Cop. No. 16

Bell

NATIONAL BUREAU OF STANDARDS REPORT

10 859

FINAL REPORT

NBS PROJECT WHERE

in support of

USAAMCA PROJECT MNPLS

by

Applied Mathematics Division

and

Computer Services Division

U.S. DEPARTMENT OF COMMERCE

NATIONAL BUREAU OF STANDARDS

NATIONAL BUREAU OF STANDARDS

The National Bureau of Standards[1] was established by an act of Congress March 3, 1901. The Bureau's overall goal is to strengthen and advance the Nation's science and technology and facilitate their effective application for public benefit. To this end, the Bureau conducts research and provides: (1) a basis for the Nation's physical measurement system, (2) scientific and technological services for industry and government, (3) a technical basis for equity in trade, and (4) technical services to promote public safety. The Bureau consists of the Institute for Basic Standards, the Institute for Materials Research, the Institute for Applied Technology, the Center for Computer Sciences and Technology, and the Office for Information Programs.

THE INSTITUTE FOR BASIC STANDARDS provides the central basis within the United States of a complete and consistent system of physical measurement; coordinates that system with measurement systems of other nations; and furnishes essential services leading to accurate and uniform physical measurements throughout the Nation's scientific community, industry, and commerce. The Institute consists of a Center for Radiation Research, an Office of Measurement Services and the following divisions:

Applied Mathematics—Electricity—Heat—Mechanics—Optical Physics—Linac Radiation[2]—Nuclear Radiation[2]—Applied Radiation[2]—Quantum Electronics[3]—Electromagnetics[3]—Time and Frequency[3]—Laboratory Astrophysics[3]—Cryogenics[3].

THE INSTITUTE FOR MATERIALS RESEARCH conducts materials research leading to improved methods of measurement, standards, and data on the properties of well-characterized materials needed by industry, commerce, educational institutions, and Government; provides advisory and research services to other Government agencies; and develops, produces, and distributes standard reference materials. The Institute consists of the Office of Standard Reference Materials and the following divisions:

Analytical Chemistry—Polymers—Metallurgy—Inorganic Materials—Reactor Radiation—Physical Chemistry.

THE INSTITUTE FOR APPLIED TECHNOLOGY provides technical services to promote the use of available technology and to facilitate technological innovation in industry and Government; cooperates with public and private organizations leading to the development of technological standards (including mandatory safety standards), codes and methods of test; and provides technical advice and services to Government agencies upon request. The Institute also monitors NBS engineering standards activities and provides liaison between NBS and national and international engineering standards bodies. The Institute consists of the following divisions and offices:

Engineering Standards Services—Weights and Measures—Invention and Innovation—Product Evaluation Technology—Building Research—Electronic Technology—Technical Analysis—Measurement Engineering—Office of Fire Programs.

THE CENTER FOR COMPUTER SCIENCES AND TECHNOLOGY conducts research and provides technical services designed to aid Government agencies in improving cost effectiveness in the conduct of their programs through the selection, acquisition, and effective utilization of automatic data processing equipment; and serves as the principal focus within the executive branch for the development of Federal standards for automatic data processing equipment, techniques, and computer languages. The Center consists of the following offices and divisions:

Information Processing Standards—Computer Information—Computer Services—Systems Development—Information Processing Technology.

THE OFFICE FOR INFORMATION PROGRAMS promotes optimum dissemination and accessibility of scientific information generated within NBS and other agencies of the Federal Government; promotes the development of the National Standard Reference Data System and a system of information analysis centers dealing with the broader aspects of the National Measurement System; provides appropriate services to ensure that the NBS staff has optimum accessibility to the scientific information of the world, and directs the public information activities of the Bureau. The Office consists of the following organizational units:

Office of Standard Reference Data—Office of Technical Information and Publications—Library—Office of International Relations.

---

[1] Headquarters and Laboratories at Gaithersburg, Maryland, unless otherwise noted; mailing address Washington, D.C. 20234.
[2] Part of the Center for Radiation Research.
[3] Located at Boulder, Colorado 80302.

