NOAA Technical Report NOS 97 NGS 26

An Inertial Survey Adjustment Program: Implementation and Validation

Rockville, Md.
November 1982

U.S.DEPARTMENT OF COMMERCE
National Oceanic and Atmospheric Administration
National Ocean Survey

NOAA Technical Publications

National Ocean Survey/National Geodetic Survey
Subseries

The National Geodetic Survey (NGS) of the National Ocean Survey (NOS), NOAA, establishes and maintains the basic national horizontal and vertical networks of geodetic control and provides Government-wide leadership in the improvement of geodetic surveying methods and instrumentation, coordinates operations to assure network development, and provides specifications and criteria for survey operations by Federal, State, and other agencies.

NGS engages in research and development for the improvement of knowledge of the figure of the Earth and its gravity field, and has the responsibility to procure geodetic data from all sources, process these data, and make them generally available to users through a central data base.

NOAA geodetic publications and relevant geodetic publications of the former U.S. Coast and Geodetic Survey are sold in paper form by the National Geodetic Information Center. To obtain a price list or to place an order, contact:

National Geodetic Information Center
C18x2
National Ocean Survey, NOAA
Rockville, MD 20852

(301) 443-8316

When placing an order, make check or money order payable to: National Geodetic Survey. Do not send cash or stamps.

Publications can also be purchased over the counter at the National Geodetic Information Center, 11400 Rockville Pike, Room 14, Rockville, Md. (Do not send correspondence to this address.)

An excellent reference source for all Government publications is the National Depository Library Program, a network of about 1,300 designated libraries. Requests for borrowing Depository Library material may be made through your local library. A free listing of libraries in this system is available from the Library Division, U.S. Government Printing Office, 5236 Eisenhower Ave., Alexandria, VA 22304 (703-557-9013).

NOAA Technical Report NOS 97 NGS 26

An Inertial Survey Adjustment Program: Implementation and Validation

Dennis G. Milbert

National Geodetic Survey
Rockville, Md.
November 1982

For sale by the National Geodetic Information Center, Rockville, MD 20852 Price: $4.70

U.S. DEPARTMENT OF COMMERCE
Malcolm Baldrige, Secretary

National Oceanic and Atmospheric Administration
John V. Byrne, Administrator

National Ocean Survey
R. Adm. H.R. Lippold, Jr., Director

CONTENTS

Abstract..... 1

Introduction..... 1

Design aims..... 1

Adjustment model..... 4

Inertial survey measurement model..... 6

Implementation..... 7

The southwest Arizona inertial survey..... 11

Test data..... 14

Adjustments..... 16

Conclusions..... 29

References..... 29

Appendix A. INERT1 data format..... 31

Appendix B. Sample INERT1 listing..... 34

Appendix C. INERT1 source code listing..... 37

Appendix D. Name table documentation..... 65

Appendix E. Inertial survey system project summary..... 82

Appendix F. Adjusted parameter values..... 84

Mention of a commercial company or product does not constitute an endorsement by the National Oceanic and Atmospheric Administration. Use for publicity or advertising purposes of information from this publication concerning proprietary products or the tests of such products is not authorized.

AN INERTIAL SURVEY ADJUSTMENT PROGRAM: IMPLEMENTATION AND VALIDATION

Dennis G. Milbert
National Geodetic Survey
National Ocean Survey, NOAA
Rockville, Maryland 20852

ABSTRACT. The field coordinates from inertial survey systems exhibit systematic error which must be reduced by postmission processing. Desirable qualities of an inertial survey least-squares adjustment are discussed, and an adjustment program using the Gregerson 12 parameter model is described. Several adjustments of inertial observations in southwest Arizona test the Gregerson model and indicate the precision of the data. Possible improvements in the observational model and adjustment program are then examined.

INTRODUCTION

It is generally accepted that the Inertial Survey Systems (ISS) currently available produce raw data which contain uncompensated systematic errors (Hannah and Mueller 1981). Such errors left unchecked will, at best, degrade the precision of the inertial survey and, at worst, accumulate to introduce serious bias into the survey results. Postmission processing of inertial observations can extract this systematic error and produce an unbiased data set suitable for analysis and computation. Least squares adjustment provides a statistically rigorous procedure for dealing with systematic effects and for analyzing results.

This report details INERT1, a rigorous, least squares adjustment program for multiple inertial traverse runs. Both design criterion and implementation details are covered. The central design philosophy is to produce a program which simultaneously adjusts multiple inertial traverse runs and displays information for observation model validation and data analysis. But any design, however carefully constructed, must ultimately be validated against reality.

A set of inertial observations made in southwest Arizona is adjusted by INERT1. The adjustment results illustrate the utility of the Gregerson model and the effectiveness of a rigorous least squares adjustment in data analysis.

DESIGN AIMS

Before beginning the implementation of a computer program, a prudent individual makes a clear plan of what the program must produce. Such an action provides a guide for the overall effort and eases the programing process.

Observation Model

The observation model selected for INERT1 is the Gregerson 12 parameter model described in Hannah and Pavlis (1980: 20-23). This model was chosen due to the accessibility, the availability of a sample computation, and the modeling of an inertial observation as latitude difference, longitude difference, and elevation difference. Selection of a differential model is most important. Modeling of inertial survey systematic effects is still in an embryonic stage of development. As suggested by the name, INERT1 will be followed by INERT2, INERT3, etc. Each subsequent version of the program will benefit by research on models. The objective of INERT1 is not to provide the definitive inertial observation model, but to establish a tool for model validation.

Observation Variance and Covariance

As mentioned, inertial observations are assumed to be a triad of latitude difference, longitude difference, and elevation difference. It is natural, therefore, to estimate the reliability of these quantities using standard deviations of these differences. Assumptions are made that the observation triad is uncorrelated and the repetitions of any observation triad are uncorrelated. This assumption is consistent with the work of Schwarz (1980: 272-273). The standard deviations are assumed to be constant for any given observation. The program must provide for easy manipulation of standard deviations.

Progress on inertial observation precision models will only come after some measure of success has been made on inertial observation models themselves, and after a large body of inertial data has been accumulated. If inertial observation models are embryonic, then inertial observation precision models are barely past conception.

Rigorous Adjustment Procedures

The program must allow for the correct, simultaneous adjustment of multiple traverse runs. This is mandatory. Each point must possess only one triad of coordinates and all observations to that point in the adjustment must reference that unique triad. Inertial traverse surveys carry low redundancy. This in turn hampers the development of observation models, and impairs the localization of blunders (Mueller 1981). Area surveys, on the other hand, do not suffer as greatly from these deficiencies when all the observations participate simultaneously in a combined least squares adjustment.

Parameter Constraints

Flexibility can be added to the Gregerson model by providing for the constraint of any of the adjustment parameters (observation model parameters or station coordinates) to any desired value. This is easily accomplished by introducing a direct observation of the parameter in question with an arbitrarily high weight. By constraining a model parameter to zero, that term is effectively deleted from the observation model. These constraints also allow the geodesist to hold coordinates known from earlier surveys.

INERT1 does not have provision for inclusion of coordinate covariance information. When the number of stations is large enough, covariance

information is more easily carried by the actual observation from the earlier surveys, rather than by numbers from a prior adjustment. The design aim is that INERT1 shall evolve into a program that simultaneously adjusts different types of geodetic observations. Such an approach has been used in photogeodesy by El Hakim and Faig (1981).

Postadjustment Parameter Standard Deviations

The program should display the standard deviation of the adjustment parameters. This gives perspective to the precision of the survey coordinates. Of greater utility, perhaps, will be the standard deviations for the observation model parameters. Deviations that are comparable in size to the adjusted value of a given model parameter indicate a possible candidate for a constraint of that parameter to zero. Such a constraint should then be verified in a subsequent adjustment using an F test.

Length Relative Accuracies

The horizontal control standard is based upon the length relative accuracy between directly connected adjacent points (Federal Geodetic Control Committee 1974: 3). By linear error propagation, one can compute this measure for any desired pair of points. The program should provide this capability and enable the geodesist to evaluate the order of a survey.

An important product accrues from providing this feature. If the relative accuracies are computed using the a priori variance of unit weight, then the relative accuracies may be computed without knowledge of any observed values. One may predict the precision of a survey before the first measurement is made. One requires the proposed design and estimates of observation precision, nothing more. The program may then function as a planning tool as well as a postmission processing tool.

Googe Numbers

A Googe number

$$g_i = \frac{c_{ii}^2}{n_{ii}} \tag{1}$$

is the square of the diagonal element of the Cholesky factor, c_{ii}, divided by that diagonal element before Cholesky factorization, n_{ii}. It is a "normalized" measure which may be used in singularity detection (Schwarz 1978: 29-32). In addition the Googe number provides a measure of the ill-conditioning of a parameter. Such a parameter is not well determined by the data, so that one may not rely on the estimate of that parameter. These numbers are useful in evaluations of network strength.

Residual Statistics

Effective organization of residuals greatly assists in detection of outliers, identification of weighting problems, and evaluation of observation models. Residuals are displayed in a manner similar to that of the horizontal least squares adjustment program, TRAV10 (Schwarz 1978). This decision is based upon personal experience with TRAV10 data analysis. The residual display is by no means optimal. As experience is gained with rigorous inertial adjustments, better residual displays will be developed.

Modular Structure

The INERT1 program was developed using the principles of stepwise refinement (Dijkstra 1965). By choosing the refinement stages properly, it is possible to "isolate" the model from the remainder of the adjustment source code. The storage structure of the parameters can also be isolated in a similar manner. This allows for great freedom in model selection without imposing a need to rewrite the entire program. The model need only be altered in a few key places to effect the desired change.

ADJUSTMENT MODEL

The least squares adjustment model and notation used in this report are from Schwarz (1974-1975) and Uotila (1967).

An adjustment model for the method of observation equations is

$$L_a = F(X_a) \tag{2}$$

where L_a is a vector of computed observation values, X_a is a vector of coordinate and model parameters, and F is a vector of functions that describes the observations in terms of the parameters. L_a and F are vectors of length n, and X_a is a vector of length u.

The design matrix, A, is defined as

$$A = \left.\frac{\partial F}{\partial X_a}\right|_{X_a = X_0} \tag{3}$$

where A is a matrix of differential changes in the observation model with respect to the parameters, X_a, evaluated at a particular set of parameter values, X_0. A vector of observation misclosures is

$$L = L_b - L_a \tag{4}$$

where L_b is the vector of actual observations and L_a is the vector described above.

Associated with the observation vector L_b is a symmetric variance-covariance matrix Σ_{L_b}, which contains information on observation precision and correlation. The weight matrix is defined as

$$P = \sigma_0^2 \Sigma_{L_b}^{-1} \tag{5}$$

where σ_0^2 is the a priori variance of unit weight. This value is typically set to 1, as is done in this report.

The observation equation may now be written as

$$AX = L + V \tag{6}$$

where V is a vector of residual errors and X is a vector of corrections to the parameter vector X_a. The least squares estimate of X is

$$X = (A^T PA)^{-1} A^T PL \ . \tag{7}$$

Computation of this estimate is known as solution of the normal equations. The estimate provides a new set of values for our parameters by

$$X_a \leftarrow X_a + X . \tag{8}$$

If the observation model $F(X_a)$ is nonlinear (that is, A is not constant for any set of X_a), then the entire process starting with eq. (2) must be iterated until the vector X_a reaches a stationary point.

Once convergence is achieved, L_a computed from eq. (2) is the vector of adjusted observations. The vector of observation residual errors, V, is

$$V = L_a - L_b . \tag{9}$$

The a posteriori variance of unit weight, σ_0^2, is a scale factor for the observation variance-covariance matrix. It can be used to bring the observation weights into agreement with predictions of statistical theory. An estimate of the a posteriori variance of unit weight is given by

$$\hat{\sigma}_0^2 = \frac{V^T P V}{(n-u)} \tag{10}$$

where n is the number of observations (length of the vector L_b) and u is the number of parameters (length of the vector X_a).

Estimates of parameter precision and correlations are given by the adjusted parameter variance-covariance matrix, Σ_{x_a}. This matrix is computed by

$$\sum_{x_a} = \left(A^T \sum_{L_b}^{-1} A\right)^{-1} . \tag{11}$$

One may also compute the precision of any other quantity which can be derived from the parameters. Suppose one wishes to compute a vector of quantities, S,

$$S = S(X_a) \tag{12}$$

from the adjusted parameters, X_a. A matrix, G, is defined as

$$G = \left.\frac{\partial S}{\partial X_a}\right|_{X_a = X_0} \tag{13}$$

where G is a matrix of differential changes in the functions, S, with respect to the parameters, X_a, evaluated at a particular set of parameter values, X_0. By the principle of linear error propagation,

$$\sum_s = G \sum_{x_a} G^T \tag{14}$$

where Σ_s is the variance-covariance matrix of the computed quantities.

Equation (14) is important since it can compute values such as the uncertainty in length without any direct length measurements. This last uncertainty is used in the computation of the length relative accuracy, the standard of horizontal geodetic control network of the United States.

Examination of eqs. (11) and (14) shows no dependence of the variance-covariance matrices upon L_b, the vector of observations. Actual observations are not needed to estimate the precision of the survey. All that are required are a proposed set of observations, the model of the

observations, and the model of the observation precision and correlation. Any adjustment routine that computes quantities from eqs. (11) and (14) can easily be designed to serve as a planning tool as well as a data analysis tool.

INERTIAL SURVEY MEASUREMENT MODEL

As discussed earlier, the Gregerson 12 parameter model approximates inertial observations. Although this model is described in Hannah and Pavlis (1980: 20-23), the description is repeated here with supplemental comments.

The Litton Autosurveyor® inertial system produces a set of three coordinates: latitude, longitude, and elevation. Despite the output form, observations are assumed to be a triad of coordinate differences. The observation models can be written as

$$\Delta\phi_{ij} = (1 + C_1)\ (\phi_j - \phi_i) + (-C_2 - C_3 t)\ (\lambda_j - \lambda_i) + C_4\ \Sigma T^2 \tag{15}$$

$$\Delta\lambda_{ij} = (1 + C_5)\ (\lambda_j - \lambda_i) + (C_6 + C_7 t)\ (\phi_j - \phi_i) + C_8\ \Sigma T^2 \tag{16}$$

$$\Delta h_{ij} = (h_j - h_i) + (C_9 + C_{11} t)\ (\phi_j - \phi_i) + (C_{10} + C_{12} t)(\lambda_j - \lambda_i) \tag{17}$$

Here, ϕ_i is the geodetic latitude at station i, λ_i is the geodetic longitude at station i, and h_i is the orthometric height above mean sea level at station i. Thus, $\Delta\phi_{ij}$ is the difference in geodetic latitude between station i and station j, $\Delta\lambda_{ij}$ is the difference in geodetic longitude between station i and station j, and Δh_{ij} is the difference in orthometric height between station i and station j. The symbol, t, represents the mean elapsed time from the start of the traverse run to the given observation in seconds. Now, ΣT^2 is the sum of the square of the intervals in seconds between marks and zero velocity updates (ZUPT's). For example, if the interval between two marks is 450 seconds and the ZUPT interval is 100 seconds, then

$$\Sigma T^2 = 100^2 + 100^2 + 100^2 + 100^2 + 50^2 = 42{,}500 \text{ seconds}^2. \tag{18}$$

Inertial systems also measure time, and the Gregerson model accepts such times as being error free.

Finally, C_1 through C_{12} are model parameters. If five traverse runs are made, the 5 x 12 = 60 model parameters, plus the coordinate triads for each station, must be carried as unknowns in the adjustment.

The first comment deals with the structure of the Gregerson Model. The form of the latitude difference and the longitude difference, eqs. (15) and (16), is identical. The measurement is modified by a scale factor (C_1 and C_5), and an off-track effect is introduced by the second term in both equations. The C_4 and C_8 coefficients control an observation bias which behaves as the sum of square time intervals. These two equations are, therefore, an extension of the two-dimensional general affine transformation,

$$X' = AX + BY + C \tag{19}$$

$$Y' = DX + EY + F\ . \tag{20}$$

By differencing the transformation,

$$\Delta X' = A \quad \Delta X + B \quad \Delta Y \tag{21}$$

$$\Delta Y' = D \quad \Delta X + E \quad \Delta Y. \tag{22}$$

Such a transformation provides for a separate scale factor on each axis. The coordinate system may be rotated and translated. Orthogonality between the coordinate axes is not preserved (Merchant 1977). Clearly, the Gregerson model also possesses such qualities.

On the other hand, the elevation difference, eq. (17), does not possess a scale factor. Systematic error terms are functions only of latitude and longitude difference. It is seen that vertical observations get a special model.

It must be stressed that these equations do not attempt to model the physical system of gyroscopes and accelerometers in an ISS. Rather, these equations model the systematic error of the measurements from a Litton Autosurveyor unit. Such measurements have already been corrected for some systematic effects and processed through a Kalman filter inside the Autosurveyor before they are available to a user. Any deficiencies in the Autosurveyor's internal model will appear as systematic effects which should be removed by postprocessing. Errors in point identification and update coordinates will also produce systematic effects. The Gregerson model attempts to correct these discrepancies.

IMPLEMENTATION

Following the description of the equations for the mathematics of the least-squares adjustment, and the equations to model inertial observations, it is still necessary to embody these equations into a computer program. The manner in which this is done will spell the difference between an easy-to-use program and an unworkable monster.

Modular Structure

The design criterion of modular structure impacts the program implementation to a high degree. Such modules perform a specific task and make a minimal number of "assumptions" about processes in other parts of the program. Communication between modules is performed chiefly through parameter lists. Thus, careful control of parameter values ensures correct functioning between modules. While a given module may invoke a number of other modules, the sharing of a module by other modules is to be avoided. This prevents a substantial change in a given module from requiring modifications to other parts of the program (Turner 1980).

Perhaps the most effective technique for development of reliable computer programs is stepwise refinement (Dijkstra 1965). In this technique the programer breaks a given problem into component subproblems. The dissection continues until each subproblem can be easily solved without resorting to further dissection. Stepwise refinement thus provides to programers a mechanism for working with arbitrarily large programs.

An example of stepwise refinement is seen in SUBROUTINE INERT1 in appendix C. This subroutine controls the entire inertial adjustment. It reads data,

forms and solves normal equations, updates parameters, monitors progress of the adjustment, and displays the adjustment results. The routine comprises only 40 lines of code, including comments. The tasks of reading data, solving normals, and result display are subproblems of the total inertial adjustment problem.

During problem dissection, knowledge of parameter storage is required throughout the program at all problem levels. This requirement runs against the need for modules to be isolated from one another and not be called throughout the program. A special type of routine called a service subprogram solves this difficulty. A service subprogram is a completely predictable, reliable routine that performs a single service (Turner 1980: 275). Service subprograms provide a natural place to perform functions that are invoked in many places and whose implementation may vary. As discussed in the design criterion, the programer may wish to modify parameter storage or the inertial observation model. Service subprograms isolate the modifications from the remainder of the program.

Examples of service subprograms are found in INTEGER FUNCTION IUNSTA and INTEGER FUNCTION IUNPRM. These functions "worry" about how the parameters are stored, and pass this information to the calling routine in a general way. In a similar fashion SUBROUTINE COMPOB computes an inertial observation based on the Gregerson model. Only COMPOB and FORMC contain the observation model. By taking such care in program inplementation, easily modifiable code results. Because of this, as more advanced inertial observation models are developed, they are easily installed in the program.

Observation Types

A description of the inertial data set formats may be found in appendix A. The data are grouped into five categories with a parameter record at the beginning of the set.

The first data group contains initial positions. These are starting values used in the iterative least squares adjustment. Since the design aims require comparisons of old coordinates observed by classical techniques and new coordinates observed inertially, positions shifts are also computed from the initial to the adjusted values.

The next group contains position constraints. These are applied as coordinate observations whose weights may be indicated in the record. If any coordinate is left blank, then that particular coordinate will not be constrained. This technique allows any combination of coordinates to be constrained.

Model parameter constraints are processed in a manner similar to position constraints. However, only one constraint may be applied with a given record.

Three different types of records comprise the inertial observation category. The run header record indicates the beginning of a new traverse run. As required by the design aims, any number of traverse runs may be simultaneously adjusted. A mark record contains the observation at a given point. An update record holds the values which updated the ISS with a new coordinate.

A number of comments can be made now. Inertial observations are coordinate differences. An observation is produced by differencing a mark record from a prior mark record, update record, or header record. The number of inertial observation triads equals the number of mark records. The mark record at a given point must precede the update record, if such an update record is needed at the point. A point which was updated does not need to be held fixed in the adjustment. An update record can be seen to merely update coordinates used to compute coordinate difference observations. An update record does not represent an observation itself. The run header record behaves much the same way. However, it also allocates a new set of 12 model parameters. This gives the geodesist flexibility to decide if a single set of parameters adequately models a double run traverse or if two parameter sets are needed.

Finally, computation records indicate which stations have length relative accuracy computations. These computations indicate the order of the survey, so they hold great importance. The mathematics of these computations are described later in this report.

These data formats provide an initial representation of the inertial observations commensurate with the Gregerson model. As progress is made in inertial models and important data elements are identified, the data formats will have to change accordingly.

Name Table

INERT1 invokes a collection of routines designed to store, retrieve, and manipulate tables of information. In this application a table is used to store 30 character names for stations. A name table is a tool which associates a unique number with a unique name. One may then use the number as an index of an array. This gives the geodesist flexibility in modifying or combining sets of data, since actual station names may be used.

Examination of the source listing in appendix C shows the functions NEWT, SEEK, SIZE, PUTVAL, SETCUR, and GETKEY are not defined. These are table routines, and they are described in appendix D. The table routines and documentation were written by John F. Isner of NGS.

Normal Equation Accumulation and Solution

INERT1 also invokes a subroutine package for the accumulation, solution, and inversion of normal equations. This package is described in Dillinger (1981). It provides for solution of large, sparse systems using variable bandwidth storage in memory and random access files for backing storage (Jennings 1977). Three different reorder algorithms are included to minimize memory requirements. INERT1 uses the "banker's" algorithm reported in Snay (1976). The inverse may also be computed within the profile. This provides availability of variance and covariance information necessary to error propagation studies without imposing a prohibitive computational burden on the program.

The routines NABGEN, ADCON, NEWORD, BIBB, ADOBS, FLUSHQ, ELEM, SETEL, REDUCE, SOLVE, and INVRSE are parts of the equation package, and not defined in the INERT1 source code.

Parameter Output

A key part of the implementation of a program is the selection and display of information. INERT1 displays data to help determine both the quality of the inertial observations and the appropriateness of the observational model. A sample adjustment run may be found in appendix B. The first major section of the listing deals with the adjustment parameters. These parameters may be divided into two groups: coordinate parameters and inertial observation model parameters.

The adjusted values of the coordinate parameters are shown in units of degrees, minutes, and seconds for geodetic latitude and longitude, and in units of meters for orthometric height above mean sea level. Associated with each position is its standard deviation computed from eq. (11). This section also displays the Googe number for each coordinate parameter (Schwarz 1978: 29-32). The numbers range from 1 to 0, where 1 indicates a strong solution, and smaller values indicate progressively weaker solutions. The adjusted values of the model parameters are shown in the natural units implied by the Gregerson model. Standard deviations and Googe numbers associated with the model parameters are displayed. In addition, a unitless number, parameter value divided by parameter standard deviation, is also computed. These numbers indicate how close a given model parameter may be to 0. If a parameter is close to 0, then it may be eliminated from the Gregerson model. One sees how these numbers assist in observation model research.

A section displays the shifts in coordinates from the initial position to the new, adjusted positions. These numbers have no utility whatsoever if the initial positions are from some approximation technique. However, if the initial positions are computed from an adjustment of classical observations, then the shifts can indicate systematic differences between classical and inertial observations. Similar shifts identified geodetic control deficiencies in Hannah and Mueller (1981). Shifts are shown for each coordinate axis, for the horizontal shift, and for the total spatial shift. Azimuths of the horizontal shift are measured clockwise from the north.

Residual Output

The first major section of the residual display lists each residual. Latitude and longitude residuals are shown in units of both seconds and meters. A unitless number, residual divided by the a priori observation standard deviation, is computed for each residual. These numbers may be considered quasi-normalized residuals.

A fully normalized residual is a residual divided by the standard deviation of that particular residual. The standard error may be computed by error propagation through eq. (14). Fully normalized residuals are not computed in INERT1 due to the heavy computational burden of producing standard deviations of residuals. A number of points made in Pope (1976) and Snay (1978) must be repeated here. Residuals are not mutually independent; they are correlated. Pooling residuals from a survey will not necessarily produce a Gaussian distribution of residuals. Quasi-normalized residuals are smaller in absolute value than fully normalized residuals. For these reasons, quasi-normalized residuals must be interpreted with care.

Excessive magnitudes of quasi-normalized residuals indicate possible blunders or observation model problems. To assist in locating large residuals, INERT1 accumulates and displays the observation numbers of the 20 greatest quasi-normalized residuals. Various residual statistics are also computed and displayed in this section. This information allows evaluation of the a priori observation standard deviations.

Length Relative Accuracy Output

The final section of INERT1 computes and displays the length relative accuracy between selected points. These numbers are extremely important since they define the order of a horizontal control survey. A length accuracy may be computed in two different ways. If coordinates from prior surveys are available for two selected points, a length shift may be computed by differencing the distance between the old coordinates and the distance between the new coordinates. The value of the distance divided by the distance shift provides one "observation" of the length relative accuracy. These length accuracies are generally not available unless one specifically resurveys a network. This computation of length relative accuracy requires some estimate be made of distance shift attributable to observation error in the old survey positions.

A length relative accuracy may also be computed by linear error propagation using eq. (14). This method does not require coordinates from old surveys. However, it is extremely sensitive to the a priori estimates of observation standard deviation. Propagated relative accuracies depend upon good knowledge of observation model and precision.

INERT1 computes length relative accuracies using both methods. One sigma, 2 sigma, and 3 sigma values of the propagated length relative accuracy show the statistically estimated precision for increasingly larger confidence intervals. One sigma corresponds to a 68 percent confidence interval, 2 sigma corresponds to a 95 percent interval, and 3 sigma corresponds to a 99 percent interval. The "observed" length shift is always computed, but only has meaning if initial positions are available from a previous survey and one has knowledge of length shift attributable to the previous survey precision. Since propagated length relative accuracies represent statistical estimates, they do not require a high degree of precision. Despite this fact, lengths are computed on an ellipsoid, and the error propagation through eq. (14) uses ellipsoidal coefficients found in Rapp (1977) and Schwarz (1978).

THE SOUTHWEST ARIZONA INERTIAL SURVEY

The National Geodetic Survey (NGS) and Span International, Inc. of Scottsdale, Ariz., jointly conducted a test of a SPANMARK® Inertial System (Litton Autosurveyor) from March 18 to April 1, 1981. A discussion of test site selection, site preparation, and ISS field procedures may be found in Leigh (1981). That discussion is summarized here.

Test Site Criteria

Accuracy of existing geodetic control was the overriding concern for the test. Ideally, a control grid accurate to one part per million between adjacent stations for latitude, longitude, and elevation was desired. No geodetic control in the United States meets this description. Emphasis on horizontal control accuracy naturally led to consideration of the transcontinental traverse (TCT). The TCT is accurate in scale to within one part per million when compared with Doppler satellite observations (Gergen 1979). No other geodetic control, horizontal or vertical, approaches the accuracy of the TCT.

Secondary criteria were test site proximity to Scottsdale, Ariz., and available geodetic control in 80-kilometer straight lines along meridians and parallels. Again, the TCT at the California-Arizona border fulfilled these criteria admirably. Figure 1 presents a sketch of the test site.

Absence of a grid pattern in the TCT is not a critical factor. The high accuracy ensures that any horizontal systematic errors are due to failure of the ISS observation model, and not to conventional survey errors. If the ISS model is accurate, then grid patterns may be easily simulated by linear error propagation (eq. 14).

Minimally Constrained Adjustment

To provide the best estimate of the horizontal control points, the Horizontal Network Division of NGS computed a simultaneous, minimally constrained, least squares adjustment of all the horizontal and astronomic observations in the area of the test site. Station LANG 1960, near the junction of the TCT near Yuma, Ariz., was the only constrained point. The adjustment contained 586 stations and 4699 observations, and produced a variance of unit weight of 1.668 with 2283 degrees of freedom.

During portions of the test, the ISS unit was run beyond the boundary of the original test site. To provide reliable coordinates for the evaluation of these supplemental points, a second adjustment was performed over a more extensive area. This adjustment contained 829 stations and 6521 observations, producing a variance of unit weight of 1.594 with 3137 degrees of freedom. These new coordinates were available after conclusion of the test and have been used in the subsequent analysis. Table 1 lists adjusted coordinates for a number of stations in the ISS test area.

When comparing coordinates from the two adjustments, KOFA NORTH BASE 1947 shifted 0.13 meter west by southwest when the data in the larger area were added. Examination of other coordinate pairs on the north-south line found shifts proportional to distance from LANG 1960. The north-south line pivoted counterclockwise very slightly in the second adjustment. No such pivot was detected along the east-west line.

Standard Operating Procedure

A helicopter carried the ISS unit during the test. The unit was aligned each morning prior to the day's survey operations. Each day the team would

repeatedly survey the north-south or the east-west leg of the test site. Each traverse survey consisted of a forward run immediately followed by a reverse run. On some days three pairs of runs were successfully measured.

Before each traverse run, the unit measured presurvey calibration stations located west of LANG 1960 RM 4. The unit was updated at these presurvey points as well as at LANG 1960 RM 4, STOVAL RM 5 RM A 1971, and KOFA NORTH BASE 1947. As mentioned before, the coordinates used for updates at these points were computed in the first minimally constrained adjustment. All other adjusted coordinates were withheld from the ISS measurement team until after the test.

