORMAT (I3, 13X, F7.3, 3X, 3F11.5, 5X, 3F8.3) 00001720
C 800 FORMAT (/ , 11X, 3F9.4, 4X, 3(F7.3, 7X)) 00001730
C 900 FORMAT (7X, A4)                          00001740
C 905 FORMAT (57X, F8.3)                      00001750
C 1000 FORMAT (4X, A4)                        00001760
C 1100 FORMAT (11X, 3F9.4, 3X, 3D14.5, 3X, 3D14.5) 00001770
C 2000 FORMAT (11X, 3F11.4, 3X, 3F12.5, 3X, 3F14.5) 00001780
C 4000 FORMAT (15X, 3F11.4, 3X, 3F12.5)      00001790
:                                              00001800
: OUTPUT FORMATS                              00001810
C 100 FORMAT (1H1, 10X, 'FORCES FROM ATB MODEL' /) 00001820
C3120 FORMAT (' I=' , I2, ' W=' , F7.3, ' XYZ=' , 3F10.5, ' ABC=' , 3F7.3) 00001830
C3808 FORMAT (1H , I2, I3, 2X, (T11,6(E16.7, 3X))) 00001840
      END                                       00001850
SDATA                                         00001860
2                                              00001870
2  0.0  0.0  0.0                             00001880
4  0.0  0.0  0.0                             00001890
C$STOP                                        00001900
C$END                                        00001910
/*                                           00001920
/**PW=BONE                                    00001930
/*                                           00001940

```

```

//FORCE2 JOB DU.D08.AQ0221,BRANDEAU,T=(,10),P=45,M=(2,0) 00000010
// EXEC WATFIV 00000020
//GO.PT09F001 DD DSN=DU.D08.AQ0221.BRANDEAU.DATA.TWO,DISP=SHR 00000030
//GO.PT08F001 DD DSN=DU.D08.AQ0221.BRANDEAU.ATBDATA.TWO,DISP=SHR 00000040
//GO.SYSIN DD * 00000050
$JOB 00000060
$OPTIONS NOEXT,CCOMP=2,NOCHECK,DEC 00000070
$OPTIONS NOLIST 00000080
CHARACTER * 10 IDUM, DUMB 00000090
CHARACTER * 3 ISEG1, ISEG2 00000100
REAL PANEL (32,8,3,6), SEGMENT (32,8,3,6), X1 (3), X2 (3) 00000110
INTEGER CHEKP (32,8), CHEKS (32,8), ID(3), UCOUNT, NP(4) 00000120
CHARACTER ITEST*3(8) / 'RUL', 'RLL', 'LUL', 'LLL', 'RUA', 'RLA', 'LUA' 00000130
* 'LLA' / 00000140
DATA CHEKP,CHEKS / 512 * 0 / 00000150
DATA PANEL,SEGMENT / 9216 * 0.0 / 00000160
DATA NP / 58, 92, 92, 92 / 00000170
***** 00000180
: EXPLANATION OF VARIABLES -- 00000190
: PANEL (I,J,K,L) AND SEGMENT (I,J,K,L) CONTAIN CONTACT DATA 00000200
: I = TIME STEP 00000210
: J = LIMB OR MEMBER NUMBER 00000220
: K = CONTACT NUMBER 00000230
: L = 1 TO 6 00000240
: L = 1 IS THE NUMBER OF THE PANEL OR SEGMENT CONTACTED 00000250
: -- SEGMENT NUMBERS ARE IN ATB MODEL CODE, NOT BREAK CODE 00000260
: L = 2 IS THE NORMAL FORCE 00000270
: L = 3 IS THE FRICTION FORCE 00000280
: L = 4 TO 6 ARE THE X,Y,Z COORDINATES OF THE CONTACT POINT 00000290
: 00000300
: SEGMENT CONTACT POINTS ARE IN ATB LOCAL COORDINATES 00000310
: PANEL CONTACT POINTS ARE IN ATB INERTIAL COORDINATES 00000320
: 00000330
: CHEKP (I,J) AND CHEKS (I,J) ARE CONTACT COUNTERS FOR EACH LIMB 00000340
: AND TIME STEP 00000350
: I = TIME STEP 00000360
: J = LIMB OR MEMBER NUMBER 00000370
: 00000380
: THE VALUE STORED IN LOCATION CHEKP OR CHEKS (I,J) IS THE NUMBER 00000390
: OF CONTACTS FOR THAT LIMB AND TIME STEP THAT HAD NON-ZERO 00000400
: FORCES. CONTACTS WITH ZERO FORCES ARE NOT STORED IN THE ARRAY 00000410
: OR WRITTEN TO THE DISK OR PRINTER. 00000420
: 00000430
: NP = THE NUMBER OF RECORDS THAT OCCUR BETWEEN PAGE HEADERS 00000440
: ON THE ATB OUTPUT FILE. THIS MUST BE CHANGED AS THE 00000450
: TIME STEP THEY USE CHANGES. THIS IS NOT THE SAME BETWEEN 00000460
: EACH HEADER BLOCK. 00000470
: UCOUNT = THE NUMBER OF CONTACT FILES TO BE LOOKED FOR. THE 00000480
: NUMBER OF FILES FOUND IS COUNTED, NOT THE NUMBER OF 00000490
: INDIVIDUAL CONTACTS. UCOUNT DOES NOT INCLUDE FILES HAVING 00000500

```

```

ALL ZERO FORCES OR SEGMENTS THAT ARE NOT OF INTEREST.
TMAX = MAXIMUM TIME TO BE FOUND.
FORMAT # 3416 MUST ALSO BE CHANGED TO SKIP THE PROPER NUMBER OF
RECORDS IN EACH FILE IF THE FILE IS OF NO INTEREST.
*****
ON ERROP GOTO 255
DLT = 0.01      ; NT = 31;      TMAX = 300.0
IW = 3      ;   IR = 1      ;   IDISK1 = 9 ;   IDISK2 = 8
WRITE (IW, 100)
IP = 0      ;   IDT = INT (10000. * DLT)      ; ICOUNT = 0 ; UCOUNT = 9
      DO 200 I = 1, 500
READ (IDISK1, 103, END=255) IDUM
IF (IDUM .EQ. 'SEGMENT NO') GO TO 210
IF (IDUM .NE. 'VEHICLE PA') GO TO 200

----- PANEL VS. SEGMENT CONTACT -----

READ (IDISK1, 102) IPAN1, ISEG1, IPAN2, ISEG2
IFLAG1 = IFLAG2 = 0

CHECK FOR SEGMENT OF INTEREST

DO 110 J = 1, 8
IF (ISEG1 .NE. ITEST(J) .AND. ISEG2 .NE. ITEST(J)) GO TO 110
IF (ISEG1 .NE. ITEST(J)) GO TO 105
JHOLD1 = J      ; IFLAG1 = 1
105 IF (ISEG2 .NE. ITEST(J)) GO TO 109
JHOLD2 = J      ; IFLAG2 = 2
109 IFLAG3 = IFLAG1 + IFLAG2
IF (IFLAG3 - 3) 110, 112, 110
110 CONTINUE

IF (IFLAG1 + IFLAG2) 111, 111, 112

SKIP ENTIRE FILE

111 READ (IDISK1,3416) DUMB
GO TO 200
112 READ (IDISK1,113) DUMB
NUMPAG = NP(1)
ICOUNT = ICOUNT + 1
DO 155 J = 1,11
IF (J .EQ. 1) GO TO 120
READ (IDISK1, 114) DUMB
120 DO 150 JL= 1, NUMPAG
READ (IDISK1, 115) T, FN1, FF1, X1, FN2, FF2, X2
IT = INT (10. * T)

```

```

00000510
00000520
00000530
00000540
00000550
00000560
00000570
00000580
00000590
00000600
00000610
00000620
00000630
00000640
00000650
00000660
00000670
00000680
00000690
00000700
00000710
00000720
00000730
00000740
00000750
00000760
00000770
00000780
00000790
00000800
00000810
00000820
00000830
00000840
00000850
00000860
00000870
00000880
00000890
00000900
00000910
00000920
00000930
00000940
00000950
00000960
00000970
00000980
00000990
00010000

```

```

      ITT = IT / IDT + 1                                00001010
      IF ((IT/IDT * IDT) .NE. IT) GO TO 150            00001020
      IF (IFLAG2) 500, 130, 129                        00001030
29      IF (FN2 .EQ. 0.0 .AND. FF2 .EQ. 0.0) GO TO 131 00001040
      CHEKP (ITT, JHOLD2) = TEMPCT = CHEKP (ITT, JHOLD2) + 1 00001050
      PANEL (ITT, JHOLD2, TEMPCT, 1) = FLOAT (IPAN2)    00001060
      PANEL (ITT, JHOLD2, TEMPCT, 2) = FN2              00001070
      PANEL (ITT, JHOLD2, TEMPCT, 3) = FF2             00001080
      DO 132 K4 = 1, 3                                  00001090
32      PANEL (ITT, JHOLD2, TEMPCT, K4+3) = X2 (K4)    00001100
31      IF (IPLAG1) 500, 140, 130                      00001110
30      IF (FN1 .EQ. 0.0 .AND. FF1 .EQ. 0.0) GO TO 140 00001120
      CHEKP (ITT, JHOLD1) = TEMPCT = CHEKP (ITT, JHOLD1) + 1 00001130
      PANEL (ITT, JHOLD1, TEMPCT, 1) = FLOAT (IPAN1)    00001140
      PANEL (ITT, JHOLD1, TEMPCT, 2) = FN1              00001150
      PANEL (ITT, JHOLD1, TEMPCT, 3) = FF1             00001160
      DO 133 K4 = 1, 3                                  00001170
133      PANEL (ITT, JHOLD1, TEMPCT, K4+3) = X1 (K4)    00001180
140      IF (ITT .GE. NT .OR. T .EQ. TMAX) GO TO 200    00001190
150      CONTINUE                                       00001200
      NUMPAG = NP(J+1)                                  00001210
155      CONTINUE                                       00001220
200      CONTINUE                                       00001230
      :                                                 00001240
205      READ (IDISK1, 103, END = 255) IDUM            00001250
      IF (IDUM .NE. 'SEGMENT NO') GO TO 260            00001260
      :                                                 00001270
      : ----- SEGMENT VS. SEGMENT CONTACT ----- 00001280
      :                                                 00001290
110      BACKSPACE IDISK1                               00001300
      READ (IDISK1, 101) IDUM, IHOLD1, ISEG1, IHOLD2, ISEG2 00001310
      IFLAG1 = IFLAG2 = 0                               00001320
      JHOLD1 = JHOLD2 = 0                               00001330
      :                                                 00001340
      : CHECK FOR SEGMENT(S) OF INTEREST                00001350
      :                                                 00001360
      DO 220 J = 1, 8                                   00001370
      IF (ISEG1 .NE. ITEST(J) .AND. ISEG2 .NE. ITEST(J)) GO TO 220 00001380
      IF (ISEG1 .NE. ITEST(J)) GO TO 218               00001390
      JHOLD1 = J ; IFLAG1 = 1                           00001400
      GO TO 220                                         00001410
118      IF (ISEG2 .NE. ITEST(J)) GO TO 219            00001420
      JHOLD2 = J ; IFLAG2 = 2                           00001430
119      IFLAG3 = IFLAG1 + IFLAG2                      00001440
      IF (IFLAG3 - 3) 220, 222, 220                   00001450
220      CONTINUE                                       00001460
      :                                                 00001470
      IF (IFLAG2 + IFLAG1) 500, 221, 222              00001480
      :                                                 00001490
      : SKIP ENTIRE FILE                                00001500

```

		00001510
221	READ (IDISK1,3416) DUMB	00001520
	GO TO 260	00001530
222	READ (IDISK1,113) DUMB	00001540
	NUMPAG = NP(1)	00001550
	ICOUNT = ICOUNT + 1	00001560
	ITT = 1	00001570
	DO 252 J = 1, 11	00001580
	IF (J .EQ. 1) GO TO 224	00001590
	READ (IDISK1,223) DUMB	00001600
224	DO 250 JL = 1, NUMPAG	00001610
	READ (IDISK1, 225, END=255) T, FN1, FF1, X1, X2	00001620
	IF (FN1 .EQ. 0.0 .AND. FF1 .EQ. 0.0) GO TO 240	00001630
	IT = INT (10. * T)	00001640
	ITT = IT / IDT + 1	00001650
	IF ((IT / IDT * IDT) .NE. IT) GO TO 240	00001660
	IF (IFLAG2) 500, 230, 226	00001670
226	CHEKS (ITT, JHOLD2) = TEMPCT = CHEKS (ITT, JHOLD2) + 1	00001680
	SEGNET (ITT, JHOLD2, TEMPCT, 1) = FLOAT (JHOLD1)	00001690
	SEGNET (ITT, JHOLD2, TEMPCT, 2) = FN1	00001700
	SEGNET (ITT, JHOLD2, TEMPCT, 3) = FF1	00001710
	DO 227 K4 = 1, 3	00001720
227	SEGNET (ITT, JHOLD2, TEMPCT, K4+3) = X2 (K4)	00001730
	IF (IFLAG1) 500, 240, 230	00001740
230	CHEKS (ITT, JHOLD1) = TEMPCT = CHEKS (ITT, JHOLD1) + 1	00001750
	SEGNET (ITT, JHOLD1, TEMPCT, 1) = FLOAT (JHOLD2)	00001760
	SEGNET (ITT, JHOLD1, TEMPCT, 2) = FN1	00001770
	SEGNET (ITT, JHOLD1, TEMPCT, 3) = FF1	00001780
	DO 232 K4 = 1, 3	00001790
232	SEGNET (ITT, JHOLD1, TEMPCT, K4+3) = X1 (K4)	00001800
240	IF (ITT. GE. NT .OR. T .EQ. TMAX) GO TO 260	00001810
250	CONTINUE	00001820
	NUMPAG = NP(J+1)	00001830
252	CONTINUE	00001840
		00001850
260	IF (ICOUNT .LT. UCOUNT) GO TO 205	00001860
		00001870
	OUTPUT DATA	00001880
		00001890
255	DO 1000 J3 = 1, NT	00001900
	TIME = FLOAT (J3-1) * DLT	00001910
2	WRITE (3,2200) TIME	00001920
3	WRITE (IDISK2) TIME	00001930
	DO 1000 J4 = 1, 8	00001940
	ID(1) = J4	00001950
	IF (CHEKP(J3,J4) .EQ. 0) THEN DO	00001960
	JPAN = ID2 = ID(2) = 1	00001970
	ELSE DO	00001980
	JPAN = 6 ; ID(2) = 6 * CHEKP (J3,J4)	00001990
	ID2 = ID(2) / 6	00002000