NATIONAL BUREAU OF STANDARDS REPORT

NBS PROJECT

6300505
2050505

May 31, 1972

NBS REPORT

10 859

FINAL REPORT

NBS PROJECT WHERE

in support of

USAAMCA PROJECT MNPLS

by

Applied Mathematics Division

and

Computer Services Division

U.S. DEPARTMENT OF COMMERCE

NATIONAL BUREAU OF STANDARDS

List of Contents

1. Executive summary of report

This report summarizes the work done at the National Bureau of Standards (NBS) under its project WHERE in support of the US Army Advanced Materiel Concepts Agency's (USAAMCA) project MNPLS (Micro Navigation and Position Location System).

MNPLS is a system that utilizes a time-division multiplexed frequency that is shared by n units, all of which are synchronized by a suitable electromagnetic signal during each time interval ΔT. Each unit is assigned one or more time slots within ΔT during which it emits an electromagnetic signal. This signal in turn is received by several but not necessarily all other units in the field which measure the time of flight of the EM signal; from the knowledge of the time and the assignment of the time slots for each unit, the distance between the sending unit and the receiving unit can be inferred.

For NBS the problem to be solved was twofold:

(1) To determine the feasibility of such a system in the presence of measurement errors. If all measurements were exact three units in the plane or 4 units in three-space would suffice to locate another (or all other)units. In the presence of error more measurements are needed and the "best" solution that minimizes total error in some mathematical sense has to be found. The fundamental problem for NBS was to show that the overall error does not increase unduly as the system develops in time.

(2) To develop a computer simulation for such a system that allows the investigation of real-time scenarios with a sufficiently large number of units undergoing battle-field manoeuvres, each with prescribed movements over suitably long times. The simulation program determines the distance between any two units i and j, perturbs that distance according to suitable assumptions about the measurement errors, and supplies to the master computer (the program that determines positions from the distance measurements) the distances as though they came from actual field measurements.

The output of the program consists of the successive locations of all units, the errors between true position and calculated position, and a measure of confidence derived from the available measurements but not from the knowledge of exact position (which is only available to the simulation program but not to the position location algorithm).

NBS has designed a set of mathematical algorithms for position location determination that allow the determination of successive positions of n field units, from inputs containing only distances between units, or perhaps those distances accompanied by independent measurements of differences in height above sea level. The experiments based on actual deployments in the Boston area show that, with a suitable number of fixed units, the average error of position over all units can be kept well within acceptable limits. Based on a range-error standard derivation of 6m , and using all 83 units as reporters, the maximum error during 2 hours of actual time remained under 110m. The average error was 4.5m over all units, with only 3 units "lost" more than 8 times out of 240 tries. When only the 39 fixed units were used as reporters, the average error was 4.7m, the maximum was 40m, but 15 units were "lost" more than 50 times out of 240. (The term "locators" refers to those users that do actually receive a signal from the locatee, and do report a time-of-flight to the master computer; the term "reporters" refers to those users that have been instructed to report time-of-flight measurements to the computer. Because of "line-of-sight" requirements, not all of the latter may be able to serve as locators for any given location-operation.)

A simulation package WHERSM was designed that simulates the operation of a Micro-Navigation and Position Location System. This program allows one to stage a wide variety of movements for any battle-field scenario; it generates the inter-unit distances, perturbs the ranges according to agreed upon error distributions and provides the links to the position location algorithms which were also developed at NBS for this purpose.

The position location algorithms fall into three distinct categories:

(1) Two-dimensional algorithms. Where the terrain is flat, no altitude reference is necessary and the resulting algorithms become two-dimensional. The sum $\Sigma w_i(r_i^2 - d_i^2)^2$ is minimized, where the sum extends over some or all reporting units and the weights are determined by the estimates of variance of each of the reported distances d_i .

(2) Pseudo three-dimensional algorithms. Where the altitudes are so small in relation to the horizontal distances that the slant distances are nearly the same as the horizontal distances it becomes necessary to reduce the position location problem to the two-dimensional one above. Not only is the three-dimensional problem ill defined mathematically, but we have also a genuine multiplicity of solutions if all locators are nearly in a plane and trying to locate a unit outside the plane. The determination of heights that are small against the horizontal distance is impossible within any reasonable limits of accuracy and the altitude must therefore be determined either by barometric reference or a reference to a digitized terrain map. For example, at a distance of 1km, a range uncertainty of only 6m leads to a height uncertainity of 110m.