Procedure Variations

During the course of the test, some intentional variations in the field procedures were introduced. One variation was the deliberate introduction of

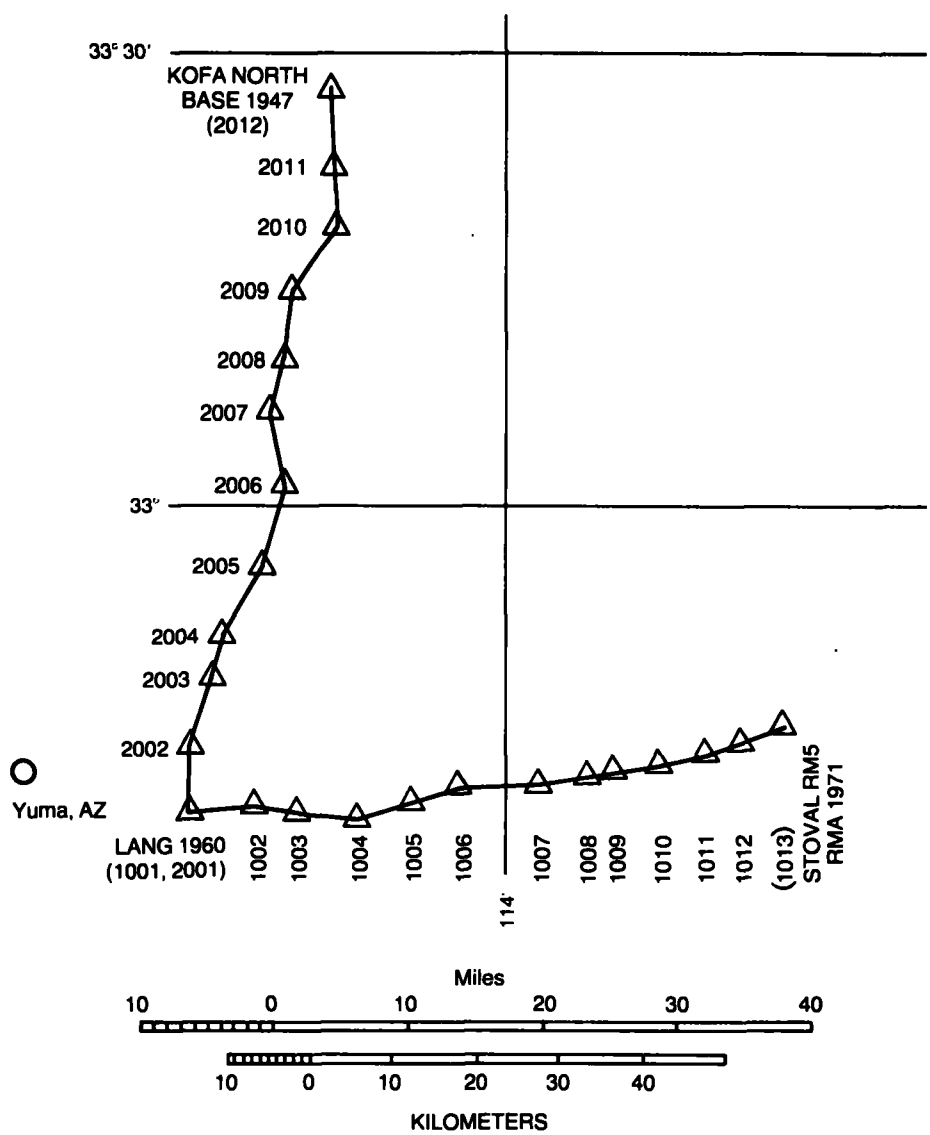


Figure 1.--Southwest Arizona test course.

Table 1.--Adjusted coordinates using conventional horizontal observations

Station name	Number	Latitude			Longitude		
LANG 1960 RM 4	1001	32°	40'	7".39636	114°	24'	29".94778
BEACON 2 1971 AZ MK 2	1002	32	40	42.73378	114	18	30.19185
OVERPASS 1934	1003	32	40	9.24811	114	15	52.09266
ADONDE 1934	1004	32	39	28.93223	114	11	29.28629
PASSO 1960 RM 2	1005	32	40	19.77996	114	6	58.91308
NAVI 1960	1006	32	40	48.51619	114	3	3.21521
GAEL 1934 RM 5	1007	32	41	46.57782	113	57	18.62695
COLFRED USGS 2 1971	1008	32	42	22.71959	113	53	33.41455
PEMB 1960	1009	32	42	40.46306	113	51	25.24326
OWL 1934 RM 4	1010	32	43	6.70717	113	48	10.69975
AWK 1960 RM 4	1011	32	43	41.28930	113	45	26.24630
KIM 1960 RM 3	1012	32	44	18.00110	113	41	46.09492
STOVAL RM 5 RM A 1971	1013	32	45	27.59854	113	38	19.83475
QUARRY	2002	32	44	38.91312	114	25	13.65760
BENCH MARK USBR 1934 RM 4	2003	32	48	39.30649	114	22	33.54397
COUNTRY WELL RM 2 RESET	2004	32	51	31.24118	114	21	32.34947
TT 6 USE 1956	2005	32	55	20.80390	114	18	49.69741
PELIGRO 1949	2006	33	1	19.93113	114	16	53.67928
HILL TOP 1949 RM 3	2007	33	6	15.20785	114	17	55.48380
INDIAN 1949 RM 2	2008	33	10	25.50005	114	16	34.20444
PGT NO 3 AMS 1971 RM 3	2009	33	14	22.15943	114	15	28.51893
CHOCO 1949 RM 2	2010	33	18	42.14580	114	12	56.57168
KOFA SOUTH BASE 1949 RM 2	2011	33	22	37.06334	114	12	59.96262
KOFA NORTH BASE 1947	2012	33	27	49.48687	114	12	59.86782

an erroneous coordinate at the far update point. The deliberate errors were made in the cross-track direction. At station 1013 (STOVAL RM 5 RM A 1971) the error was 1.85 meters south. At station 2012 (KOFA NORTH BASE 1947) the error was 6.06 meters east. These errors were made to discover the effect upon the onboard processing of the Autosurveyor, and to discover if these errors could be successfully removed by an inertial adjustment model.

Another variation was the addition of intervening stations along the traverses. This was done along both legs. In fact, on one north-south traverse run, a ZUPT was made midway between each pair of stations. These variations were made to test effects of a shorter ZUPT interval.

A variation mentioned earlier was the extension of the survey beyond the original limits of the test. One example of this variation occurred on the last day of the test. A forward run with no reverse run was made over an east-west course twice as long as that used for the normal test.

TEST DATA

The ISS unit writes the data onto cassettes of magnetic tape. Each cassette typically contains the survey information for a day or a portion of a day's operation. The Geodetic Survey of Canada (GSC) kindly agreed to reformat these data onto a reel of nine track, magnetic tape. In addition, the GSC also supplied computer listings of the smoothed coordinates and their adjusted values. For more information on the GSC processing of inertial data, see Kouba (1977). These materials arrived at NGS on November 30, 1981.

The data were organized into 25 files, one file for each cassette. A set of 79 data elements comprised each mark record. Each file held all the mark records for a particular cassette. These data were transformed into the format described in appendix B. Appendix E contains a summary of the data sets.

Data Screening

The Gregerson model (eqs. 15, 16, and 17) behaves badly for partial traverse runs containing only a forward or a reverse run (Hannah and Pavlis 1980). Only those data for complete traverse runs, forward and reverse, were retained. In some cases, gaps were discovered in the data, causing elimination of that particular traverse. I am unsure if the gaps were due to media failure, or due to problems in the translation to the nine track tape. Data on 13 different cassettes were retained.

Next, diagnostic adjustments on each complete traverse were computed. These adjustments used the Gregerson model described earlier. The results led to the rejection of a traverse run (cassettte 312) due to severe residuals. It should be noted that the subsequent traverse, cassette 324, failed due to velocity runaway of the ISS unit. The problem which led to velocity runaway on cassette 324 was detected by the Gregerson model adjustment in the prior traverse.

Finally, the data for the traverse runs observed with experimental procedures were set aside. This excluded the traverse runs over extended areas, runs with mid-traverse coordinate updates, and runs with halved ZUPT intervals.

The result was a homogeneous set of data, using known procedures, observed over a known test site. The set contained 18 complete traverse runs, forward and reverse. The data were recorded onto 11 cassettes spanning 9 different days. Ten of the traverses were observed along the east-west line, while the remaining 8 were oriented north-south. Table 2 contains a brief summary of the traverses.

Table 2.--Traverse summary

Number	Date	Cassette	Director	Comment
1	3/19	302	EW	
2	3/21	305	NS	
3	3/21	305	NS	
4	3/22	306	NS	
5	3/22	306	NS	
6	3/22	306	NS	
7	3/24	309	NS	False longitude update
8	3/26	314	NS	False presurvey update
9	3/27	315	EW	
10	3/27	316	EW	
11	3/27	317	EW	
12	3/28	318	EW	False latitude update
13	3/28	318	EW	
14	3/28	318	EW	
15	3/29	319	EW	
16	3/29	319	EW	
17	3/29	319	EW	
18	3/31	322	NS	

ADJUSTMENTS

The data from the selected traverse runs were combined with a simultaneous, minimally constrained, least squares adjustment using the Gregerson 12 parameter model. This model requires that the midpoint of a traverse be constrained. The latitudes, longitudes, and elevations of stations 1001, 1007, 1013, 2006, and 2012 were held fixed. These 15 constraints produced a minimally constrained adjustment. The a priori estimates of observation standard errors were 0.1 meter for differential latitude, differential longitude, and differential height. The a priori variance of unit weight was 1.0.

Parameter Scope

One question which has not been properly addressed in the literature concerns parameter scope. What should be the scope of each parameter in an error model? Should a set of 12 parameters for the Gregerson model cover a complete traverse run or only a forward or a reverse run? Is one set of parameters satisfactory for all traverse runs in a day? Do the 12 parameters vary in scope, with some parameters covering more traverses than others?

To begin to explore these questions, some preliminary adjustments were computed. Three data sets were extracted, each of which contained three complete traverse runs observed in a single day. Each of the three data sets was adjusted twice. The first adjustment used 12 parameters for each traverse run, totaling 36 parameters. The other adjustment used 12 parameters for the entire day. The results are summarized in table 3.

As seen in table 3, fewer parameters result in an increase in the a posteriori variance of unit weight. However, the increases are not large. These numbers support the premise that many of the parameters in the Gregerson model have the scope of a day or longer.

To test the other extreme, adjustments using 12 parameters for each forward run and 12 for each reverse run were computed. This parameterization made the adjustments very ill-conditioned, almost singular.

Midpoint Constraint

As mentioned earlier, the Gregerson model requires that both endpoints and a midpoint of a traverse be held fixed. Some experiments did indeed verify that the model is singular without a fixed midpoint. Other adjustments demonstrated it was possible to specify any other traverse point as the midpoint constraint without altering any residuals. Thus, for example, station 1006 could be constrained instead of 1007, and the residuals and variance of unit weight would remain unchanged. This result is expected in a minimally constrained adjustment.

Table 3.--Standard deviations of unit weight with different parameters

Cassette	$\hat{\sigma}_0$--36 parameters	$\hat{\sigma}_0$--12 parameters
306	2.34	2.80
318	1.90	2.37
319	1.34	1.53

As also expected by theory, the values of the adjusted coordinates do change when a different midpoint is selected. Experiments revealed that adjusted position shifts did not change dramatically when different midpoints were selected. Generally, smaller shifts were obtained on the traverse when the midpoint constraint was selected near the center of the traverse, rather than near the ends.

Combined Adjustment

Given these findings, the constraints of 1001, 1007, 1013, 2006, and 2012 were used. The constraints were applied as 15 coordinate observations, each of 0.001 meter precision for latitude and longitude, and of 1.0 meter for elevation. The different value of the elevation constraint reflects the uncertainty of the TCT elevations.

To study parameter scope more closely, each set of 12 parameters was given the scope of one complete traverse. Since the data set held 18 complete traverses, 18 x 12 = 216 parameters were used to describe the inertial data. A total of 38 points participated in the adjustment, 24 of which were the traverse points visited in each run (1001-1013,1001-2012). Since one and only one set of three coordinates was allocated to each point, the coordinates contributed another 114 unknowns to the adjustment. This led to a total of 1392 observations and constraints, 330 unknowns, and 1062 degrees of freedom in the combined adjustment. The a posteriori variance of unit weight was 3.28.

Precision

Precision is the measure of internal consistency and describes the repeatability of an observation. To examine the precision of the ISS unit, the square weighted residuals were summed $\Sigma V^T PV$ for ϕ, λ, and η in each of the 18 traverses. In addition, the square root of the square weighted residuals divided by the number of the observations $\Sigma V^T PV$ was computed for each category.

This value, the root mean square weighted residual, is a biased estimator of the a posteriori standard deviation of unit weight. It consistently underestimates the standard deviation of unit weight. However, it can be used to make comparisons between runs and between observation types. These numbers are presented in table 4.

Figure 2 plots the root mean square weighted residuals for the latitude difference observations of the traverses. Figure 3 plots similar numbers for longitude. The horizontal lines in both figures are merely the average values of these residuals. These numbers, in conjuction with the standard deviation of unit weight, indicate that the ISS unit measures latitude and elevation differences with a precision of roughly 0.175 meter, and longitude differences with a precision of around 0.2 meter.

The key point about figures 2, 3 and table 4 is that the longitude residuals tend to be larger than the latitude residuals. There are no apparent reasons for the ISS unit to operate more precisely in latitude than in longitude. A deficiency in the Gregerson longitude model difference, eq. (16), may be causing the larger residuals. The system should be equally precise in latitude and longitude.

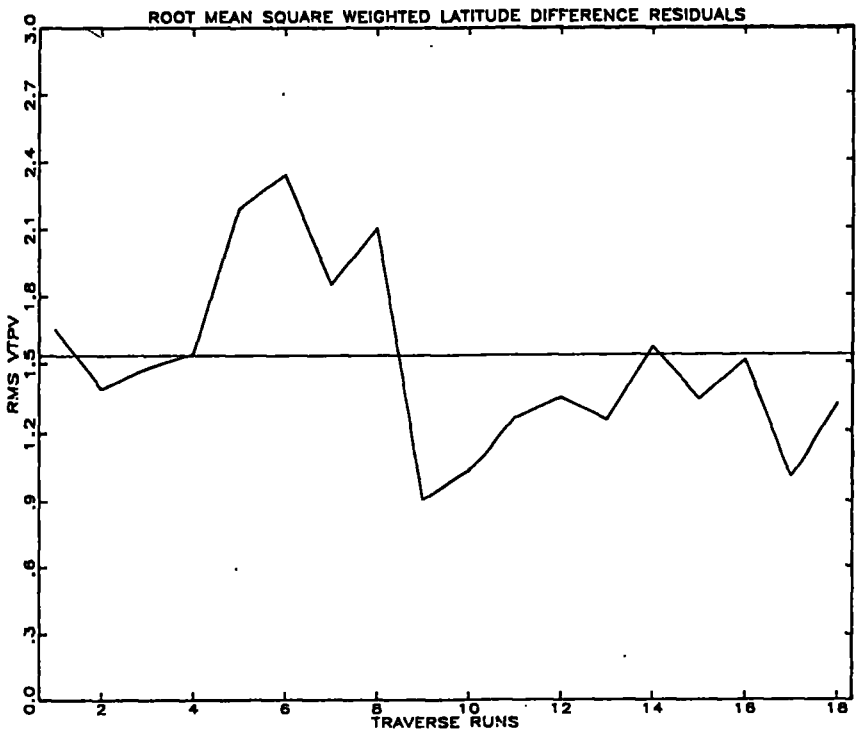


Figure 2.--Root mean square weighted latitude difference residuals.

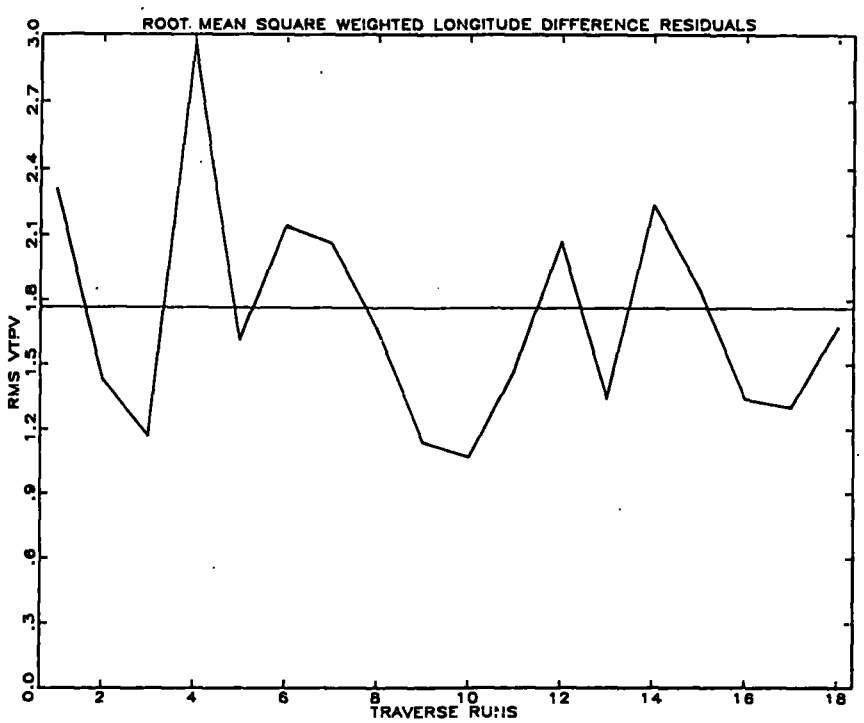


Figure 3.--Root mean square weighted longitude difference residuals.

Table 4.--Residuals categorized by traverse run and type

Run	Number of observations	Latitude rms, V^TPV	Longitude rms, V^TPV	Height rms, V^TPV
1	24	1.65	2.31	2.45
2	22	1.39	1.43	1.42
3	22	1.48	1.17	1.21
4	22	1.54	2.96	1.79
5	22	2.19	1.61	1.53
6	22	2.34	2.14	1.68
7	26	1.85	2.06	1.23
8	26	2.10	1.66	1.11
9	28	.91	1.14	1.27
10	28	1.03	1.07	1.24
11	26	1.26	1.48	.87
12	27	1.35	2.07	2.11
13	26	1.25	1.34	1.48
14	26	1.57	2.24	1.65
15	28	1.34	1.85	.98
16	28	1.51	1.34	.75
17	30	1.00	1.30	.73
18	26	1.32	1.67	1.86
Total	459	1.53	1.76	1.46

More scatter is noticeable in the longitude residuals. The residuals have the same trends. That is, high latitude residuals tend to occur with high longitude residuals. Also of interest are some of the field procedure anomalies. At traverse 8 (cassette 314), a point which did not have previously adjusted coordinates was used for presurvey initialization. A peak appears in the latitude residuals in figure 2. However, no procedural variations explain the longitude residual peaks for traverses 5, 6, and 9. The false coordinate updates at the ends of the traverse seem to have major effect on longitude (fig. 3). In traverse 7, a false longitude, and in traverse 12, a false latitude, were used for the position updates. An explanation for the traverse 4 longitude residuals has not been found.

In closing this section, some points must be made. Since a minimally constrained adjustment was performed, these results on residuals are invariant with respect to the specific choice of the constraints. Also, a simple a-priori weight model was used for the adjustment. However, before more work can be done on observation weight models, more progress is needed with the observation models themselves.

Gregerson Model Parameters

As discussed earlier, a set of 12 parameters was assigned to each of the 18 traverses. These numbers are presented in appendix F. The first column is the adjusted value of the parameter, the second is the standard deviation of that parameter computed by linear error propagation, the third column is the quotient of the first two columns, and the fourth column is the Googe number of that parameter.

The units of the parameters are quite different and complicate interpretation of the results. Normalizing the parameter values by the parameter standard deviations gives a unitless number related to a t statistic. A normalized parameter close to zero indicates that the parameter does not have a significant effect in the adjustment. One can constrain such parameters to zero, effectively removing them from the adjustment without greatly disturbing the results. Of course, any such constraint should be verified by an appropriate F test. Inspection of appendix F shows that the Gregerson parameters generally do not attain large values.

Figure 4 plots the normalized values of C_4 (latitude) for the traverse runs. Figure 5 plots the normalized values of C_8 (longitude). With a few exceptions, these values are close to zero. Recall that false update values were used in traverse 7 and 12, and that traverse 8 used an incorrect presurvey initialization coordinate. No abnormal field procedures explain the normalized C_4 value for traverse 1 or the C_8 values in traverses 2, 3, and 4. The false longitude updates in traverse 7 have a spectacular effect upon the C_8 parameter. I do not feel there is any correlation between a deliberate positional error and the ZUPT interval used in the survey. Rather, I am seeing the Gregerson model trying to absorb an unmodelled positional error as best it can. I feel it is safe to say that the C_4 and C_8 parameters in the Gregerson model are not needed.

As a test, I adjusted the three traverses on cassette 318. The unconstrained, weighted, variance sum was 593.6 with 165 degrees of freedom. Then the values of C_4 and C_8 were constrained to zero in all three terms, totaling 6 constraints. That adjustment produced a variance sum of 619.6 and 171 degrees of freedom. Our null hypothesis is that all six parameters are zero. The F statistic is

$$F_{6,165} = \frac{(619.6 - 593.6)/(171 - 165)}{593.6/165} = 1.20 \qquad (23)$$

The hypothesis can be rejected at the 25 percent level, but cannot be rejected at the 50 percent level or higher. In other words, if one states that at least one of the parameters is not equal to zero, one risks a chance between 50 percent to 75 percent of being wrong. This is evidence that the C_4 and C_8 parameters are not needed to process the data.

Accuracy

Accuracy is the measure of external consistency. Accuracy describes how well an observation agrees with observations of a different character or with a priori knowledge of the "true" observation. For example, the TCT was designed for one part per million precision. But until the intercomparison of TCT and Doppler satellite observations (Gergen 1979), it was not certain the TCT achieved one part per million accuracy. For the purposes of this test, the TCT positions may safely be considered error free.

Table 5 displays the adjusted positions computed from inertial observations and the differences between the TCT and the ISS positions in meters. The sense of the differences is TCT minus ISS, with latitude positive north and longitude positive east.

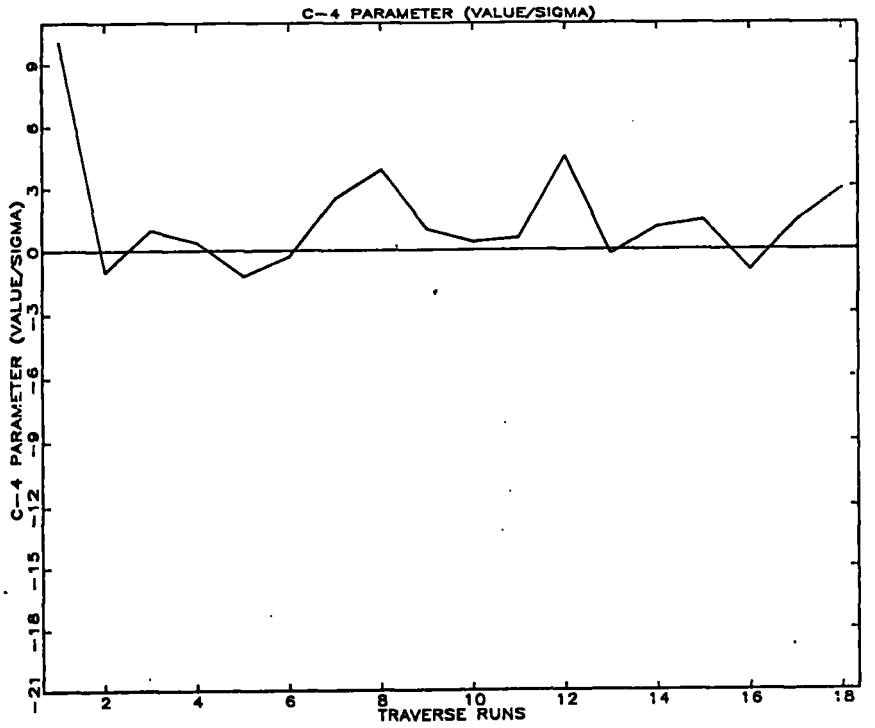


Figure 4.--Normalized C_4 values.

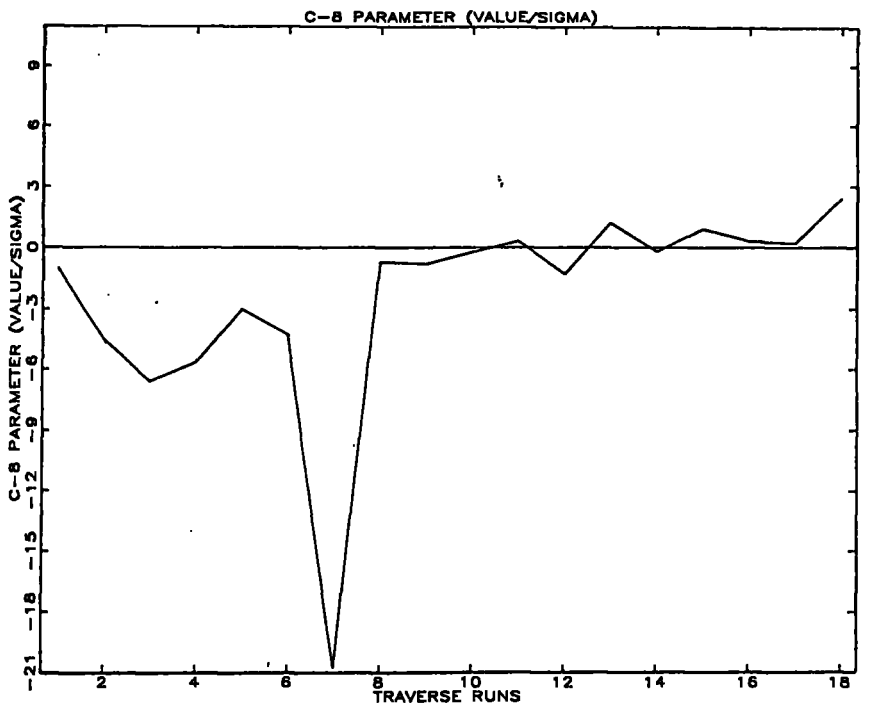


Figure 5.--Normalized C_8 values.

Table 5.--Adjusted Inertial Survey System positions and position shifts

Station	Inertial Survey System latitude	Inertial Survey System longitude	TCT Inertial Survey System latitude shift north	TCT Inertial Survey System longitude shift east
			(m)	(m)
1001	32° 40' 7".39636	114° 24' 29".94778	0.000	0.000
1002	32 40 42.72926	114 18 30.20609	-.139	-.371
1003	32 40 9.24665	114 15 52.09993	-.045	-.189
1004	32 39 28.94035	114 11 29.28443	.250	.048
1005	32 40 19.78441	114 6 58.91141	.137	.044
1006	32 40 48.51984	114 3 3.21138	.112	.100
1007	32 41 46.57782	113 57 18.62695	.000	.000
1008	32 42 22.71924	113 53 33.41661	-.011	-.054
1009	32 42 40.46243	113 51 75.24449	-.019	-.032
1010	32 43 6.70581	113 48 10.69588	-.042	.101
1011	32 43 41.28710	113 45 26.24398	-.068	.061
1012	32 44 17.99584	113 41 46.09254	-.162	.062
1013	32 45 27.59854	113 38 19.83475	.000	.000
2002	32 44 38.90550	114 25 13.64737	-.235	.266
2003	32 48 39.31093	114 22 33.53668	.137	.190
2004	32 51 31.24369	114 21 32.34895	.077	.014
2005	32 55 20.81168	114 18 49.69431	.240	.081
2006	33 1 19.93113	114 16 53.67928	.000	.000
2007	33 6 15.20220	114 17 55.47507	-.174	.226
2008	33 10 25.50101	114 16 34.19250	.030	.309
2009	33 14 22.15564	114 15 28.51281	-.117	.158
2010	33 18 42.15303	114 12 56.56818	.223	.090
2011	33 22 37.06517	114 12 59.96134	.056	.033
2012	33 27 49.48687	114 12 59.86782	.000	.000

Notice that the elevations are omitted from table 5. This omission is deliberate. The TCT elevations are not highly accurate, so elevation intercomparison is inappropriate. Figure 6 plots the longitude differences from table 5 for the east-west portion of the survey. Figure 7 plots longitude differences computed from individual adjustments of cassettes 302, 315, 316, 317, 318, and 319. These figures display ISS inaccuracies along the track of the survey. One immediately notices a westward bias around station 1002.

To determine if this bias is serious or not, the position difference is compared with standard deviations computed by linear error propagation. Table 6 lists the coordinate shifts in table 5 and their associated a posteriori standard deviation at the 95 percent level ($2\,\hat{\sigma}_\phi, 2\,\hat{\sigma}_\lambda$). In other words, statistics predict that 95 percent of the position shifts should fall within a $2\,\hat{\sigma}$ limit.

Examination of table 6 shows systematic errors in the ISS data. The errors are not severe, but they are noticeable. Systematic errors arise when the observation model does not correctly describe the measurements. The Gregerson 12 parameter model does not correctly model Litton Autosurveyor observations recorded in the 79 element mark data set.