```

//ALF .PA JOB DU.D08.AQ0221, BRANDEAU, M=(2,0) 00000010
// EXEC WATFIV 00000020
//GO.FT06F001 DD DSN=DU.D08.AQ0221.BRANDEAU.ATBDATA.ONE, DISP=SHR 00000030
//GO.FT07F001 DD DSN=DU.D08.AQ0221.BRANDEAU.ATBDATA.TWO, DISP=SHR 00000040
//GO.FT09F001 DD DSN=DU.D08.AQ0221.BRANDEAU.ATBDATA.FINAL, DISP=SHR 00000050
//GO.SYSIN DD * 00000060
:JOB 00000070
:SOPTIONS NOEXT, NOCHECK, CCOMP=0 00000080
    REAL VEH(13), FORCE(8,30), PANEL(8,24), INSEG(8,24), TIME 00000090
    REAL DUMMY(7), OUTSEG(8,72), POSIT(24), CNSTRN(8,3) 00000100
    INTEGER NT, LIMB, ID(8,3), KT(8) 00000110
    READ (6) NT, DLT, KCON4, (POSIT(I), I = 1, KCON4) 00000120
    WRITE (9) NT, DLT, KCON4, (POSIT(I), I = 1, KCON4) 00000130
: 00000140
: READ AND WRITE SEGMENT #, MASS, INERTIA, AND SEMI-MAJOR AXES. 00000150
: 00000160
: DO 10 I = 1, 8 00000170
:     READ (6) J, DUMMY 00000180
: 10 WRITE (9) J, DUMMY 00000190
: 00000200
: BEGIN LOOP FOR ALL REMAINING DATA TO BE COMBINED 00000210
: 00000220
: DO 100 I = 1, NT 00000230
: READ (6) VEH 00000240
: READ (7) TIME 00000250
: PRINT, 'TIME = ', TIME 00000260
: WRITE (9) VEH 00000270
: 00000280
: READ IN ALL DATA FOR THIS TIME STEP 00000290
: 00000300
: DO 40 J = 1, 8 00000310
:     READ (6) LIMB, K3, (FORCE(J,K), K = 1,30), (CNSTRN(J,K), K=1,K3) 00000320
:     KT(J) = K3 00000330
: 00000340
: CONVERT ROTATIONS FROM DEGREES TO RADIANS 00000350
: 00000360
: DO 20 K = 7,9 00000370
: 20 FORCE(J,K) = FORCE(J,K) / 57.29578 00000380
: READ (7) ID(J,1), ID2, ID3, (PANEL(J, KK), KK = 1, ID2), 00000390
: $ (INSEG(J, KK), KK = 1, ID3) 00000400
: ID(J,2) = ID2 ; ID(J,3) = ID3 00000410
: IF (LIMB .NE. ID(J,1)) THEN DO 00000420
:     PRINT, 'LIMB NOS. NOT EQUAL AT STEP ', J 00000430
:     PRINT, 'LIMB NO. FROM UNIT 6 =', LIMB 00000440
:     PRINT, 'LIMB NO. FROM UNIT 7 =', ID(J,1) 00000450
:     PRINT, 'STOPPING NOW' 00000460
:     STOP 00000470
: ENDIF 00000480
: 40 CONTINUE 00000490
: 00000500

```

```

COMBINE DATA FOR EACH CONTACTED SEGMENT SO THAT RELATIVE VELOCITIES CAN BE CALCULATED. INCREASE SEGMENT DATA FROM 6 TO 18 ITEMS.
DO 60 J = 1, 8
  ID3 = ID(J,3) ; ID2 = ID(J,2) ; OUTSEG(J,1) = 0.0
  IF (ID3 .EQ. 1) GO TO 55
  DO 50 K = 1, ID3, 6
    NUM = IPIX(INSEG(J,K))
    K1 = 3 * K - 2 ; K2 = K1 + 17 ; K3 = K1 + 6 ; K4 = K1 + 5
    DO 44 L = K1, K4
      OUTSEG(J,L) = INSEG(J,K+L-K1)
      IF (NUM .EQ. 0) THEN DO
        DO 45 L = K3, K4
          OUTSEG(J,L) = 0.0
        ELSE DO
          FIND PROPER LOCATION IN SEGMENT "NUM" FILE
          ID2 = 1
          WHILE (IPIX(INSEG(NUM,ID2)) .NE. J) DO
            ID2 = ID2 + 6
          ENDWHILE
          DO 48 L = 1, 3
            OUTSEG(J,K1+L+5) = FORCE(NUM,L+3)
          DO 49 L = 15, 17
            OUTSEG(J,K1+L) = INSEG(NUM,ID2+L-12)
          ENDIF
        50 CONTINUE
        ID3 = ID3 / 6 * 18
      NEGATE "OTHER FORCES" FOR EQUILIBRIUM
      55 K3 = KT(J)
      WRITE (9) ID(J,1), ID2, ID3, K3, (FORCE(J,KK), KK = 1, 30),
        $ (PANEL(J,KK), KK = 1, ID2), (OUTSEG(J,KK), KK = 1, ID3)
        $ (-CNSTRN(J,KK), KK = 1, K3)
      :2 WRITE (3,300) ID(J,1), ID2, (PANEL(J,KK), KK = 1, ID2)
      :2 WRITE (3,200) ID3, (OUTSEG(J,KK), KK = 1, ID3)
      :2 IF (K3 .GT. 1) WRITE (3,150) K3, (-CNSTRN(J,KK), KK = 1,K3)
      60 CONTINUE
      100 CONTINUE
      STOP
      :2300 FORMAT (' LIMB =',I2,I3,5X,(2(F5.0,5F8.2,5X)))
      :2200 FORMAT (' SEGS',I3,(F5.0,1X,2F6.2,2X,15F7.1))
      :2150 FORMAT ('EXTRA CONTACTS =',I4,3F12.4)
      END
    :DATA
    :END

```

```

00000510
00000520
00000530
00000540
00000550
00000560
00000570
00000580
00000590
00000600
00000610
00000620
00000630
00000640
00000650
00000660
00000670
00000680
00000690
00000700
00000710
00000720
00000730
00000740
00000750
00000760
00000770
00000780
00000790
00000800
00000810
00000820
00000830
00000840
00000850
00000860
00000870
00000880
00000890
00000900
00000910
00000920
00000930
00000940
00000950
00000960
00000970
00000980
00000990
00001000

```

```

:SOPTIONS DEC,CCOMP=1                                00000010
: PROGRAM NAME : CONTACT                              00000020
  REAL VEH(12), FORCE(30), PANEL(24), SEGMENT(72), PLANE(33) 00000030
  REAL ABC(24), XYZ(24), TIME, INERT(24), WEIGHT(8), D1(8) 00000040
  REAL RADS(3), OMEGA(3), THOLD(3), OMEGA2(3), RADS2(3) 00000050
  REAL XHOLD(3), XTEMP(3), TEMP(3), WORK(6), POSIT(24), CNSTEN(6) 00000060
  CHARACTER SEG*3(8) /'RUL','RLL','LUL','LLL','RUA','RLA','LUA', 00000070
  * 'LLA' / 00000080
  INTEGER NT, LIMB, NPAN, NSEG, KT, JS, JS2, JS3, LIMBO, LIMB1, KKK 00000090
  INTEGER KK, J1, J2, J, I, K, ICHEK, KS, KP, NUM, NUMO, NCON, KCON4 00000100
  COMMON /TRANS/ ICHEK 00000110
  INTEGER YES(8) 00000120
  DATA YES /8 * 0 / 00000130
  REAL GINCH / 386.0886 / 00000140
  DATA PLANE /-.1104,0.0,-.9939, .9744,0.0,-.2249, 00000150
  * 0.0,0.0,-1.0, .9745,0.0,-.2245, .9191,0.0,-.3939, 00000160
  * -.0499,0.0,.9988, -1.0,0.0,0.0, 0.0,-1.0,0.0, 00000170
  * 0.0,1.0,0.0, .9720,0.0,-.2350, -.6428,0.0,-.7660 / 00000180
  DATA D1 /8.8, 7.93, 8.8, 7.93, 5.44, 7.94, 5.44, 7.94 / 00000190
  EQUIVALENCE (RADS, FORCE(7)), (OMEGA, FORCE(10)) 00000200
  READ (5) NT, DLT, KCON4, (POSIT(I), I = 1, KCON4) 00000210
  IF (KCON4 .EQ. 1) GO TO 4 00000220
: 00000230
: ROTATE "OTHER CONSTRAINT FORCE" LOCATION TO X-AXIAL COORDINATES 00000240
: WITH THE ORIGIN AT THE PROXIMAL JOINT 00000250
: 00000260
  DO 3 I = 1, KCON4, 4 00000270
    CALL CHANGE (POSIT, I+1, 24, D1(POSIT(I))) 00000280
  3 YES(POSIT(I)) = I + 1 00000290
: 00000300
  4 WRITE (9) NT, DLT, KCON4, (POSIT(I), I = 1, KCON4) 00000310
  WRITE (9) D1(1), D1(2), D1(5), D1(6) 00000320
: 00000330
: READ SEGMENT WEIGHTS, INERTIAS, AND SEMI-MAJOR AXES IN LOCAL X-AXIAL 00000340
: SEGMENT COORDINATES. 00000350
: 00000360
  J1 = -2 00000370
  DO 5 J = 1, 8 00000380
  J1 = J1 + 3 00000390
  J2 = J1 + 2 00000400
  READ (5) I, WEIGHT(J), (INERT(K), K=J1,J2), (ABC(K), K = J1, J2) 00000410
  WRITE (9) I, WEIGHT(J), (ABC(K), K=J1,J2) 00000420
: 00000430
: SQUARE SEMI-MAJOR AXES OF ELLIPSOIDS FOR LATER USE 00000440
: 00000450
  DO 5 K = J1, J2 00000460
  5 ABC(K) = ABC(K) * ABC(K) 00000470
: 00000480
  DO 350 KKK = 1, NT 00000490
  READ (5) TIME, VEH 00000500

```