(3) Fully three-dimensional algorithms are being used when the altitudes are comparable to the horizontal distances. Again, a penalty-function approach is used that minimizes $\Sigma w_i(d_i^2 - r_i^2)^2$ with suitably determined weights depending on the variance of the measurement and the sum being taken over some or all reporting units.

The following results were obtained:

o The test runs conducted on the Boston deployment show that stability of the position-location process is indeed achievable as long as some units remain fixed.

o Computer size for position location:

Runs were all carried out on the UNIVAC 1108 at NBS. For this machine, the formula for computing the number of computer words required by the position-location algorithms as they are currently programmed is as follows:

$$26N + 7R + 3874,$$

where N = number of units in the field, and R = the maximum number of ranges being reported.

As a matter of possible interest, the number of additional words required by the simulation program (into which the position-location algorithms were inserted) is:

$$91N + N_S + 4002,$$

where N_S = the number of subcycles in a cycle of the MNPLS (see Working Paper No. 9) . Furthermore, the preprocessor to the simulation requires 39,714 computer words.

o Timing for position location operation:

Several runs are available from which the time required by the algorithms can be estimated. One run was done using LSS(3d), with 20 locators;

it required approximately 16.5ms per location operation. A run with 15 locators using slant range reduction took 8-12ms ; with 20 locators, the computation took 14 to 18ms. For the Boston deployment, using 39 reporters (but probably less than 15 locators on the average because of intervisibility restrictions), 8 to 14ms were required; using 83 reporters, 17 to 25ms were required. Since the algorithms are not optimized for running time, it is reasonable to expect some decrease in these times for the operational system.

2. Description of general approach used in this report.

(1) Simulation.

The simulation of the battlefield scenario is done in our NBS-developed computer program WHERSM. This program provides for a "real life" framework for MNPLS through its capability for moving all field units along prescribed paths, or subjecting these to random changes in their direction of movement. WHERSM generates all the inter-unit distances(ranges); checks, if necessary, for intervisibility (by look up in an intervisibility matrix) ; disturbs the exact ranges according to known error distribution laws; and transmitts these "measurements" to the position location algorithms that we developed here at NBS and which are described below.

The simulation package also provides a facility for monitoring the operation of such a system under a variety of movement scenarios; and ultimately it provides for a variety of outputs that are desired for both checking out the feasibility of the entire system and what might be called the ultimate output that would be useful to the field commander who is interested in the whereabouts of his units. For details of WHERSM, the reader is referred to working paper No. 9 in the appendix of this report.

(2) Algorithms.

The basic LSS algorithm that we finally used after numerous experiments designed to eliminate competing ones is the so-called "least-squares squared" algorithm, which minimizes the penalty function

$$E = \sum_i w_i (d_i^2 - r_i^2)^2$$

where i is taken over all of the locators and w_i is a weight that increases with decreasing variance of the observation r_i, the reported distance of the locatee from the locator with index i. The quantity d_i is the calculated distance between (x_i, y_i, z_i) and (x,y,z) the estimated position of the locatee. The solution (x,y,z) is obtained iteratively from the equations $\partial E/\partial x = 0$, $\partial E/\partial y = 0$, $\partial E/\partial z = 0$ once a starting point is known.

As long as a trajectory is known for a given locatee, the initial guess is simply obtained from extrapolation. If the unit is lost, however, a set of linear equations is solved (LSL method). Details of the LSL method can be found in NBS report 10663: First Interim Progress Report on NBS Project WHERE in support of USA AMCA Project MNPLS.

The appendix contains a list of variables used in the simulation code. Since this is the part of the NBS system that interfaces with the outside world, we believe that those are the only variables that have to be listed. The variables in the algorithm parts are internal and are thus not referencable from outside.