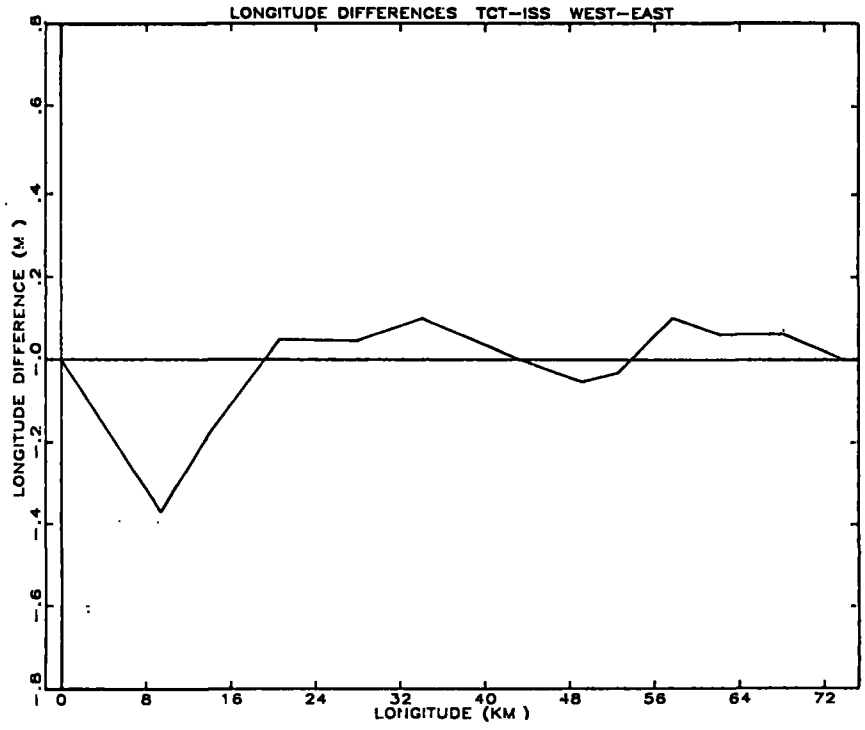


Figure 6.--Combined west-east longitude differences.

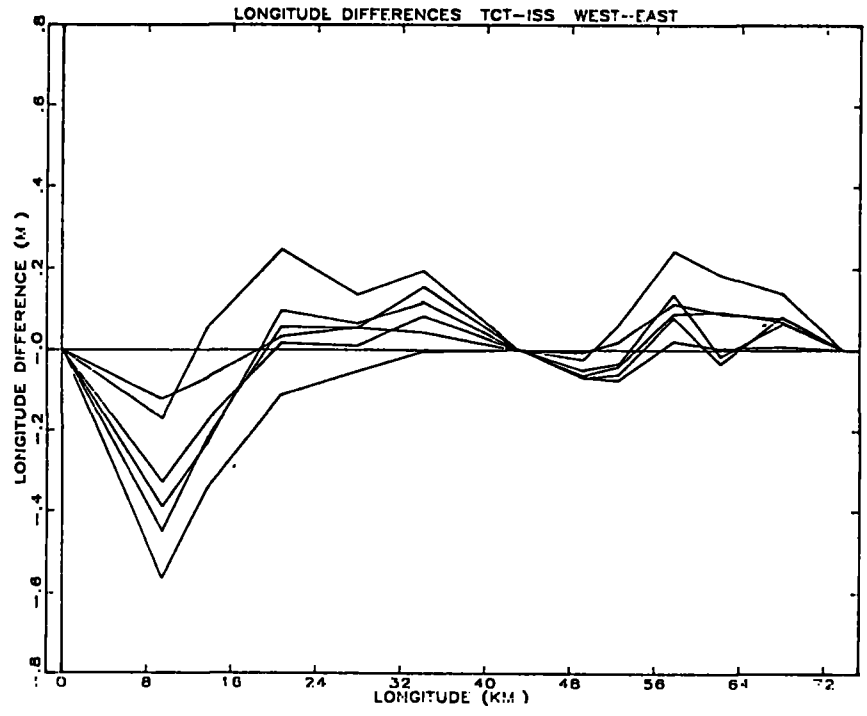


Figure 7.--Individual west-east longitude differences.

Table 6.--Position shifts and position precisions (2 $\hat{\sigma}$)

Number	Latitude shift	Latitude precision (2 $\hat{\sigma}$)	Longitude shift	Longitude precision (2 $\hat{\sigma}$)
			(m)	(m)
1001	0.000	0.004	0.000	0.004
1002	-.139	.105	-.371	.105
1003	-.045	.116	-.189	.116
1004	.250	.170	.048	.170
1005	.137	.123	.044	.123
1006	.112	.094	.100	.094
1007	.000	.004	.000	.004
1008	-.011	.076	-.054	.076
1009	-.019	.098	-.032	.098
1010	-.042	.109	.101	.109
1011	-.068	.105	.061	.105
1012	-.162	.091	.062	.091
1013	.000	.004	.000	.004
2002	-.235	.163	.266	.163
2003	.137	.119	.190	.119
2004	.077	.127	.014	.127
2005	.240	.091	.081	.091
2006	.000	.004	.000	.004
2007	-.174	.138	.226	.138
2008	.030	.123	.309	.123
2009	-.117	.116	.158	.116
2010	.223	.138	.090	.138
2011	.056	.098	.033	.098
2012	.000	.004	.000	.004

In a manner similar to figures 6 and 7, figures 8 and 9 display latitude differences across the track of the west-east survey. Figures 10 and 11 show latitude differences along the track of the south-north traverse. And figures 12 and 13 illustrate longitude differences across the track of the south-north survey.

The systematic errors are most evident in the cross-track of the south-north traverse. The ISS errors have a definite eastward bias. This reinforces evidence presented earlier in this report. The Gregerson model is deficient in modeling longitude observations from the ISS unit. Systematic errors also disturb the latitudes, but predominate in longitude.

Figure 13, which plots the cross-track systematic errors from the south-north runs, has a particularly large curve which reaches 0.8 meter at station 2007. This curve corresponds to cassette 309, and this traverse had a deliberate longitude update. Clearly, the Gregerson model has difficulty in capturing the errors introduced by such distortions. The second largest curve comes from cassette 314. Field procedures were violated when those observations were made. A presurvey update was made at station 2003. Since the true coordinates were withheld until after the test, the position used for the update was in error. This error amounted to 0.5 meter in longitude. This violation in field procedure could explain the systematic errors in cassette 314. The remaining traverses, 305, 306, and 322, conform to the test field procedures. Even disregarding 309 and 314, which account for two complete runs, the remaining data comprise six complete runs, and still demonstrate an eastward bias.

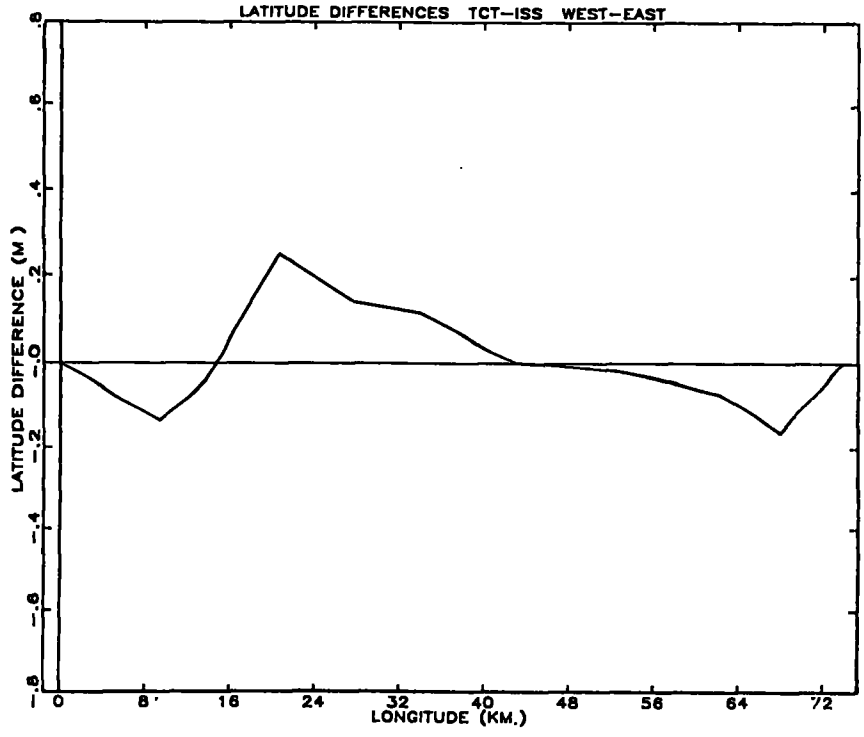


Figure 8.--Combined west-east latitude differences.

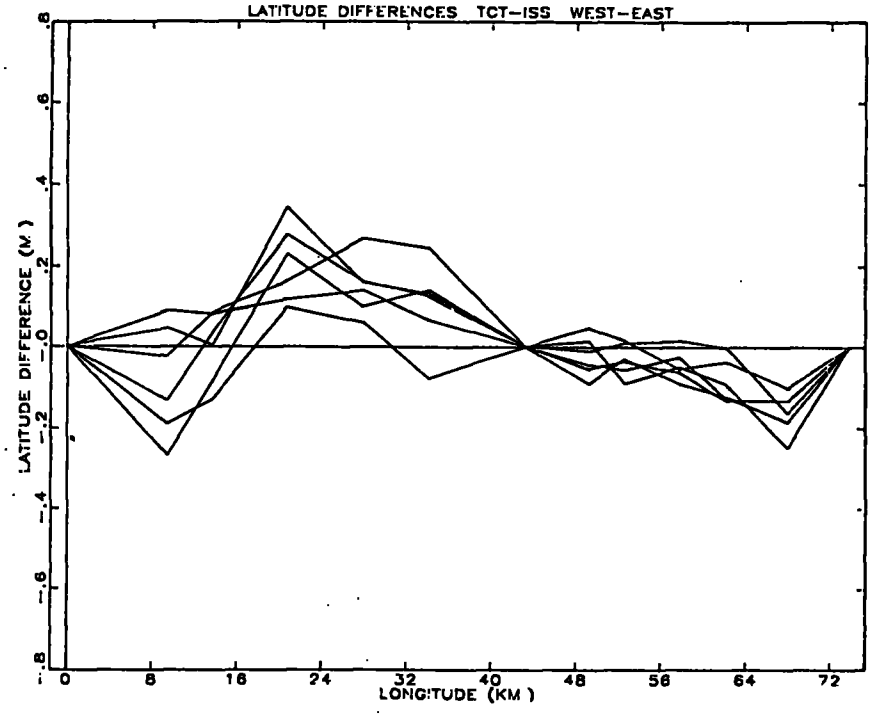


Figure 9.--Individual west-east latitude differences.

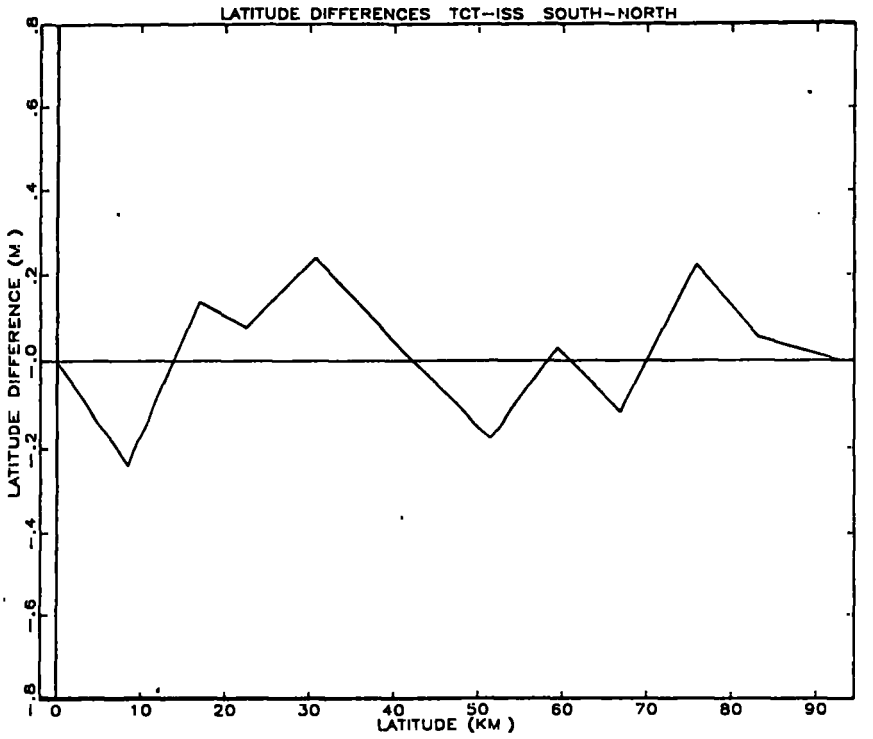


Figure 10.--Combined south-north latitude differences.

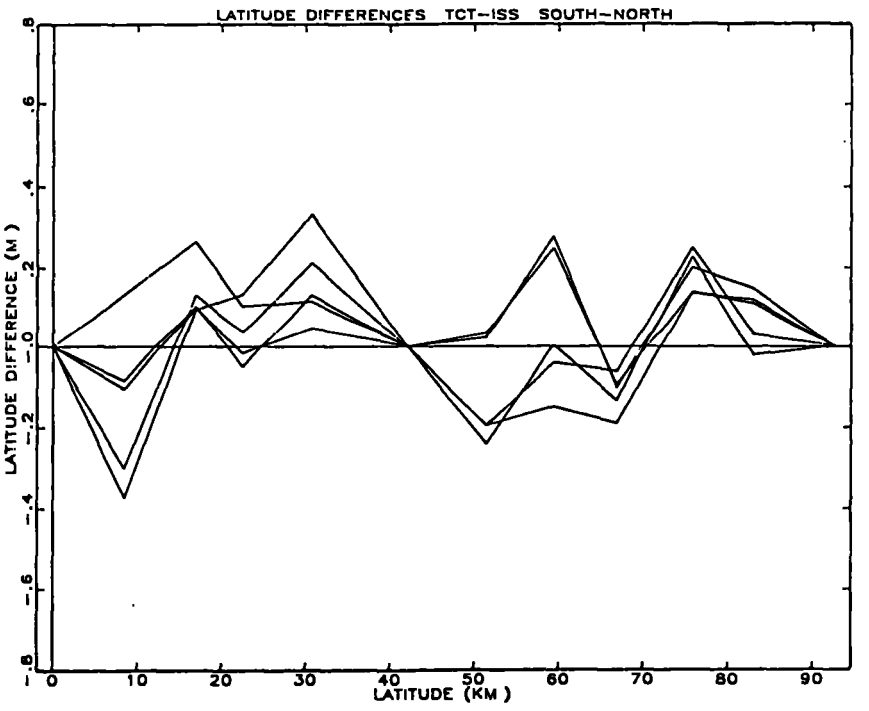


Figure 11.--Individual south-north latitude differences.

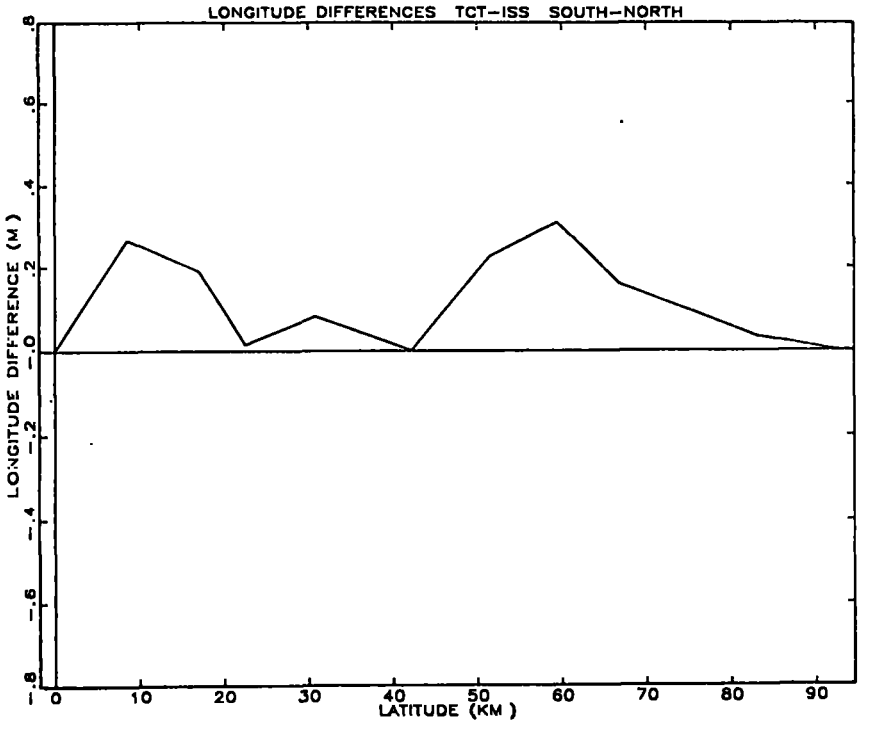


Figure 12.--Combined south-north longitude differences.

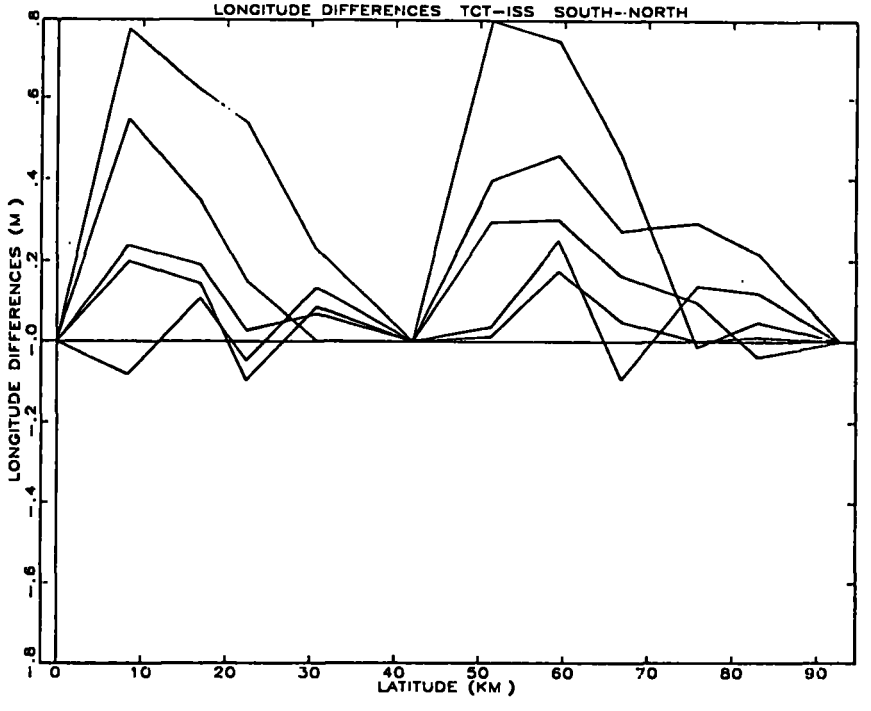


Figure 13.--Individual south-north longitude differences.

As mentioned earlier in this report, the standard for horizontal control is the length relative accuracy between directly connected adjacent points. This is the official standard for classification of control networks. A length relative accuracy is expressed as a ratio, 1:r, where r is computed as

$$r = \frac{\ell_x - \ell_s}{\ell_s} \tag{24}$$

and ℓ_s is the computed distance between the known survey coordinates, and ℓ_x is the computed distance between the survey coordinates to be evaluated. The length relative accuracies for the ISS test are presented in table 7.

The length relative accuracies range from 1:24538 to 1:2115481. Since the classification standard is expressed as a minimum value of the length relative accuracy, the worst length relative accuracy, 1:24528, must be used to classify the test. This ISS test, processed by the Gregerson model, meets the second order, class II standard, 1:20000. It does not qualify for second order, class I, or any higher order work.

The presence of systematic error is also evident when examining the length relative accuracies. The right hand column of table 7 lists twice the a-posteriori standard deviation of the adjusted length. These values are computed by error propagation using the method discussed earlier in this report. Statistics predict these values would only be exceeded 2.3 percent of the time.

Table 7.--Length relative accuracies

Line	$\lvert \ell_x - \ell_s \rvert$	ℓ_s	1:r	$2\,\hat{\sigma}_{\ell_s}$
	(m)	(m)		(m)
1001-1002	0.385	9436	1:24538	0.091
1002-1003	.153	4247	1:27689	.087
1003-1004	.181	6960	1:38413	.105
1004-1005	.029	7217	1:245558	.080
1005-1006	.052	6204	1:118739	.065
1006-1007	.120	9153	1:76271	.080
1007-1008	.055	5971	1:109243	.062
1008-1009	.020	3383	1:170778	.065
1009-1010	.128	5131	1:40205	.065
1010-1011	.045	4413	1:97701	.062
1011-1012	.017	5843	1:346177	.062
1012-1013	.003	5782	1:2155481	.043
1001-2002	.269	8441	1:31422	.163
2002-2003	.286	8497	1:29681	.109
2003-2004	.108	5530	1:51409	.083
2004-2005	.174	8239	1:47385	.112
2005-2006	.253	11466	1:45407	.087
2006-2007	.211	9236	1:43835	.134
2007-2008	.218	7993	1:36592	.094
2008-2009	.177	7486	1:42331	.091
2009-2010	.275	8923	1:32455	.141
2010-2012	.166	7238	1:43707	.091
2011-2013	.056	9625	1:17023	.098

CONCLUSIONS

Experience with INERT1 validated the original design aims. The correct, simultaneous adjustment of multiple traverse runs clearly highlighted data problems. The provision for weighted constraints on both station coordinates and model parameters enabled statistical tests of the C_4 and C_8 terms of the Gregerson model. The various residual statistics aided evaluation of observation precision. Display of position shifts allowed evaluation of model accuracy since reliable initial positions were available.

It may further be concluded that the Gregerson model does not accurately model Litton Autosurveyor observations made under the prescribed field procedures. Possible areas for model improvement would be variation of the number of model parameters, variation of parameter scope, inclusion of a specific term for heading misalignment, and allowance for variations in deflection of the vertical.

REFERENCES

Dijkstra, E., 1965: Programming considered as a human activity. Proceedings of the 1965 IFIP Congress, New York, North-Holland Publishing Co., Amsterdam, The Netherlands.

Dillinger, William H., 1981: Subroutine package for large, sparse, least-squares problems. NOAA Technical Memorandum, NOS NGS 29, National Geodetic Information Center, Rockville, Md. 20852, 18 pp.

El Hakim, S. F. and Faig, W., 1981: A combined adjustment of geodetic and photogrammetric observations. Photogrammetric Engineering and Remote Sensing, 47(1), 93-99.

Federal Geodetic Control Committee, 1974: Classification, Standards of Accuracy, and General Specifications of Geodetic Control Surveys. Superintendent of Documents, U. S. Government Printing Office, Washington, D.C., 12 pp.

Gergen, John G., 1979: The relationship of Doppler satellite positions to the U.S. Transcontinental Traverse. Proceedings of the Second International Geodetic Symposium on Satellite Doppler Positioning, Austin, Texas, January 22-26, 1979, Defense Mapping Agency, Washington D.C., 689-694.

Hannah, John and Mueller, Ivan I., 1981: Improvement of inertial surveys through post-mission network adjustment and self-calibration. Proceedings of the Second International Symposium on Inertial Technology for Surveying and Geodesy, Banff, Alberta, Canada, June 1-5, 1981, Canadian Institute of Surveying, Ottawa, 177-189.

Hannah, John and Pavlis, Despina E., 1980: Post-mission adjustment techniques for inertial surveys. Reports of the Department of Geodetic Science, Report No. 305, Department of Geodetic Science, Ohio State University, Columbus, Ohio, 94 pp.

Jennings, Alan, 1977: Matrix Computation for Engineers and Scientists. John-Wiley & Sons, New York, N. Y., 330 pp.

Kouba, J. 1977: Geodetic adjustment of inertial surveys. Proceedings of the First International Symposium on Inertial Technology for Surveying and Geodesy, Ottawa, Canada, October 12-14, 1977, Canadian Institute of Surveying, Ottawa, 162-187.

Leigh, George E., 1981: Preliminary data analysis of inertial survey system test in southwest Arizona. Proceedings of the Second Inertial Symposium on Inertial Technology for Surveying and Geodesy, Banff, Alberta, Canada, June 1-5, 1981, Canadian Institute of Surveying, Ottawa, 415-430.

Merchant, Dean C., 1977: Introduction to advanced photogrammetry (lecture notes), Department of Geodetic Science, Ohio State University, Columbus, Ohio.

Mueller, Ivan I., 1981: Inertial survey systems in the geodetic arsenal. Proceedings of the Second International Symposium on Inertial Technology for Surveying and Geodesy, Banff, Alberta, Canada, June 1-5, 1981, Canadian Institute of Surveying, Ottawa, 11-33.

Pope, Alan J., 1976: The statistics of residuals and the detection of outliers. NOAA Technical Report, NOS 65 NGS 1, National Geodetic Information Center, Rockville, Md. 20852, 133 pp.

Rapp, Richard H., 1977: Geometric geodesy, volume I (lecture notes), Department of Geodetic Science, Ohio State University, Columbus, Ohio.

Schwarz, Charles R., 1974-75: Adjustment computations (lecture notes), Department of Geography and Regional Science, George Washington University, Washington D.C.

Schwarz, Charles R., 1978: TRAV10 horizontal network adjustment program. NOAA Technical Memorandum, NOS NGS 12, 52 pp., National Geodetic Information Center., Rockville, Md. 20852.

Schwarz, Klaus Peter, 1980: Error propagation in inertial positioning. The Canadian Surveyor, 34(3), 265-276.

Snay, Richard A., 1976: Reducing the profile of sparse symmetric matricies. NOAA Technical Memorandum, NOS NGS 4, National Geodetic Information Center, Rockville, Md. 20852, 24 pp.

Snay, Richard A. (National Geodetic Survey, National Ocean Survey, National Oceanic and Atmospheric Administration, U. S. Department of Commerce, Rockville, Md.) 1978: A short study of tellurometer observations in the Kentucky-Tennessee test block (unpublished).

Turner, Joshua, 1980: The structure of modular programs. Communications of the ACM, 23(5), 272-277.

Uotila, Urho A., 1967: Introduction to adjustment computations with matrices (lecture notes), Department of Geodetic Science, Ohio State University, Columbus, Ohio.

APPENDIX A.--INERT1 DATA FORMAT

INERT1 DATA FORMATS

The INERT1 data format is structured into a "deck" of 6 record types.

	INERT1 parameter record	(mandatory)
'D'	Initial positions	(mandatory)
'K'	Constrained positions	(mandatory)
'N'	Constrained model parameters	(optional)
'I'	Inertial observations	(mandatory)
'Q'	Length relative accuracy computation	(optional)

There is only one INERT1 parameter record and it must be the first record in the data set. The remaining record types may be intermixed somewhat, but keeping them grouped by types in the order above will guarantee no structure problems. The inertial observations MUST be chronological and grouped together by traverse run.

INERT1 Parameter Record (First record in deck)

01-05	Mean ZUPT Interval, units of seconds, optional, default--220 seconds	(integer, right justified)
06-10	Number of Position Names, required, no default	(integer, right justified)
11-15	Number of Traverse Runs, required, no default	(integer, right justified)
16-25	Semi-Major Axis, optional, default 6378137.000	(real, 3 implied decimals)
26-43	Square First Eccentricity, optional, default 0.00669438002290341567	(real, 18 implied decimals)
44-44	Longitude Code (Code: blank--positive east	nonblank--positive west)
45-90	Reserved	

Initial Position Record

01-01	D	
02-06	Reserved	
07-36	Position Name, required	(alphanumeric, left justified)
37-37	Latitude Sign, +/-, optional, default +	
38-39	Degrees Latitude, default 0	(integer, right justified)
40-41	Minutes Latitude, default 0	(integer, right justified)
42-48	Seconds Latitude, units of 0.00001 seconds	(integer, right justified)
49-49	Longitude Sign, +/-, optional, default +	
50-52	Degrees Longitude, default 0	(integer, right justified)
53-54	Minutes Longitude, default 0	(integer, right justified)
55-61	Seconds Longitude, units of 0.00001 seconds	(integer, right justified)
62-68	Elevation, units of millimeters, default 0	(integer, right justified)
69-90	Reserved	

Comment: Each name must be unique. The total number of these cards MUST exactly match the number on 06-10 in the parameter card. No name may appear in later cards that is not in an Initial Position Card.

Constrained Position Record

01-01	K	
02-06	Reserved	
07-36	Position Name, required	(alphanumeric, left justified)
37-37	Latitude Sign, +/-, optional, default +	
38-39	Degrees Latitude, default 0	(integer, right justified)
40-41	Minutes Latitude, default 0	(integer, right justified)
42-48	Seconds Latitude, units of 0.00001 seconds	(integer, right justified)
49-49	Longitude Sign, +/-, optional, default +	
50-52	Degrees Longitude, default 0	(integer, right justified)
53-54	Minutes Longitude, default 0	(integer, right justified)
55-61	Seconds Longitude, units of 0.00001 seconds	(integer, right justified)
62-68	Elevation, units of millimeters, default 0	(integer, right justified)
69-74	Latitude Standard Deviation, units of millimeters, default 1 mm.	(integer,right justified)
75-80	Longitude Standard Deviation, units of millimeters, default 1 mm.	(integer,right justified)
81-86	Elevation Standard Deviation, units of millimeters, default 1 mm.	(integer,right justified)
87-90	Reserved	

Constrained Model Parameter Record

01-01	N
02-02	Reserved
03-06	Traverse Run Number, required, >=1, (integer, right justified)
07-08	Parameter Number, required, 1-12 (integer, right justified)
09-20	Parameter Value, default 0 (real, implied decimal between 11 and 12)
21-30	Parameter Standard Deviation, required (real, implied decimal between 22 and 23)
31-90	Reserved

Comment: The traverse run number in 03-06 may not exceed the number in 11-15 of the parameter record above.