```

01 PRINT, 'TIME =', TIME 00000510
WRITE (9) TIME 00000520
DO 350 KK = 1, 8 00000530
READ (5) LIMB, NPAN, NSEG, NCON, FORCE, (PANEL(I), I = 1, NPAN) 00000540
* , (SEGNET(I), I = 1, NSEG), (CNSTRN(I), I = 1, NCON) 00000550
LIMBO = 3 * LIMB - 3 00000560
2 00000570
03 PRINT, '*****' 00000580
03 PRINT, 'SEGMENT IS ', LIMB, ' OR ', SEG(LIMB) 00000590
03 PRINT, '*****' 00000600
03 PRINT, 'ROTATION ANGLES (RADIANS)' 00000610
03 WRITE (3,500) (FORCE(I), I = 7, 9) 00000620
ICHECK = -1 00000630
KS = 0 00000640
KP = 0 00000650
IF (NSEG .EQ. 1) GO TO 50 00000660
----- 00000670
03 SEGMENT - SEGMENT CONTACT 00000680
03 ----- 00000690
03 SEGMENT CONTACT POINTS IN SEGMENT COORDINATES 00000700
03 00000710
KS = NSEG / 18 00000720
DO 40 J = 1, KS 00000730
JXYZ = 6 * J - 5 00000740
JS = 1 + (J-1) * 18 00000750
JS2 = JS + 1 00000760
JS3 = JS + 2 00000770
NUM = IFIX (SEGNET(JS)) 00000780
03 PRINT, ' ' 00000790
03 PRINT, 'SEGMENT', LIMB, ' CONTACTING SEGMENT', NUM 00000800
03 PRINT, 'NORMAL FORCE =', SEGNET(JS2) 00000810
03 PRINT, 'FRICTION FORCE =', SEGNET(JS3) 00000820
03 PRINT, 'CONTACT AT ATB LOCAL XYZ', (SEGNET(JS+I), I = 3, 5) 00000830
03 ----- 00000840
03 NORMAL CONTACT FORCES 00000850
03 ----- 00000860
03 00000870
03 FIND NORMAL TO ELLIPSOID AT POINT OF CONTACT 00000880
03 00000890
HOLD = 0.0 00000900
DO 10 I = 1, 3 00000910
HOLD2 = SEGNET(JS3+I) / ABC(LIMBO+I) 00000920
TEMP(I) = HOLD2 00000930
WORK(I+3) = HOLD2 00000940
10 HOLD = HOLD + HOLD2 * HOLD2 00000950
HOLD = SQRT (HOLD) 00000960
03 00000970
03 MULTIPLY UNIT NORMAL BY NORMAL FORCE SCALAR - STORE IN "WORK(1-3)" 00000980
03 00000990
DO 15 I = 1, 3 00001000

```

15	WORK(I) = -TEMP(I) * SEGMET(JS2) / HOLD	00001010
	IF (SEGMET(JS3) .EQ. 0.0) GO TO 39	00001020
	-----	00001030
	FRICITION CONTACT FORCES	00001040
	-----	00001050
		00001060
	ASSIGN DATA FOR SEGMENT "NUM" TO STORAGE VECTORS	00001070
		00001080
	DO 20 I = 1, 3	00001090
	JSI = JS + I	00001100
	XTEMP(I) = SEGMET(JSI + 14)	00001110
	THOLD(I) = SEGMET(JSI + 5)	00001120
	RADS2(I) = SEGMET(JSI + 8)	00001130
	OMEGA2(I) = SEGMET(JSI + 11)	00001140
20	TEMP(I) = SEGMET(JSI + 2)	00001150
		00001160
	CONSTRUCT RELATIVE VELOCITY VECTOR IN SEGMENT "LIMB" COORDINATES	00001170
	FOR SEGMENT "LIMB" CONTACTING SEGMENT "NUM".	00001180
	VECTORS "OMEGA2", "RADS2", "THOLD" & "XTEMP" CONTAIN DATA FOR	00001190
	SEGMENT "NUM".	00001200
		00001210
	CALL CROSS (OMEGA2, XTEMP, XHOLD)	00001220
29	PRINT, 'W2 CROSS R2=', XHOLD	00001230
	CALL ROT (XHOLD, 1, 3, RADS2, -1)	00001240
29	PRINT, 'SAME VECTOR ROTATED INTO INERTIAL', XHOLD	00001250
	ICHECK = -1	00001260
	CALL CROSS (OMEGA, TEMP, XTEMP)	00001270
29	PRINT, 'W CROSS R=', XTEMP	00001280
	DO 25 I = 1, 3	00001290
25	THOLD(I) = FORCE(I+3) - THOLD(I) - XHOLD(I)	00001300
	CALL ROT (THOLD, 1, 3, RADS, +1)	00001310
		00001320
	DO 30 I = 1, 3	00001330
	THOLD(I) = THOLD(I) + XTEMP(I)	00001340
30	TEMP(I) = WORK(I+3)	00001350
29	PRINT, 'RELATIVE VELOCITY VECTOR IN LOCAL', THOLD	00001360
		00001370
	PROJECT VELOCITY VECTOR ONTO TANGENT PLANE TO SEGMENT "LIMB"	00001380
		00001390
	CALL VECTOR (TEMP, THOLD, XHOLD)	00001400
29	PRINT, 'SAME PROJECTED INTO PLANE OF SEG.', XHOLD	00001410
		00001420
	MULTIPLY PROJECTED VELOCITY BY FRICTION SCALAR AND COMBINE	00001430
	ALL FORCES INTO "WORK (1-3)"	00001440
		00001450
	DO 35 I = 1, 3	00001460
35	WORK(I) = WORK(I) + XHOLD(I) * SEGMET(JS3)	00001470
	CONVERT SEGMENT Z-AXIAL COORDINATES TO X-AXIAL	00001480
		00001490
39	XYZ(JXYZ) = SEGMET(JS+5) + D1(LIMB)	00001500

XYZ (JXYZ+1) = SEGMET (JS+3)	00001510
XYZ (JXYZ+2) = SEGMET (JS+4)	00001520
XYZ (JXYZ+3) = WORK (3)	00001530
XYZ (JXYZ+4) = WORK (1)	00001540
XYZ (JXYZ+5) = WORK (2)	00001550
40 CONTINUE	00001560
50 IF (NPAN .EQ. 1) GO TO 130	00001570

SEGMENT - PANEL CONTACT	00001580

PANEL CONTACT DATA IS IN INERTIAL COORDINATES.	00001600
KP = NPAN / 6	00001610
K1 = KS + 1	00001620
K2 = K1 + KP - 1	00001630
DO 120 J1 = K1, K2	00001640
JKYZ = 6 * J1 - 5	00001650
J = J1 - KS	00001660
JS = 1 + (J-1) * 6	00001670
JS2 = JS + 1	00001680
JS3 = JS + 2	00001690
CORRECT PANEL CONTACT POINT FOR VEHICLE MOTION	00001700
DO 55 I2 = 1, KP	00001710
DO 55 I = 1, 3	00001720
NUM = 6 * (J-1) + I + 3	00001730
55 PANEL(NUM) = PANEL(NUM) + VEH(I)	00001740
13 NUM = IFIX(PANEL(JS))	00001750
13 PRINT, ' '	00001760
13 PRINT, 'SEGMENT', LIMB, ' CONTACTING PANEL', NUM	00001770
13 PRINT, 'NORMAL FORCE =', PANEL(JS2)	00001780
13 PRINT, 'FRICTION FORCE =', PANEL(JS3)	00001790
13 PRINT, 'CONTACT AT INERTIAL XYZ', (PANEL(JS+I), I = 3, 5)	00001800
FIND VECTOR FROM CENTER OF ELLIPSOID TO CONTACT POINT	00001810
IN INERTIAL COORDINATES.	00001820
DO 60 I = 1, 3	00001830
TEMP(I) = PANEL(JS3+I) - FORCE(I)	00001840
60 THOLD(I) = TEMP(I)	00001850
15 PRINT, 'INERTIAL CONTACT VECTOR BEFORE TRANSFORMATION'	00001860
15 WRITE (3,500) TEMP	00001870
TRANSFORM CONTACT VECTOR FROM INERTIAL TO SEGMENT COORDINATES	00001880
CALL ROT (TEMP, 1, 3, RADS, 1)	00001890
15 PRINT, 'AFTER TRANSFORMATION TO SEGMENT LOCAL'	00001900
15 WRITE (3,500) TEMP	00001910
15 PRINT, ' '	00001920
	00001930
	00001940
	00001950
	00001960
	00001970
	00001980
	00001990
	00002000

	STORE CONTACT POINT IN X-AXIAL COORDINATES	00002010
		00002020
		00002030
	XYZ(JXYZ) = TEMP(3) + D1(LIMB)	00002040
	XYZ(JXYZ+1) = TEMP(1)	00002050
	XYZ(JXYZ+2) = TEMP(2)	00002060
	-----	00002070
	NORMAL CONTACT FORCES	00002080
	-----	00002090
	NUM = IPIX (PANEL(JS))	00002100
	NUM0 = 3 * NUM - 3	00002110
	DO 70 I = 1, 3	00002120
	TEMP(I) = PLANE(NUM0+I)	00002130
	WORK(I+3) = 0.0	00002140
	XTEMP(I) = OMEGA(I)	00002150
	70 WORK(I) = TEMP(I) * PANEL(JS2)	00002160
		00002170
	IF (PANEL(JS3) .EQ. 0.0) GO TO 95	00002180
	-----	00002190
	FRICTION CONTACT FORCES	00002200
	-----	00002210
	FIND VELOCITY VECTOR OF CONTACT POINT IN INERTIAL COORDINATES.	00002220
		00002230
	CALL ROT2 (XTEMP, 1, 3, -1)	00002240
	CALL CROSS (XTEMP, THOLD, XHOLD)	00002250
	PRINT, 'ANGULAR VELOCITY OF C.G.'	00002260
C6	WRITE (3,500) XTEMP	00002270
C6	PRINT, 'LINEAR VELOCITY DUE TO ROTATION -- OMEGA CROSS R'	00002280
C6	WRITE (3,500) XHOLD	00002290
C6	PRINT, 'TRANSLATIONAL VELOCITY'	00002300
C6	WRITE (3,500) (FORCE(I), I = 4, 6)	00002310
C6	DO 80 I = 1, 3	00002320
	80 XHOLD(I) = XHOLD(I) + FORCE(I+3)	00002330
C6	PRINT, 'TOTAL VELOCITY VECTOR'	00002340
C6	WRITE (3,500) XHOLD	00002350
		00002360
		00002370
	PROJECT VELOCITY VECTOR ONTO TANGENT PLANE	00002380
		00002390
	CALL VECTOR (TEMP, XHOLD, THOLD)	00002400
C6	PRINT, 'UNIT VECTOR OF VELOCITY PROJECTED ONTO TANGENT PLANE'	00002410
C6	WRITE (3,500) THOLD	00002420
		00002430
	MULTIPLY NEG. UNIT VECTOR BY FRICTION SCALAR TO GET FRICTION VECTOR	00002440
		00002450
	DO 90 I = 1, 3	00002460
	90 WORK(I+3) = -THOLD(I) * PANEL(JS3)	00002470
		00002480
	CREATE TOTAL FORCE VECTOR IN INERTIAL COORDINATES	00002490
		00002500

95	DO 100 I = 1, 3	00002510
100	TEMP(I) = WORK(I) + WORK(I+3)	00002520
06	PRINT, 'TOTAL FORCE VECTOR IN INERTIAL COORDINATES'	00002530
06	WRITE (3,500) TEMP	00002540
		00002550
	TRANSFORM TOTAL FORCE VECTOR TO SEGMENT Z-AXIAL COORDINATES.	00002560
		00002570
	CALL ROT2 (TEMP, 1, 3, 1)	00002580
05	PRINT, 'AFTER CHANGING TO X-AXIAL COORDINATES'	00002590
05	WRITE (3,500) TEMP	00002600
05	PRINT, ' '	00002610
		00002620
	CONVERT SEGMENT Z-AXIAL COORDINATES TO X-AXIAL.	00002630
		00002640
	XYZ(JXYZ+3) = TEMP(3)	00002650
	XYZ(JXYZ+4) = TEMP(1)	00002660
	XYZ(JXYZ+5) = TEMP(2)	00002670
120	CONTINUE	00002680
	-----	00002690
	JOINT FORCES & TORQUES; LINEAR & ANGULAR ACCELERATIONS	00002700
	-----	00002710
	IN BOTH INERTIAL AND LOCAL COORDINATES	00002720
	LOCAL : ANGULAR VELOCITIES, ACCELERATIONS, U2 ARRAY	00002730
	INERTIAL : JOINT FORCES & TORQUES, LINEAR ACCEL., U1 ARRAY	00002740
		00002750
130	KT = KS + KP	00002760
	KT6 = KT * 6	00002770
		00002780
	TRANSFORM INERTIAL VECTORS TO SEGMENT X-AXIAL LOCAL VECTORS.	00002790
		00002800
04	PRINT, 'W, FORCES, TORQUES, ALPHA, ACCEL., U1, U2 IN ORIG. COORDS'	00002810
04	WRITE (3,600) (FORCE(I), I=10,30)	00002820
	DO 134 J = 13, 25, 3	00002830
	IF (J .EQ. 19) GO TO 134	00002840
	CALL ROT (FORCE, J, 30, RADS, +1)	00002850
	CALL CHANGE (FORCE, J, 30, 0.0)	00002860
134	CONTINUE	00002870
		00002880
	ROTATE LOCAL VECTORS TO X-AXIAL	00002890
		00002900
	DO 136 J = 10, 28, 9	00002910
136	CALL CHANGE (FORCE, J, 30, 0.0)	00002920
04	PRINT, 'SAME ITEMS IN SEGMENT X-AXIAL COORDINATES'	00002930
04	WRITE (3,600) (FORCE(J), J = 10, 30)	00002940
	IF (YES(LIMB)) 140, 140, 138	00002950
		00002960
	TRANSFORM "OTHER CONSTRAINT FORCE" TO SEGMENT LOCAL X-AXIAL	00002970
		00002980
138	CALL ROT (CNSTRN, 1, 6, RADS, +1)	00002990
	CALL CHANGE (CNSTRN, 1, 6, 0.0)	00003000

IF (KT .LE. 1) GO TO 200	00003510
K = KT - 1	00003520
DO 170 I = 1, KT1	00003530
J1 = 6 * I - 5	00003540
HOLD = XYZ(J1)	00003550
I1 = I + 1	00003560
NUM = J1	00003570
DO 158 J = I1, KT	00003580
J2 = 6 * J - 5	00003590
IF (XYZ(J2) .GE. HOLD) GO TO 158	00003600
NUM = J2	00003610
HOLD = XYZ(J2)	00003620
158 CONTINUE	00003630
IF (NUM .EQ. J1) GO TO 170	00003640
J1 = J1 - 1	00003650
NUM = NUM - 1	00003660
DO 160 K = 1, 6	00003670
J3 = 6 * K - 5	00003680
HOLD = XYZ(J1+K)	00003690
XYZ(J1+K) = XYZ(NUM+K)	00003700
160 XYZ(NUM+K) = HOLD	00003710
170 CONTINUE	00003720
200 IF (KT) 205, 205, 210	00003730
205 KT = 1	00003740
XYZ(1) = 0.0	00003750
GO TO 300	00003760
210 KT = KT * 6	00003770
300 CONTINUE	00003780
34 PRINT, 'CONTACT FORCES'	00003790
34 PRINT, 'NUMBER OF DATA ITEMS =', KT	00003800
34 K2 = KT / 6	00003810
34 IF (KT .GT. 1) WRITE (3,600) (XYZ(I), I = 1, KT)	00003820
:	00003830
:	00003840
:	00003850
WRITE (9) LIMB,KT,NCON,(FORCE(I),I = 10,24),(FORCE(I),I = 28,30),	00003860
\$ (XYZ(I),I = 1, KT), (CNSTRN(I), I = 1, NCON)	00003870
350 CONTINUE	00003880
STOP	00003890
500 FORMAT(1H ,3F10.4)	00003900
600 FORMAT(1H ,3F10.4, ' --- ',3F10.4)	00003910
END	00003920
SUBROUTINE ROT (V, IZ, N, P, K)	00003930
:	00003940
:	00003950
:	00003960
SUBROUTINE TO TRANSFORM VECTOR V THROUGH P RADIANS INTO ANOTHER	00003970
COORDINATE SYSTEM. K DETERMINES WHETHER THE TRANSFORMATION IS FROM	00003980
INERTIAL TO SEGMENT OR VICE-VERSA :	00003990
K = 1 INERTIAL TO SEGMENT TRANSFORMATION	00004000
K = -1 SEGMENT TO INERTIAL	

REAL R (3,3), V(N), VN(3), P(3)	00004010
COMMON /TRANS/ ICHEK	00004020
IF (ICHEK .EQ. 1) GO TO 10	00004030
ICHEK = 1	00004040
C1 = COS (P(1))	00004050
C2 = COS (P(2))	00004060
C3 = COS (P(3))	00004070
S1 = SIN (P(1))	00004080
S2 = SIN (P(2))	00004090
S3 = SIN (P(3))	00004100
R(1,1) = C2 * C1	00004110
R(2,1) = S3 * S2 * C1 - S1 * C3	00004120
R(3,1) = C3 * S2 * C1 + S1 * S3	00004130
R(1,2) = C2 * S1	00004140
R(2,2) = S3 * S2 * S1 + C1 * C3	00004150
R(3,2) = C3 * S2 * S1 - C1 * S3	00004160
R(1,3) = -S2	00004170
R(2,3) = S3 * C2	00004180
R(3,3) = C3 * C2	00004190
ENTRY ROT2 (V, IZ, N, K)	00004200
10 IZ1 = IZ - 1	00004210
IF (K) 40, 60, 20	00004220
20 DO 30 I = 1, 3	00004230
VN(I) = 0.0	00004240
DO 30 J = 1, 3	00004250
30 VN(I) = VN(I) + R(I,J) * V(IZ1+J)	00004260
GO TO 60	00004270
40 DO 50 I = 1, 3	00004280
VN(I) = 0.0	00004290
DO 50 J = 1, 3	00004300
50 VN(I) = VN(I) + R(J,I) * V(J)	00004310
60 DO 70 I = 1, 3	00004320
70 V(IZ1+I) = VN(I)	00004330
RETURN	00004340
END	00004350
SUBROUTINE VECTOR (N, V, C)	00004360
	00004370
	00004380
	00004390
	00004400
	00004410
	00004420
	00004430
	00004440
	00004450
	00004460
	00004470
	00004480
	00004490
	00004500

REAL N(3), V(3), C(3), D(3), HOLD	
HOLD = 0.0	
CALL CROSS (N, V, C)	
CALL CROSS (C, N, D)	
DO 10 J = 1, 3	
10 HOLD = HOLD + D(J) * D(J)	
HOLD = SQRT (HOLD)	
IF (HOLD .EQ. 0.0) RETURN	
DO 20 J = 1, 3	

20	C(J) = D(J) / HOLD	00004510
	RETURN	00004520
	END	00004530
	SUBROUTINE CROSS (A, B, C)	00004540
		00004550
	SUBROUTINE TO TAKE THE CROSS PRODUCT OF A CROSS B, AND RETURN	00004560
	IT IN VECTOR C. A ,B AND C ARE ALL VECTORS OF LENGTH 3.	00004570
		00004580
	REAL A(3), B(3), C(3)	00004590
	C(1) = A(2) * B(3) - A(3) * B(2)	00004600
	C(2) = -A(1) * B(3) + A(3) * B(1)	00004610
	C(3) = A(1) * B(2) - A(2) * B(1)	00004620
	RETURN	00004630
	END	00004640
	SUBROUTINE CHANGE (TEMP, I, N, D)	00004650
		00004660
	CHANGES Z-AXIAL TO X-AXIAL COORDINATES. SHIFTS NEW X BY	00004670
	D TO ALLOW TRANSFER OF OFIGIN TO PROXIMAL JOINT.	00004680
		00004690
	REAL TEMP(N)	00004700
	HOLD1 = TEMP(I)	00004710
	HOLD2 = TEMP(I+1)	00004720
	HOLD3 = TEMP(I+2)	00004730
	TEMP(I) = HOLD3 + D	00004740
	TEMP(I+1) = HOLD1	00004750
	TEMP(I+2) = HOLD2	00004760
	RETURN	00004770
	END	00004780
	SUBROUTINE FOR (K)	00004790
	DO 10 J = 1, K	00004800
10	READ (5)	00004810
	RETURN	00004820
	END	00004830
	SUBROUTINE BACK (K)	00004840
	DO 10 J = 1, K	00004850
10	BACKSPACE 5	00004860
	RETURN	00004870
	END	00004880

FINDS AREA OF BONE AT VARIOUS LENGTHS	00000010
INTEGER TITLE(19)	00000020
REAL A(40), D(40), DNORM(40), X(40), Y(40), Z(40), U, JP	00000030
READ (1,100) TITLE	00000040
WRITE (3,150) TITLE	00000050
PRINT, 'NORM LENGTH, LENGTH, AREA'	00000060
PPRINT, ' '	00000070
K = 1	00000080
10 READ (1,*,END=20) D(K), A(K)	00000090
K = K + 1	00000100
GO TO 10	00000110
20 K = K - 1	00000120
DO 30 I = 1, K	00000130
DNORM(I) = D(I) / D(K)	00000140
30 WRITE (3,200) DNORM(I), D(I), A(I)	00000150
CALL SPLINE (K, DNORM, A, X, Y, Z)	00000160
	00000170
EVALUATE @ 11 POINTS NORMALIZED	00000180
	00000190
PRINT, ' '	00000200
PRINT, ' '	00000210
PRINT, 'NORM LENGTH & AREA EVALUATED BY SEVAL'	00000220
PRINT, ' '	00000230
DO 40 I = 1, 11	00000240
U = FLOAT(I-1) / 10.0	00000250
TEMP = SEVAL (K, U, DNORM, A, X, Y, Z)	00000260
40 WRITE (3,200) U, TEMP	00000270
STOP	00000280
100 FORMAT (18A4)	00000290
150 FORMAT (' ID = ',18A4)	00000300
200 FORMAT (3F10.3)	00000310
END	00000320
SUBROUTINE SPLINE (N, X, Y, B, C, D)	00000330
INTEGER N	00000340
REAL X(N), Y(N), B(N), C(N), D(N)	00000350
INTEGER NM1, IB, I	00000360
REAL T	00000370
NM1 = N-1	00000380
IF (N .LT. 2) RETURN	00000390
IF (N .LT. 3) GO TO 50	00000400
D(1) = X(2) - X(1)	00000410
C(2) = (Y(2) - Y(1))/D(1)	00000420
DO 10 I = 2, NM1	00000430
D(I) = X(I+1) - X(I)	00000440
B(I) = 2.*(D(I-1) + D(I))	00000450
C(I+1) = (Y(I+1) - Y(I))/D(I)	00000460
C(I) = C(I+1) - C(I)	00000470
10 CONTINUE	00000480
B(1) = -D(1)	00000490
B(N) = -D(N-1)	00000500

C(1) = 0.	00000510
C(N) = 0.	00000520
IF (N .EQ. 3) GO TO 15	00000530
C(1) = C(3)/(X(4)-X(2)) - C(2)/(X(3)-X(1))	00000540
C(N) = C(N-1)/(X(N)-X(N-2)) - C(N-2)/(X(N-1)-X(N-3))	00000550
C(1) = C(1)*D(1)**2/(X(4)-X(1))	00000560
C(N) = -C(N)*D(N-1)**2/(X(N)-X(N-3))	00000570
15 DO 20 I = 2, N	00000580
T = D(I-1)/B(I-1)	00000590
B(I) = B(I) - T*D(I-1)	00000600
C(I) = C(I) - T*C(I-1)	00000610
20 CONTINUE	00000620
C(N) = C(N)/B(N)	00000630
DO 30 IB = 1, NM1	00000640
I = N-IB	00000650
C(I) = (C(I) - D(I)*C(I+1))/B(I)	00000660
30 CONTINUE	00000670
B(N) = (Y(N) - Y(NM1))/D(NM1) + D(NM1)*(C(NM1) + 2.*C(N))	00000680
DO 40 I = 1, NM1	00000690
B(I) = (Y(I+1) - Y(I))/D(I) - D(I)*(C(I+1) + 2.*C(I))	00000700
D(I) = (C(I+1) - C(I))/D(I)	00000710
C(I) = 3.*C(I)	00000720
40 CONTINUE	00000730
C(N) = 3.*C(N)	00000740
D(N) = D(N-1)	00000750
RETURN	00000760
50 B(1) = (Y(2)-Y(1))/(X(2)-X(1))	00000770
C(1) = 0.	00000780
D(1) = 0.	00000790
B(2) = B(1)	00000800
C(2) = 0.	00000810
D(2) = 0.	00000820
RETURN	00000830
END	00000840
REAL FUNCTION SEVAL(N, U, X, Y, B, C, D)	00000850
INTEGER N	00000860
REAL U, X(N), Y(N), B(N), C(N), D(N)	00000870
INTEGER I, J, K	00000880
REAL DX	00000890
DATA I/1/	00000900
IF (I .GE. N) I = 1	00000910
IF (U .LT. X(I)) GO TO 10	00000920
IF (U .LE. X(I+1)) GO TO 30	00000930
10 I = 1	00000940
J = N+1	00000950
20 K = (I+J)/2	00000960
IF (U .LT. X(K)) J = K	00000970
IF (U .GE. X(K)) I = K	00000980
IF (J .GT. I+1) GO TO 20	00000990
30 DX = U - X(I)	00001000

SEVAL = Y(I) + DX*(B(I) + DX*(C(I) + DX*D(I)))
RETURN
END

SDATA
FEMUR #1 -- RIGHT DISTAL VS. AVG. INERTIA
1.0 .226
2.0 .092
3.0 .076
4.0 .075
5.0 .074
6.0 .069
7.2 .067
8.2 .065
9.2 .066
10.2 .067
11.2 .073
12.2 .085
13.2 .105
14.2 .249
14.2 1.818

00001010
00001020
00001030
00001040
00001050
00001060
00001070
00001080
00001090
00001100
00001110
00001120
00001130
00001140
00001150
00001160
00001170
00001180
00001190
00001200