3. Review of analyses to date.

3.1. General.

As originally conveived, the testing was to comprise two phases: initial testing to choose the best position-location algorithm, and final verification testing to determine how well the chosen algorithm would work on the Boston deployment. This neat plan did not fit the real-life flow of events, however: every stone turned over revealed two more yet unturned. The process of developing a new system is bound to contain such loops. For example, one does not know how to optimize the algorithm until one knows the system design, but the system design is sensitive to the effectiveness of the chosen algorithm. Thus the two phases gradually blurred into each other. Furthermore, in the limited time allotted for this study, it seemed that effort would be better spent getting reasonable answers to more of the real questions, rather than "optimizing" on a smaller subset of the questions while assuming particular (and questionable) answers to the others.

For these reasons, a strictly chronological account of our analyses would not be very helpful. Instead, it seems reasonable to describe the test results by first building a framework of the questions to be answered, and then describing the tests performed and the answers obtained. The next section consists of a set of questions; the following section presents the answers to date, as gleaned from the experimental program and from certain theoretical studies undertaken specifically to support that program.

3.2. Questions pertinent to the MNPL system.

The general question that constitutes the focus of a feasibility study is: Will the proposed system perform satisfactorily under the conditions of

its intended use? This question is however a very broad one; it asks essentially whether the final outcome of the feasibility study is "yes", "no" or "maybe", and thus does not give much guidance in conducting the study. Thus one is led to pose narrower queries:

a. <u>General Questions</u>

(1) Under highly optimistic conditions, will the system work? (If not, terminate the study.)

(2) Under what reasonable near-minimum conditions can the system be made to work?

(3) How well does it work?

b. <u>More Specific Questions</u>

The next question leads to a whole set of sub-questions, which are listed along with the main one.

(4) What position-location algorithm(s) should be used? For each candidate algorithm, consider:

(a) Computer time required?

(b) One-step accuracy?

(c) Error propagation (sensitivity to locators' position errors)?

(d) Sensitivity to occasional very large errors in the range measurements?

(e) Affected by dimensionality?

(f) Need an initial estimate of position?

The next set of questions refers to restrictions placed on the units in the system.

(5) How many (if any) units must remain fixed?

(6) Must they be permanently fixed in known positions?

(7) How many reporters are required, to keep the errors small?

(8) Will ground units have to carry altimeters? Instead, will a digitized map, stored in the computer, suffice?

(9) How accurate must altimeters be (both ground and airborne)?

Questions (10) - (19) refer to design of the algorithm.

(10) How often should units report?

(11) How many aircraft are required, in order to use a three-dimensional algorithm on ground units?

(12) How should one handle near-planarity (when any aircraft present are flying very low)?

(13) When should the algorithm:

(a) ignore height differences?
(b) reduce slant ranges to horizontal distances?
(c) use a 3-dimensional method?
(d) use 3d for aircraft, slant range reduction for ground units?

(14) How should the locators be chosen?

(15) How weight the locators?

(a) What is the distribution of range errors?
(b) How handle the occasional occurrence of range measurements with extremely large errors?
(c) How estimate the accuracy of an estimated position?

(16) Should the locators be "relocated" ? (i.e., should the locators' positions be adjusted, based on discrepancies between measured ranges and calculated ranges to the best estimate of locatee's position?)

(17) Use only fixed units as reporters? How soon after they become fixed?

(18) Can the algorithm determine (satisfactorily) which units are fixed? If not, could a moving/stopped indicator bit be included in the transmission?

(19) After the system has been running for a while, and errors have accumulated, is it possible to "restart" the system ---i.e. to treat the set of (currently) fixed units in toto, simultaneously, allowing all position coordinates to vary, in order to converge to good estimates for their positions?

3.3. Conduct, progress, and results of the test program.

The first few questions, above, suggest a beginning emphasis on favorable conditions. The rationale is that, when very little is known about a system, more is learned faster by starting with conditions where the system works tolerably well, and then investigating the effects of departures from these conditions. Accordingly, the first major project was to devise, develop, program, and test several candidate algorithms for determining the position of a locatee, using the (perturbed) ranges to a set of locators, and assuming the locators' positions to be known exactly. That is, attention was focussed first on questions(4) . Some of these algorithms were suitable for 2 or 3 dimensions, others were usable in 2 dimensions only. Methods considered were: Linear Method (LM), Smallest Tangent Circle (STC), Least Squares Linear (LSL), Minmax, Least Squares(LS), and Least Squares Squared (LSS). The first two methods use exactly 3 locators in 2 dimensions, or 4 locators in 3 dimensions. The next two are generalizations of the first two,

respectively, to an arbitrary number of locators. Least squares and LSS are penalty-function methods. All are fully described in earlier sections, in Working Papers Nos. 7 and 8, and/or in NBS Report 10663, First Interim Progress Report on Project WHERE in support of USAAMCA Project MNPLS.