Inertial Observation Run Header Record

01-01	I	
02-02	R	
03-06	Traverse Run Number, required, >=1,	(integer, right justified)
07-36	Position Name, required	(alphanumeric, left justified)
37-37	Latitude Sign, +/-, optional, default +	
38-39	Degrees Latitude, default 0	(integer, right justified)
40-41	Minutes Latitude, default 0	(integer, right justified)
42-48	Seconds Latitude, units of 0.00001 seconds	(integer, right justified)
49-49	Longitude Sign, +/-, optional, default +	
50-52	Degrees Longitude, default 0	(integer, right justified)
53-54	Minutes Longitude, default 0	(integer, right justified)
55-61	Seconds Longitude, units of 0.00001 seconds	(integer, right justified)
62-68	Elevation, units of millimeters, default 0	(integer, right justified)
69-70	Hour of Observation, default 0	(integer, right justified)
71-72	Minute of Observation, default 0	(integer, right justified)
73-74	Second of Observation, default 0	(integer, right justified)
75-90	Reserved	

Comment: The traverse run number in 03-06 may not exceed the number in 11-15 of the parameter record above.

Inertial Observation Mark Record

01-01	I	
02-02	M	
03-06	Reserved	
07-36	Position Name, required	(alphanumeric, left justified)
37-37	Latitude Sign, +/-, optional, default +	
38-39	Degrees Latitude, default 0	(integer, right justified)
40-41	Minutes Latitude, default 0	(integer, right justified)
42-48	Seconds Latitude, units of 0.00001 seconds	(integer, right justified)
49-49	Longitude Sign, +/-, optional, default +	
50-52	Degrees Longitude, default 0	(integer, right justified)
53-54	Minutes Longitude, default 0	(integer, right justified)
55-61	Seconds Longitude, units of 0.00001 seconds	(integer, right justified)
62-68	Elevation, units of millimeters, default 0	(integer, right justified)
69-70	Hour of Observation, default 0	(integer, right justified)
71-72	Minute of Observation, default 0	(integer, right justified)
73-74	Second of Observation, default 0	(integer, right justified)
75-79	Latitude Difference Standard Deviation, units of millimeters, default 100 mm.	(integer, right justified)
80-84	Longitude Difference Standard Deviation, units of millimeters, default 100 mm.	(integer, right justified)
85-89	Elevation Difference Standard Deviation, units of millimeters, default 100 mm.	(integer, right justified)
90-90	Reserved	

Inertial Observation Update Record

01-01	I	
02-02	U	
03-36	Reserved	
37-37	Latitude Sign, +/-, optional, default +	
38-39	Degrees Latitude, default 0	(integer, right justified)
40-41	Minutes Latitude, default 0	(integer, right justified)
42-48	Seconds Latitude, units of 0.00001 seconds	(integer, right justified)
49-49	Longitude Sign, +/-, optional, default +	
50-52	Degrees Longitude, default 0	(integer, right justified)
53-54	Minutes Longitude, default 0	(integer, right justified)
55-61	Seconds Longitude, units of 0.00001 seconds	(integer, right justified)
62-68	Elevation, units of millimeters, default 0	(integer, right justified)
69-90	Reserved	

Length Relative Accuracy Computation Record

01-01	Q	
02-06	Reserved	
07-36	Position Name, required	(alphanumeric, left justified)
37-66	Position Name, required	(alphanumeric, left justified)
67-90	Reserved	

Comment: The names must NOT be identical.

SAMPLE DATA SET

```
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890

  220    7    1
D     JACK                          +32523935510-1060710302801243044
D     MONUMENT 14                   +33000845770-1060830054401258138
D     IPS 3                         +33015460550-1060856266901257820
D     OASIS                         +33043583740-1060854167901265633
D     VALLEY ASTRO                  +33062146590-1061610970901221461
D     B.M.G-48                      +33065438040-1061851284101219781
D     SALT                          +33071222320-1062147993001234945
K     JACK                          +32523935510-1060710302801243044
K     OASIS                         +33043583740-1060854167901265633
K     SALT                          +33071222320-1062147993001234945
N    110  0000000000  10000000
IR   1JACK                          +32523935500-1060710303001243044081959
IM    MONUMENT 14                   +33000846200-1060830020001257950084009
IM    IPS 3                         +33015461600-1060856219001257230084850
IM    OASIS                         +33043584100-1060854096001265026085702
IM    VALLEY ASTRO                  +33062152100-1061611052001220633091214
IM    B.M.G-48                      +33065445400-1061851429001218826092110
IM    SALT                          +33071231800-1062148210001233699092945
IU                                  +33071222400-1062147994001234945
IM    B.M.G-48                      +33065434900-1061851227001219636094538
IM    VALLEY ASTRO                  +33062140600-1061610863001221319095338
IM    OASIS                         +33043570200-1060853909001265611100724
IM    IPS 3                         +33015446600-1060856048001257710101600
IM    MONUMENT 14                   +33000831800-1060829842001257606102400
IM    JACK                          +32523920100-1060710165001242209104147
Q     JACK                          MONUMENT 14
Q     JACK                          IPS 3
Q     VALLEY ASTRO                  SALT
Q     B.M.G-48                      JACK
Q     JACK                          B.M.G-48
Q     JACK                          VALLEY ASTRO
Q     MONUMENT 14                   IPS 3
Q     MONUMENT 14                   VALLEY ASTRO
Q     MONUMENT 14                   B.M.G-48
Q     IPS 3                         OASIS
Q     OASIS                         VALLEY ASTRO
Q     VALLEY ASTRO                  B.M.G-48
Q     B.M.G-48                      SALT

12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
```

Discussion

The inertial observation section of the INERT1 data set may be thought of as a series of sub-sections, one for each traverse run, back to back. Each traverse run must begin with an inertial run header record, followed by the appropriate number of inertial mark records. Inertial update records appear where appropriate, and they appear after the corresponding mark record for a given point. The inertial update record resets the position for the point at the prior inertial mark record. The time of the update event is assumed to be the time of that prior mark record. A traverse run does NOT need to end with an inertial update record, since a subsequent traverse run will be initialized by its own run header record.

Although the inertial observation format records positions, the INERT1 program extracts position DIFFERENCES as observations between sequential points. Therefore the position standard deviation at a particular inertial observation mark record is refering to the position difference between that point and the run header, update, or mark observation preceeding it. It can be seen that the run header record and the update record are nothing more than mechanisms to reset the most recent position from which a positon difference is to be computed. Additionally, the run header tells the INERT1 adjustment program to initialize model parameters for a new traverse run.

The constrained position record may constrain latitude and/or longitude and/or elevation to a known value. If any of these fields is completely blank, then that particular unknown will not be constrained. Thus, only elevation or only horizontal coordinates may be constrained at a point.

APPENDIX B.--SAMPLE INERT1 LISTING

```
INERT - JANUARY 21, 1983 / 06:33:41                    NATIONAL GEODETIC SURVEY
PROGRAM INERT1                                         INERTIAL ADJUSTMENT PROGRAM

        A= 6378206.400
       E2= .006768657997291000
       LONGITUDES POSITIVE EAST


THE UNKNOWNS HAVE BEEN REORDERED,
     D     JACK                      +32523935510-1060710302801243044
     D     MONUMENT 14               +33000845770-1060830054401258138
     D     IPS 3                     +33015460550-1060856266901257820
     D     OASIS                     +33043583740-1060854167901265633
     D     VALLEY ASTRO              +33062146590-1061610970901221461
     D     B.M.G-48                  +33065438040-1061851284101219781
     D     SALT                      +33071222320-1062147993001234945
   3 K     JACK                      +32523935510-1060710302801243044
   6 K     OASIS                     +33043583740-1060854167901265633
   9 K     SALT                      +33071222320-1062147993001234945
     IR   1JACK                      +32523935500-1060710303001243044081959
  12 IM    MONUMENT 14               +33000846200-1060830020001257950084009      .132     .893
  15 IM    IPS 3                     +33015461600-1060856219001257230084850      .323    1.243
  18 IM    OASIS                     +33043584100-1060854096001265026085702      .111    1.865
  21 IM    VALLEY ASTRO              +33062152100-1061611052001220633091214     1.697   -2.103
  24 IM    B.M.G-48                  +33065445400-1061851429001218826092110     2.267   -3.757
  27 IM    SALT                      +33071231800-1062148210001233699092945     2.920   -5.626
     IU                              +33071222400-1062147994001234945
  30 IM    B.M.G-48                  +33065434900-1061851227001219636094538      -.967    1.480
  33 IM    VALLEY ASTRO              +33062140600-1061610863001221319095338    -1.845    2.798
  36 IM    OASIS                     +33043570200-1060853909001265611100724    -4.171    6.715
  39 IM    IPS 3                     +33015446600-1060856048001257710101600    -4.297    5.681
  42 IM    MONUMENT 14               +33000831800-1060829842001257606102400    -4.304    5.514
  45 IM    JACK                      +32523920100-1060710165001242209104147    -4.747    3.582
     Q     JACK                      MONUMENT 14
     Q     JACK                      IPS 3
     Q     VALLEY ASTRO              SALT
     Q     B.M.G-48                  JACK
     Q     JACK                      B.M.G-48
     Q     JACK                      VALLEY ASTRO
     Q     MONUMENT 14               IPS 3
     Q     MONUMENT 14               VALLEY ASTRO
     Q     MONUMENT 14               B.M.G-48
     Q     IPS 3                     OASIS
     Q     OASIS                     VALLEY ASTRO
     Q     VALLEY ASTRO              B.M.G-48
     Q     B.M.G-48                  SALT
USING THE BANKERS ALGORITHM
THERE ARE   33  UNKNOWNS IN COMPONENT    1
                                        STARTING NODE IS     28
THE TOTAL NUMBER OF ELEMENTS IN THE ORIGINAL MATRIX EXCLUDING THE ABSOLUTE COLUMN IS      561
EXCLUDING THE ABSOLUTE COLUMN THE REORDERED MATRIX HAS                                    433
THE ELEMENTS IN THE ABSOLUTE COLUMN AND ANY ADDITIONAL VECTORS  TOTAL                      34
AT ITERATION # 0 THE RMS CORRECTION IS              .003 METERS
ADJUSTED POSITONS

                         LATITUDE          LONGITUDE    HEIGHT    STD.DEV.(METERS)              GOOGE

   1 JACK          32 52 39.35510  -106  7 10.30280  1243.044    .001  .001  .001    1.0+000  1.0+000  1.0+000
   2 MONUMENT 14   33  0  8.45801  -106  8 30.05442  1258.139    .068  .068  .068    5.5-001  5.5-001  7.5-001
   3 IPS 3         33  1 54.60576  -106  8 56.26698  1257.820    .059  .059  .059    9.9-001  9.9-001  1.0+000
   4 OASIS         33  4 35.83740  -106  8 54.16790  1265.633    .001  .001  .001    1.0+000  1.0+000  1.0+000
   5 VALLEY ASTRO  33  6 21.46577  -106 16 10.97075  1221.461    .065  .065  .064    6.0-001  8.5-001  8.7-001
   6 B.M.G-48      33  6 54.38031  -106 18 51.28430  1219.779    .061  .060  .059    9.7-001  6.9-001  7.2-001
   7 SALT          33  7 12.22320  -106 21 47.99300  1234.945    .001  .001  .001    1.0+000  1.0+000  1.0+000
 ADJUSTED PARAMETERS

 RUN=  1
```

	VALUE	STD.DEV.	VAL/SIG	GOOGE
1	-5.33-006	5.77-006	-.9	1.00+000
2	9.92-005	1.97-005	5.0	8.66-001
3	1.11-005	4.44-006	2.5	1.20-001
4	-7.88-011	4.58-011	-1.7	4.17-001
5	3.94-004	5.41-006	72.9	9.17-001
6	1.56-004	1.07-005	14.7	3.71-001
7	8.43-006	2.04-006	4.1	1.00+000
8	-9.75-011	4.50-011	-2.2	4.26-001
9	-1.08+002	5.48+001	-2.0	2.81-001
10	1.66+002	9.44+001	1.8	1.12-001
11	3.50+001	9.45+000	3.7	7.26-001
12	-1.72+001	2.02+001	-.8	9.88-001

ADJUSTED POSITION SHIFTS

		LATITUDE SEC	LATITUDE METERS	LONGITUDE SEC	LONGITUDE METERS	AZM. DEG	HORIZ. METERS	ELEV. METER	TOTAL METER
1	JACK	.00000	.000	.00000	.000	0	.000	.000	.000
2	MONUMENT 14	.00031	.009	-.00002	-.001	0	.009	.001	.010
3	IPS 3	.00026	.008	-.00008	-.002	-14	.008	.000	.008
4	OASIS	.00000	.000	.00000	.000	0	.000	.000	.000
5	VALLEY ASTRO	-.00013	-.004	.00015	.004	137	.006	.000	.006
6	B.M.G-48	-.00009	-.003	-.00020	-.005	-117	.006	-.002	.006
7	SALT	.00000	.000	.00000	.000	0	.000	.000	.000

RESIDUALS

		COMPUTED	OBSERVED	V=C-O SEC.	V=C-O METER	V/SD			
1	LAT.	32 52 39.35510	32 52 39.35510	.00000	.000	.0	JACK		
2	LON.	-106 7 10.30280	-106 7 10.30280	.00000	.000	.0	JACK		
3	HT.	1243.044	1243.044		.000	.0	JACK		
4	LAT.	33 4 35.83740	33 4 35.83740	.00000	.000	.0	OASIS		
5	LON.	-106 8 54.16790	-106 8 54.16790	.00000	.000	.0	OASIS		
6	HT.	1265.633	1265.633		.000	.0	OASIS		
7	LAT.	33 7 12.22320	33 7 12.22320	.00000	.000	.0	SALT		
8	LON.	-106 21 47.99300	-106 21 47.99300	.00000	.000	.0	SALT		
9	HT.	1234.945	1234.945		.000	.0	SALT		
10	1 DEL LAT.	0 7 29.10483	0 7 29.10700	-.00217	-.067	-.7	JACK		MONUMENT 14
11	1 DEL LON.	0 -1 19.71580	0 -1 19.71700	.00120	.037	.4	JACK		MONUMENT 14
12	1 DEL HT.	14.846	14.906		-.060	-.6	JACK		MONUMENT 14
13	1 DEL LAT.	0 1 46.14853	0 1 46.15400	-.00547	-.169	-1.7	MONUMENT 14		IPS 3
14	1 DEL LON.	0 0-26.20709	0 0-26.19900	-.00809	-.250	-2.5	MONUMENT 14		IPS 3
15	1 DEL HT.	-.366	-.720		.354	3.5	MONUMENT 14		IPS 3
16	1 DEL LAT.	0 2 41.22891	0 2 41.22500	.00391	.121	1.2	IPS 3		OASIS
17	1 DEL LON.	0 0 2.12576	0 0 2.12300	.00276	.085	.9	IPS 3		OASIS
18	1 DEL HT.	7.784	7.796		-.012	-.1	IPS 3		OASIS
19	1 DEL LAT.	0 1 45.68091	0 1 45.68000	.00091	.028	.3	OASIS		VALLEY ASTRO
20	1 DEL LON.	0 -7 16.96019	0 -7 16.95600	-.00419	-.130	-1.3	OASIS		VALLEY ASTRO
21	1 DEL HT.	-44.434	-44.393		-.041	-.4	OASIS		VALLEY ASTRO
22	1 DEL LAT.	0 0 32.93459	0 0 32.93300	.00159	.049	.5	VALLEY ASTRO		B.M.G-48
23	1 DEL LON.	0 -2 40.37283	0 -2 40.37700	.00417	.129	1.3	VALLEY ASTRO		B.M.G-48
24	1 DEL HT.	-1.764	-1.807		.043	.4	VALLEY ASTRO		B.M.G-48
25	1 DEL LAT.	0 0 17.86633	0 0 17.86400	.00233	.072	.7	B.M.G-48		SALT
26	1 DEL LON.	0 -2 56.77708	0 -2 56.78100	.00392	.121	1.2	B.M.G-48		SALT
27	1 DEL HT.	15.084	14.873		.211	2.1	B.M.G-48		SALT
28	1 DEL LAT.	0 0-17.87267	0 0-17.87500	.00233	.072	.7	SALT		B.M.G-48
29	1 DEL LON.	0 2 56.77092	0 2 56.76700	.00392	.121	1.2	SALT		B.M.G-48
30	1 DEL HT.	-15.098	-15.309		.211	2.1	SALT		B.M.G-48
31	1 DEL LAT.	0 0-32.94141	0 0-32.94300	.00159	.049	.5	B.M.G-48		VALLEY ASTRO
32	1 DEL LON.	0 2 40.36817	0 2 40.36400	.00417	.129	1.3	B.M.G-48		VALLEY ASTRO
33	1 DEL HT.	1.726	1.683		.043	.4	B.M.G-48		VALLEY ASTRO
34	1 DEL LAT.	0 -1 45.70309	0 -1 45.70400	.00091	.028	.3	VALLEY ASTRO		OASIS
35	1 DEL LON.	0 7 16.94981	0 7 16.95400	-.00419	-.130	-1.3	VALLEY ASTRO		OASIS
36	1 DEL HT.	44.251	44.292		-.041	-.4	VALLEY ASTRO		OASIS
37	1 DEL LAT.	0 -2 41.23209	0 -2 41.23600	.00391	.121	1.2	OASIS		IPS 3
38	1 DEL LON.	0 0 -2.13624	0 0 -2.13900	.00276	.085	.9	OASIS		IPS 3
39	1 DEL HT.	-7.913	-7.901		-.012	-.1	OASIS		IPS 3
40	1 DEL LAT.	0 -1 46.15347	0 -1 46.14800	-.00547	-.169	-1.7	IPS 3		MONUMENT 14
41	1 DEL LON.	0 0 26.19791	0 0 26.20600	-.00809	-.250	-2.5	IPS 3		MONUMENT 14

```
  42  1 DEL HT.            .250            -.104            .354  3.5 IPS 3                          MONUMENT 14
  43  1 DEL LAT.   0 -7 29.11917    0 -7 29.11700 -.00217  -.067  -.7 MONUMENT 14                    JACK
  44  1 DEL LON.   0  1 19.67820    0  1 19.67700  .00120   .037   .4 MONUMENT 14                    JACK
  45  1 DEL HT.          -15.457          -15.397          -.060  -.6 MONUMENT 14                    JACK
RESIDUAL STATISTICS


OBSERVATION NUMBERS OF 20 GREATEST QUASI-NORMALIZED RESIDUALS (V/SD)
    42    15    14    41    27    30    40    13    35    20    32    23    26    29    37  16    38  17    28  25


 TOTAL=   45        NO-CHECK=  9
 MAX V=  3.5-001    MAX V/SD=  3.541
 MIN V= -6.0-002    MIN V/SD=  -.060
MEAN V=  2.2-002   MEAN V/SD=   .233

              N      VTPV      RMS   MEAN ABS
                              VTPV   RESIDUAL
DELTA LAT    12      11.2      .96       .084 (METERS)
DELTA LON    12      23.9     1.41       .125 (METERS)
DELTA   H    12      35.5     1.72       .120 (METERS)
    OTHER     9        .0      .00       .000
    TOTAL    45      70.5     1.25


        LATITUDE                        LONGITUDE                        HEIGHT                        OTHER
RUN    N   VTPV    RMS   MEAN        N   VTPV    RMS   MEAN        N   VTPV    RMS   MEAN  N   VTPV    RMS   MEAN
                   VTPV    V                     VTPV    V                     VTPV    V                VTPV    V
  1   12   11.2    .96   .084       12   23.9   1.41   .125       12   35.5   1.72   .120  0     .0     .00   .000
ELLIPSOIDAL LENGTH RELATIVE ACCURACIES   (USING A-PRIORI WEIGHTS)

JACK                          MONUMENT 14    S= 13989.  SIGMA= .068  LENG.REL.ACCURACY=1:  207020/  103510/   69006
                                                 LENGTH SHIFT= .009  LENG.REL.ACCURACY=1: 1482864

JACK                          IPS 3          S= 17325.  SIGMA= .059  LENG.REL.ACCURACY=1:  293659/  146829/   97886
                                                 LENGTH SHIFT= .008  LENG.REL.ACCURACY=1: 2135917

VALLEY ASTRO                  SALT           S=  8877.  SIGMA= .055  LENG.REL.ACCURACY=1:  162564/   81282/   54188
                                                 LENGTH SHIFT= .004  LENG.REL.ACCURACY=1: 1988700

B.M.G-48                      JACK           S= 32015.  SIGMA= .057  LENG.REL.ACCURACY=1:  557200/  278600/  185733
                                                 LENGTH SHIFT= .001  LENG.REL.ACCURACY=1:41704123

JACK                          B.M.G-48       S= 32015.  SIGMA= .057  LENG.REL.ACCURACY=1:  557200/  278600/  185733
                                                 LENGTH SHIFT= .001  LENG.REL.ACCURACY=1:41704123

JACK                          VALLEY ASTRO   S= 28956.  SIGMA= .062  LENG.REL.ACCURACY=1:  463378/  231689/  154459
                                                 LENGTH SHIFT=-.005  LENG.REL.ACCURACY=1: 5319079

MONUMENT 14                   IPS 3          S=  3340.  SIGMA= .062  LENG.REL.ACCURACY=1:   54033/   27016/   18011
                                                 LENGTH SHIFT=-.001  LENG.REL.ACCURACY=1: 2709366

MONUMENT 14                   VALLEY ASTRO   S= 16584.  SIGMA= .084  LENG.REL.ACCURACY=1:  196538/   98269/   65512
                                                 LENGTH SHIFT=-.013  LENG.REL.ACCURACY=1: 1323191

MONUMENT 14                   B.M.G-48       S= 20399.  SIGMA= .080  LENG.REL.ACCURACY=1:  253811/  126905/   84603
                                                 LENGTH SHIFT=-.004  LENG.REL.ACCURACY=1: 5424531

IPS 3                         OASIS          S=  4967.  SIGMA= .059  LENG.REL.ACCURACY=1:   83892/   41946/   27964
                                                 LENGTH SHIFT=-.008  LENG.REL.ACCURACY=1:  631463

OASIS                         VALLEY ASTRO   S= 11786.  SIGMA= .055  LENG.REL.ACCURACY=1:  213827/  106913/   71275
                                                 LENGTH SHIFT=-.005  LENG.REL.ACCURACY=1: 2463654

VALLEY ASTRO                  B.M.G-48       S=  4278.  SIGMA= .051  LENG.REL.ACCURACY=1:   84331/   42165/   28110
                                                 LENGTH SHIFT= .009  LENG.REL.ACCURACY=1:  472373

B.M.G-48                      SALT           S=  4614.  SIGMA= .051  LENG.REL.ACCURACY=1:   91113/   45556/   30371
                                                 LENGTH SHIFT=-.005  LENG.REL.ACCURACY=1:  958686

DEGREES OF FREEDOM      =   12
VARIANCE SUM            =     70.5
STD.DEV.OF UNIT WEIGHT  =   2.42
VARIANCE OF UNIT WEIGHT =   5.88
END OF PROCESSING
```