```
DOUBLE PRECISION FUNCTION ZEROIN(AX,BX,F,TOL)
DOUBLE PRECISION AX,BX,F,TOL
```

```
A ZERO OF THE FUNCTION F(X) IS COMPUTED IN THE INTERVAL AX,BX .
```

```
INPUT..
```

```
AX      LEFT ENDPOINT OF INITIAL INTERVAL
BX      RIGHT ENDPOINT OF INITIAL INTERVAL
F       FUNCTION SUBPROGRAM WHICH EVALUATES F(X) FOR ANY X IN
        THE INTERVAL AX,BX
TOL     DESIRED LENGTH OF THE INTERVAL OF UNCERTAINTY OF THE
        FINAL RESULT ( .GE. 0.0D0)
```

```
OUTPUT..
```

```
ZEROIN ABCISSA APPROXIMATING A ZERO OF F IN THE INTERVAL AX,BX
```

```
IT IS ASSUMED THAT F(AX) AND F(BX) HAVE OPPOSITE SIGNS
WITHOUT A CHECK. ZEROIN RETURNS A ZERO X IN THE GIVEN INTERVAL
AX,BX TO WITHIN A TOLERANCE  $4 * \text{MACHEPS} * \text{ABS}(X) + \text{TOL}$ , WHERE MACHEPS
IS THE RELATIVE MACHINE PRECISION.
```

```
THIS FUNCTION SUBPROGRAM IS A SLIGHTLY MODIFIED TRANSLATION OF
THE ALGOL 60 PROCEDURE ZERO GIVEN IN RICHARD BRENT, ALGORITHMS FOR
MINIMIZATION WITHOUT DERIVATIVES, PRENTICE - HALL, INC. (1973).
```