Tests were first conducted in two dimensions. STC turned out considerably more accurate than LM, but slower; LS and LSS agreed closely, and were more accurate than any others, but slower than all but Minmax. Since Minmax was no more accurate, and several times more time-consuming, than LSS, it was dropped from further consideration. Also, since LM was inferior to STC, it was dropped. Thus the survivors at this point were: STC, LSL, LS, and LSS.

The next sequence of tests was directed to question (4)(c). For these runs, errors were applied to the positions of the locators, corresponding to a point in time after the system has been operating long enough to accumulate such errors. In these runs, STC fell down markedly. LS and LSS did better than LSL, but the latter was kept since it alone (of the three methods still in contention) does not need a starting value — i.e., is not an iterative method — and therefore can provide a position estimate when the locatee first joins the net or has been out of touch for some time.

Three-dimensional tests confirmed the pattern of results above, and also accented the importance of geometry: when the locators are nearly coplanar (collinear in 2 dimensions), LSL often produces a very poor estimate, because the planes (lines) that it tries to fit are nearly parallel. LS and LSS also have problems, but not as severe as LSL. These problems were

accentuated in 3d, simply because locators on the surface of the earth are relatively coplanar; the corresponding situation in 2d would have the locators strung out nearly on a single line, a less likely situation.

In terms of computer time LSL was best, with LSS next. Since LSS was devised to perform like LS but faster, these two were then compared for accuracy. 72 trials were run on each method, with 18 users scattered over an 8km by 16km area, with altitudes to 9km, for each of two different values of σ (the standard deviation of the range error). Each of the 18 units served as locatee 4 times for each method and each value of σ. The results were as follows: For $\sigma = 1m$, the position estimates were in error by as much as 3.32m, but the difference between the LS estimate and the LSS estimate was never more than 0.0004m. For $\sigma = 10m$, the errors reached 31.9m, but the two methods gave estimates that differed by no more than 0.045m. As a result of this test, it was concluded that LS and LSS could be used interchangeably so far as accuracy is concerned. Since LSS was considerably faster, LS was dropped. The final result thus was: Use LSS; if no starting value (initial estimate of position) is available, use LSL to obtain one. (Complications develop when there is a choice between 2- and 3-dimensional versions, as we will see.)

The only part of Question (4) yet unanswered is (4)(d): sensitivity to occasional large errors (outliers) in the measured ranges. This is a severe problem for STC and Minmax, since STC uses only three ranges and Minmax concentrates on minimizing the maximum discrepancy. For LSL, LS, and LSS, the matter is treated in a statistical sense, as is commonly done in

fitting regression curves. The true position is estimated, and then the discrepancies between measured and calculated ranges are examined. Any discrepancies significantly larger than the average are attributed to faulty measurements. These measurements are ignored, and the cycle repeated. This approach is sure to detect all really bad errors, provided they occur infrequently, unless there is exactly the minimum number of ranges necessary to produce an estimate. It was been shown to be quite efficient, in the sense that the occasional rejection of measurements "apparently" but not really contaminated with large extraneous errors does not appreciably degrade the accuracy of the estimation procedure. The actual cutoff point for rejection is a variable that will need further investigation as part of a system development effort.

Questions 5,6, and 7 pertain to the set of reporters. (The term "locators" refers to those users that do actually recieve a signal from the locatee, and do report a time-of-flight to the master computer; the term "reporters" refers to those users that have been instructed to report time-of-flight measurements to the computer. Due to the requirement for a "line-of-sight", not all of the latter may be able to serve as locators for any given location-operation.) In order to obtain broad(rather than situation-specific) answers to these questions, it was necessary to set up a number of different scenarios, with different numbers of users and different proportions stopped. This diversity was created by employing several kinds of "randomness" . Users were assigned to randomly chosen starting points within a specified rectangle, with aircraft heights specified or randomly chosen;

sequences of azimuths were chosen randomly over a 90° range, and azimuth change times were drawn from an exponential distribution. Stopping and starting times were drawn from exponential distributions, chosen to achieve specified values for the expected (long-term average) proportion of users stopped at any given instant, and for the mean times spent in "fixed" status and in "moving". Intervisibilities were similarly randomized, to achieve specified values for the average time two users maintain communication and the average time they remain out of communication. These features were used in various combinations to investigate most of the remaining questions. The results of those investigations are presented below.