APPENDIX C. INERT1 LISTING

```
      IMPLICIT REAL*8(A-H,O-Z)
      LOGICAL LONFLG
      DIMENSION IW(15000),AR(7500)
      EQUIVALENCE(AR(1),IW(1))
      COMMON/CONST/PI,PI2,RAD,LONFLG,ZERO
      COMMON/ELLIP/A,E2
      COMMON/PARMS/ZUPT,NSTA,NRUN,NUNK,ITAB,NOBS
C
      ISPACE=15000
C
C *** DEFINE CONSTANTS AND DEFAULT ELLIPSOID
C
      A=6378137.D0
      E2=6.69438002290341567D-3
      PI2=2.D0*DATAN(1.D0)
      PI=2.D0*PI2
      RAD=180./PI
      ZERO=0.D0
C
C *** DEFINE SCRATCH FILES
C
      IU0=3
      IU02=4
      IUH=7
      IUS=8
      CALL START(IDUMMY,IDUMMY)
C
C *** GET PARAMETER CARD (FIRST RECORD)
C
      CALL GETPRM
C
C *** ALLOCATE STORAGE
C
      NSTAS=3*NSTA
      NUNK=NSTAS+12*NRUN
      NSTATS=12*NRUN
      N1=NUNK+NSTAS
      N2=N1+NUNK
      N3=N2+NSTATS
      I3=N3+N3
C
C *** INITIALIZE PARAMETERS AND SHIFTS
C
      DO 2 I=1,N1
    2 AR(I)=0.D0
C
C *** INITIALIZE NAME TABLE
C
      ITAB=NEWT(0,NSTA,0,8,103)
C
C *** PERFORM ADJUSTMENT
C
      CALL INERT1(AR(1),AR(NUNK+1),AR(N1+1),AR(N2+1),IW(I3+1),ISPACE-I3,
     *            IU0,IU02,IUH,IUS)
C
C *** END OF ADJUSTMENT
C
      WRITE(6,3)
    3 FORMAT('0END OF PROCESSING')
      STOP
      END
      SUBROUTINE GETPRM
C
C *** GET PARAMETER CARD AND CHECK DEFAULT VALUES
C
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*1 CODE
      LOGICAL LONFLG
      COMMON/CONST/PI,PI2,RAD,LONFLG,ZERO
      COMMON/PARMS/ZUPT,NSTA,NRUN,NUNK,ITAB,NOBS
```

```
      COMMON/ELLIP/A,E2
C
      READ(11,1,END=666) ZUPT,NSTA,NRUN,AX,EX,CODE
    1 FORMAT(F5.0,2I5,F10.3,F18.18,A1)
C
      IF(ZUPT.LE.0.D0) ZUPT=220.D0
C
      IF(NSTA.LE.0) THEN
        WRITE(6,2) NSTA
    2   FORMAT('0NSTA ILLEGAL VALUE--',I5)
        CALL FERR
      ENDIF
C
      IF(NRUN.LE.0) THEN
        WRITE(6,3) NRUN
    3   FORMAT('0NRUN ILLEGAL VALUE--',I5)
        CALL FERR
      ENDIF
C
      IF(AX.GT.0.D0) A=AX
C
      IF(EX.GT.0.D0) E2=EX
C
C *** FLAG    BLANK--POSITIVE EAST
C ***      NONBLANK--POSITIVE WEST (U.S.)
C
      IF(CODE.EQ.' ') THEN
        LONFLG=.TRUE.
      ELSE
        LONFLG=.FALSE.
      ENDIF
C
      RETURN
C
  666 WRITE(6,4)
    4 FORMAT('0NO INPUT DATA')
      CALL FERR
C
      RETURN
      END
      SUBROUTINE INERT1(A,SHIFTS,GOOGE,STATS,IW,ISPACE,IUO,IUO2,IUH,IUS)
C
C *** ADJUST INERTIAL DATA
C
      IMPLICIT REAL*8(A-H,O-Z)
      LOGICAL CONVRG,LONFLG
      DIMENSION A(1),IW(1),SHIFTS(1),GOOGE(1),STATS(1)
      COMMON/CONST/PI,PI2,RAD,LONFLG,ZERO
      COMMON/PARMS/ZUPT,NSTA,NRUN,NUNK,ITAB,NOBS
      COMMON/ELLIP/AX,E2
C
C *** HEADING
C
      WRITE(6,10) AX,E2
   10 FORMAT('1',53X,'NATIONAL GEODETIC SURVEY'/' PROGRAM INERT1',
     *       T53,'INERTIAL ADJUSTMENT PROGRAM',T124,'PAGE   1'//
     *       10X,'A=',F12.3/9X,'E2=',F20.18)
C
      IF(LONFLG) THEN
        WRITE(6,11)
   11   FORMAT(9X,'LONGITUDES POSITIVE EAST'//)
      ELSE
        WRITE(6,12)
   12   FORMAT(9X,'LONGITUDES POSITIVE WEST'//)
      ENDIF
C
C *** READ DATA AND FORM OBS EQS AND CONNECTION MATRIX
C *** REORDER FOR MINIMUM PROFILE
C
      CALL FIRST(IUO,IUS,IUH,A,IW,ISPACE)
      TOL=.01D0
      STOL=1.D-9
      ITER=0
      MAXITR=5
```

```
145.      C
146.      C *** FORM NORMALS AND SOLVE
147.      C
148.        100 CALL NORMAL(IUO,IW,ISPACE,GOOGE,ITER,STOL)
149.      C
150.      C *** UPDATE UNKNOWNS AND TEST CONVERGENCE
151.      C
152.            IF(CONVRG(A,SHIFTS,TOL,ITER)) THEN
153.              CALL FINAL(IUO,A,SHIFTS,GOOGE,STATS,STOL)
154.            ELSE
155.              IF(ITER.GE.MAXITR) THEN
156.                WRITE(6,1)
157.          1     FORMAT('0SLOWLY CONVERGING SOLUTION')
158.                RETURN
159.              ELSE
160.                CALL FORMOB(IUO,IUO2,A)
161.                ITER=ITER+1
162.                GO TO 100
163.              ENDIF
164.            ENDIF
165.      C
166.            RETURN
167.            END
168.            SUBROUTINE FIRST(IUO,IUS,IUH,A,IW,ISPACE)
169.      C
170.      C *** READ DATA AND WRITE FIRST OBSERVATION EQUATIONS
171.      C
172.            IMPLICIT REAL*8 (A-H,O-Z)
173.            DIMENSION A(1),IW(1)
174.            INTEGER SIZE
175.            LOGICAL GETCRD
176.            CHARACTER*90 CARD
177.            CHARACTER*1 CC1
178.            COMMON/PARMS/ZUPT,NSTA,NRUN,NUNK,ITAB,NOBS
179.      C
180.            NOBS=0
181.      C
182.      C *** INITIALIZE CONNECTION MATRIX
183.      C
184.            CALL NABGEN(IW,ISPACE,NUNK,IUS,IUH)
185.      C
186.      C *** READ THE DATA
187.      C
188.        100 IF(GETCRD(CARD,CC1)) THEN
189.              IF(CC1.EQ.'D') THEN
190.                CALL INITP(CARD,A)
191.              ELSEIF(CC1.EQ.'K') THEN
192.                CALL FIXEDP(CARD,IUO,A)
193.              ELSEIF(CC1.EQ.'I') THEN
194.                CALL INERTL(CARD,IUO,A)
195.              ELSEIF(CC1.EQ.'N') THEN
196.                CALL FIXPRM(CARD,IUO,A)
197.              ELSE
198.                CALL ADDACC(CARD)
199.              ENDIF
200.              GO TO 100
201.            ENDIF
202.      C
203.      C *** ALL OBSERVATIONS WRITTEN, VERIFY CORRECT # OF STATIONS
204.      C
205.            IF(SIZE(ITAB).NE.NSTA) THEN
206.              WRITE(6,1) SIZE(ITAB),NSTA
207.          1   FORMAT(' 0TABLE SIZE=',I5,' NOT EQUAL TO STATIONS=',I5)
208.              CALL FERR
209.            ENDIF
210.      C
211.      C *** REORDER THE UNKNOWNS (SNAY)
212.      C
213.            IREORD=3
214.            CALL NEWORD(IREORD)
215.      C
216.            RETURN
217.            END
218.            LOGICAL FUNCTION GETCRD(CARD,CC1)
219.      C
```

```
C *** GET A CARD WITH CC1 OF D,K,I,N,Q
C
      CHARACTER*90 CARD
      CHARACTER*1 CC1
C
      GETCRD=.TRUE.
  100 READ(11,1,END=666) CARD
    1 FORMAT(A90)
      CC1=SUBSTR(CARD,1,1)
C
      IF(CC1.EQ.'I'.OR.CC1.EQ.'D'.OR.CC1.EQ.'K'.OR.CC1.EQ.'N'.
     *   OR.CC1.EQ.'Q') RETURN
C
      GO TO 100
C
C *** END OF FILE
C
  666 GETCRD=.FALSE.
C
      RETURN
      END
      SUBROUTINE INITP(CARD,A)
C
C *** PROCESS AN INITIAL POSITION
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(1)
      INTEGER SEEK,SIZE
      CHARACTER*90 CARD
      CHARACTER*32 NAME
      COMMON/PARMS/ZUPT,NSTA,NRUN,NUNK,ITAB,NOBS
C
C *** IF NAME IS DUPLICATE--BOMB||
C
      NAME=SUBSTR(CARD,7,30)
      ISTA=SEEK(ITAB,NAME)
      IF(ISTA.LE.SIZE(ITAB)) THEN
        WRITE(6,1) CARD,ISTA
    1   FORMAT(1X,A90,' ** DUPLICATE NAME',I5)
        CALL FERR
      ENDIF
C
C *** IF NAMES OVERFLOW TABLE--BOMB|
C
      IF(ISTA.GT.NSTA) THEN
        WRITE(6,2) CARD,NSTA
    2   FORMAT(1X,A90,' EXCEEDS MAX SIZE OF',I5)
        CALL FERR
      ENDIF
C
C *** LOAD NAME INTO TABLE
C
      CALL PUTVAL(ITAB,DUMMY)
C
C *** EXTRACT INITIAL VALUES FROM CARD
C
      DECODE(3,CARD) I1,M1,S1,I2,M2,S2,H
    3 FORMAT(36X,I3,I2,F7.5,I4,I2,F7.5,F7.3)
      CALL GETRAD(I1,M1,S1,GLAT)
      CALL GETRAD(I2,M2,S2,GLON)
      A(IUNSTA(ISTA,1))=GLAT
      A(IUNSTA(ISTA,2))=GLON
      A(IUNSTA(ISTA,3))=H
C
C *** LIST THE INPUT
C
      WRITE(6,4) CARD
    4 FORMAT(7X,A90)
C
      RETURN
      END
      SUBROUTINE GETRAD(ID,IM,S,VAL)
C
C *** CONVERT DEG, MIN, SEC TO RADIANS
```

```
C
      IMPLICIT REAL*8(A-H,O-Z)
      LOGICAL LONFLG
      COMMON/CONST/PI,PI2,RAD,LONFLG,ZERO
C
      VAL=DFLOAT(IABS(ID))
      VAL=VAL+DFLOAT(IABS(IM))/60.D0
      VAL=VAL+DABS(S)/3600.0
      VAL=VAL/RAD
      IF(ID.NE.0) THEN
        VAL=DSIGN(VAL,DFLOAT(ID))
      ELSEIF(IM.NE.0) THEN
        VAL=DSIGN(VAL,DFLOAT(IM))
      ELSE
        VAL=DSIGN(VAL,S)
      ENDIF
C
      RETURN
      END
      INTEGER FUNCTION IUNSTA(ISTA,I)
C
C *** DETERMINE UNKNOWN NUMBER OF STATION COORDINATE
C
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/PARMS/ZUPT,NSTA,NRUN,NUNK,ITAB,NOBS
C
C *** STATIONS STORED THREEWISE--LAT,LON,H
C
      IF(ISTA.LE.0.OR.ISTA.GT.NSTA.OR.I.LE.0.OR.I.GT.3) THEN
        WRITE(6,1) ISTA,I,NSTA
    1   FORMAT('0ILLEGAL VALUES IN IUNSTA',3I5)
        CALL FERR
      ELSE
        IUNSTA=(ISTA-1)*3+I
      ENDIF
C
      RETURN
      END
      INTEGER FUNCTION IUNPRM(IRUN,I)
C
C *** DETERMINE UNKNOWN NUMBER OF PARAMETERS
C
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/PARMS/ZUPT,NSTA,NRUN,NUNK,ITAB,NOBS
C
C *** STATIONS STORED THREEWISE THEN BY RUN PARAMETERS STORED TWELVEWISE
C
      IF(IRUN.LE.0.OR.IRUN.GT.NRUN.OR.I.LE.0.OR.I.GT.12) THEN
        WRITE(6,1) IRUN,I,NRUN
    1   FORMAT('0ILLEGAL VALUES IN IUNPRM',3I5)
        CALL FERR
      ELSE
        IUNPRM=3*NSTA+(IRUN-1)*12+I
      ENDIF
C
      RETURN
      END
      SUBROUTINE INVIUN(IUNK,ICODE,I,J)
C
C *** GIVEN AN UNKNOWN INDEX NUMBER, GET STATION/RUN NUMBER
C
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/PARMS/ZUPT,NSTA,NRUN,NUNK,ITAB,NOBS
C
C *** INDEX IS A STATION INDEX
C
      NSTAS=3*NSTA
      IF(IUNK.GT.0.AND.IUNK.LE.NSTAS) THEN
        ICODE=0
        I=(IUNK-1)/3
        J=IUNK-I*3
        I=I+1
C
C *** INDEX IS A RUN INDEX
C
```

```
369.            ELSEIF(IUNK.GT.NSTAS.AND.IUNK.LE.NUNK) THEN
370.              ICODE=1
371.              K=IUNK-NSTAS
372.              I=(K-1)/12
373.              K=K-I*12
374.              I=I+1
375.      C
376.      C *** ILLEGAL INPUT
377.      C
378.            ELSE
379.              WRITE(6,1) IUNK
380.          1   FORMAT('0ILLEGAL VALUE IN INVIUN',I5)
381.              CALL FERR
382.            ENDIF
383.      C
384.            RETURN
385.            END
386.            SUBROUTINE FIXEDP(CARD,IUO,A)
387.      C
388.      C *** CONSTRAIN A POSITION
389.      C
390.            IMPLICIT REAL*8(A-H,O-Z)
391.            CHARACTER*90 CARD
392.            CHARACTER*32 NAME
393.            CHARACTER*13 ALON
394.            CHARACTER*12 ALAT
395.            CHARACTER*7 AH
396.            CHARACTER*6 ASLAT,ASLON,ASH
397.            INTEGER SEEK,SIZE
398.            LOGICAL LONFLG
399.            DIMENSION IUNK(10),C(10),A(1)
400.            COMMON/CONST/PI,PI2,RAD,LONFLG,ZERO
401.            COMMON/PARMS/ZUPT,NSTA,NRUN,NUNK,ITAB,NOBS
402.      C
403.      C *** DEFINE DEFAULT STANDARD ERROR AND A RADIUS OF CURVATURE
404.      C
405.            SIGDEF=.001D0
406.            R=RADIUS(45.D0,PI2)
407.      C
408.      C *** EXTRACT THE FIELDS
409.      C
410.            NAME=SUBSTR(CARD,7,30)
411.            ALAT=SUBSTR(CARD,37,12)
412.            ALON=SUBSTR(CARD,49,13)
413.            AH=SUBSTR(CARD,62,7)
414.            ASLAT=SUBSTR(CARD,69,6)
415.            ASLON=SUBSTR(CARD,75,6)
416.            ASH=SUBSTR(CARD,81,6)
417.      C
418.      C *** CHECK FOR NAME
419.      C
420.            ISTA=SEEK(ITAB,NAME)
421.            IF(ISTA.GT.SIZE(ITAB)) THEN
422.              WRITE(6,1) CARD
423.          1   FORMAT(1X,A90,'** UNKNOWN NAME')
424.              CALL FERR
425.            ENDIF
426.      C
427.      C *** WRITE CONSTRAINT OBSERVATION EQUATIONS
428.      C
429.            IF(ALAT.NE.' ') THEN
430.              KIND=13
431.              DECODE(2,ALAT) I1,M1,S1
432.          2   FORMAT(I3,I2,F7.5)
433.              CALL GETRAD(I1,M1,S1,OBSB)
434.              CALL FORMIU(KIND,ISTA,1,IRUN,IUNK,LENG)
435.              CALL FORMC(KIND,C,IUNK,A,DUMMY,DUMMY)
436.              CALL COMPOB(KIND,OBSO,IUNK,A,DUMMY,DUMMY)
437.      C
438.      C *** SAVE RADIUS OF CURVATURE IN R FOR LONGITUDE CONSTRAINTS
439.      C
440.              R=RADIUS(OBSB,ZERO)
441.              IF(ASLAT.EQ.' ') THEN
442.                SD=SIGDEF/R
```

```
443.              ELSE
444.                DECODE(3,ASLAT) SD
445.        3       FORMAT(F6.3)
446.                SD=DABS(SD)/R
447.              ENDIF
448.              CALL ADCON(IUNK,LENG)
449.              WRITE(IUO) KIND,IUNK,C,OBSO,OBSB,SD,DUMMY,DUMMY,LENG
450.              NOBS=NOBS+1
451.            ENDIF
452.      C
453.            IF(ALON.NE.' ') THEN
454.              KIND=14
455.              DECODE(4,ALON) I2,M2,S2
456.        4     FORMAT(I4,I2,F7.5)
457.              CALL GETRAD(I2,M2,S2,OBSB)
458.              CALL FORMIU(KIND,ISTA,2,IRUN,IUNK,LENG)
459.              CALL FORMC(KIND,C,IUNK,A,DUMMY,DUMMY)
460.              CALL COMPOB(KIND,OBSO,IUNK,A,DUMMY,DUMMY)
461.              IF(ASLON.EQ.' ') THEN
462.                SD=SIGDEF/R
463.              ELSE
464.                DECODE(3,ASLON) SD
465.                SD=DABS(SD)/R
466.              ENDIF
467.              CALL ADCON(IUNK,LENG)
468.              WRITE(IUO) KIND,IUNK,C,OBSO,OBSB,SD,DUMMY,DUMMY,LENG
469.              NOBS=NOBS+1
470.            ENDIF
471.      C
472.            IF(AH.NE.' ') THEN
473.              KIND=15
474.              DECODE(5,AH) H
475.        5     FORMAT(F7.3)
476.              OBSB=H
477.              CALL FORMIU(KIND,ISTA,3,IRUN,IUNK,LENG)
478.              CALL FORMC(KIND,C,IUNK,A,DUMMY,DUMMY)
479.              CALL COMPOB(KIND,OBSO,IUNK,A,DUMMY,DUMMY)
430.              IF(ASH.EQ.' ') THEN
481.                SD=SIGDEF
482.              ELSE
483.                DECODE(3,ASH) SD
484.              ENDIF
485.              CALL ADCON(IUNK,LENG)
486.              WRITE(IUO) KIND,IUNK,C,OBSO,OBSB,SD,DUMMY,DUMMY,LENG
487.              NOBS=NOBS+1
488.            ENDIF
489.      C
490.      C *** LIST THE INPUT
491.      C
492.            WRITE(6,6) NOBS,CARD
493.        6   FORMAT(1X,I5,1X,A90)
494.      C
495.            RETURN
496.            END
497.            DOUBLE PRECISION FUNCTION RADIUS(GLAT,AZ)
498.      C
499.      C *** COMPUTE RADIUS OF CURVATURE ALONG AZIMUTH ON ELLIPSOID
500.      C
501.            IMPLICIT REAL*8(A-H,O-Z)
502.            LOGICAL LONFLG
503.            COMMON/CONST/PI,PI2,RAD,LONFLG,ZERO
504.            COMMON/ELLIP/A,E2
505.      C
506.            SLAT=DSIN(GLAT)
507.            SLAT2=SLAT*SLAT
508.            W=DSQRT(1.D0-E2*SLAT2)
509.      C
510.            IF(AZ.EQ.ZERO) THEN
511.              RADIUS=A*(1.D0-E2)/(W*W*W)
512.            ELSEIF(AZ.EQ.PI2) THEN
513.              RADIUS=A/W
514.            ELSE
515.              SAZ=DSIN(AZ)
516.              CAZ=DCOS(AZ)
517.              SAZ2=SAZ*SAZ
518.              CAZ2=CAZ*CAZ
```

```
519.              EM=A*(1.D0-E2)/(W*W*W)
520.              EN=A/W
521.              RADIUS=1.D0/(SAZ2/EN+CAZ2/EM)
522.            ENDIF
523.      C
524.            RETURN
525.            END
526.            SUBROUTINE FORMIU(KIND,I,J,IRUN,IUNK,LENG)
527.      C
528.      C *** LOAD UNKNOWN NUMBER VECTOR WITH UNKNOWN NUMBERS
529.      C
530.            IMPLICIT REAL*8(A-H,O-Z)
531.            DIMENSION IUNK(10)
532.            DIMENSION LENGS(18)/1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,8,8,10/
533.            COMMON/PARMS/ZUPT,NSTA,NRUN,NUNK,ITAB,NOBS
534.      C
535.      C *** KIND 1 TO 12 ARE SIMPLE PARAMETERS (I=IRUN, J=IPARM)
536.      C
537.            IF(KIND.GE.1.AND.KIND.LE.12.AND.I.GE.1.AND.I.LE.NRUN.AND.
538.           *   J.GE.1.AND.J.LE.12) THEN
539.              IUNK(1)=IUNPRM(I,J)
540.      C
541.      C *** KIND 13 TO 15 ARE COORDINATE CONSTRAINTS (I=ISTA, J=TYPE)
542.      C
543.            ELSEIF(KIND.GE.13.AND.KIND.LE.15.AND.I.GE.1.AND.I.LE.NSTA.
544.           *       AND.J.GE.1.AND.J.LE.3) THEN
545.              IUNK(1)=IUNSTA(I,J)
546.      C
547.      C *** KIND 16 IS DIFFERENTIAL LATITUDE
548.      C
549.            ELSEIF(KIND.EQ.16.AND.I.GE.1.AND.I.LE.NSTA.AND.
550.           *       J.GE.1.AND.J.LE.NSTA) THEN
551.              IUNK(1)=IUNSTA(I,1)
552.              IUNK(2)=IUNSTA(J,1)
553.              IUNK(3)=IUNSTA(I,2)
554.              IUNK(4)=IUNSTA(J,2)
555.              IUNK(5)=IUNPRM(IRUN,1)
556.              IUNK(6)=IUNPRM(IRUN,2)
557.              IUNK(7)=IUNPRM(IRUN,3)
558.              IUNK(8)=IUNPRM(IRUN,4)
559.      C
560.      C *** KIND 17 IS DIFFERENTIAL LONGITUDE
561.      C
562.            ELSEIF(KIND.EQ.17.AND.I.GE.1.AND.I.LE.NSTA.AND.
563.           *       J.GE.1.AND.J.LE.NSTA) THEN
564.              IUNK(1)=IUNSTA(I,1)
565.              IUNK(2)=IUNSTA(J,1)
566.              IUNK(3)=IUNSTA(I,2)
567.              IUNK(4)=IUNSTA(J,2)
568.              IUNK(5)=IUNPRM(IRUN,5)
569.              IUNK(6)=IUNPRM(IRUN,6)
570.              IUNK(7)=IUNPRM(IRUN,7)
571.              IUNK(8)=IUNPRM(IRUN,8)
572.      C
573.      C
574.      C *** KIND 18 IS DIFFERENTIAL HEIGHT
575.      C
576.            ELSEIF(KIND.EQ.18.AND.I.GE.1.AND.I.LE.NSTA.AND.
577.           *       J.GE.1.AND.J.LE.NSTA) THEN
578.              IUNK(1)=IUNSTA(I,1)
579.              IUNK(2)=IUNSTA(J,1)
580.              IUNK(3)=IUNSTA(I,2)
581.              IUNK(4)=IUNSTA(J,2)
582.              IUNK(5)=IUNSTA(I,3)
583.              IUNK(6)=IUNSTA(J,3)
584.              IUNK(7)=IUNPRM(IRUN,9)
585.              IUNK(8)=IUNPRM(IRUN,10)
586.              IUNK(9)=IUNPRM(IRUN,11)
587.              IUNK(10)=IUNPRM(IRUN,12)
588.      C
589.            ELSE
590.              WRITE(6,1) KIND,I,J,IRUN
591.          1   FORMAT('0ILLEGAL VALUES IN FORMIU=',4I10)
592.              CALL FERR
593.            ENDIF
```

```
C
C *** GET LENGTH OF UNKNOWN VECTOR
C
      LENG=LENGS(KIND)
C
      RETURN
      END
      SUBROUTINE FORMC(KIND,C,IUNK,A,T,ST)
C
C *** COMPUTE THE OBS EQUATION COEFFICIENTS FROM THE PARAMETER
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(1),IUNK(10),C(10)
C
C *** KIND 1 TO 12 ARE SIMPLE PARAMETER CONSTRAINTS
C
      IF(KIND.GE.1.AND.KIND.LE.12) THEN
        C(1)=1.D0
C
C *** KIND 13 TO 15 ARE SIMPLE COORDINATE CONSTRAINTS
C
      ELSEIF(KIND.GE.13.AND.KIND.LE.15) THEN
        C(1)=1.D0
C
C *** KIND 16 IS DIFFERENTIAL LATITUDE
C
      ELSEIF(KIND.EQ.16) THEN
        C(2)=1.D0+A(IUNK(5))
        C(1)=-C(2)
        C(3)=A(IUNK(6))+A(IUNK(7))*T
        C(4)=-C(3)
        C(5)=A(IUNK(2))-A(IUNK(1))
        C(6)=-(A(IUNK(4))-A(IUNK(3)))
        C(7)=T*C(6)
        C(8)=ST
C
C *** KIND 17 IS DIFFERENTIAL LONGITUDE
C
      ELSEIF(KIND.EQ.17) THEN
        C(2)=A(IUNK(6))+A(IUNK(7))*T
        C(1)=-C(2)
        C(4)=1.D0+A(IUNK(5))
        C(3)=-C(4)
        C(5)=A(IUNK(4))-A(IUNK(3))
        C(6)=A(IUNK(2))-A(IUNK(1))
        C(7)=T*C(6)
        C(8)=ST
C
C *** KIND 18 IS DIFFERENTIAL HEIGHT
C
      ELSEIF(KIND.EQ.18) THEN
        C(2)=A(IUNK(7))+A(IUNK(9))*T
        C(1)=-C(2)
        C(4)=A(IUNK(8))+A(IUNK(10))*T
        C(3)=-C(4)
        C(5)=-1.D0
        C(6)=1.D0
        C(7)=A(IUNK(2))-A(IUNK(1))
        C(8)=A(IUNK(4))-A(IUNK(3))
        C(9)=T*C(7)
        C(10)=T*C(8)
      ELSE
        WRITE(6,1) KIND
    1   FORMAT('0ILLEGAL KIND IN FORMC =',I5)
        CALL FERR
      ENDIF
C
      RETURN
      END
      SUBROUTINE COMPOB(KIND,OBS0,IUNK,A,T,ST)
C
C *** COMPUTE THE OBSERVATION FROM THE PARAMETERS
C
```

```
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(1),IUNK(10)
C
C *** KIND 1 TO 12 ARE SIMPLE PARAMETER CONSTRAINTS
C
      IF(KIND.GE.1.AND.KIND.LE.12) THEN
        OBS0=A(IUNK(1))
C
C *** KIND 13 TO 15 ARE SIMPLE COORDINATE CONSTRAINTS
C
      ELSEIF(KIND.GE.13.AND.KIND.LE.15) THEN
        OBS0=A(IUNK(1))
C
C *** KIND 16 IS DIFFERENTIAL LATITUDE
C
      ELSEIF(KIND.EQ.16) THEN
        DELLAT=A(IUNK(2))-A(IUNK(1))
        DELLON=A(IUNK(4))-A(IUNK(3))
        E1=A(IUNK(5))*DELLAT
        E2=-A(IUNK(6))*DELLON
        E3=-A(IUNK(7))*T*DELLON
        E4=A(IUNK(8))*ST
        OBS0=DELLAT+E1+E2+E3+E4
C
C *** KIND 17 IS DIFFERENTIAL LONGITUDE
C
      ELSEIF(KIND.EQ.17) THEN
        DELLAT=A(IUNK(2))-A(IUNK(1))
        DELLON=A(IUNK(4))-A(IUNK(3))
        E1=A(IUNK(5))*DELLON
        E2=A(IUNK(6))*DELLAT
        E3=A(IUNK(7))*T*DELLAT
        E4=A(IUNK(8))*ST
        OBS0=DELLON+E1+E2+E3+E4
C
C *** KIND 18 IS DIFFERENTIAL HEIGHT
C
      ELSEIF(KIND.EQ.18) THEN
        DELLAT=A(IUNK(2))-A(IUNK(1))
        DELLON=A(IUNK(4))-A(IUNK(3))
        DELH=A(IUNK(6))-A(IUNK(5))
        E1=A(IUNK(7))*DELLAT
        E2=A(IUNK(8))*DELLON
        E3=A(IUNK(9))*T*DELLAT
        E4=A(IUNK(10))*T*DELLON
        OBS0=DELH+E1+E2+E3+E4
      ELSE
        WRITE(6,1) KIND
    1   FORMAT('0ILLEGAL KIND IN COMPOB=',I5)
        CALL FERR
      ENDIF
C
      RETURN
      END

      SUBROUTINE INERTL(CARD,IUO,A)
C
C *** PROCESS THE INERTIAL OBSERVATIONS
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(1)
      CHARACTER*90 CARD
      CHARACTER*1 CC2
C
      CC2=SUBSTR(CARD,2,1)
C
      IF(CC2.EQ.'R') THEN
        CALL HEADER(CARD)
      ELSEIF(CC2.EQ.'M') THEN
        CALL MARK(CARD,IUO,A)
      ELSEIF(CC2.EQ.'U') THEN
        CALL INUPDT(CARD)
      ENDIF
C
      RETURN
```

```
      END
      SUBROUTINE HEADER(CARD)
C
C *** PROCESS AN INERTIAL RUN HEADER CARD
C
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*90 CARD
      CHARACTER*32 NAME
      INTEGER SEEK,SIZE
      COMMON/OLD/GLAT1,GLON1,H1,ISTA,T0,T1,IRUN
      COMMON/PARMS/ZUPT,NSTA,NRUN,NUNK,ITAB,NOBS
C
C *** EXTRACT THE NECESSARY FIELDS
C
      DECODE(1,CARD) IRUN,NAME,I1,M1,S1,I2,M2,S2,H1,IH,IM,IS
    1 FORMAT(2X,I4,A30,I3,I2,F7.5,I4,I2,F7.5,F7.3,3I2)
C
C *** VERIFY RANGES
C
      IF(IRUN.LE.0.OR.IRUN.GT.NRUN) THEN
        WRITE(6,2) CARD,IRUN,NRUN
    2   FORMAT(1X,A90,'** BAD RUN NUM',2I5)
        CALL FERR
      ENDIF
C
      ISTA=SEEK(ITAB,NAME)
      IF(ISTA.GT.SIZE(ITAB)) THEN
        WRITE(6,3) CARD
    3   FORMAT(1X,A90,'  NON-POS NAME')
        CALL FERR
      ENDIF
C
C *** GET VALUES
C
      T0=(IH*60+IM)*60+IS
      CALL GETRAD(I1,M1,S1,GLAT1)
      CALL GETRAD(I2,M2,S2,GLON1)
C
C *** INITIALIZE TIME VARIABLE
C
      T1=T0
C
C *** LIST THE INPUT
C
      WRITE(6,4) CARD
    4 FORMAT(7X,A90)
C
      RETURN
      END
      SUBROUTINE MARK(CARD,IUO,A)
C
C *** PROCESS AN INERTIAL 'MARK' OBSERVATION
C
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*90 CARD
      CHARACTER*32 NAME
      CHARACTER*5 ASDLAT,ASDLON,ASDH
      INTEGER SEEK,SIZE
      LOGICAL LONFLG
      DIMENSION A(1),IUNK(10),C(10)
      COMMON/PARMS/ZUPT,NSTA,NRUN,NUNK,ITAB,NOBS
      COMMON/CONST/PI,PI2,RAD,LONFLG,ZERO
      COMMON/OLD/GLAT1,GLON1,H1,ISTA,T0,T1,IRUN
C
C *** DEFAULT DELTA POSITION (METERS)
C
      SDDEFL=0.100D0
C
C *** EXTRACT THE FIELDS
C
      DECODE(1,CARD) NAME,I1,M1,S1,I2,M2,S2,H2,IH,IM,IS,
     *               ASDLAT,ASDLON,ASDH,SDLAT,SDLON,SDH
    1 FORMAT(6X,A30,I3,I2,F7.5,I4,I2,F7.5,F7.3,3I2,3A5,T75,3F5.3)
C
```

```
815.            JSTA=SEEK(ITAB,NAME)
816.            IF(JSTA.GT.SIZE(ITAB)) THEN
817.              WRITE(6,2) CARD
818.        2     FORMAT(1X,A90,' NON-POS NAME')
819.              CALL FERR
820.            ENDIF
821.      C
822.      C *** TRANSLATE FIELDS
823.      C
824.            T2=(IH*60+IM)*60+IS
825.            CALL GETRAD(I1,M1,S1,GLAT2)
826.            CALL GETRAD(I2,M2,S2,GLON2)
827.      C
828.      C *** COMPUTE MEAN ELAPSED TIME (T) AND SUM OF SQUARE INTERVALS (ST)
829.      C
830.            T=((T2-T0)+(T1-T0))/2000.
831.            NT=INT((T2-T1)/ZUPT)
832.            RT=T2-T1-NT*ZUPT
833.            ST=(ZUPT*ZUPT*NT+RT*RT)/1000.
834.      C
835.      C *** FORM AND WRITE DIFFERENTIAL LAT OBS EQ
836.      C
837.            KIND=16
838.      C
839.            CALL FORMIU(KIND,ISTA,JSTA,IRUN,IUNK,LENG)
840.            CALL FORMC(KIND,C,IUNK,A,T,ST)
841.            CALL COMPOB(KIND,OBS0,IUNK,A,T,ST)
842.      C
843.            OBSB=GLAT2-GLAT1
844.      C
845.            IF(ASDLAT.EQ.' ') THEN
846.              SD=SDDEFL/RADIUS((GLAT1+GLAT2)/2.D0,ZERO)
847.            ELSE
848.              SD=SDLAT/RADIUS((GLAT1+GLAT2)/2.D0,ZERO)
849.            ENDIF
850.      C
851.            CALL ADCON(IUNK,LENG)
852.            WRITE(IUO) KIND,IUNK,C,OBS0,OBSB,SD,T,ST,LENG
853.            NOBS=NOBS+1
854.      C
855.      C *** FORM AND WRITE DIFFERENTIAL LONG OBS EQ
856.      C
857.            KIND=17
858.      C
859.            CALL FORMIU(KIND,ISTA,JSTA,IRUN,IUNK,LENG)
860.            CALL FORMC(KIND,C,IUNK,A,T,ST)
861.            CALL COMPOB(KIND,OBS0,IUNK,A,T,ST)
862.      C
863.            OBSB=GLON2-GLON1
864.      C
865.            IF(ASDLON.EQ.' ') THEN
866.              SD=SDDEFL/RADIUS((GLAT1+GLAT2)/2.D0,PI2)
867.            ELSE
868.              SD=SDLON/RADIUS((GLAT1+GLAT2)/2.D0,PI2)
869.            ENDIF
870.      C
871.            CALL ADCON(IUNK,LENG)
872.            WRITE(IUO) KIND,IUNK,C,OBS0,OBSB,SD,T,ST,LENG
873.            NOBS=NOBS+1
874.      C
875.      C *** FORM AND WRITE DIFFERENTIAL HEIGHT OBS EQ
876.      C
877.            KIND=18
878.      C
879.            CALL FORMIU(KIND,ISTA,JSTA,IRUN,IUNK,LENG)
880.            CALL FORMC(KIND,C,IUNK,A,T,ST)
881.            CALL COMPOB(KIND,OBS0,IUNK,A,T,ST)
882.      C
883.            OBSB=H2-H1
884.      C
885.            IF(ASDH.EQ.' ') THEN
886.              SD=SDDEFL
887.            ELSE
888.              SD=SDH
```

```
      ENDIF
C
      CALL ADCON(IUNK,LENG)
      WRITE(IUO) KIND,IUNK,C,OBSO,OBSB,SD,T,ST,LENG
      NOBS=NOBS+1
C
C *** UPDATE THE COORDINATES AND TIME
C
      ISTA=JSTA
      GLAT1=GLAT2
      GLON1=GLON2
      H1=H2
      T1=T2
C
C *** LIST THE INPUT
C
      CALL GETM(GLAT2,GLON2,A(IUNSTA(JSTA,1)),A(IUNSTA(JSTA,2)),GMLAT,
     *          GMLON)
      WRITE(6,3) NOBS,CARD,GMLAT,GMLON
    3 FORMAT(1X,I5,1X,A90,2F10.3)
C
      RETURN
      END
      SUBROUTINE GETM(GLA2,GLO2,GLA1,GLO1,GMLA,GMLO)
C
C *** DISPLAY RAW DIFFERENCES
C
      IMPLICIT REAL*8(A-H,O-Z)
      LOGICAL FLAG
      COMMON/CONST/PI,PI2,RAD,FLAG,ZERO
C
      IF(GLA1.NE.GLA2) THEN
        CALL ELIPIN(GLA1,GLO1,GLA2,GLO1,GMLA,FAZ,BAZ)
        IF(FAZ.GT.PI2.AND.FAZ.LT.3.D0*PI2) GMLA=-GMLA
      ELSE
        GMLA=0.D0
      ENDIF
C
      IF(GLO1.NE.GLO2) THEN
        CALL ELIPIN(GLA1,GLO1,GLA1,GLO2,GMLO,FAZ,BAZ)
        IF(FAZ.GT.PI) GMLO=-GMLO
      ELSE
        GMLO=0.D0
      ENDIF
C
      RETURN
      END
      SUBROUTINE INUPDT(CARD)
C
C *** PROCESS AN INERTIAL 'UPDATE' OBSERVATION
C
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*90 CARD
      COMMON/OLD/GLAT1,GLON1,H1,ISTA,T0,T1,IRUN
C
C *** EXTRACT THE UPDATE FIELDS WHICH PASS THRU COMMON /OLD/
C
      DECODE(1,CARD) I1,M1,S1,I2,M2,S2,H1
    1 FORMAT(36X,I3,I2,F7.5,I4,I2,F7.5,F7.3)
C
      CALL GETRAD(I1,M1,S1,GLAT1)
      CALL GETRAD(I2,M2,S2,GLON1)
C
C *** LIST THE INPUT
C
      WRITE(6,2) CARD
    2 FORMAT(7X,A90)
C
      RETURN
      END
      SUBROUTINE FIXPRM(CARD,IUO,A)
C
C *** CONSTRAIN A PARAMETER
C
      IMPLICIT REAL*8(A-H,O-Z)
```

```
      CHARACTER*90 CARD
      CHARACTER*12 VALUE
      CHARACTER*10 SIGMA
      DIMENSION IUNK(10),C(10),A(1)
      COMMON/PARMS/ZUPT,NSTA,NRUN,NUNK,ITAB,NOBS
C
C *** EXTRACT THE FIELDS
C
      VALUE=SUBSTR(CARD,9,12)
      SIGMA=SUBSTR(CARD,21,10)
      DECODE(1,CARD) IRUN,IPARM,OBSB,SD
    1 FORMAT(2X,I4,I2,F12.10,F10.8)
C
C *** MAKE SURE CORRECT RANGE ON QUANTITIES
C
      IF(IRUN.LE.0.OR.IRUN.GT.NRUN.OR.IPARM.LE.0.OR.IPARM.GT.12) THEN
        WRITE(6,2) IRUN,IPARM,NRUN
    2   FORMAT('0ILLEGAL VALUES IN FIXPRM',3I5)
        CALL FERR
      ENDIF
      IF(VALUE.EQ.' '.OR.SIGMA.EQ.' ') THEN
        WRITE(6,3) CARD
    3   FORMAT(1X,A90,' ILLEGAL VALUE OR SIGMA')
        CALL FERR
      ENDIF
C
C *** WRITE THE CONSTRAINT OBSERVATION EQ
C
      KIND=IPARM
      CALL FORMIU(KIND,IRUN,IPARM,IDUMMY,IUNK,LENG)
      CALL FORMC(KIND,C,IUNK,A,DUMMY,DUMMY)
      CALL COMPOB(KIND,OBS0,IUNK,A,DUMMY,DUMMY)
      CALL ADCON(IUNK,LENG)
      WRITE(IUO) KIND,IUNK,C,OBS0,OBSB,SD,DUMMY,DUMMY,LENG
      NOBS=NOBS+1
C
C *** LIST THE INPUT
C
      WRITE(6,4) NOBS,CARD
    4 FORMAT(1X,I5,1X,A90)
C
      RETURN
      END
      SUBROUTINE ADDACC(CARD)
C
C *** ADD CONNECTIONS FOR ACCURACIES
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION IU(6)
      INTEGER SEEK,SIZE
      CHARACTER*90 CARD
      CHARACTER*32 NAME1,NAME2
      COMMON/PARMS/ZUPT,NSTA,NRUN,NUNK,ITAB,NOBS
C
      LENG=6
C
C *** INSURE BOTH NAMES POSITONED
C
      NAME1=SUBSTR(CARD,7,30)
      NAME2=SUBSTR(CARD,37,30)
      I=SEEK(ITAB,NAME1)
      J=SEEK(ITAB,NAME2)
      MAX=SIZE(ITAB)
C
      IF(I.LE.MAX.AND.J.LE.MAX) THEN
        CALL CONECG(I,J,IU)
        CALL ADCON(IU,LENG)
      ENDIF
C
C *** LIST THE INPUT
C
      WRITE(6,1) CARD
    1 FORMAT(7X,A90)
C
      RETURN
```

```
      END
      SUBROUTINE CONECG(I,J,IU)
C
C *** FILL CONNECTION MATRIX FOR TWO STATIONS
C
      DIMENSION IU(6)
C
      IU(1)=IUNSTA(I,1)
      IU(2)=IUNSTA(I,2)
      IU(3)=IUNSTA(I,3)
      IU(4)=IUNSTA(J,1)
      IU(5)=IUNSTA(J,2)
      IU(6)=IUNSTA(J,3)
C
      RETURN
      END
      SUBROUTINE NORMAL(IUO,IW,ISPACE,GOOGE,ITER,STOL)
C
C *** FORM AND SOLVE NORMAL EQUATIONS
C
      IMPLICIT REAL*8(A-H,O-Z)
      LOGICAL FLAG
      DIMENSION IW(1),GOOGE(1),IUNK(10),C(10)
      COMMON/PARMS/ZUPT,NSTA,NRUN,NUNK,ITAB,NOBS
C
C *** REWIND OBSERVATION EQUATIONS AND INITIALIZE BIBB
C
      REWIND IUO
      CALL BIBB(ISPACE,IW,NUNK,ITER)
C