```
DOUBLE PRECISION A,B,C,D,E,EPS,FA,FB,FC,TOL1,XM,P,Q,R,S
DOUBLE PRECISION DABS,DSIGN
```

```
COMPUTE EPS, THE RELATIVE MACHINE PRECISION
```

```
EPS = 1.0D0
10 EPS = EPS/2.0D0
TOL1 = 1.0D0 + EPS
IF (TOL1 .GT. 1.0D0) GO TO 10
```

```
INITIALIZATION
```

```
A = AX
B = BX
FA = F(A)
FB = F(B)
```

```
BEGIN STEP
```

```
20 C = A
FC = FA
```

```
D = B - A
E = D
30 IF (DABS(FC) .GE. DABS(FB)) GO TO 40
A = B
B = C
C = A
FA = FB
FB = FC
FC = FA
```

```
:
: CONVERGENCE TEST
```

```
40 TOL1 = 2.0D0*EPS*DABS(B) + 0.5D0*TOL
XM = .5*(C - B)
IF (DABS(XM) .LE. TOL1) GO TO 90
IF (FB .EQ. 0.0D0) GO TO 90
```

```
:
: IS BISECTION NECESSARY
```

```
IF (DABS(E) .LT. TOL1) GO TO 70
IF (DABS(FA) .LE. DABS(FB)) GO TO 70
```

```
:
: IS QUADRATIC INTERPOLATION POSSIBLE
```

```
IF (A .NE. C) GO TO 50
```

```
:
: LINEAR INTERPOLATION
```

```
S = FB/FA
P = 2.0D0*XM*S
Q = 1.0D0 - S
GO TO 60
```

```
:
: INVERSE QUADRATIC INTERPOLATION
```

```
50 Q = FA/PC
R = FB/FC
S = FB/FA
P = S*(2.0D0*XM*Q*(Q - R) - (B - A)*(R - 1.0D0))
Q = (Q - 1.0D0)*(R - 1.0D0)*(S - 1.0D0)
```

```
:
: ADJUST SIGNS
```

```
60 IF (P .GT. 0.0D0) Q = -Q
P = DABS(P)
```

```
:
: IS INTERPOLATION ACCEPTABLE
```

```
IF ((2.0D0*P) .GE. (3.0D0*XM*Q - DABS(TOL1*Q))) GO TO 70
IF (P .GE. DABS(0.5D0*E*Q)) GO TO 70
```

```
E = D
D = P/Q
GO TO 80
```

```
:
: BISECTION
:
```

```
70 D = XM
E = D
```

```
: COMPLETE STEP
:
```

```
80 A = B
FA = FB
IF (DABS(D) .GT. TOL1) B = B + D
IF (DABS(D) .LE. TOL1) B = B + DSIGN(TOL1, XM)
FB = F(B)
IF ((FB*(FC/DABS(FC))) .GT. 0.0D0) GO TO 20
GO TO 30
```

```
: DONE
:
```

```
90 ZEROIN = B
RETURN
END
```

```
SUBROUTINE SPLINE (N, X, Y, B, C, D)
INTEGER N
DOUBLE PRECISION X(N), Y(N), B(N), C(N), D(N)
```

```
THE COEFFICIENTS B(I), C(I), AND D(I), I=1,2,...,N ARE COMPUTED
FOR A CUBIC INTERPOLATING SPLINE
```

$$S(X) = Y(I) + B(I)*(X-X(I)) + C(I)*(X-X(I))**2 + D(I)*(X-X(I))**3$$

```
FOR X(I) .LE. X .LE. X(I+1)
```

```
INPUT..
```

```
N = THE NUMBER OF DATA POINTS OR KNOTS (N.GE.2)
X = THE ABSCISSAS OF THE KNOTS IN STRICTLY INCREASING ORDER
Y = THE ORDINATES OF THE KNOTS
```

```
OUTPUT..
```

```
B, C, D = ARRAYS OF SPLINE COEFFICIENTS AS DEFINED ABOVE.
```

```
USING P TO DENOTE DIFFERENTIATION,
```

```
Y(I) = S(X(I))
B(I) = SP(X(I))
C(I) = SPP(X(I))/2
D(I) = SPPP(X(I))/6 (DERIVATIVE FROM THE RIGHT)
```

```
THE ACCOMPANYING FUNCTION SUBPROGRAM SEVAL CAN BE USED
TO EVALUATE THE SPLINE.
```

```
INTEGER NM1, IB, I
DOUBLE PRECISION T
```

```
NM1 = N-1
IF ( N .LT. 2 ) RETURN
IF ( N .LT. 3 ) GO TO 50
```

```
SET UP TRIDIAGONAL SYSTEM
```

```
B = DIAGONAL, D = OFFDIAGONAL, C = RIGHT HAND SIDE.
```

```
D(1) = X(2) - X(1)
C(2) = (Y(2) - Y(1))/D(1)
DO 10 I = 2, NM1
  D(I) = X(I+1) - X(I)
  B(I) = 2.*(D(I-1) + D(I))
  C(I+1) = (Y(I+1) - Y(I))/D(I)
  C(I) = C(I+1) - C(I)
```

10 CONTINUE

END CONDITIONS. THIRD DERIVATIVES AT X(1) AND X(N)
OBTAINED FROM DIVIDED DIFFERENCES

B(1) = -D(1)

B(N) = -D(N-1)

C(1) = 0.

C(N) = 0.

IF (N .EQ. 3) GO TO 15

C(1) = C(3)/(X(4)-X(2)) - C(2)/(X(3)-X(1))

C(N) = C(N-1)/(X(N)-X(N-2)) - C(N-2)/(X(N-1)-X(N-3))

C(1) = C(1)*D(1)**2/(X(4)-X(1))

C(N) = -C(N)*D(N-1)**2/(X(N)-X(N-3))

FORWARD ELIMINATION

15 DO 20 I = 2, N

T = D(I-1)/B(I-1)

B(I) = B(I) - T*D(I-1)

C(I) = C(I) - T*C(I-1)

20 CONTINUE

BACK SUBSTITUTION

C(N) = C(N)/B(N)

DO 30 IB = 1, NM1

I = N-IB

C(I) = (C(I) - D(I)*C(I+1))/B(I)

30 CONTINUE

C(I) IS NOW THE SIGMA(I) OF THE TEXT

COMPUTE POLYNOMIAL COEFFICIENTS

B(N) = (Y(N) - Y(NM1))/D(NM1) + D(NM1)*(C(NM1) + 2.*C(N))

DO 40 I = 1, NM1

B(I) = (Y(I+1) - Y(I))/D(I) - D(I)*(C(I+1) + 2.*C(I))

D(I) = (C(I+1) - C(I))/D(I)

C(I) = 3.*C(I)

40 CONTINUE

C(N) = 3.*C(N)

D(N) = D(N-1)

RETURN

50 B(1) = (Y(2)-Y(1))/(X(2)-X(1))

C(1) = 0.

D(1) = 0.

B(2) = B(1)

C(2) = 0.

D(2) = 0.
RETURN
END

```
DOUBLE PRECISION FUNCTION SEVAL(N, U, X, Y, B, C, D)
INTEGER N
DOUBLE PRECISION U, X(N), Y(N), B(N), C(N), D(N)
```

```
THIS SUBROUTINE EVALUATES THE CUBIC SPLINE FUNCTION
```

```
SEVAL = Y(I) + B(I)*(U-X(I)) + C(I)*(U-X(I))**2 + D(I)*(U-X(I))**3
```

```
WHERE X(I) .LT. U .LT. X(I+1), USING HORNER'S RULE
```

```
IF U .LT. X(1) THEN I = 1 IS USED.
IF U .GE. X(N) THEN I = N IS USED.
```

```
INPUT..
```

```
N = THE NUMBER OF DATA POINTS
U = THE ABSCISSA AT WHICH THE SPLINE IS TO BE EVALUATED
X,Y = THE ARRAYS OF DATA ABSCISSAS AND ORDINATES
B,C,D = ARRAYS OF SPLINE COEFFICIENTS COMPUTED BY SPLINE
```

```
IF U IS NOT IN THE SAME INTERVAL AS THE PREVIOUS CALL, THEN A
BINARY SEARCH IS PERFORMED TO DETERMINE THE PROPER INTERVAL.
```

```
INTEGER I, J, K
DOUBLE PRECISION DX
DATA I/1/
IF ( I .GE. N ) I = 1
IF ( U .LT. X(I) ) GO TO 10
IF ( U .LE. X(I+1) ) GO TO 30
```

```
BINARY SEARCH
```

```
10 I = 1
J = N+1
20 K = (I+J)/2
IF ( U .LT. X(K) ) J = K
IF ( U .GE. X(K) ) I = K
IF ( J .GT. I+1 ) GO TO 20
```

```
EVALUATE SPLINE
```

```
30 DX = U - X(I)
SEVAL = Y(I) + DX*(B(I) + DX*(C(I) + DX*D(I)))
RETURN
END
```

DOUBLE PRECISION FUNCTION FMIN (AX, BX, F, TOL)
DOUBLE PRECISION AX, BX, F, TOL

AN APPROXIMATION X TO THE POINT WHERE F ATTAINS A MINIMUM ON
THE INTERVAL (AX, BX) IS DETERMINED.

INPUT..

AX LEFT ENDPOINT OF INITIAL INTERVAL
BX RIGHT ENDPOINT OF INITIAL INTERVAL
F FUNCTION SUBPROGRAM WHICH EVALUATES F(X) FOR ANY X
IN THE INTERVAL (AX, BX)
TOL DESIRED LENGTH OF THE INTERVAL OF UNCERTAINTY OF THE FINAL
RESULT (.GE. 0.0D0)

OUTPUT..

FMIN ABCISSA APPROXIMATING THE POINT WHERE F ATTAINS A MINIMUM

THE METHOD USED IS A COMBINATION OF GOLDEN SECTION SEARCH AND
SUCCESSIVE PARABOLIC INTERPOLATION. CONVERGENCE IS NEVER MUCH SLOWER
THAN THAT FOR A FIBONACCI SEARCH. IF F HAS A CONTINUOUS SECOND
DERIVATIVE WHICH IS POSITIVE AT THE MINIMUM (WHICH IS NOT AT AX OR
BX), THEN CONVERGENCE IS SUPERLINEAR, AND USUALLY OF THE ORDER OF
ABOUT 1.324....

THE FUNCTION F IS NEVER EVALUATED AT TWO POINTS CLOSER TOGETHER
THAN $\text{EPS} \cdot \text{ABS}(\text{FMIN}) + (\text{TOL}/3)$, WHERE EPS IS APPROXIMATELY THE SQUARE
ROOT OF THE RELATIVE MACHINE PRECISION. IF F IS A UNIMODAL
FUNCTION AND THE COMPUTED VALUES OF F ARE ALWAYS UNIMODAL WHEN
SEPARATED BY AT LEAST $\text{EPS} \cdot \text{ABS}(X) + (\text{TOL}/3)$, THEN FMIN APPROXIMATES
THE ABCISSA OF THE GLOBAL MINIMUM OF F ON THE INTERVAL AX, BX WITH
AN ERROR LESS THAN $3 \cdot \text{EPS} \cdot \text{ABS}(\text{FMIN}) + \text{TOL}$. IF F IS NOT UNIMODAL,
THEN FMIN MAY APPROXIMATE A LOCAL, BUT PERHAPS NON-GLOBAL, MINIMUM TO
THE SAME ACCURACY.

THIS FUNCTION SUBPROGRAM IS A SLIGHTLY MODIFIED VERSION OF THE
ALGOL 60 PROCEDURE LOCALMIN GIVEN IN RICHARD BRENT, ALGORITHMS FOR
MINIMIZATION WITHOUT DERIVATIVES, PRENTICE - HALL, INC. (1973).

DOUBLE PRECISION A, B, C, D, E, EPS, XM, P, Q, R, TOL1, TOL2, U, V, W
DOUBLE PRECISION FU, FV, FW, FX, X
DOUBLE PRECISION DABS, DSQRT, DSIGN

C IS THE SQUARED INVERSE OF THE GOLDEN RATIO

$C = 0.5D0 \cdot (3. - \text{DSQRT}(5.0D0))$

: EPS IS APPROXIMATELY THE SQUARE ROOT OF THE RELATIVE MACHINE
: PRECISION.

EPS = 1.0D00
10 EPS = EPS/2.0D00
TOL1 = 1.0D0 + EPS
IF (TOL1 .GT. 1.0D00) GO TO 10
EPS = DSQRT(EPS)

:
: INITIALIZATION

A = AX
B = BY
V = A + C*(B - A)
W = V
X = V
E = 0.0D0
FX = F(X)
FV = FX
FW = FX

:
: MAIN LOOP STARTS HERE

20 XM = 0.5D0*(A + B)
TOL1 = EPS*DABS(X) + TOL/3.0D0
TOL2 = 2.0D0*TOL1

:
: CHECK STOPPING CRITERION

IF (DABS(X - XM) .LE. (TOL2 - 0.5D0*(B - A))) GO TO 90

: IS GOLDEN-SECTION NECESSARY

IF (DABS(E) .LE. TOL1) GO TO 40

: FIT PARABOLA

R = (X - W)*(FX - FV)
Q = (X - V)*(FX - FW)
P = (X - V)*Q - (X - W)*R
Q = 2.0D00*(Q - R)
IF (Q .GT. 0.0D0) P = -P
Q = DABS(Q)
R = E
E = D

: IS PARABOLA ACCEPTABLE

30 IF (DABS(P) .GE. DABS(0.5D0*Q*R)) GO TO 40
IF (P .LE. Q*(A - X)) GO TO 40

IF (P .GE. Q*(B - X)) GO TO 40

A PARABOLIC INTERPOLATION STEP

D = P/Q
U = X + D

F MUST NOT BE EVALUATED TOO CLOSE TO AX OR BX

IF ((U - A) .LT. TOL2) D = DSIGN(TOL1, XM - X)
IF ((B - U) .LT. TOL2) D = DSIGN(TOL1, XM - X)
GO TO 50

A GOLDEN-SECTION STEP

40 IF (X .GE. XM) E = A - X
IF (X .LT. XM) E = B - X
D = C*E

F MUST NOT BE EVALUATED TOO CLOSE TO X

50 IF (DABS(D) .GE. TOL1) U = X + D
IF (DABS(D) .LT. TOL1) U = X + DSIGN(TOL1, D)
FU = F(U)

UPDATE A, B, V, W, AND X

IF (FU .GT. FX) GO TO 60
IF (U .GE. X) A = X
IF (U .LT. X) B = X
V = W
FV = FW
W = X
FW = FX
X = U
PX = FU
GO TO 20

60 IF (U .LT. X) A = U
IF (U .GE. X) B = U
IF (FU .LE. FW) GO TO 70
IF (W .EQ. X) GO TO 70
IF (FU .LE. FV) GO TO 80
IF (V .EQ. X) GO TO 80
IF (V .EQ. W) GO TO 80
GO TO 20

70 V = W
FV = FW
W = U
FW = FU
GO TO 20