If all the reporters are moving, strange things can happen. Small (random) errors in locating a given reporter propagate into the location of other reporters, so that each remains well-located relative to the others but the whole cluster is displaced(and perhaps rotated) far from its true position. This effect is reasonably well understood, and has been clearly demonstrated. One case involved 30 users, moving more or less in the same direction. 12 users were reporting, with perfect intervisibility (2 dimensions). By the time 100 location operations had been done on each, they had moved 0.9 to 4.1km . The estimated position of the cluster as a whole was 3.3km from its true location, but each element was located within 150m of its true position relative to the others. (Range errors were less than 2m in this case.) The problem can be seen clearly in the following simple example: If all of a collection of fixed users suddenly begin to move, each with the same velocity vector, or all of a collection of users moving with a common

velocity vector suddenly stop, no algorithm based on inter-unit ranges can see the change, because the inter-unit ranges don't change. Therefore, the algorithm will leave the users fixed in the first case, and will continue to move them in the second. This is a special case, of course, but the same principle applies to the average velocity vector when the units begin to move with different velocities - or more generally, to the average <u>change</u> in the velocity vector when they (more or less) all change direction at about the same time.

These results made it plain that one needs some users fixed. The next question was, how many?

Several runs were made with different sets of fixed users, under two conditions: first, that the computer does not know which users are fixed, and second, that it does know(and can use) this information. The first run, with range errors distributed uniformly from -3m to +3m , involved 15 motorized and 15 foot units; in each set of 15, 5 were fixed and reporting, 5 were moving and reporting, and the remaining 5 were moving but not reporting. After 220 cycles (109 minutes), the xy errors ranged from 500m to 10,240m, and were distributed into all four quadrants (more specifically, 14 users were placed NW of their true locations, 5 SW,10 SE, and 1 NE). For the second run, the subcycle Δt was reduced by a factor of 5, to see if locating the users more often would improve the accuracy. After 613 cycles (60 min), the errors ranged from 570m to 22000m. (The errors grew more slowly per cycle, but faster in real time.) Next, 4 of the fixed users were assumed "known" - i.e., no estimation of their positions was allowed - to provide an

anchor of sorts. After 239 cycles (116 min.), the average error over all users was 9.4m , and the maximum was 23m ; over all cycles, the average was 7.2m , the maximum 82m . But there is some evidence that the errors fluctuate considerably with time: in the set of ten cycles prior to the above (i.e., cycles 221-230), the average was 23m ; the maximum 69m !

At this point, relocation (Question 16)was tried. (Refer to Working Paper No. 8 for technical details.) Tentative conclusions from the dozen or so runs are: In the beginning, relocation increases the errors. Eventually, the errors without relocation become larger than those with relocation; however, this did not show up until about 2 hours of real time had been simulated, by which time the fast-moving users had separated from the slow-moving and fixed users by about 20km . (At this time, the overall average and maximum errors were 19m and 557m with relocation, and 34m and 1036m without. Least Squares was also tried; the average errors were comparable, but the maxima were 570 and 1695 respectively.)

Perhaps more instructive were the shorter runs with differing values of the fraction mentioned in Working Paper No. 8, which measures the extent to which relocation is carried. Runs were done with this fraction set at 0.5, 0.2, 0.1, and 0 (i.e., no relocation). The results showed steady improvement as the fraction went down, with smallest errors occurring when no relocation was done. This shows clearly that there is no advantage to relocation for the first 100 cycles (50 minutes) at least. Still open (and reasonable) is the possibility that relocation should be instituted for a time, now and then, with a small fraction, to help remove accumulated errors.