C *** LOAD OBSERVATION EQUATIONS AND FORM NORMALS
C
      DO 1 I=1,NOBS
      READ(IUO) KIND,IUNK,C,OBSO,OBSB,SD,TMEAN,SSQT,LENG
      EL=OBSB-OBSO
      VAR=SD*SD
      P=1.D0/VAR
    1 CALL ADOBS(IUNK,C,LENG,EL,P)
      CALL FLUSHQ
C
C *** INITIALIZE GOOGE NUMBERS
C
      DO 2 I=1,NUNK
      GOOGE(I)=ELEM(I,I,FLAG)
      IF(.NOT.FLAG) THEN
        WRITE(6,3) I
    3   FORMAT('0FATAL ERROR IN NORMALS',I5)
        CALL FERR
      ENDIF
    2 CONTINUE
C
C *** REDUCE THE NORMALS
C
      CALL SETEL(IDUMMY,0)
      CALL REDUCE(NSING,IPOINT,STOL,IDUMMY)
      CALL SETEL(IDUMMY,0)
C
C *** TEST FOR SINGULARITY
C
      IF(NSING.GT.0) CALL SINGUL(IW,NSING,IPOINT,STOL)
C
C *** NO SINGULARITY -- COMPUTE GOOGE NUMBERS
C
      DO 4 I=1,NUNK
      TEMP=ELEM(I,I,FLAG)
C
      IF(.NOT.FLAG) THEN
        WRITE(6,5) I
    5   FORMAT('0 FATAL ERROR DOWN IN NORMALS',I5)
        CALL FERR
      ENDIF
C
    4 GOOGE(I)=1.D0/(GOOGE(I)*TEMP*TEMP)
C
C *** COMPUTE THE SOLUTION
```

```
C
      CALL SETEL(IDUMMY,0)
      CALL SOLVE(NSING,IPOINT,STOL,IDUMMY)
      CALL SETEL(IDUMMY,0)
C
      RETURN
      END
      SUBROUTINE SINGUL(IW,NSING,IPOINT,STOL)
C
C *** FATAL TERMINATE DUE TO SINGULARITY
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION IW(1)
C
      WRITE(6,1) STOL
    1 FORMAT('0THE FOLLOWING UNKNOWNS FALL BELOW TOLERANCE OF ',D8.2)
C
      DO 2 I=1,NSING
    2 WRITE(6,3) IW(IPOINT+I-1)
    3 FORMAT(10X,I5)
C
      WRITE(6,4)
    4 FORMAT('0THE FOLLOWING DUMP IS INTENTIONAL')
      CALL FERR
C
      RETURN
      END
      LOGICAL FUNCTION CONVRG(A,SHIFTS,TOL,ITER)
C
C *** UPDATE THE UNKNOWNS AND TEST FOR CONVERGENCE
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(1),SHIFTS(1)
      LOGICAL FLAG,LONFLG
      COMMON/CONST/PI,PI2,RAD,LONFLG,ZERO
      COMMON/PARMS/ZUPT,NSTA,NRUN,NUNK,ITAB,NOBS
C
      SSQ=0.D0
      N3=NSTA*3
C
C *** LOOP OVER ALL UNKNOWNS
C
      DO 2 I=1,NUNK
      X=ELEM(I,NUNK+1,FLAG)
      IF(.NOT.FLAG) THEN
        WRITE(6,1)
    1   FORMAT('0PROGRAMMER ERROR IN CONVRG')
        CALL FERR
      ENDIF
C
C *** UPDATE UNKNOWNS
C
      A(I)=A(I)+X
C
C *** ACCUMULATE SUM OF SQUARES AT STATIONS AND SHIFTS
C
      CALL INVIUN(I,ICODE,ISTA,ITYPE)
      IF(ICODE.EQ.0) THEN
        SHIFTS(I)=SHIFTS(I)+X
        IF(ITYPE.EQ.1) THEN
          X=X*RADIUS(A(I),ZERO)
        ELSEIF(ITYPE.EQ.2) THEN
          X=X*RADIUS(A(I-1),PI2)
        ENDIF
        SSQ=SSQ+X*X
      ENDIF
    2 CONTINUE
C
C *** END OF LOOP OVER UNKNOWNS
C
C *** TEST FOR CONVERGENCE
C
      VAL=SQRT(SSQ/N3)
      IF(VAL.LE.TOL) THEN
        CONVRG=.TRUE.
```

```
1189.          ELSE
1190.            CONVRG=.FALSE.
1191.          ENDIF
1192.     C
1193.     C *** PRINT RESIDUALS
1194.     C
1195.          WRITE(6,3) ITER,VAL
1196.        3 FORMAT('0AT ITERATION #',I2,' THE RMS CORRECTION IS ',F17.3,
1197.         *       ' METERS')
1198.     C
1199.          RETURN
1200.          END
1201.          SUBROUTINE FINAL(IUO,A,SHIFTS,GOOGE,STATS,STOL)
1202.     C
1203.     C *** LIST THE ADJUSTMENT RESULTS
1204.     C
1205.          IMPLICIT REAL*8(A-H,O-Z)
1206.          LOGICAL FLAG
1207.          DIMENSION A(1),SHIFTS(1),GOOGE(1),STATS(1)
1208.          COMMON/PARMS/ZUPT,NSTA,NRUN,NUNK,ITAB,NOBS
1209.     C
1210.     C *** GET VARIANCE OF UNIT WEIGHT (A-PRIORI IS 1.0)
1211.     C
1212.          SUMPVV=ELEM(NUNK+1,NUNK+1,FLAG)
1213.          IDOF=NOBS-NUNK
1214.          IF(IDOF.EQ.0) THEN
1215.            VARUWT=SUMPVV
1216.          ELSE
1217.            VARUWT=SUMPVV/IDOF
1218.          ENDIF
1219.          SIGUWT=DSQRT(VARUWT)
1220.     C
1221.     C *** INVERT WITHIN PROFILE
1222.     C
1223.          CALL SETEL(IDUMMY,0)
1224.          CALL INVRSE(NSING,IPOINT,STOL,IDUMMY)
1225.          CALL SETEL(IDUMMY,0)
1226.     C
1227.     C *** LIST ADJUSTED POSITIONS AND PARAMETERS AND POSITION SHIFTS
1228.     C
1229.          CALL ADJPOS(A,SHIFTS,GOOGE)
1230.     C
1231.     C *** LIST ADJUSTED OBSERVATIONS AND RESIDUALS
1232.     C
1233.          CALL RESID(IUO,A,STATS)
1234.     C
1235.     C *** COMPUTE ACCURACIES AND PRINT VARIANCE
1236.     C
1237.          REWIND 11
1238.          CALL ACCUR(A,SHIFTS,SIGUWT,SUMPVV,IDOF,VARUWT)
1239.     C
1240.          RETURN
1241.          END
1242.          SUBROUTINE ADJPOS(A,SHIFTS,GOOGE)
1243.     C
1244.     C *** LIST ADJUSTED POSITIONS AND PARAMETERS
1245.     C
1246.          IMPLICIT REAL*8(A-H,O-Z)
1247.          LOGICAL FLAG,LONFLG
1248.          CHARACTER*32 NAME
1249.          DIMENSION A(1),SHIFTS(1),GOOGE(1)
1250.          COMMON/PARMS/ZUPT,NSTA,NRUN,NUNK,ITAB,NOBS
1251.          COMMON/CONST/PI,PI2,RAD,LONFLG,ZERO
1252.     C
1253.     C *** HEADING
1254.     C
1255.          WRITE(6,1)
1256.        1 FORMAT('1ADJUSTED POSITONS'//T43,'LATITUDE',T60,'LONGITUDE',T73,
1257.         *'HEIGHT',T83,'STD.DEV.(METERS)',T114,'GOOGE'/)
1258.     C
1259.     C *** GET POSITIONS
1260.     C
```

```
1261.          DO 2 ISTA=1,NSTA
1262.          CALL SETCUR(ITAB,ISTA)
1263.          CALL GETKEY(ITAB,NAME)
1264.          IUNK1=IUNSTA(ISTA,1)
1265.          IUNK2=IUNSTA(ISTA,2)
1266.          IUNK3=IUNSTA(ISTA,3)
1267.    C
1268.          CALL GETDMS(A(IUNK1),ID1,IM1,S1)
1269.          CALL GETDMS(A(IUNK2),ID2,IM2,S2)
1270.          H=A(IUNK3)
1271.    C
1272.          SD1=DSQRT(ELEM(IUNK1,IUNK1,FLAG))*RADIUS(A(IUNK1),ZERO)
1273.          SD2=DSQRT(ELEM(IUNK2,IUNK2,FLAG))*RADIUS(A(IUNK1),PI2)
1274.          SD3=DSQRT(ELEM(IUNK3,IUNK3,FLAG))
1275.    C
1276.    C *** LIST POSITONS, PRECISION, CONDITIONING
1277.    C
1278.        2 WRITE(6,3) ISTA,NAME,ID1,IM1,S1,ID2,IM2,S2,H,SD1,SD2,SD3,
1279.         *           GOOGE(IUNK1),GOOGE(IUNK2),GOOGE(IUNK3)
1280.        3 FORMAT(I4,1X,A30,2I3,F9.5,I6,I3,F9.5,F10.3,2X,3F6.3,2X,
1281.         *       3(1PD9.1))
1282.    C
1283.    C *** PARAMETER HEADING
1284.    C
1285.          WRITE(6,4)
1286.        4 FORMAT('1ADJUSTED PARAMETERS'/)
1287.    C
1288.    C *** LIST PARAMETERS, PRECISION, CONDITIONING
1289.    C
1290.          DO 5 IRUN=1,NRUN
1291.          WRITE(6,6) IRUN
1292.        6 FORMAT('0RUN=',I3//T14,'VALUE',T21,'STD.DEV.',T32,'VAL/SIG',
1293.         *       T44,'GOOGE'/)
1294.          DO 7 I=1,12
1295.          IUNK=IUNPRM(IRUN,I)
1296.          SIGMA=DSQRT(ELEM(IUNK,IUNK,FLAG))
1297.        7 WRITE(6,8) I,A(IUNK),SIGMA,A(IUNK)/SIGMA,GOOGE(IUNK)
1298.        8 FORMAT(I3,5X,2(1PD10.2),0PF10.1,1PD10.2)
1299.        5 WRITE(6,9)
1300.        9 FORMAT('0')
1301.    C
1302.    C *** POSITION SHIFT HEADING
1303.    C
1304.          WRITE(6,10)
1305.       10 FORMAT('1ADJUSTED POSITION SHIFTS'//T47,'LATITUDE',T68,
1306.         *       'LONGITUDE',T84,'AZM.',T90,'HORIZ.',T99,'ELEV.',
1307.         *       T107,'TOTAL'/T45,'SEC',T52,'METERS',T66,'SEC',T74,
1308.         *       'METERS',T85,'DEG',T90,'METERS',T99,'METER',T107,'METER'/)
1309.    C
1310.    C *** LIST POSITION SHIFTS
1311.    C
1312.          DO 11 ISTA=1,NSTA
1313.          CALL SETCUR(ITAB,ISTA)
1314.          CALL GETKEY(ITAB,NAME)
1315.          SLATR=SHIFTS(IUNSTA(ISTA,1))
1316.          SLONR=SHIFTS(IUNSTA(ISTA,2))
1317.          SH=SHIFTS(IUNSTA(ISTA,3))
1318.          SLATS=SLATR*RAD*3600.D0
1319.          SLONS=SLONR*RAD*3600.D0
1320.          GLA1=A(IUNSTA(ISTA,1))
1321.          GLO1=A(IUNSTA(ISTA,2))
1322.          CALL GETM(GLA1,GLO1,GLA1-SLATR,GLO1-SLONR,SLATM,SLONM)
1323.          SLATM2=SLATM*SLATM
1324.          SLONM2=SLONM*SLONM
1325.          SHFTXY=DSQRT(SLATM2+SLONM2)
1326.          IF(.NOT.LONFLG) SLONM=-SLONM
1327.          IF(DABS(SLATM).LE.1.D-3.OR.DABS(SLONM).LE.1.D-3) THEN
1328.            ISHFTZ=0
1329.          ELSE
1330.            ISHFTZ=DATAN2(SLONM,SLATM)*RAD
1331.          ENDIF
1332.          SHFTOT=DSQRT(SLATM2+SLONM2+SH*SH)
1333.       11 WRITE(6,12) ISTA,NAME,SLATS,SLATM,SLONS,SLONM,ISHFTZ,SHFTXY,
1334.         *            SH,SHFTOT
```

```
   12 FORMAT(I3,2X,A30,2(F14.5,F8.3),I8,3F8.3)
C
      RETURN
      END
      SUBROUTINE GETDMS(VAL,ID,IM,S)
C
C *** CONVERT RADIANS TO DEG, MIN, SEC
C
      IMPLICIT REAL*8(A-H,O-Z)
      LOGICAL LONFLG
      COMMON/CONST/PI,PI2,RAD,LONFLG,ZERO
C
      S=DABS(VAL*RAD)
      ID=IDINT(S)
      S=(S-ID)*60.D0
      IM=IDINT(S)
      S=(S-IM)*60.D0
      IF(ID.NE.0) THEN
        ID=ISIGN(ID,IDINT(VAL*RAD))
      ELSEIF(IM.NE.0) THEN
        IM=ISIGN(IM,IDINT(VAL*RAD*60.D0))
      ELSE
        S=DSIGN(S,VAL)
      ENDIF
C
      RETURN
      END
      SUBROUTINE RESID(IUO,A,STATS)
C
C *** LIST ADJUSTED OBSERVATIONS AND RESIDUALS
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(1),IUNK(10),C(10),STATS(1)
      CHARACTER*32 NAME1,NAME2
      LOGICAL LONFLG
      COMMON/PARMS/ZUPT,NSTA,NRUN,NUNK,ITAB,NOBS
      COMMON/CONST/PI,PI2,RAD,LONFLG,ZERO
C
C *** DEFAULT INITIAL RADIUS OF CURVATURE
C
      R=RADIUS(45.D0,PI2)
C
C *** HEADING
C
      WRITE(6,1)
    1 FORMAT('1RESIDUALS'/T24,'COMPUTED',T42,'OBSERVED',T57,'V=C-O',
     *T67,'V/SD'/T54,'SEC.',T60,'METER')
C
C *** INITIALIZE RESIDUAL STATISTICS
C
      CALL RSINIT(STATS)
C
C *** LOOP OVER THE OBSERVATION EQUATIONS
C
      REWIND IUO
      DO 100 IOBS=1,NOBS
      READ(IUO) KIND,IUNK,C,OBSO,OBSB,SD,TMEAN,SSQT,LENG
C
C *** GET COMPUTED OBSERVATION
C
      CALL FORMC(KIND,C,IUNK,A,TMEAN,SSQT)
      CALL COMPOB(KIND,OBSO,IUNK,A,TMEAN,SSQT)
C
C *** RESIDUAL=COMPUTED-OBSERVED
C
      V=OBSO-OBSB
      VSD=V/SD
C
C *** LIST THE RESIDUAL
C
      IF(KIND.GE.1.AND.KIND.LE.12) THEN
        CALL INVIUN(IUNK(1),IDUM1,IRUN,IDUM2)
        WRITE(6,12) IOBS,IRUN,OBSO,OBSB,V,VSD
   12   FORMAT(1X,I5,I3,' PARM.     ',2E20.10,E20.3,F10.1)
```

```
1409.          ELSEIF(KIND.EQ.13) THEN
1410.            R=RADIUS(OBSB,ZERO)
1411.            VSEC=V*RAD*3600.D0
1412.            VMET=V*R
1413.            CALL GETDMS(OBS0,ID1,IM1,S1)
1414.            CALL GETDMS(OBSB,ID2,IM2,S2)
1415.            CALL GETNAM(IUNK(1),NAME1)
1416.            IRUN=0
1417.            WRITE(6,13) IOBS,ID1,IM1,S1,ID2,IM2,S2,VSEC,VMET,VSD,NAME1
1418.       13   FORMAT('0',I4,'    LAT.    ',I4,I3,F9.5,I5,I3,F9.5,F8.5,F7.3,
1419.         *         F5.1,1X,A50)
1420.          ELSEIF(KIND.EQ.14) THEN
1421.            VSEC=V*RAD*3600.D0
1422.            VMET=V*R
1423.            CALL GETDMS(OBS0,ID1,IM1,S1)
1424.            CALL GETDMS(OBSB,ID2,IM2,S2)
1425.            CALL GETNAM(IUNK(1),NAME1)
1426.            IRUN=0
1427.            WRITE(6,14) IOBS,ID1,IM1,S1,ID2,IM2,S2,VSEC,VMET,VSD,NAME1
1428.       14   FORMAT(I5,'    LON.    ',I4,I3,F9.5,I5,I3,F9.5,F8.5,F7.3,
1429.         *         F5.1,1X,A30)
1430.          ELSEIF(KIND.EQ.15) THEN
1431.            CALL GETNAM(IUNK(1),NAME1)
1432.            IRUN=0
1433.            WRITE(6,15) IOBS,OBS0,OBSB,V,VSD,NAME1
1434.       15   FORMAT(I5,'    HT.     ',F16.3,F17.3,F15.3,F5.1,1X,A30)
1435.          ELSEIF(KIND.EQ.16) THEN
1436.            VSEC=V*RAD*3600.D0
1437.            R=RADIUS((A(IUNK(1))+A(IUNK(2)))/2.D0,ZERO)
1438.            VMET=V*R
1439.            CALL GETDMS(OBS0,ID1,IM1,S1)
1440.            CALL GETDMS(OBSB,ID2,IM2,S2)
1441.            CALL GETNAM(IUNK(1),NAME1)
1442.            CALL GETNAM(IUNK(2),NAME2)
1443.            CALL INVIUN(IUNK(8),IDUM1,IRUN,IDUM2)
1444.            WRITE(6,16) IOBS,IRUN,ID1,IM1,S1,ID2,IM2,S2,VSEC,VMET,VSD,NAME1,
1445.         *              NAME2
1446.       16   FORMAT('0',I4,I3,' DEL LAT.',I4,I3,F9.5,I5,I3,F9.5,F8.5,F7.3,
1447.         *         F5.1,2(1X,A30))
1448.          ELSEIF(KIND.EQ.17) THEN
1449.            VSEC=V*RAD*3600.D0
1450.            R=RADIUS((A(IUNK(1))+A(IUNK(2)))/2.D0,PI2)
1451.            VMET=V*R
1452.            CALL GETDMS(OBS0,ID1,IM1,S1)
1453.            CALL GETDMS(OBSB,ID2,IM2,S2)
1454.            CALL GETNAM(IUNK(1),NAME1)
1455.            CALL GETNAM(IUNK(2),NAME2)
1456.            CALL INVIUN(IUNK(8),IDUM1,IRUN,IDUM2)
1457.            WRITE(6,17) IOBS,IRUN,ID1,IM1,S1,ID2,IM2,S2,VSEC,VMET,VSD,NAME1,
1458.         *              NAME2
1459.       17   FORMAT(I5,I3,' DEL LON.',I4,I3,F9.5,I5,I3,F9.5,F8.5,F7.3,
1460.         *         F5.1,2(1X,A30))
1461.          ELSE
1462.            CALL GETNAM(IUNK(1),NAME1)
1463.            CALL GETNAM(IUNK(2),NAME2)
1464.            CALL INVIUN(IUNK(8),IDUM1,IRUN,IDUM2)
1465.            WRITE(6,18) IOBS,IRUN,OBS0,OBSB,V,VSD,NAME1,NAME2
1466.       18   FORMAT(I5,I3,' DEL HT. ',F16.3,F17.3,F15.3,F5.1,2(1X,A30))
1467.          ENDIF
1468.     C
1469.     C *** ACCUMULATE RESIDUAL STATISTICS
1470.     C
1471.          CALL RSTAT(V,VSD,R,IOBS,KIND,IRUN,STATS)
1472.      100 CONTINUE
1473.     C
1474.     C *** LIST RESIDUAL STATISTICS
1475.     C
1476.          CALL RSOUT(STATS)
1477.     C
1478.          RETURN
1479.          END
1480.          SUBROUTINE GETNAM(IUNK,NAME)
1481.     C
1482.     C *** GET NAME FOR AN UNKNOWN INDEX NUMBER
```

```
C
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*32 NAME
      COMMON/PARMS/ZUPT,NSTA,NRUN,NUNK,ITAB,NOBS
C
      CALL INVIUN(IUNK,ICODE,I,J)
      IF(ICODE.NE.0) THEN
        WRITE(6,1) IUNK
    1   FORMAT('0ILLEGAL VALUE IN GETNAM',I5)
        CALL FERR
      ELSE
        CALL SETCUR(ITAB,I)
        CALL GETKEY(ITAB,NAME)
      ENDIF
C
      RETURN
      END
      SUBROUTINE RSINIT(STATS)
C
C *** INITIALIZE RESIDUAL STATISTICS
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION STATS(1)
      COMMON/PARMS/ZUPT,NSTA,NRUN,NUNK,ITAB,NOBS
      COMMON/RESTAT/VSD20(20),I20(20),N0,N1,N2,N3,N4,VMAX,VMIN,
     * VSDMAX,VSDMIN,VSUM,VSDSUM,VSD21,VSD22,VSD23,VSD24,VABS1,VABS2,
     * VABS3,VABS4
C
      DO 1 I=1,20
      VSD20(I)=0.D0
    1 I20(I)=0
C
      NSTATS=12*NRUN
      DO 2 I=1,NSTATS
    2 STATS(I)=0.D0
C
      N0=0
      N1=0
      N2=0
      N3=0
      N4=0
C
      VMAX=-1.D100
      VMIN=1.D100
      VSDMAX=-1.D100
      VSDMIN=1.D100
      VSUM=0.D0
      VSDSUM=0.D0
C
      VSD21=0.D0
      VSD22=0.D0
      VSD23=0.D0
      VSD24=0.D0
C
      VABS1=0.D0
      VABS2=0.D0
      VABS3=0.D0
      VABS4=0.D0
C
      RETURN
      END
      SUBROUTINE RSTAT(V,VSD,R,IOBS,KIND,IRUN,STATS)
C
C *** ACCUMULATE RESIDUAL STATISTICS
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION STATS(1)
      COMMON/RESTAT/VSD20(20),I20(20),N0,N1,N2,N3,N4,VMAX,VMIN,
     * VSDMAX,VSDMIN,VSUM,VSDSUM,VSD21,VSD22,VSD23,VSD24,VABS1,VABS2,
     * VABS3,VABS4
C
C *** ACCUMULATE NO-CHECK
C
      IF(V.EQ.0.D0) N0=N0+1
```

```
1557.      C
1558.      C *** ACCUMULATE EXTREMA
1559.      C
1560.            IF(V.GT.VMAX) VMAX=V
1561.            IF(VSD.GT.VSDMAX) VSDMAX=VSD
1562.            IF(V.LT.VMIN) VMIN=V
1563.            IF(VSD.LT.VSDMIN) VSDMIN=V
1564.      C
1565.      C *** ACCUMULATE SUM
1566.      C
1567.            VABS=DABS(V)
1568.            VSD2=VSD*VSD
1569.            VSUM=VSUM+V
1570.            VSDSUM=VSDSUM+VSD
1571.      C
1572.      C *** ACCUMULATE BY KIND
1573.      C
1574.            CALL ADSTAT(KIND,IRUN,STATS,VSD2,VABS,R)
1575.            IF(KIND.EQ.16) THEN
1576.              N1=N1+1
1577.              VABS1=VABS1+VABS*R
1578.              VSD21=VSD21+VSD2
1579.            ELSEIF(KIND.EQ.17) THEN
1580.              N2=N2+1
1581.              VABS2=VABS2+VABS*R
1582.              VSD22=VSD22+VSD2
1583.            ELSEIF(KIND.EQ.18) THEN
1584.              N3=N3+1
1585.              VABS3=VABS3+VABS
1586.              VSD23=VSD23+VSD2
1587.            ELSE
1588.              N4=N4+1
1589.              VABS4=VABS4+VABS
1590.              VSD24=VSD24+VSD2
1591.            ENDIF
1592.      C
1593.      C *** ACCUMULATE 20 LARGEST
1594.      C
1595.            CALL ACUM20(IOBS,VSD)
1596.      C
1597.            RETURN
1598.            END
1599.            SUBROUTINE ADSTAT(KIND,IRUN,STATS,VSD2,VABS,R)
1600.      C
1601.      C *** ACCUMULATE STATISTICS BY RUN
1602.      C
1603.            IMPLICIT REAL*8(A-H,O-Z)
1604.            DIMENSION STATS(1)
1605.      C
1606.      C *** DO NOT PROCESS POSITION CONSTRAINTS
1607.      C
1608.            IF(IRUN.GT.0) THEN
1609.              N=(IRUN-1)*12
1610.              IF(KIND.EQ.16) THEN
1611.                STATS(N+1)=STATS(N+1)+1.D0
1612.                STATS(N+2)=STATS(N+2)+VABS*R
1613.                STATS(N+3)=STATS(N+3)+VSD2
1614.              ELSEIF(KIND.EQ.17) THEN
1615.                STATS(N+4)=STATS(N+4)+1.D0
1616.                STATS(N+5)=STATS(N+5)+VABS*R
1617.                STATS(N+6)=STATS(N+6)+VSD2
1618.              ELSEIF(KIND.EQ.18) THEN
1619.                STATS(N+7)=STATS(N+7)+1.D0
1620.                STATS(N+8)=STATS(N+8)+VABS
1621.                STATS(N+9)=STATS(N+9)+VSD2
1622.              ELSE
1623.                STATS(N+10)=STATS(N+10)+1.D0
1624.                STATS(N+11)=STATS(N+11)+VABS
1625.                STATS(N+12)=STATS(N+12)+VSD2
1626.              ENDIF
1627.            ENDIF
1628.      C
1629.            RETURN
1630.            END
```

```
      SUBROUTINE ACUM20(IOBS,VSD)
C
C *** ACCUMULATE A LARGE RESIDUAL
C
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/RESTAT/VSD20(20),I20(20),N0,N1,N2,N3,N4,VMAX,VMIN,
     * VSDMAX,VSDMIN,VSUM,VSDSUM,VSD21,VSD22,VSD23,VSD24,VABS1,VABS2,
     * VABS3,VABS4
C
C *** DEAL ONLY WITH ABSOLUTE VALUES .GT. SMALLEST
C
      V=DABS(VSD)
      IF(V.LE.VSD20(20)) RETURN
C
C *** POINT TO NEW ARRAY LOCATION
C
      IP=IPOINT(V)
C
C *** SHIFT ARRAY CONTENTS AND LOAD ARRAYS
C
      CALL VSHIFT(IP,IOBS,V)
C
      RETURN
      END
      INTEGER FUNCTION IPOINT(V)
C
C *** LOCATE NEW POSITON IN THE MAX V ARRAY
C
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/RESTAT/VSD20(20),I20(20),N0,N1,N2,N3,N4,VMAX,VMIN,
     * VSDMAX,VSDMIN,VSUM,VSDSUM,VSD21,VSD22,VSD23,VSD24,VABS1,VABS2,
     * VABS3,VABS4
C
C *** FIND NEW POSITION (CHECK MAX VALUES FIRST)
C
      DO 1 I=1,20
      IPOINT=I
      IF(V.GT.VSD20(I)) RETURN
    1 CONTINUE
C
C *** FALL THRU LOOP--NOT A MAXIMAL RESIDUAL
C
      IPOINT=21
C
      RETURN
      END
      SUBROUTINE VSHIFT(IP,IOBS,V)
C
C *** SHIFT ARRAY CONTENTS AND LOAD ARRAYS
C
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/RESTAT/VSD20(20),I20(20),N0,N1,N2,N3,N4,VMAX,VMIN,
     * VSDMAX,VSDMIN,VSUM,VSDSUM,VSD21,VSD22,VSD23,VSD24,VABS1,VABS2,
     * VABS3,VABS4
C
C *** PROTECT AGAINST ILLEGAL VALUES
C
      IF(IP.LE.0.OR.IP.GE.21) RETURN
C
C *** SHIFT ARRAY CONTENTS
C
      IF(IP.LT.20) THEN
        DO 1 I=19,IP,-1
        VSD20(I+1)=VSD20(I)
    1   I20(I+1)=I20(I)
      ENDIF
C
C *** LOAD THE SHIFTED ARRAY
C
      VSD20(IP)=V
      I20(IP)=IOBS
C
      RETURN
      END
      SUBROUTINE RSOUT(S)
```

```
1706.      C
1707.      C *** LIST RESIDUAL STATISTICS
1708.      C
1709.            IMPLICIT REAL*8(A-H,O-Z)
1710.            DIMENSION S(1)
1711.            COMMON/PARMS/ZUPT,NSTA,NRUN,NUNK,ITAB,NOBS
1712.            COMMON/RESTAT/VSD20(20),I20(20),N0,N1,N2,N3,N4,VMAX,VMIN,
1713.           * VSDMAX,VSDMIN,VSUM,VSDSUM,VSD21,VSD22,VSD23,VSD24,VABS1,VABS2,
1714.           * VABS3,VABS4
1715.      C
1716.      C *** COMPLETE COMPUTATION OF STATS
1717.      C
1718.            N=N1+N2+N3+N4
1719.            VSD2=VSD21+VSD22+VSD23+VSD24
1720.            VMEAN=VSUM/N
1721.            VSDMN=VSDSUM/N
1722.      C
1723.            RMSV=DSQRT(VSD2/N)
1724.            RMSV1=DSQRT(VSD21/N1)
1725.            RMSV2=DSQRT(VSD22/N2)
1726.            RMSV3=DSQRT(VSD23/N3)
1727.            RMSV4=DSQRT(VSD24/N4)
1728.      C
1729.            ABSV1=VABS1/N1
1730.            ABSV2=VABS2/N2
1731.            ABSV3=VABS3/N3
1732.            ABSV4=VABS4/N4
1733.      C
1734.      C *** HEADING
1735.      C
1736.            WRITE(6,1)
1737.          1 FORMAT('1RESIDUAL STATISTICS'/)
1738.      C
1739.      C *** MAXIMUM RESIDUALS
1740.      C
1741.            WRITE(6,2)
1742.          2 FORMAT(' OBSERVATION NUMBERS OF 20 GREATEST QUASI-NORMALIZED',
1743.           *        ' RESIDUALS (V/SD)')
1744.            WRITE(6,3) I20
1745.          3 FORMAT(1X,20I6//)
1746.      C
1747.      C *** EXTREMA
1748.      C
1749.            WRITE(6,4) N,N0,VMAX,VSDMAX,VMIN,VSDMIN,VMEAN,VSDMN
1750.          4 FORMAT('  TOTAL=',I5,T22,'NO-CHECK=',I3/
1751.           *        '  MAX V=',1PD9.1,T22,'MAX V/SD=',0PF7.3/
1752.           *        '  MIN V=',1PD9.1,T22,'MIN V/SD=',0PF7.3/
1753.           *        ' MEAN V=',1PD9.1,T21,'MEAN V/SD=',0PF7.3///)
1754.      C
1755.      C *** STATS
1756.      C
1757.            WRITE(6,5) N1,VSD21,RMSV1,ABSV1,
1758.           *           N2,VSD22,RMSV2,ABSV2,
1759.           *           N3,VSD23,RMSV3,ABSV3,
1760.           *           N4,VSD24,RMSV4,ABSV4,
1761.           *           N,VSD2,RMSV
1762.          5 FORMAT(T16,'N',T23,'VTPV',T33,'RMS',T39,'MEAN ABS'/
1763.           *        T32,'VTPV',T39,'RESIDUAL'/
1764.           *        ' DELTA LAT',I6,F10.1,F9.2,F11.3,' (METERS)'/
1765.           *        ' DELTA LON',I6,F10.1,F9.2,F11.3,' (METERS)'/
1766.           *        ' DELTA   H',I6,F10.1,F9.2,F11.3,' (METERS)'/
1767.           *        '     OTHER',I6,F10.1,F9.2,F11.3/
1768.           *        '     TOTAL',I6,F10.1,F9.2//)
1769.      C
1770.      C *** RUN STATS
1771.      C
1772.            WRITE(6,6)
1773.          6 FORMAT(T10,'LATITUDE',T40,'LONGITUDE',T70,'HEIGHT',T100,'OTHER'/
1774.           *        ' RUN',T9,'N',T13,'VTPV',T21,'RMS',T27,'MEAN',
1775.           *              T39,'N',T43,'VTPV',T51,'RMS',T57,'MEAN',
1776.           *              T69,'N',T73,'VTPV',T81,'RMS',T87,'MEAN',
1777.           *              T99,'N',T103,'VTPV',T111,'RMS',T117,'MEAN'/
1778.           *                                  T20,'VTPV',T28,'V',
1779.           *                                  T50,'VTPV',T58,'V',
```

```
     *                                   T80,'VTPV',T88,'V',
     *                                   T110,'VTPV',T118,'V')
      DO 7 I=1,NRUN
      J=(I-1)*12
      N1=DABS(S(J+1))
      N2=DABS(S(J+4))
      N3=DABS(S(J+7))
      N4=DABS(S(J+10))
    7 WRITE(6,8) I,N1,S(J+3),DSQRT(S(J+3)/N1),S(J+2)/N1,
     *             N2,S(J+6),DSQRT(S(J+6)/N2),S(J+5)/N2,
     *             N3,S(J+9),DSQRT(S(J+9)/N3),S(J+8)/N3,
     *             N4,S(J+12),DSQRT(S(J+12)/N4),S(J+11)/N4
    8 FORMAT(I4,4(I5,F7.1,F7.2,F7.3,4X))
C
      RETURN
      END
      SUBROUTINE ACCUR(A,SHIFTS,SIGUWT,SUMPVV,IDOF,VARUWT)
C
C *** COMPUTE AND LIST ACCURACIES
C
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER SEEK,SIZE
      CHARACTER*90 CARD
      CHARACTER*32 NAME1,NAME2
      CHARACTER*1 CC1
      LOGICAL GETCRD
      DIMENSION A(1),SHIFTS(1)
      COMMON/PARMS/ZUPT,NSTA,NRUN,NUNK,ITAB,NOBS
C
C *** HEADING
C
      WRITE(6,1)
    1 FORMAT('1ELLIPSOIDAL LENGTH RELATIVE ACCURACIES',
     *       3X,'(USING A-PRIORI WEIGHTS)'/)
C
C *** LOOP OVER ALL CARDS FOR ACCURACIES
C
  100 IF(GETCRD(CARD,CC1)) THEN
        IF(CC1.EQ.'Q') THEN
C
C *** INSURE BOTH ENDS POSTIONED
C
          NAME1=SUBSTR(CARD,7,30)
          NAME2=SUBSTR(CARD,37,30)
          I=SEEK(ITAB,NAME1)
          J=SEEK(ITAB,NAME2)
          MAX=SIZE(ITAB)
          IF(I.LE.MAX.AND.J.LE.MAX) THEN
            CALL RELACC(A,SHIFTS,I,J,DST,SIGDST,IR,DELTA,IR2)
            WRITE(6,2) NAME1,NAME2,DST,SIGDST,IR,IR/2,IR/3,DELTA,IR2
    2       FORMAT(1X,2(A30,1X),'S=',F7.0,'  SIGMA=',F5.3,
     *             '  LENG.REL.ACCURACY=1:',2(I8,'/'),I8/
     *             T68,'LENGTH SHIFT=',F5.3,
     *             '  LENG.REL.ACCURACY=1:',I8/)
          ENDIF
        ENDIF
        GO TO 100
      ENDIF
C
C *** PRINT VARIANCE OF UNIT WEIGHT
C
      WRITE(6,3) IDOF,SUMPVV,SIGUWT,VARUWT
    3 FORMAT('0DEGREES OF FREEDOM       =',I5/
     *       ' VARIANCE SUM             =',F9.1/
     *       ' STD.DEV.OF UNIT WEIGHT   =',F7.2/
     *       ' VARIANCE OF UNIT WEIGHT  =',F7.2)
C
      RETURN
      END
      SUBROUTINE RELACC(A,SHIFTS,I,J,DST,SIGDST,IR,DELTA,IR2)
C
C *** COMPUTE RELATIVE ACCURACIES ON SPHERE
C
      IMPLICIT REAL*8(A-H,O-Z)
      LOGICAL FLAG,LONFLG
```

```
      DIMENSION A(1),SHIFTS(1)
      COMMON/CONST/PI,PI2,RAD,LONFLG,ZERO
C
C *** GET PARAMETER INDICIES
C
      IP1=IUNSTA(I,1)
      IE1=IUNSTA(I,2)
      IP2=IUNSTA(J,1)
      IE2=IUNSTA(J,2)
C
C *** GET VARIANCE-COVARIANCE ELEMENTS
C
      P1P1=ELEM(IP1,IP1,FLAG)
      IF(.NOT.FLAG) CALL ERROR(I,J,IP1,IE1,IP2,IE2,1)
      P1E1=ELEM(IP1,IE1,FLAG)
      IF(.NOT.FLAG) CALL ERROR(I,J,IP1,IE1,IP2,IE2,2)
      P1P2=ELEM(IP1,IP2,FLAG)
      IF(.NOT.FLAG) CALL ERROR(I,J,IP1,IE1,IP2,IE2,3)
      P1E2=ELEM(IP1,IE2,FLAG)
      IF(.NOT.FLAG) CALL ERROR(I,J,IP1,IE1,IP2,IE2,4)
      E1E1=ELEM(IE1,IE1,FLAG)
      IF(.NOT.FLAG) CALL ERROR(I,J,IP1,IE1,IP2,IE2,5)
      P2E1=ELEM(IP2,IE1,FLAG)
      IF(.NOT.FLAG) CALL ERROR(I,J,IP1,IE1,IP2,IE2,6)
      E1E2=ELEM(IE1,IE2,FLAG)
      IF(.NOT.FLAG) CALL ERROR(I,J,IP1,IE1,IP2,IE2,7)
      P2P2=ELEM(IP2,IP2,FLAG)
      IF(.NOT.FLAG) CALL ERROR(I,J,IP1,IE1,IP2,IE2,8)
      P2E2=ELEM(IP2,IE2,FLAG)
      IF(.NOT.FLAG) CALL ERROR(I,J,IP1,IE1,IP2,IE2,9)
      E2E2=ELEM(IE2,IE2,FLAG)
      IF(.NOT.FLAG) CALL ERROR(I,J,IP1,IE1,IP2,IE2,10)
C
C *** COMPUTE DISTANCE
C
      CALL ELIPIN(A(IP1),A(IE1),A(IP2),A(IE2),DST,FAZ,BAZ)
C
C *** COMPUTE DIFFERENTIAL COEFFICIENTS
C
      EM1=RADIUS(A(IP1),ZERO)
      EM2=RADIUS(A(IP2),ZERO)
      EN2=RADIUS(A(IP2),PI2)
C
      D=-EM1*DCOS(FAZ)
      B=EN2*DCOS(A(IP2))*DSIN(BAZ)
      C=-EM2*DCOS(BAZ)
C
C *** TEST FOR LONGITUDE POSITIVE WEST
C
      IF(.NOT.LONFLG) THEN
        B=-B
      ENDIF
C
C *** COMPUTE TERMS OF ERROR PROPAGATION
C
      T1=P1P1
      T2=P1E1+P1E1-P1E2-P1E2
      T3=E1E1+E2E2-E1E2-E1E2
      T4=P1P2+P1P2
      T5=P2E1+P2E1-P2E2-P2E2
      T6=P2P2
C
C *** COMPUTE SIGMA (NOT VARIANCE) OF DISTANCE
C
      SIGDST=DSQRT(D*D*T1+D*B*T2+B*B*T3+D*C*T4+B*C*T5+C*C*T6)
C
C *** COMPUTE R FOR PROPORTIONAL ACCURACY
C
      IR=IDINT(DST/SIGDST)
C
C *** COMPUTE THE ORIGINAL LENGTH (TRUE LENGTH)
C
      AP1=A(IP1)-SHIFTS(IP1)
      AP2=A(IP2)-SHIFTS(IP2)
```

```
      AE1=A(IE1)-SHIFTS(IE1)
      AE2=A(IE2)-SHIFTS(IE2)
      CALL ELIPIN(AP1,AE1,AP2,AE2,DTRUE,AZ,BAZ)
C
C *** COMPUTE SHIFT IN LENGTH FOR PROPORTIONAL ERROR
C
      DELTA=DST-DTRUE
      IR2=IDINT(DST/DABS(DELTA))
C
      RETURN
      END
      SUBROUTINE ELIPIN(GLAT1,E1,GLAT2,E2,S,FAZ,BAZ)
C
C *** SOLVE INVERSE GEODETIC PROBLEM
C *** AZIMUTHS CLOCKWISE FROM NORTH
C *** FLAG -- TRUE  LONGITUDE POSITIVE EAST
C ***      -- FALSE LONGITUDE POSITIVE WEST (U.S.)
C
      IMPLICIT REAL*8(A-H,O-Z)
      LOGICAL FLAG
      COMMON/CONST/PI,PI2,RAD,FLAG,ZERO
      COMMON/ELLIP/A,EE
      DATA TOL/.5D-13/
C
C *** CONVERT LONGITUDES TO POSITIVE WEST
C
      IF(FLAG) THEN
        GLON1=-E1
        GLON2=-E2
      ELSE
        GLON1=E1
        GLON2=E2
      ENDIF
C
      R=DSQRT(1.D0-EE)
      F=1.D0-R
C
      TU1=R*DSIN(GLAT1)/DCOS(GLAT1)
      TU2=R*DSIN(GLAT2)/DCOS(GLAT2)
      CU1=1.D0/DSQRT(TU1*TU1+1.D0)
      SU1=CU1*TU1
      CU2=1.D0/DSQRT(TU2*TU2+1.D0)
      S=CU1*CU2
      BAZ=S*TU2
      FAZ=BAZ*TU1
C
      W=0.D0
  100 X=GLON1-GLON2+W
      SX=DSIN(X)
      CX=DCOS(X)
      TU1=CU2*SX
      TU2=SU1*CU2*CX-BAZ
      SY=DSQRT(TU1*TU1+TU2*TU2)
      CY=S*CX+FAZ
      Y=DATAN2(SY,CY)
      SA=S*SX/SY
      C2A=-SA*SA+1.D0
      CZ=FAZ+FAZ
      IF(C2A.GT.0.D0) CZ=-CZ/C2A+CY
      E=CZ*CZ*2.D0-1.D0
      C=((-3.D0*C2A+4.D0)*F+4.D0)*C2A*F/16.D0
      D=W
      W=((E*CY*C+CZ)*SY*C+Y)*SA
      W=(1.D0-C)*W*F
      IF(DABS(D-W).GT.TOL) GO TO 100
C
      FAZ=DATAN2(-TU1,TU2)
      BAZ=DATAN2(CU1*SX,BAZ*CX-SU1*CU2)
      FAZ=FAZ+PI
      BAZ=BAZ+PI
      IF(FAZ.LT.0.D0) FAZ=FAZ+PI+PI
      IF(BAZ.LT.0.D0) BAZ=BAZ+PI+PI
      X=DSQRT((1.D0/R/R-1.D0)*C2A+1.D0)+1.D0
      X=(X-2.D0)/X
```

```
      C=1.D0-X
      C=(X*X/4.D0+1.D0)/C
      D=(0.375D0*X*X-1.D0)*X
      X=E*CY
      S=1.D0-E-E
      S=((((SY*SY*4.D0-3.D0)*S*CZ*D/6.D0-X)*D/4.D0+CZ)*SY*D+Y)*C*A*R
C
      RETURN
      END
      SUBROUTINE ERROR(I,J,IP1,IE1,IP2,IE2,K)
C
C *** PRINT ERROR MESSAGE
C
      WRITE(6,1) I,J,IP1,IE1,IP2,IE2,K
    1 FORMAT(7I10/' ELEMENTS NOT IN PROFILE')
      CALL FERR
C
      RETURN
      END
      SUBROUTINE FORMOB(IU0,IU02,A)
C
C *** REFORM OBS EQUATIONS USING MOST RECENT PARAMETERS
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(1),IUNK(10),C(10)
      COMMON/PARMS/ZUPT,NSTA,NRUN,NUNK,ITAB,NOBS
C
C *** LOOP OVER THE OBSERVATIONS
C
      REWIND IU0
      REWIND IU02
      DO 1 I=1,NOBS
      READ(IU0) KIND,IUNK,C,OBS0,OBSB,SD,TMEAN,SSQT,LENG
      CALL FORMC(KIND,C,IUNK,A,TMEAN,SSQT)
      CALL COMPOB(KIND,OBS0,IUNK,A,TMEAN,SSQT)
    1 WRITE(IU02) KIND,IUNK,C,OBS0,OBSB,SD,TMEAN,SSQT,LENG
      REWIND IU0
      REWIND IU02
C
C *** EXCHANGE PRIMARY/SECONDARY OBS EQ FILE INDICATOR
C
      ITEMP=IU0
      IU0=IU02
      IU02=ITEMP
C
      RETURN
      END
```