```
80 V = U  
   PV = FU  
   GO TO 20
```

```
.....  
END OF MAIN LOOP
```

```
90 PHIN = X  
   RETURN  
   END
```

```

*****00000010
PROGRAM NAME : MAPGRAF *00000020
WRITTEN BY : J.F. BRANDEAU *00000030
COMPILER(S) : WATFIV (DOUBLE PRECISION) *00000040
----- *00000050
PURPOSE : USE COEFFICIENTS X TO PRODUCE DATA POINTS THAT WILL BE READ *00000060
BY SAS TO PRODUCE A 3-D PLOT OF A STRESS-STRAIN SURFACE. PROGRAM *00000070
CREATES A GRID OF STRAIN VS. LOG STRAIN RATE, AND USES SUBROUTINE *00000080
ZEROIN TO SOLVE FOR THE VALUE OF STRESS FOR EACH UNIQUE COMBINATION. *00000090
THE DENSITY OF THE GRID, THE RANGE OF THE GRID, AND THE NON- *00000100
PERMISSIBLE REGION ARE ALL CONTROLLABLE. THE NON-PERMISSIBLE REGION *00000110
OF THE GRID WILL HAVE NO POINTS ON IT, TO PREVENT EXTRAPOLATION PAST *00000120
EXPERIMENTAL LIMITS. THE SHAPE OF THE EDGE IS CALCULATED USING A *00000130
SPLINE CURVE FIT TO THE MAX VALUES OF STRAIN FOR EACH OBSERVED STRESS *00000140
THIS IS DONE BY SUBROUTINE SPLINE AND SEVAL. *00000150
----- *00000160
VARIABLES : *00000170
A1 & B1 : A1 IS THE MAX OBSERVED VALUES OF STRAIN, ONE FOR EACH VALUE *00000180
OF STRAIN RATE B1. B1 MUST START WITH LOWEST VALUE OF STRAIN RATE *00000190
FIRST, OR THE SPLINE WILL BE INCORRECTLY CALCULATED. *00000200
K : THE NUMBER OF UNIQUE STRAIN RATES OBSERVED, ALSO THE NUMBER OF *00000210
ELEMENTS I EACH OF A1 & B1. *00000220
SIGMAX : GREATER THAN THE EXPECTED MAX VALUE OF STRESS (KSI) TO BE *00000230
USED AS THE UPPER LIMIT OF SEARCH FOR ZEROIN. FOR EACH UNIQUE *00000240
STRAIN RATE THIS MAY BE CORRECTED DOWNWARD BY THE PROGRAM TO PREVENT *00000250
OVERFLOWS. THIS IS SET BY THE USER. *00000260
A,B,N,D,C & COEFF : A,B,N,D, & C ARE THE MANTISSA OF THE COEFFICIENTS *00000270
FOR THE EQUATION, AND COEFF IS THE EXPONENTS. THE PRODUCTS *00000280
A * COEFF (1), B * COEFF (2), ETC... ARE THE COEFFICIENTS. *00000290
X1 : THE STARTING POINT FOR VALUES OF STRAIN. THIS MUST BE GREATER *00000300
THAN ZERO, AS THE EQUATION IS INDETERMINATE @ STRAIN = 0.0. *00000310
Y1 : THE STARTING POINT FOR LOG RATE. Y1 = -3 IS A STARTING POINT OF *00000320
STRAIN RATE = 0.001 / SEC. *00000330
DY : STEP LENGTH IN LOG RATE (OR Y) DIRECTION. *00000340
DX : " " " STRAIN (OR X) " *00000350
YMAX : MAX VALUE OF LOG RATE TO BE USED & IS ASSIGNED BY THE USER. *00000360
THOLD = MAX VALUE OF STRAIN TO USE, & IS SET BY PROGRAM TO EQUAL *00000370
MAX VALUE IN A1. *00000380
*00000390
*00000400
*00000410
*00000420
*00000430
*00000440
*00000450
*00000460
*00000470
*00000480
*00000490
*00000500

```

```

: TOL : CONVERGENCE CRITERIA FOR ZEROIN. *00000510
: IX : # OF POINTS ON STRAIN (X) AXIS. *00000520
: IY : " " " " LOG RATE (Y) " *00000530
: THE TOTAL GRID WILL CONSIST OF IY * IX - NON-PERMISSIBLE POINTS. *00000540
: C1,D1,E1 : WORK VECTORS FOR SPLINE & SEVAL. MUST NOT BE CHANGED. *00000550
: ----- *00000560
: I/O REQUIREMENTS : *00000570
: FILE #5 : MANTISSA OF COEFFICIENT VECTOR AND EXPONENT OF COEFFICIENT *00000580
: VECTOR ON TWO RECORDS. *00000590
: ----- *00000600
: OPTIONS : NONE *00000610
: ***** *00000620
: IMPLICIT REAL * 8 (A-H, N, O-Z) 00000630
: EXTERNAL FUNCT 00000640
: DIMENSION Y(40), X(40), YE(40), COEFF(5) 00000650
: DIMENSION A1(6), B1(6), C1(6), D1(6), E1(6) 00000660
: COMMON X, Y, I, J, A, B, C, D, N, RATEB, RATED, TLOG 00000670
: COMMON /SUB1/ GEPS 00000680
: DATA A1/6.90D-3, 6.5D-3, 5.7D-3, 5.1D-3, 10.5D-3, 9.2D-3/ 00000690
: DATA B1 /.001, 0.10, 10.0, 150.0, 0.0, 0.0/ 00000700
:
: SET PROGRAM PARAMETERS 00000710
:
: K = 4 00000720
: YMAX = 2.3 00000730
: SIGMAX = 25.0D0 00000740
: X1 = 0.05D-3 ; Y1 = -3.0D0 00000750
: TOL = 1.0D-8 ; IX = 30 ; IY = 30 00000760
: THOLD = 0.0 00000770
: DO 5 J = 1, K 00000780
: IF (A1(J) .GT. THOLD) THOLD = A1(J) 00000790
: 5 B1(J) = DLOG10 (B1(J)) 00000800
:
: CALL SPLINE (K, B1, A1, C1, D1, E1) 00000810
: READ (5,*) A, B, N, D, C 00000820
: READ (5,*) COEFF 00000830
: A = A * COEFF(1) ; B = B * COEFF(2) ; N = N * COEFF(3) 00000840
: D = D * COEFF(4) ; C = C * COEFF(5) 00000850
: PRINT, 'A ',A ; PRINT, 'B ',B ; PRINT, 'N ',N 00000860
: PRINT, 'D ',D ; PRINT,'C ',C 00000870
:
: CHECK FOR VALUES THAT WILL CAUSE OVER/UNDER FLOW 00000880
:
: IF (A .EQ. 0.0D0) THEN DO 00000890
: TLOG = -80.0D0 00000900
:
: ***** *00000910

```

ELSE DO	00001010
TLOG = DLOG10 (A)	00001020
ENDIF	00001030
DX = (THOLD - X1) / DFLOAT (IX-1)	00001040
DY = (YMAX - Y1) / DFLOAT (IY-1)	00001050
DO 10 J = 1, IY	00001060
Y(J) = DFLOAT(J-1) * DY + Y1	00001070
10 YE(J) = 10.0 ** Y(J)	00001080
DO 20 I = 1, IX	00001090
20 X(I) = DFLOAT(I-1) * DX + X1	00001100
CALCULATE MACHINE EPSILON FOR ZEROIN	00001110
GEPS = 1.0D0	00001120
24 GEPS = GEPS/2.0D0	00001130
TOL1 = 1.0D0 + GEPS	00001140
IF (TOL1 .GT. 1.0D0) GO TO 24	00001150
BEGIN SOLUTIONS	00001160
DO 40 J = 1, IY	00001170
CORRECT BX IF NEEDED TO PREVENT OVERFLOW	00001180
BX = SIGMAX	00001190
RATEB = DLOG10 (YE(J) ** B)	00001200
25 AX = N * DLOG10 (BX) + RATEB	00001210
IF (AX .GT. 75.0D0) THEN DO	00001220
BX = BX - 0.5D0	00001230
GO TO 25	00001240
ENDIF	00001250
RATEB = YE(J) ** B	00001260
RATED = C * YE(J) ** D	00001270
Z = X(1) * RATED / 1.3	00001280
TEMP = SEVAL (K, Y(J), B1, A1, C1, D1, E1)	00001290
ALLOW FOR SPLINE RIPPLE	00001300
IF (TEMP .GT. (THOLD * 0.97D0)) TEMP = THOLD	00001310
DO 30 I = 1, IX	00001320
IF (X(I) .GT. TEMP) GO TO 40	00001330
AX = Z	00001340
Z = ZEROIN (AX, BX, FUNCT, TOL)	00001350
WRITE (6,400) X(I), Y(J), YE(J), Z	00001360
30 CONTINUE	00001370
40 CONTINUE	00001380
STOP	00001390
400 FORMAT (1H ,5(1PD13.5))	00001400
	00001410
	00001420
	00001430
	00001440
	00001450
	00001460
	00001470
	00001480
	00001490
	00001500

END	00001510
DOUBLE PRECISION FUNCTION FUNCT(STRESS)	00001520
IMPLICIT REAL * 8 (A-H, N, O-Z)	00001530
DIMENSION Y(40), X(40)	00001540
COMMON X, Y, I, J, A, B, C, D, N, RATEB, RATED, TLOG	00001550
T1 = STRESS / RATED	00001560
IF (STRESS .GT. 1.0D-1) GO TO 10	00001570
IF ((N * DLOG10(STRESS)) .GT. -60.0D0) GO TO 10	00001580
HOLD = -25.0D0	00001590
GO TO 15	00001600
10 T2 = STRESS ** N	00001610
HOLD = DLOG10 (T2)	00001620
15 IF ((HOLD + TLOG) .LT. -17.0D0) THEN DO	00001630
FUNCT = T1 - X(I)	00001640
ELSE DO	00001650
FUNCT = T1 + A * T2 * RATEB - X(I)	00001660
ENDIF	00001670
25 RETURN	00001680
END	00001690
!SOPTIONS NOLIST	00001700
DOUBLE PRECISION FUNCTION ZEROIN(AX,BX,F,TOL)	00001710
DOUBLE PRECISION AX,BX,F,TOL	00001720
DOUBLE PRECISION A,B,C,D,E, EPS,FA,FB,FC,TOL1,XM,P,Q,R,S	00001730
DOUBLE PRECISION DABS,DSIGN	00001740
COMMON /SUB1/ EPS	00001750
A = AX	00001760
B = BX	00001770
FA = F(A)	00001780
FB = F(B)	00001790
20 C = A	00001800
FC = FA	00001810
D = B - A	00001820
E = D	00001830
30 IF (DABS(FC) .GE. DABS(FB)) GO TO 40	00001840
A = B	00001850
B = C	00001860
C = A	00001870
FA = FB	00001880
FB = FC	00001890
FC = FA	00001900
40 TOL1 = 2.0D0*EPS*DABS(B) + 0.5D0*TOL	00001910
XM = .5*(C - B)	00001920
IF (DABS(XM) .LE. TOL1) GO TO 90	00001930
IF (FB .EQ. 0.0D0) GO TO 90	00001940
IF (DABS(E) .LT. TOL1) GO TO 70	00001950
IF (DABS(FA) .LE. DABS(FB)) GO TO 70	00001960
IF (A .NE. C) GO TO 50	00001970
S = FB/FA	00001980
P = 2.0D0*XM*S	00001990
Q = 1.0D0 - S	00002000

GO TO 60	00002010
50 Q = FA/FC	00002020
R = FB/FC	00002030
S = FB/FA	00002040
P = S*(2.0D0*XM*Q*(Q - R) - (B - A)*(R - 1.0D0))	00002050
Q = (Q - 1.0D0)*(R - 1.0D0)*(S - 1.0D0)	00002060
60 IF (P .GT. 0.0D0) Q = -Q	00002070
P = DABS(P)	00002080
IF ((2.0D0*P) .GE. (3.0D0*XM*Q - DABS(TOL1*Q))) GO TO 70	00002090
IF (P .GE. DABS(0.5D0*E*Q)) GO TO 70	00002100
E = D	00002110
D = P/Q	00002120
GO TO 80	00002130
70 D = XM	00002140
E = D	00002150
80 A = B	00002160
FA = FB	00002170
IF (DABS(D) .GT. TOL1) B = B + D	00002180
IF (DABS(D) .LE. TOL1) B = B + DSIGN(TOL1, XM)	00002190
FB = F(B)	00002200
IF ((FB*(FC/DABS(FC))) .GT. 0.0D0) GO TO 20	00002210
GO TO 30	00002220
90 ZERGIN = B	00002230
RETURN	00002240
END	00002250
SUBROUTINE SPLINE (N, X, Y, B, C, D)	00002260
INTEGER N	00002270
DOUBLE PRECISION X(N), Y(N), B(N), C(N), D(N)	00002280
INTEGER NM1, IB, I	00002290
DOUBLE PRECISION T	00002300
NM1 = N-1	00002310
IF (N .LT. 2) RETURN	00002320
IF (N .LT. 3) GO TO 50	00002330
D(1) = X(2) - X(1)	00002340
C(2) = (Y(2) - Y(1))/D(1)	00002350
DO 10 I = 2, NM1	00002360
D(I) = X(I+1) - X(I)	00002370
B(I) = 2.*(D(I-1) + D(I))	00002380
C(I+1) = (Y(I+1) - Y(I))/D(I)	00002390
C(I) = C(I+1) - C(I)	00002400
10 CONTINUE	00002410
B(1) = -D(1)	00002420
B(N) = -D(N-1)	00002430
C(1) = 0.	00002440
C(N) = 0.	00002450
IF (N .EQ. 3) GO TO 15	00002460
C(1) = C(3)/(X(4)-X(2)) - C(2)/(X(3)-X(1))	00002470
C(N) = C(N-1)/(X(N)-X(N-2)) - C(N-2)/(X(N-1)-X(N-3))	00002480
C(1) = C(1)*D(1)**2/(X(4)-X(1))	00002490
C(N) = -C(N)*D(N-1)**2/(X(N)-X(N-3))	00002500

15 DO 20 I = 2, N	00002510
T = D(I-1)/B(I-1)	00002520
B(I) = B(I) - T*D(I-1)	00002530
C(I) = C(I) - T*C(I-1)	00002540
20 CONTINUE	00002550
C(N) = C(N)/B(N)	00002560
DO 30 IB = 1, NM1	00002570
I = N-IB	00002580
C(I) = (C(I) - D(I)*C(I+1))/B(I)	00002590
30 CONTINUE	00002600
B(N) = (Y(N) - Y(NM1))/D(NM1) + D(NM1)*(C(NM1) + 2.*C(N))	00002610
DO 40 I = 1, NM1	00002620
B(I) = (Y(I+1) - Y(I))/D(I) - D(I)*(C(I+1) + 2.*C(I))	00002630
D(I) = (C(I+1) - C(I))/D(I)	00002640
C(I) = 3.*C(I)	00002650
40 CONTINUE	00002660
C(N) = 3.*C(N)	00002670
D(N) = D(N-1)	00002680
RETURN	00002690
50 B(1) = (Y(2)-Y(1))/(X(2)-X(1))	00002700
C(1) = 0.	00002710
D(1) = 0.	00002720
B(2) = B(1)	00002730
C(2) = 0.	00002740
D(2) = 0.	00002750
RETURN	00002760
END	00002770
DOUBLE PRECISION FUNCTION SEVAL(N, U, X, Y, B, C, D)	00002780
INTEGER N	00002790
DOUBLE PRECISION U, X(N), Y(N), B(N), C(N), D(N)	00002800
INTEGER I, J, K	00002810
DOUBLE PRECISION DX	00002820
DATA I/1/	00002830
IF (I .GE. N) I = 1	00002840
IF (U .LT. X(I)) GO TO 10	00002850
IF (U .LE. X(I+1)) GO TO 30	00002860
10 I = 1	00002870
J = N+1	00002880
20 K = (I+J)/2	00002890
IF (U .LT. X(K)) J = K	00002900
IF (U .GE. X(K)) I = K	00002910
IF (J .GT. I+1) GO TO 20	00002920
30 DX = U - X(I)	00002930
SEVAL = Y(I) + DX*(B(I) + DX*(C(I) + DX*D(I)))	00002940
RETURN	00002950
END	00002960

```

C$OPTIONS DEC,CCOMB=19
C PROGRAM NAME: STRESS
REAL DUMMY(24), OUT(6), X(15), L95(4), YOI(15)
REAL ROJ, AREA(17), DET, SUBJL, SUBJW, L1L2, W1W2, DELX(5)
REAL INER1, INER2, MASS, SIG(2), FORCE(6,4), MOMENT(3)
REAL A, B, C, CG(15), WEIGHT(15), INERT(45), DI(4)
REAL OMEGA(3), FJ(3), TJ(3), ALPHA(3), ACCEL(3), U2(3)
REAL POSIT(24), INSTRN(6)
INTEGER SEG(8), CHECK(4), NCON, KCON, YES, RT6, J, K, NSEGS, NT, I
INTEGER ODD
EQUIVALENCE (DUMMY(1), FORCE(1)), (T, TEMP), (S2, TEMP, SUBJL)
EQUIVALENCE (A1, S1, SUBJW), (INER2, SUBJL)
COMMON /05/ FORCE
COMMON /06/ BONEP(8), BONEA(8), BONEI(8), L1L2, W1W2, A1, INER1,
RORIG, INER2, R0
DATA L95 /18.89, 15.98, 11.02, 11.0 /
DATA SEG /'RUL', 'ELL', 'LUL', 'LLL', 'RUA', 'RLA', 'LUA', 'LLA' /
DATA GINCH / 386.0886 /, PI / 3.141593 /, W95 / 217.0 /
DET (A,B) = SQRT (A * A + B * B / 2.)
S1MAX = 0.0
S2MAX = 0.0
TMAX = 0.0
WRITE (8,5)
5 FORMAT (' INPUT SEGMENT # TO BE ANALYZED (INTEGER 1-7) ')
DO 7 I = 1, 8
WRITE (8,6) I, SEG(I)
6 FORMAT ('H', I3, ' = ', A4)
7 CONTINUE
READ (1,*) J
NSEGS = 7
READ (5) NT, DLT, KCON4, (POSIT(I), I = 1, KCON4)
YES = 0
DO 8 I = 1, KCON4, 4
IF (POSIT(I) .NE. J) GO TO 8
YES = I
GO TO 9
8 CONTINUE
9 FIND MASS, DI AND SEMI-MAJOR AXES OF SEGMENT J FROM FILE #5
9 READ (5) J
IF (J .NE. I) CALL FOR (J-1, 5)
READ (5) K, MASS, A, B, C
IF (J .NE. 8) CALL FOR (8-J, 5)
IF (K .NE. J) PRINT, 'LOCKING FOR SEGMENT FILE', J, ' FOUND', K
WRITE (3,150) J, SEG(J), NT, NSEGS
WRITE (3,155) MASS, A, B, C
IL = YES + 1
IR = YES + 3

```

AD-A112 456

DUKE UNIV DURHAM NC DEPT OF MECHANICAL ENGINEERING A--ETC F/8 6/19
DEVELOPMENT OF A STRAIN RATE DEPENDENT LONG BONE INJURY CRITERI--ETC(U)
JAN 82 T K HIGHT

AFOSR-81-0062

UNCLASSIFIED

AFOSR-TR-82-0138

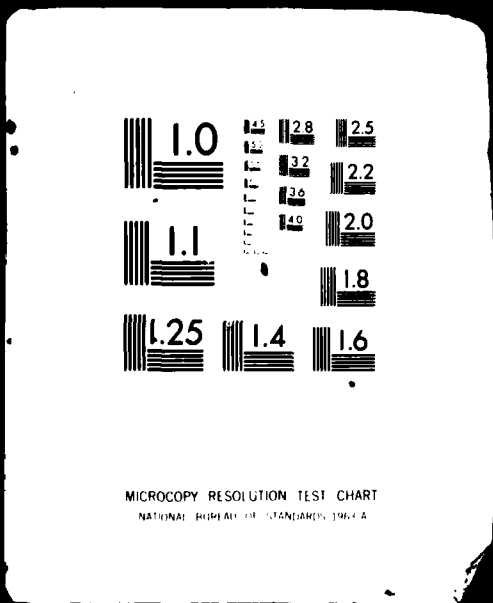
NL

2 of 2
1-1-82



END
DATE
INDEXED
1-1-82
DTIC

2 OF 2
AD
A112458



	IF (YES .GT. 0) WRITE (3,156) (POSIT(I), I = IL, IR)	00000510
C	SET JNUM TO EQUAL PROPER COUNTER IN GEOMETRY FILE #5.	00000520
C	READ (3,*) CHECK	00000530
	GO TO (10, 20, 30, 40, 30, 40), J	00000540
	10 JNUM = 1	00000550
	ODD = 1	00000560
	GO TO 50	00000570
	20 JNUM = 2	00000580
	ODD = -1	00000590
	CALL FOR (CHECK(1), 6)	00000600
	GO TO 50	00000610
	30 JNUM = 3	00000620
	ODD = 1	00000630
	CALL FOR (CHECK(1) + CHECK(2), 6)	00000640
	GO TO 50	00000650
	40 JNUM = 4	00000660
	ODD = -1	00000670
	CALL FOR (CHECK(1) + CHECK(2) + CHECK(3), 6)	00000680
	GO TO 50	00000690
	50 WRITE (3,157) DI(JNUM), L95(JNUM)	00000700
C	CREATE PERCENT-OF-LENGTH POINTS TO BE USED BETWEEN 20 & 80	00000710
C	SOLVE FOR NEEDED VALUES AT EACH FREE-BODY SECTION.	00000720
C	VOL = 4. * PI * A * B * B / 3.	00000730
	R0 = MASS / VOL	00000740
	A2 = A * A	00000750
	A3 = A * A2	00000760
	A4 = A * A3	00000770
	B2 = B * B	00000780
	B3 = B * B2	00000790
	WRITE (3,+25)	00000800
	DO 60 I = 1, NSEGS	00000810
	K = 3 * I - 2	00000820
	DELX(I) = 0.20 * (R0 * (A2 * B2 - A3 * B3) / (A2 * B2 - A3 * B3) + 0.50 / BLOCK (NSEGS - 1))	00000830
	X(I) = DELX(I) * L95(JNUM)	00000840
C	XI WITH RESPECT TO C.G. OF FULL ELLIPSOID	00000850
C	X1 = X(I) - DI(JNUM)	00000860
	X2 = X1 * X1	00000870
	X3 = X2 * X1	00000880
	X4 = X3 * X1	00000890
C	FIND WEIGHT (OR MASS) OF FREE-BODY SECTION I.	00000900
C	WEIGHT(I) = R0 * PI * B2 * (A2 * X1 - X3) + 2 * A3 * X1 / A2	00000910
C	FIND C.G. OF FREE-BODY SECTION I RELATIVE TO C.G. OF ELLIPSOID	00000920
C		00000930
		00000940
		00000950
		00000960
		00000970
		00000980
		00000990
		00001000
		00001010

```

CG(I) = 3. * (2.*X2*A2 - X4 - A4) / (3.*A2*X1 - X3 + 2.*A3) / 4. 00001020
C
C FIND I(XX) OF FREE-BODY SECTION I ABOUT POINT X(I) 00001030
C 00001040
TEMP = PI * RO * B2 * B2 / (2. * A4) * 00001060
$ (A4*X1 + X4*X1/5. - 2.*A2*X3/3. + 8.*A2*A3/15.) 00001070
INERT(K) = TEMP / GINCH 00001080
C
C FIND I(YY) = I(ZZ) OF FREE-BODY SECTION I ABOUT POINT X(I) 00001090
C 00001100
TEMP = RO * PI * B2 * (X3* (1./3. - X2 / (30.*A2)) 00001120
$ + X1*B2/4. * PI - 2.*X2/(3.*A2) + X4/(5.*A4)) 00001130
$ + 2. / 15. * (A3 + A*B2) + A2*X1/2. + 2.*X2*1/3. ) 00001140
INERT(K+1) = TEMP / GINCH 00001150
INERT(K+2) = INERT(K+1) 00001160
WRITE (3,45) I, X1, HEIGHT(I), G(I), INERT(K), INERT(K+1) 00001170
C
60 CONTINUE 00001180
C 00001190
C FIND AREAS AND INERTIA PROPERTIES -- USE DETAILED DATA IF 00001200
C AVAILABLE (CHECK(JNUM) = +1) -- ELSE USE CONSTANT CROSS-SECTION 00001210
C HOLLOW TUBE FOR BONE (CHECK(JNUM) = -1). 00001220
C 00001230
C IF (CHECK(JNUM) .GT. 0) GO TO 80 00001240
C 00001250
C CONSTANT CROSS-SECTION 00001260
C 00001270
C 00001280
READ (6,*1) A1, TEMP4, TEMPI 00001290
TEMP2 = TEMP4 * TEMP4 00001300
TEMP3 = TEMPI * TEMPI 00001310
A1 = PI * (TEMP3 - TEMP2) 00001320
TEMP = PI * (TEMP3 * TEMP3 - TEMP2 * TEMP2) / 4. 00001330
WRITE (3,160) TEMP4, TEMPI, A1, TEMP 00001340
TEMP = TEMPI / TEMP 00001350
DO 70 I = 1, NSEGS 00001360
AREKIT = A1 00001370
70 Y0(I) = TEMP 00001380
GO TO 95 00001390
C
C VARIABLE CROSS-SECTION WITH SLIDING TO 95TH PERCENTILE 00001400
C 00001410
80 READ (6,*1) SUBJL, SUBJW 00001420
K = CHECK(JNUM) - 1 00001430
DO 85 I = 1, K 00001440
85 READ (6,*1) BONEP(I), BONEA(I), BONEI(I) 00001450
K = I 00001460
LIL2 = SUBJL / L95(JNUM) 00001470
W95 = SUBJW / W95 00001480
WRITE (3,170) SUBJW, SUBJL, W95, L95(JNUM) 00001490
WRITE (3,175) 00001500
00001510

```