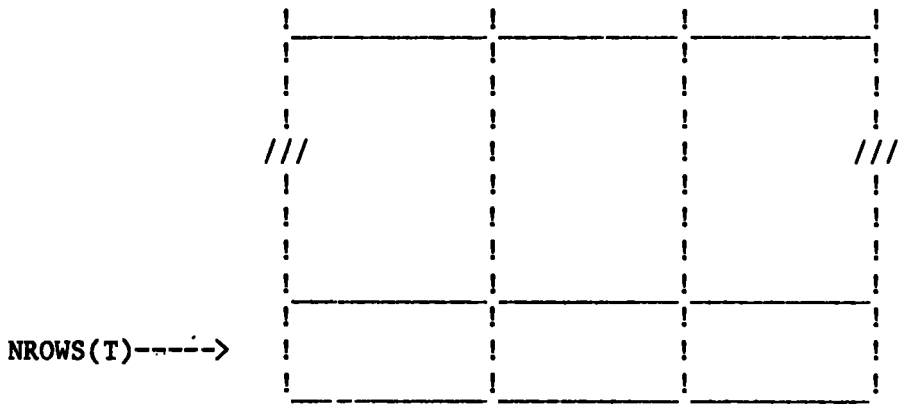
APPENDIX D.--NAME TABLE DOCUMENTATION

1. INTRODUCTION

The HASHTABLES package provides all functions necessary to create, manipulate, and destroy objects of "type" HASHTABLE. In this sense, a particular hash table is an object of "abstract data type", and HASHTABLES is the "type manager" for objects of type HASHTABLE.

A hash table may be conceptualized in terms of those functions which return aspects of its current "state" but do not have any affect on the state. In section 2 (where the functions are described) functions of this class are called "V-functions" because they give the ability to "View" particular aspects of the state. The following diagram and the informal descriptions of the V-functions below it provide the conceptual foundation for understanding the formal descriptions in section 2.

```
                                  !            !            !
                                  !<KEYLEN(T)>!<VALLEN(T)>!
                                  !            !            !
                    ____________________________________________
                    !            !            !            !
                    !            !            !            !
                    !____________!____________!____________!
                    !            !            !            !
                    !            !            !            !
                    !____________!____________!____________!
                    !            !            !            !
                    !            !            !            !
                   ///           !            !           ///
                    !            !            !            !
                    !            !            !            !
                    !____________!____________!____________!
                    !            !            !            !
CURREN(T)----->     !            ! GETKEY(T) ! GETVAL(T) !
                    !____________!____________!____________!
                    !            !            !            !
                    !            !            !            !
                   ///           !            !           ///
                    !            !            !            !
                    !____________!____________!____________!
                    !            !            !            !
         I----->    ! ISDEL(T,I) !            !            !
                    !____________!____________!____________!
                    !            !            !            !
                    !            !            !            !
                   ///           !            !           ///
                    !            !            !            !
                    !____________!____________!____________!
                    !            !            !            !
  SIZE(T)----->     !            !            !            !
```




```
VALIDT(T)........Indicator of the existence of table T
SIZE(T)..........Number of rows put in table so far
NROWS(T).........Integer number of rows to which table may grow
CURREN(T)........The "current" row number
GETKEY(T)........"key" part of the current row
GETVAL(T)........"value" part of the current row
KEYLEN(T)........Length of GETKEY(T)
VALLEN(T)........Length of GETVAL(T)
ISDEL(T,I).......Value of the delete flag for row I
HASHUB(T)........Upper bound on range of hash function
HFUNC(T).........Address of user-supplied hash function
                 (0 if none supplied)
```

2. FORMAL SPECIFICATION

In this section each O-, V-, and OV- function is described formally. A formal description consists of the following elements:

1. Type of function and its invocation rule

   There are 3 types of functions:

   O- (Operate-) functions affect the state of an object

   V- (View-) functions have no effects, but allow some aspect of the state of an object to be observed

   OV- (Operate- and View-) functions are provided as a convenience since they are equivalent to a hypothetical pair of O- and V- functions invoked in series. Because these two functions would always be used together, it makes sense to combine them (an example is the SEEK function).

   The invocation rule is a schema for invoking the function given in an Algol-like P.D.L. where the assignment operator is indicated by ':=' and is distinct from the equality operator '=' found in the EFFECTS and EXCEPTIONS sections.

2. POSSIBLE VALUES

For V- and OV- functions, this section gives the range of values returned by the function. For example, NEWT returns a value in the range of "capabilities", which is itself implementation-dependent, while SEEK returns an integer between 1 and an object-dependent upper limit.

3. PARAMETERS

Parameters are values expected by the function. Parameter values are not altered by functions. For each parameter, its type and intuitive meaning is stated.

4. EFFECTS

This section is included only for O- and OV-functions (because V-functions have no effects). An effect is a "well-formed formula" whose terms are constant symbols and V-functions. A wff becomes "true" as a result of a successful invocation of the function.

The formulas contain the following special symbols:

=	equality
not	negation
>	greater than
<	less than
>=	greater-than-or-equal-to
<=	less-than-or-equal-to
and	logical conjunction
or	logical disjunction
true,false	true,false
'	the quote operator preceding a function reference refers to the value of the function prior to the current set of effects.
there exists an S such that	existential quantification
for all A such that	universal quantification

5. EXCEPTIONS

An exception is a well-formed formula whose truth at the time of invocation results in the abnormal termination of any program calling the function. The notational conventions for writing exceptions are the same as for EFFECTS.

6. NOTES

Additional informal discussion is often included to clarify the semantics of a function.

7. FORTRAN EXAMPLE

An occasional FORTRAN program segment is included to illustrate the proper (or improper) use of a function.

```
************************************
*                                  *
* OV-FUNCTION T:=NEWT(A,N1,N2,N3,N4) *
*                                  *
************************************
```

POSSIBLE VALUES:

T.....A "capability" for a hash table

PARAMETERS:

A.....The address of a hash routine to be used with this table. If 0, a system-supplied routine will be used.

N1.....An upper bound on SIZE(T)
N2.....The fixed length of a "value" (in words)
N3.....The fixed length of a "key" (in words)
N4.....An upper bound on the range of the hash function

EFFECTS:

VALIDT(T)
HFUNC(T)=A
NROWS(T)=N1+1 (see note)
VALLEN(T)=N2
KEYLEN(T)=N3
INDLEN(T)=N4
CURREN(T)=0
SIZE(T)=0

EXCEPTIONS:

not [N1 >= 1 and N2 >= 0 and N3 >= 1 and N4 >= 1]

FORTRAN EXAMPLE:

```
EXTERNAL HASH1
.
.
T1=NEWT(HASH1,100,2,10,500)
T2=NEWT(0,1000,0,2,2000)
T3=NEWT(0,10,0,1,1)
.
.
CALL SUB(T1)
```

NOTES:

The significance of the term "capability" can be understood by

considering the above example. T1 is a "capability" for one of the three tables since its possession gives the owner EVERYTHING it needs to know in order to manipulate the table. In the example, table T1 is passed by passing a single variable. In a conventional "data structures" approach, considerably more information would have to be passed in order to give SUB "full capability" with respect to the table.

```
************************
*                      *
* V-FUNCTION N:=SIZE(T) *
*                      *
************************
```

POSSIBLE VALUES:

N.....Integer number of rows in table

PARAMETERS:

T.....A "capability" for a hash table

EXCEPTIONS:

not VALIDT(T)

NOTES:

SIZE(T) gives the number of rows the user has already placed in the table. This is not to be confused with NROWS(T) which is an upper limit on the size to which the table may grow.

```
*************************
*                       *
* V-FUNCTION N:=NROWS(T) *
*                       *
*************************
```

POSSIBLE VALUES:

N.....Integer number of rows to which table may grow

PARAMETERS:

T.....A "capability" for a hash table

EXCEPTIONS:

not VALIDT(T)

NOTES:

As described by effect #3 of NEWT, NROWS(T) is initially

set to one more than the user's maximum requested table size. Thus if the user calls NEWT with N1=100, NEWT will reserve 101 rows and NROWS(T) will be 101. No O-function (except NEWT) affects the value of NROWS(T), so it remains constant for a given table. The additional row is reserved for the proper operation of SEEK when the table contains 100 rows (in the present example).

```
**************************
*                        *
* V-FUNCTION L:=KEYLEN(T) *
*                        *
**************************
```

POSSIBLE VALUES:

L.....Integer number of words in "key" portion of row

PARAMETERS:

T.....A "capability" for a hash table

EXCEPTIONS:

not VALIDT(T)

NOTES:

keys in the user program may be of any type. Within HASHTABLES, keys are compared and manipulated in integral words. Thus if character keys are used, the user program should take care of consistent padding of keys out to the full declared size of the key.

FORTRAN EXAMPLE:

```
.
.
NAMES=NEWT(0,100,4,2,200)
.
.
I=SEEK(NAMES,'J DOE')   (*)
.
.
J=SEEK(NAMES,'J DOE    ')   (**)
.
.
```

In this example, the result of (*) is uncertain since the key is not padded to the declared size of 2 words. The correct version is given in (**).

```
***************************
*                         *
*                         *
* V-FUNCTION L:=VALLEN(T) *
*                         *
***************************
```

POSSIBLE VALUES:

L.....Integer number of words in "value" portion of row

PARAMETERS:

T.....A "capability" for a hash table

EXCEPTIONS:

not VALIDT(T)

NOTES:

HASHTABLES manipulates both keys and values in integral words. Thus, values length is always stated in whole words, regardless of the type and length of the actual values in the user program. This fact demands cautious handling of values having non-integral word lengths such as character strings (see FORTRAN EXAMPLE).

FORTRAN EXAMPLE:

```
.
.
T1=NEWT(0,100,4,2,200)
.
.
CALL PUTVAL(T1,'PUMPERNICKEL')   (*)
.
.
CALL PUTVAL(T1,'PUMPERNICKEL    ')   (**)
.
.
```

The last word of the value stored in the table would have undefined contents in the case of (*). The problem is fixed in (**) by padding the value to 16 characters.

```
***************************
*                         *
* V-FUNCTION M:=HASHUB(T) *
*                         *
***************************
```

POSSIBLE VALUES:

M.....Integer upper bound of hash function range

PARAMETERS:

T.....A "capability" for a hash table

EXCEPTIONS:

not VALIDT(T)

NOTES:

Parameter N4 in the call to NEWT establishes the range of the hash function to be used with a table. No O-function has any effect on this range, so it remains fixed for the life of the table.

If the built-in hash function is elected, it will transform keys into the range [1,HASHUB(T)]. If a user-provided hash function is used, that function must return a result in the same range or an exception will be raised.

```
***************************
*                         *
* V-FUNCTION A:=HFUNC(T)  *
*                         *
***************************
```

POSSIBLE VALUES:

A.....A value in the range of system addresses OR 0

PARAMETERS:

T.....A "capability" for a hash table

EXCEPTIONS:

not VALIDT(T)

NOTES:

A possible use of HFUNC would be to determine whether or not a user-defined hash function is associated with a table. Such information might be useful to a general-purpose routine used to analyze the relative efficiency of a variety of hashing algorithms.

```
*****************************
*                           *
* OV-FUNCTION R:=SEEK(T,ARG) *
*                           *
*****************************
```

POSSIBLE VALUES:

R.....Integer "row number" in range [1,SIZE(T)+1]

PARAMETERS:

```
T.......A "capability" for a hash table
ARG.....Argument to be used in table lookup
```

EFFECTS:

```
IF
    there exists an S such that
        CURREN(T)=S and KEYVAL(T)=ARG
THEN
    R=S
    CURREN(T)=S
ELSE
    R=SIZE(T)+1
    CURRENT(T)=R
    GETKEY(T)=ARG
    ISDEL(T,R)=false
ENDIF
```

EXCEPTIONS:

```
not VALIDT(T)
```

FORTRAN EXAMPLE:

```
.
.
IF(SEEK(T1,'JOHN DOE').LE.SIZE(T))THEN
    .
    .
    successful lookup logic
    .
    .
ELSE
    .
    .
    unsuccessful lookup logic
    .
    .
ENDIF
```

```
****************************
*                          *
* V-FUNCTION VAL:=GETVAL(T) *
*                          *
****************************
```

POSSIBLE VALUES:

```
VAL.....A value in the range of table values
```

PARAMETERS:

T.....A "capability" for a hash table

EXCEPTIONS:

not VALIDT(T)
not [1<=CURRENT(T)<=SIZE(T)]

NOTES:

Because of the limitations on the "type" of values returned by FORTRAN functions, GETVAL is implemented as a subroutine with 'VAL' as the second parameter (see EXAMPLE).

FORTRAN EXAMPLE;

```
        .
        .
        IF(SEEK(NAMES,'JOHN DOE').LE.SIZE(NAMES))THEN
            CALL GETVAL(NAMES,RECORD)
            EMPNO=RECORD(1)
            AGE=RECORD(2)
            DECODE(100,RECORD(3))ADDRESS
100         FORMAT(A32)
            .
            .
```

```
***************************
*                         *
* V-FUNCTION K:=GETKEY(T) *
*                         *
***************************
```

POSSIBLE VALUES:

K.....A value in the range of table keys

PARAMETERS:

T.....A "capability" for a hash table

EXCEPTIONS:

not VALIDT(T)
not [1<=CURREN(T)<=SIZE(T)]

NOTES:

Because of the limitations on the "type" of values returned by FORTRAN functions, GETKEY is implemented as a subroutine with 'KEY' as the second parameter (see EXAMPLE).

FORTRAN EXAMPLE:

```
        .
        .
        CALL SETCUR(NAMES,I)
        CALL GETKEY(NAMES,NAME)
        .
        .
```

```
****************************
*                          *
* O-FUNCTION PUTVAL(T,VAL) *
*                          *
****************************
```

POSSIBLE VALUES: none

PARAMETERS:

```
    T.......A "capability" for a hash table
    VAL.....A value in the range of table values
```

EFFECTS:

```
    GETVAL(T)=VAL
    SIZE(T)='SIZE(T)+1
```

EXCEPTIONS:

```
    not VALIDT(T)
    not [ CURREN(T)='SIZE(T)+1 and 'SIZE(T)<NROWS(T) ]
```

NOTES:

As EXCEPTION #2 indicates, PUTVAL can only be used to store a value in a new row. In fact, it is PUTVAL which causes the row to "become" part of the table.
In fact, it is PUTVAL that CAUSES the row to become part of the table.

A value in an existing row must be changed by CHGVAL.

FORTRAN EXAMPLE:

```
        .
        .
        IF(SEEK(NAMES,'JOHN DOE') .EQ. SIZE(NAMES)+1)THEN
            RECORD(1)=EMPNO
            RECORD(2)=AGE
            ENCODE(100,RECORD(3))ADDRESS
100         FORMAT(A32)
            CALL PUTVAL(NAMES,RECORD)
            .
            .
```

****

```
************************
*                      *
* O-FUNCTION DELETE(T) *
*                      *
************************
```

POSSIBLE VALUES: none

PARAMETERS:

T.....A "capability" for a hash table

EFFECTS:

ISDEL[T,'CURREN(T)]=true
CURREN(T)=0

EXCEPTIONS:

not VALIDT(T)
not [1 <= 'CURREN(T)<=SIZE(T)]

```
***************************
*                         *
* V-FUNCTION R:=CURREN(T) *
*                         *
***************************
```

POSSIBLE VALUES:

R.....Integer "row number" in range [1,SIZE(T)+1]

PARAMETERS:

T.....A "capability" for a hash table

EXCEPTIONS:

not VALIDT(T)

```
*************************
*                       *
* O-FUNCTION SETCUR(T,R) *
*                       *
*************************
```

POSSIBLE VALUES: none

PARAMETERS:

T.....A "capability" for a hash table
R.....Integer "row number"

EFFECTS:

```
        CURREN(T)=R

    EXCEPTIONS:

        not VALIDT(T)
        ISDEL(T,R)
        not [ 1 <=R<=SIZE(T) ]

****************************
*                          *
* V-FUNCTION B:=ISDEL(T,R) *
*                          *
****************************

    POSSIBLE VALUES:

        B.......true,false

    PARAMETERS:

        T.......A "capability" for a hash table
        R.......INTEGER "row number"

    EXCEPTIONS:

        not VALIDT(T)
        not [ 1<=R<=SIZE(T) ]

    NOTES:

        Because a deleted row can never be made current by any O-operation,
        a function (ISDEL) must be provided to test whether a row is deleted
        or not.

    FORTRAN EXAMPLE:

        .
        .
        IF(.NOT.ISDEL(T,I))THEN
            CALL SETCUR(T,I)
            CALL GETVAL(T,VAL)
            .
            .

****************************
*                          *
* O-FUNCTION CHGVAL(T,VAL) *
*                          *
****************************

    POSSIBLE VALUES: none

    PARAMETERS:
```

```
    T.......A "capability" for a hash table
    VAL.....A "value" in the range of table values

EFFECTS:

    GETVAL(T)=VAL

EXCEPTIONS:

    not VALIDT(T)
    not [ 1<=CURREN(T)<=SIZE(T) ]

NOTES:

    CHGVAL can be used to change the value in any row which is
    already part of the table, while PUTVAL may only be used for
    putting a value into a new row.

FORTRAN EXAMPLE:

    .
    .
    IF(SEEK(NAMES,'JOHN ROE').LE.SIZE(NAMES))THEN
        CALL GHGVAL(NAMES,'JOHN DOE')
        .
        .
```

```
**********************
*                    *
* O-FUNCTION CLEAR(T) *
*                    *
**********************
```

```
POSSIBLE VALUES: none

PARAMETERS:

    T.......A "capability" for a hash table

EFFECTS:

    CURREN(T)=0
    SIZE(T)=0

EXCEPTIONS:

    not VALIDT(T)

NOTES:

    This function is appropriate when it is desired to reuse
    a table.  No table resources are freed as the result of a
    CLEAR.
```

```
*********************
*                   *
* O-FUNCTION ABOR(T) *
*                   *
*********************
```

POSSIBLE VALUES: none

PARAMETERS:

T.....A "capability" for a hash table

EFFECTS:

not VALIDT(T)

EXCEPTIONS:

not 'VALIDT(T)

NOTES:

ABOR is appropriate when a table is no longer needed and it is desired to return its resources (e.g. its memory, its "capability") to the type manager.

FORTRAN EXAMPLE:

```
.
.
T=NEWT(.......)
.
.
CALL ABOR(T)
.
.
PRINT *,VALIDT(T)   (*)
.
.
CALL SETCUR(T,I)    (**)
.
.
```

The first print (*) will cause ".FALSE." to be written. The call to CLEAR will result in an abort since T is no longer a valid capability.

```
***************************
*                         *
* V-FUNCTION B:=VALIDT(T) *
*                         *
***************************
```

POSSIBLE VALUES:

B.....true,false

PARAMETERS:

T.....An arbitrary value

EXCEPTIONS: none

NOTES:

VALIDT is the only function without exceptions, and should be used in "mutually suspicious" subsystems.

3. USAGE GUIDELINES

HASHTABLES performs hashing with collisions resolved by chaining. It contains a built-in hashing function which should do a fairly good job of randomizing keys which are themselves fairly random. In the event that the keys have a distribution about which something is known in advance, it may be preferable for the user to write a hashing function which does a better job of randomizing the keys than the built-in function does.

For example, suppose a key consists of two alphabetic characters (e.g., the abbreviations of U.S. states). An EXACT (1-1, onto) hashing function could be devised which maps the 676 possible pairs (AA,AB,...,ZZ) onto the integers (1,2,...,676). This function would result in no collisions (optimal SEEK efficiency) as long as N4 were set greater than or equal to 676 in the call to NEWT. Specifying N4 as less than 676 would introduce the possibility of collisions (degrading SEEK efficiency). Specifying N4 as 1 would mean 100% collision probability, and a SEEK would result in an O(n) (linear) scan of the entire table.

There are many trade-offs to be made involving space and time. Space requirements are equal to

N1(2+N2+N3)+N4

while time is directly proportional to N4 and the "goodness" of the hashing function for a particular set of keys. Because N1, N2, and N3 are probably fixed in a given application, attention should be focused on the choice of N4 and the design of an appropriate hashing function when table processing is a critical part of an application. In an application where a few hundred SEEKs are required, the choice of N4 is not critical, and the built-in hashing function is more than adequate. For a few thousand SEEKs, the value of N4 ought to be increased to the same order of magnitude as N1; 2*N1 is perhaps a good choice. Only when the number of SEEKs is expected to approach the tens or hundreds of thousands should N4 be made much larger or should a user-defined hashing function be considered.

NOTE: For very small tables (1 <= SIZE(T) <= 10) calling NEWT with N4 =1 may result in better performance than calling with N4>1, since the cost of hashing itself may exceed that of a full table search. SEEK does not actually invoke a built-in or user-defined hashing function when N4=1.

A user-supplied hashing function must be parameterless. It must communicate with HASHTABLES by means of a common block labelled HASH, with variables arranged as follows:

```
COMMON/HASH/RESULT,M,STRL,STRING(10)
```

Each time the application program does a SEEK, the user-defined function will be called. On entry to the user function, the variables in HASH will be set as follows:

M	will contain INDLEN(T)
STRL	will contain KEYLEN(T)
STRING	will contain the search argument (up to 10 words max)

Before returning, the user function must place the result of its work in the integer variable RESULT, making sure that its value is between [1,M].

3/18 -- Cassette 300
east-west, forward run only, system malfunction

3/19 -- Cassette 301
east-west, forward run only

3/19 -- Cassette 302
east-west, two complete runs, operator error at 1013 update on second run

3/20 -- Cassette 303
east-west, three complete runs, data gap in reverse run, entire second, and entire third run

3/21 -- Cassette 304
observations only over pre-survey calibration points

3/21 -- Cassette 305
north-south, two complete runs

3/22 -- Cassette 306
north-south, three complete runs

3/22 -- Cassette 307
observations on points 3001-3005

3/23 -- Cassette 308
north-south, two complete runs, data gap in reverse run and entire second traverse

3/24 -- Cassette 309
north-south, one complete run, deliberate error in longitude update

Cassette 310
this cassette was not used

3/24 -- Cassette 311
north-south, two complete runs, data gap in reverse run of first traverse and entire second traverse

3/25 -- Cassette 312
north-south, two complete runs, first run has ZUPTs midway between each mark. Second run has midway ZUPTs on the forward run only.

3/25 -- Cassette 313
north-south, forward run only, system malfunction

3/26 -- Cassette 314
north-south, one complete run, incorrect update at 2003

3/27 -- Cassette 315
east-west, two complete runs, data gap in reverse run of first traverse, second traverse intact

3/27 -- Cassette 316
east-west, one complete run

3/27 -- Cassette 317
east-west, one complete run

3/28 -- Cassette 318
east-west, three complete runs, deliberate error in latitude update of first traverse

3/29 -- Cassette 319
east-west, three complete runs

3/30 -- Cassette 320
north-south, two traverse runs over extended north-south line, data gap at end of second traverse

3/30 -- Cassette 321
north-south, two partial traverses, first traverses incomplete, insufficient fuel, second traverse incomplete, too dark

3/31 -- Cassette 322
north-south, one complete traverse

3/31 -- Cassette 323
east-west, one complete traverse, rejected due to severe residuals.

3/31 -- Cassette 324
north-south, one partial traverse, system malfunction

4/1 -- Cassette 325
east-west, one forward run over extended east-west line

APPENDIX F.--ADJUSTED PARAMETER VALUES

RUN= 1

	VALUE	STD.DEV.	VAL/SIG	GOOGE
1	1.31-004	2.55-005	5.1	5.34-001
2	-7.43-006	8.61-006	-.9	1.21-001
3	-1.88-005	2.48-006	-7.6	8.15-001
4	1.11-009	1.09-010	10.2	1.00+000
5	-2.47-005	3.96-006	-6.2	5.37-001
6	1.15-004	4.16-005	2.8	1.86-001
7	-5.40-005	1.12-005	-4.8	9.38-001
8	-7.03-011	7.14-011	-1.0	1.00+000
9	-1.13+002	2.51+003	.0	2.21-001
10	-4.53+001	3.27+002	-.1	1.99-001
11	-1.32+002	7.21+001	-1.8	6.07-001
12	5.63+000	1.04+001	.5	1.00+000

RUN= 2

	VALUE	STD.DEV.	VAL/SIG	GOOGE
1	1.87-005	3.87-006	4.8	1.00+000
2	3.56-005	1.76-005	2.0	6.45-001
3	1.58-005	4.04-006	3.9	2.88-001
4	-6.93-011	6.87-011	-1.0	7.35-001
5	8.63-005	1.21-005	7.1	1.00+000
6	-5.16-005	9.27-006	-5.6	6.45-001
7	4.32-006	2.61-006	1.7	2.38-001
8	-4.66-010	1.04-010	-4.5	3.16-001
9	4.02+001	1.27+002	.3	1.00+000
10	-9.87+002	7.90+002	-1.2	6.45-001
11	1.00+001	1.15+001	.9	2.38-001
12	-1.12+001	2.69+001	-.4	1.93-001

RUN= 3

	VALUE	STD.DEV.	VAL/SIG	GOOGE
1	2.06-005	3.87-006	5.3	1.00+000
2	7.54-005	1.78-005	4.2	6.45-001
3	6.23-006	4.06-006	1.5	2.85-001
4	6.90-011	7.06-011	1.0	7.30-001
5	1.11-004	1.21-005	9.1	1.00+000
6	-2.74-005	9.89-006	-2.8	6.45-001
7	-1.26-005	2.85-006	-4.4	2.36-001
8	-7.64-010	1.16-010	-6.6	2.67-001
9	1.02+002	1.27+002	.8	1.00+000
10	-1.10+003	7.90+002	-1.4	6.45-001
11	-5.69+000	1.15+001	-.5	2.36-001
12	-8.90+000	2.70+001	-.3	1.91-001

RUN= 4

	VALUE	STD.DEV.	VAL/SIG	GOOGE
1	2.41-005	3.87-006	6.2	1.00+000
2	2.13-005	1.74-005	1.2	6.45-001
3	2.34-005	4.31-006	5.4	2.96-001
4	3.03-011	6.79-011	.4	7.25-001
5	6.37-005	1.22-005	5.2	1.00+000
6	-2.96-005	8.64-006	-3.4	6.45-001
7	5.24-007	2.70-006	.2	2.47-001
8	-5.64-010	9.92-011	-5.7	3.42-001
9	2.83+002	1.27+002	2.2	1.00+000
10	-1.22+003	7.89+002	-1.5	6.45-001
11	1.82+001	1.21+001	1.5	2.47-001
12	2.91+001	2.83+001	1.0	2.02-001

RUN= 5

	VALUE	STD.DEV.	VAL/SIG	GOOGE
1	1.67-005	3.87-006	4.3	1.00+000
2	1.40-005	1.79-005	.8	6.45-001
3	2.41-005	4.59-006	5.2	2.90-001
4	-9.22-011	7.97-011	-1.2	7.03-001
5	5.72-005	1.21-005	4.7	1.00+000
6	-1.92-006	9.77-006	-.2	6.45-001
7	-8.54-006	3.21-006	-2.7	2.41-001
8	-3.88-010	1.30-010	-3.0	2.61-001
9	3.32+002	1.27+002	2.6	1.00+000
10	-1.12+003	7.90+002	-1.4	6.45-001
11	-2.20-001	1.28+001	.0	2.41-001
12	1.45+001	3.00+001	.5	1.93-001

RUN= 6

	VALUE	STD.DEV.	VAL/SIG	GOOGE
1	2.81-005	3.88-006	7.2	1.00+000
2	6.98-005	1.77-005	3.9	6.45-001
3	1.87-005	4.28-006	4.4	2.95-001
4	-2.26-011	7.42-011	-.3	7.18-001
5	8.00-005	1.22-005	6.6	1.00+000
6	-3.80-005	8.86-006	-4.3	6.45-001
7	1.30-006	2.80-006	.5	2.46-001
8	-4.84-010	1.13-010	-4.3	3.05-001
9	3.44+002	1.27+002	2.7	9.65-001
10	-1.05+003	7.90+002	-1.3	6.23-001
11	-4.74+000	1.21+001	-.4	2.46-001
12	5.44+000	2.83+001	.2	1.96-001

RUN= 7

	VALUE	STD.DEV.	VAL/SIG	GOOGE
1	2.36-005	3.85-006	6.1	9.97-001
2	3.49-005	1.89-005	1.8	7.14-001
3	3.26-005	3.83-006	8.5	2.75-001
4	1.61-010	6.42-011	2.5	7.73-001
5	9.81-005	1.26-005	7.8	9.91-001
6	1.32-005	1.04-005	1.3	7.19-001
7	5.24-006	2.59-006	2.0	2.20-001
8	-2.15-009	1.03-010	-20.8	2.96-001
9	3.47+002	1.28+002	2.7	9.97-001
10	-1.24+003	7.91+002	-1.6	7.14-001
11	1.10+001	1.04+001	1.1	2.20-001
12	-6.56+000	2.46+001	-.3	2.08-001

RUN= 8

	VALUE	STD.DEV.	VAL/SIG	GOOGE
1	3.35-005	3.85-006	8.7	9.97-001
2	1.29-004	1.90-005	6.8	7.14-001
3	-3.92-006	4.30-006	-.9	2.81-001
4	2.80-010	7.24-011	3.9	7.37-001
5	1.27-004	1.26-005	10.1	9.90-001
6	3.72-006	1.01-005	.4	7.19-001
7	-9.06-006	2.91-006	-3.1	2.27-001
8	-8.05-011	1.16-010	-.7	2.86-001
9	2.60+002	1.28+002	2.0	9.97-001
10	-1.27+003	7.90+002	-1.6	7.14-001
11	2.17+001	1.15+001	1.9	2.27-001
12	1.94+001	2.71+001	.7	2.11-001

RUN= 9

	VALUE	STD.DEV.	VAL/SIG	GOOGE
1	9.78-005	2.34-005	4.2	6.29-001
2	2.14-005	1.03-005	2.1	9.32-002
3	-9.95-006	3.24-006	-3.1	7.70-001
4	1.25-010	1.21-010	1.0	9.98-001
5	4.90-006	3.95-006	1.2	6.41-001
6	1.50-004	3.63-005	4.1	2.34-001
7	-3.15-005	9.50-006	-3.3	9.52-001
8	-5.29-011	6.93-011	-.8	1.00+000
9	-1.48+002	2.51+003	-.1	2.05-001
10	1.38+002	3.27+002	.4	2.29-001
11	-1.83+002	5.90+001	-3.1	7.46-001
12	3.58+001	1.15+001	3.1	1.00+000

RUN= 10

	VALUE	STD.DEV.	VAL/SIG	GOOGE
1	6.88-005	2.34-005	2.9	6.39-001
2	1.89-005	1.21-005	1.6	6.41-002
3	-4.80-006	3.79-006	-1.3	7.99-001
4	6.39-011	1.44-010	.4	1.00+000
5	4.09-006	3.95-006	1.0	1.00+000
6	1.98-004	3.73-005	5.3	1.90-001
7	-4.54-005	9.73-006	-4.7	6.96-001
8	-1.55-011	7.21-011	-.2	9.99-001
9	-8.86+001	2.51+003	.0	1.93-001
10	1.25+002	3.27+002	.4	2.14-001
11	-1.40+002	6.00+001	-2.3	7.14-001
12	3.26+001	1.17+001	2.8	9.93-001

RUN= 11

	VALUE	STD.DEV.	VAL/SIG	GOOGE
1	1.13-004	2.32-005	4.9	6.18-001
2	1.89-005	1.08-005	1.7	7.76-002
3	-2.30-006	3.78-006	-.6	7.81-001
4	9.07-011	1.50-010	.6	1.00+000
5	8.79-007	3.95-006	.2	6.19-001
6	2.27-004	3.42-005	6.6	2.48-001
7	-5.20-005	9.46-006	-5.5	9.56-001
8	2.10-011	7.80-011	.3	1.00+000
9	-8.52+001	2.50+003	.0	7.19-001
10	1.31+002	3.27+002	.4	2.17-001
11	-2.07+002	5.93+001	-3.5	2.46-001
12	3.51+001	1.23+001	2.9	1.00+000

RUN= 12

	VALUE	STD.DEV.	VAL/SIG	GOOGE
1	5.71-005	2.32-005	2.5	6.30-001
2	1.42-005	1.24-005	1.1	5.90-002
3	6.47-006	3.98-006	1.6	8.30-001
4	6.96-010	1.53-010	4.5	1.00+000
5	1.22-005	3.95-006	3.1	6.31-001
6	2.54-004	3.75-005	6.8	1.96-001
7	-8.17-005	1.00-005	-8.2	9.61-001
8	-1.05-010	7.82-011	-1.3	1.00+000
9	2.10+002	2.51+003	.1	6.95-001
10	9.34+001	3.27+002	.3	1.91-001
11	-3.01+002	6.14+001	-4.9	2.05-001
12	5.71+001	1.25+001	4.6	1.00+000

RUN= 13

	VALUE	STD.DEV.	VAL/SIG	GOOGE
1	7.55-005	2.33-005	3.2	6.12-001
2	5.14-006	1.12-005	.5	7.44-002
3	6.40-006	3.96-006	1.6	7.70-001
4	-3.88-011	1.58-010	-.2	1.00+000
5	-8.89-007	3.95-006	-.2	6.20-001
6	1.49-004	3.41-005	4.4	2.50-001
7	-4.51-005	9.61-006	-4.7	9.46-001
8	9.44-011	8.00-011	1.2	1.00+000
9	6.17+001	2.50+003	.0	2.16-001
10	1.71+002	3.27+002	.5	2.43-001
11	-2.39+002	6.03+001	-4.0	7.22-001
12	3.35+001	1.26+001	2.7	1.00+000

RUN= 14

	VALUE	STD.DEV.	VAL/SIG	GOOGE
1	3.56-005	2.32-005	1.5	6.16-001
2	3.37-005	1.13-005	3.0	7.28-002
3	-1.02-005	3.85-006	-2.7	7.85-001
4	1.66-010	1.53-010	1.1	1.00+000
5	-4.23-006	3.95-006	-1.1	6.20-001
6	2.45-004	3.44-005	7.1	2.43-001
7	-7.91-005	9.47-006	-8.4	9.49-001
8	-1.18-011	7.82-011	-.2	1.00+000
9	8.18+001	2.50+003	.0	2.15-001
10	1.84+002	3.27+002	.6	2.41-001
11	-2.52+002	5.89+001	-4.3	7.17-001
12	2.40+001	1.23+001	2.0	1.00+000

RUN= 15

	VALUE	STD.DEV.	VAL/SIG	GOOGE
1	8.15-005	2.33-005	3.5	6.41-001
2	9.96-006	1.32-005	.8	5.43-002
3	2.74-006	4.31-006	.6	8.62-001
4	2.32-010	1.62-010	1.4	1.00+000
5	1.11-007	3.95-006	.0	6.41-001
6	1.99-004	3.81-005	5.2	1.98-001
7	-2.75-005	1.07-005	-2.6	9.73-001
8	7.38-011	8.02-011	.9	1.00+000
9	-1.24+002	2.51+003	.0	8.80-001
10	1.19+002	3.27+002	.4	5.97-001
11	-1.40+002	6.54+001	-2.1	2.18-001
12	3.47+001	1.30+001	2.7	2.02-001

RUN= 16

	VALUE	STD.DEV.	VAL/SIG	GOOGE
1	3.83-005	2.34-005	1.6	1.00+000
2	6.59-006	1.26-005	.5	6.43-001
3	6.75-006	3.72-006	1.8	2.14-001
4	-1.40-010	1.39-010	-1.0	2.31-001
5	-5.89-006	3.95-006	-1.5	6.40-001
6	2.16-004	3.76-005	5.7	9.96-001
7	-2.17-005	9.48-006	-2.3	1.99-001
8	1.81-011	6.94-011	.3	9.99-001
9	-6.08+002	2.51+003	-.2	6.37-001
10	1.15+002	3.27+002	.4	1.00+000
11	-1.27+001	5.90+001	-.2	2.10-001
12	3.28+001	1.15+001	2.8	2.14-001

RUN= 17

	VALUE	STD.DEV.	VAL/SIG	GOOGE
1	4.83-005	2.35-005	2.1	6.56-001
2	2.88-006	1.19-005	.2	7.49-002
3	2.99-006	3.48-006	.9	7.99-001
4	1.67-010	1.32-010	1.3	9.89-001
5	-1.88-006	4.03-006	-.5	6.83-001
6	1.19-004	3.76-005	3.2	2.21-001
7	-6.90-006	9.41-006	-.7	9.56-001
8	1.06-011	7.04-011	.2	1.00+000
9	-7.37+002	2.51+003	-.3	9.74-001
10	8.17+001	3.27+002	.2	6.82-001
11	5.28+001	5.78+001	.9	2.21-001
12	3.50+001	1.14+001	3.1	2.23-001

RUN= 18

	VALUE	STD.DEV.	VAL/SIG	GOOGE
1	9.77-006	3.85-006	2.5	9.97-001
2	9.54-005	1.89-005	5.0	7.14-001
3	-4.69-006	3.38-006	-1.4	2.79-001
4	1.69-010	5.74-011	2.9	7.59-001
5	1.13-004	1.26-005	9.0	9.90-001
6	-2.36-005	1.05-005	-2.3	7.19-001
7	-9.15-006	2.39-006	-3.8	2.25-001
8	2.28-010	9.56-011	2.4	2.71-001
9	3.60+002	1.28+002	2.8	9.97-001
10	-8.75+002	7.90+002	-1.1	7.14-001
11	1.60+001	9.14+000	1.7	2.25-001
12	-1.72+001	2.16+001	-.8	2.10-001

* U.S. GOVERNMENT PRINTING OFFICE: 1983 380-997/594

U.S. DEPARTMENT OF COMMERCE
National Oceanic and Atmospheric Administration
National Ocean Survey
National Geodetic Survey, C18x2
Rockville, Maryland 20852

OFFICIAL BUSINESS

POSTAGE AND FEES PAID
U.S. DEPARTMENT OF COMMERCE
COM-210
THIRD CLASS MAIL

U.S.MAIL