```

DO 90 I = 1, NSEGS                                00001520
CALL INTERP (K, JNUM, DELX(I), AREA(I), YO(I))    00001530
WRITE (3,180) DELX(I), X(I), AREA(I), INER2, RO   00001540
C4 WRITE (3,200) DELX(I), X(I), AREA(I), INER2, RO 00001550
90 CONTINUE                                        00001560
C4 STOP                                            00001570
C-----
C BEGIN MAIN LOOP FOR ALL TIME STEPS FOR LINE I.  00001590
C-----
95 DO I45 KK = I, NT                               00001600
C-----
C FIND APPLIED LOADS ON LINE I FROM FILE #5.      00001620
C-----
READ (5) TIME                                      00001640
IF (J.NE. 1) CALL FOR (J-1, 5)                    00001650
READ (5) TIME, KT, NCON, OMEGA, F1, F2, ALPHA, ACCEL, UZ 00001660
$ (DUMMY(I), I = 1, KT), (CNSTRN(I), I = 1, NCON) 00001670
IF (J.NE. 8) CALL FOR (8-J, 5)                    00001680
WRITE (3,200) TIME                                  00001690
C1 WRITE (1,2) F1, F2, ALPHA, ACCEL, OMEGA, UZ     00001700
C1 2 FORMAT (1H, 'APPLIED LOADS', 6F12.5, /, (15, 6F12.5)) 00001720
KT6 = KT / 6                                       00001730
IF (KT6.EQ. 0) GO TO 96                            00001740
WRITE (3,500) KT6, DUMMY(I), I = 1, KT, A1        00001750
C1 PRINT, (DUMMY(I), I = 1, KT)                    00001760
C1 96 IF (YES.EQ. 0) GO TO 100                      00001770
C1 PRINT, "OTHER CONSTRAINT FORCES"               00001780
C1 PRINT, POSIT(I), I = 1, 3, CNSTRN(I), I = 1, 3 00001790
C-----
C BEGIN LOOP FOR EACH FREE-BODY SECTION I.         00001800
C CALCULATE STRESSES ON EACH FREE-BODY SECTION USING MOHR'S CIRCLE 00001810
C AND STATIC EQUILIBRIUM                          00001820
C-----
100 WRITE (3,250)                                   00001840
DO I45 I = 1, NSEGS                                00001850
RO1 = YO(I) / X(I)                                  00001860
XFORCE = 0.0                                        00001870
DO I05 K = 1, 3                                     00001880
105 MOMENT(K) = 0.0                                  00001890
TE (YES, I, 0) GO TO 107                            00001900
C-----
C FIND MOMENTS AND SUM FORCES DUE TO "OTHER CONSTRAINT FORCES" 00001920
C-----
XPOINT = X(I) - POSIT(YES)                          00001950
IF (XPOINT) 107, 107, 106                          00001960
106 MOMENT(1) = POSIT(YES+2) * CNSTRN(3) - POSIT(YES+3) * CNSTRN(2) 00001970
MOMENT(2) = XPOINT * CNSTRN(3) + POSIT(YES+3) * CNSTRN(1) 00001980
MOMENT(3) = -XPOINT * CNSTRN(2) - POSIT(YES+2) * CNSTRN(1) 00001990
XFORCE = CNSTRN(1)                                  00002000
107 IF (KT.EQ. 1) GO TO 120                          00002010

```

```

                                00002020
FIND MOMENTS & FORCES DUE TO CONTACTS                                00002030
                                00002040
                                00002050
                                00002060
                                00002070
                                00002080
                                00002090
                                00002100
                                00002110
                                00002120
                                00002130
                                00002140
                                00002150
                                00002160
                                00002170
                                00002180
                                00002190
                                00002200
                                00002210
                                00002220
                                00002230
                                00002240
                                00002250
                                00002260
                                00002270
                                00002280
                                00002290
                                00002300
                                00002310
                                00002320
                                00002330
                                00002340
                                00002350
                                00002360
                                00002370
                                00002380
                                00002390
                                00002400
                                00002410
                                00002420
                                00002430
                                00002440
                                00002450
                                00002460
                                00002470
                                00002480
                                00002490
                                00002500
                                00002510

```


C	SOLVE MOHR'S CIRCLE FOR EACH END OF DIAMETER OF TUBE	00002520
C	DO 125 K2 = 1, 2	00002530
	K = K2 * RZ	00002540
	TEMP = DET (TAU, SIG(K2))	00002550
	TEMP2 = SIG(K2) / 2.	00002560
	SA = TEMP2 + TEMP	00002570
	OUT(K) = SA	00002580
	SB = TEMP2 - TEMP	00002590
	OUT(K+1) = SB	00002600
	TI = TEMP	00002610
	OUT(K+2) = TI	00002620
	125 CONTINUE	00002630
C	SI = AMAXI (OUT(1), OUT(4))	00002640
	S2 = AMINC (OUT(2), OUT(5))	00002650
	T = AMAXI (OUT(3), OUT(6))	00002660
	IF (S1 .LT. SIMAX) GO TO 130	00002670
	SIMAX = S1	00002680
	XSI = X(T)	00002690
	130 IF (S2 .GT. S2MAX) GO TO 135	00002700
	S2MAX = S2	00002710
	XS2 = X(T)	00002720
	135 IF (T .LT. TMAX) GO TO 140	00002730
	TMAX = T	00002740
	XT = X(T)	00002750
	140 WRITE (3,300) DELX(T), X(T), OUT	00002760
C9	WRITE (7,400) TIME=DELX(T), X(T), OUT	00002770
	145 CONTINUE	00002780
	STOP	00002790
C	150 FORMAT (1P, 'STRESS HISTORY FOR SEGMENT #', I2, ' ', '83.7',	00002800
	\$ ' THERE ARE', I3, ' TIME STEPS',	00002810
	\$ ' I3, ' FREE-BODY SECTIONS WILL BE USED',	00002820
	155 FORMAT (1H-, 'SEGMENT MASS =', F10.3, ' LBS. /' ' SEMI-MAJOR AXES',	00002830
	\$ ' (X,Y) ARE: ', 3F10.3)	00002840
	157 FORMAT (1P, 'PROXIMAL JOINT TO C.G. DISTANCE =', F10.2, ' IN. /'	00002850
	\$ ' PROXIMAL JOINT TO DISTAL JOINT LENGTH =', F10.2, ' IN. /'	00002860
	156 FORMAT (1P, 'OTHER CONSTRAINT FORCE' LOCATIONS', 3F10.3)	00002870
	\$ ' PROXIMAL JOINT TO DISTAL JOINT LENGTH =', F10.2, ' IN. /'	00002880
	160 FORMAT (1P, 'BONE GEOMETRY -- CONSTANT CROSS-SECTION HOLLOW TUBE',	00002890
	\$ ' INSIDE RADIUS =', F6.3, ' IN. /'	00002900
	\$ ' OUTSIDE RADIUS =', F6.3, ' IN. /'	00002910
	\$ ' AREA =', F6.3, ' SQ. IN. /'	00002920
	\$ ' 2ND MOMENT OF INERTIA =', F6.3, ' IN. ** 4 /'	00002930
	170 FORMAT (1P, 'GEOMETRY DATA FOR THIS LIMB -- VARIABLE CROSS-SECTION',	00002940
	\$ ' HOLLOW TUBE', ' BASED ON EXPERIMENTAL DATA: /'	00002950
	\$ ' SUBJECT: ' ' HEIGHT =', F7.0, ' LBS. ' LIMB LENGTH =',	00002960
	\$ ' F7.2, ' IN. /' ' 95TH PERC. MAN : ' ' WEIGHT =', F7.0, ' LBS. /'	00002970
	\$ ' LIMB LENGTH =', F7.2, ' IN. /'	00002980
		00002990
		00003000
		00003010

```

175 FORMAT (IHO, I5, 'X', L', I13, 'X(IN)', T26, 'A(IN**2)', T39, 00003020
$ 'I(IN**4)', T51, 'OUTSIDE R. FINI')/ 00003030
180 FORMAT (IH, F7.3, 4(F9.3, 5X)) 00003040
200 FORMAT (I, TIME = 'F12.2', SECONDS) 00003050
250 FORMAT (IHO, I5, 'L', I13, 'X(IN)', T29, 'S1', T39, 'S2', T48, 'TAJ T', 00003060
$ T64, 'S18', T73, 'S28', T82, 'TAU B')/ 00003070
300 FORMAT (IH, F7.3, F9.3, 5X, 3F10.2, 5X, 3F10.2) 00003080
400 FORMAT (IH, 9F15.5) 00003090
425 FORMAT (IH, I4, 'I', I12, 'X (FROM C.G.)', T29, 'MASS (LB)', T41, 00003100
$ 'CG (FROM C.G.)', T62, 'I(XX)', T72, 'I(YY) = I(ZZ)')/ 00003110
450 FORMAT (IH, I3, 2X, 5F15.4) 00003120
500 FORMAT (IH, I1, 'CONTACT TCSE/THIS TIME STEP', 00003130
$ 'OCCUR AT X (FROM PROXIMAL) =', 4F6.2, '/') 00003140
END 00003150
C. 00003160
SUBROUTINE XPROD 00003170
REAL A(3), B(6,4) 00003180
COMMON /Q57/ B 00003190
C 00003200
A(1) = A(1) + X * B(6,1) + B(4,1) * B(5,1) 00003210
C 00003220
A(2) = A(2) + X * B(6,2) + B(4,2) * B(5,2) 00003230
C 00003240
A(3) = A(3) + X * B(6,3) + B(4,3) * B(5,3) 00003250
C 00003260
RETURN 00003270
END 00003280
C. 00003290
SUBROUTINE INTERP (K, J, X, A2, Y0) 00003300
REAL L1(2), I1, I2 00003310
COMMON /Q67/ BP(8), BA(8), BI(8), L1(2), WIW2, A1, I1, RORIG, I2, 00003320
$ R01 00003330
DATA PI /3.1415926/ 00003340
C 00003350
C FIND BOUNDING INTERVAL 00003360
C 00003370
DO 10 I = K, 6 00003380
IF (BP(I) .GT. X) GO TO 20 00003390
10 CONTINUE 00003400
20 IML = I 00003410
C 00003420
C INTERPOLATE FOR VALUES OF AREA AND INERTIA 00003430
C 00003440
PERC = (X - BP(IML)) / (BP(I) - BP(IML)) 00003450
A1 = BA(IML) + PERC * (BA(I) - BA(IML)) 00003460
I1 = BI(IML) + PERC * (BI(I) - BI(IML)) 00003470
C3 PRINT, 'A1, I1', A1, I1 00003480
K = IML 00003490
C 00003500
C FIND ORIGINAL OUTSIDE RADIUS FOR HOLLOW TUBE 00003510

```

	00003520
SORT (2. * (AI / 4. * PI * FI / AI) / PI)	00003530
ORIGINAL 'OUTSIDE RADIUS', RORIG	00003540
	00003550
TO 95-TH PERCENTILE SIZE	00003560
	00003570
IW2 * LIL2 * RORIG / I1 * PI / 4.)	00003580
/ (X1 * Y1)	00003590
/ (X1 * Y1)	00003600
X1, Y1, Z1, X1, Y1, Z1	00003610
(Z1 + SQRT(Z1 * Z1 + 8. * Y1)) / 4.	00003620
NEW 'OUTSIDE RADIUS' GROU	00003630
UT * RORIG	00003640
QRT (S0 - Y1)	00003650
N * RIN	00003660
* (S0 - X1)	00003670
* (S0 * S0 - X1 * X1) / 4.	00003680
OUT / I2	00003690
A2, I2, YOI, A2, I2, YOI	00003700
	00003710
	00003720
	00003730
LINE FOR (K, IFILE)	00003740
LE, EQ, AT GO TO 20	00003750
= 1, K	00003760
FILE)	00003770
	00003780
* 1, K	00003790
FILE, *)	00003800
	00003810