However, if there are sufficiently many fixed reporters, one can probably do better by periodically going into a survey mode wherein all the fixed units' positions are simultaneously adjusted, effectively "restarting" the system.

At this point, it seemed clear that relocation was not worth its price (more than doubling the algorithm's computer time). This conclusion was checked once more, in the next set of runs, under more realistic conditions, and was supported by the results: the errors were smaller (at 100 cycles) without relocation.

Parallel efforts by ECOM had meanwhile produced a "best engineering judgment" range-error distribution, which can be adequately modeled as "Gaussian with variance 36 m^2 ; mean 0 , except that a random 1% of measurements are inflated by about 7 meters." There will also be occasional extremely large errors; a simple preprocessor can weed most of these out in the field system, by simply checking whether the range measurement is compatible with the last known position and the velocity potential of the user. This error distribution was implemented in the balance of our tests. Also implemented were moving boundaries, which can be set to advance at a speed appropriate for foot-soldiers; any motorized unit which reaches the boundary will be reflected off it, so that the motorized units do not stray far from the foot units. Additional features implemented at this time were: a moving/stopped indicator; calculation of weights which reflect the estimated accuracy of a user's estimated position; the ability to choose as locators those users with greatest weights, or those which have been stopped

for a certain number of location-operations; improvement of estimates for the positions of fixed users, by combining successive estimates; a functional representation of terrain; use of an independent measure of the z-coordinate (the height), with random errors; algorithms to reduce slant-range distances to planar (xy-plane) distances, using the independent measure of height difference between locator and locatee; and tests to determine whether a particular location-operation should be treated by slant-range reduction or by a 3-dimensional method. Again, these features were used in various combinations for the runs recounted below.

Before these remaining runs are described, the rational used to "weight" the various range measurements will be explained. No claim is made that these weights are optimal, but it is felt that they are reasonably close to the best values, which is sufficient for a feasibility study. Suppose half the locators had the same x-coordinate as the locatee, and half had the same y-coordinate. Then (assuming ranges to be large relative to range errors)the first half of the locators is useless in determining the x-coordinate, and the second half is useless in determining the y-coordinate. Thus if there are n locators, $n/2$ are used for each coordinate, and so we have $n/2$ estimates (assumed independent) of each of the two coordinates of the locatee's position. Consider for definiteness the x-coordinate. If the locators' position errors are unbiased (i.e., have mean value equal to zero), the best estimate of the locatee's x-coordinate is a weighted average of these $n/2$ values: weighted by the reciprocals of the variances of these observations. Since the observational error is the sum of the locator's x-coordinate error

and the range measurement error, its variance is the sum of the position error variance (determined when the locator was last located) and the measurement error variance (= 36 m^2) . Thus one can calculate an estimate of the x-coordinate error variance, which in turn is used to calculate the weight for this user when he serves as a locator.

[There are two considerations which should be mentioned here. First, locators are not in general split so nicely along perpendicular lines. Does this matter? Of course it does, in the following sense: if more locators serve to estimate x than y, then x is likely to be better known - i.e., to have smaller variance — and y will be less well known. On the average, however, things even out: if azimuths (from locatee to locators)are uniformly distributed from 0 to 360° , the <u>average</u> (or expected) variance in a given direction turns out to be the value derived above. The second consideration is: What if the estimated variances for the locators are wrong? Two consequences follow: (a) To the extent that the variances are <u>different</u> multiples of the assumed values, the estimate of position is less than optimum (because the <u>true</u> relative variances should be used to get the estimate); and to the extent that the variances are (as a group) larger (or smaller) than assumed, the estimated variance of the locatee's position will be too small (or large, respectively). Point (a) is not likely to be important, since it would take very large discrepancies to affect the estimate noticeably. Point (b), on the other hand, should at least be investigated. One technique is to act as though the true variances are proportional to the estimated values, with an unknown constant of proportionality (say c) . Then c can be estimated, and used to produce a fair variance estimate. Specifically, if each

range observation has variance $c\sigma_i^2$ (σ_i^2 = location variance + range measurement variance), then the best estimate for the locatee can be found without considering c. If d_i represents the discrepancy between the calculated range from this position to the assumed position of the i-th locator and the corresponding measured range, then $[\frac{1}{n-2} \Sigma d_i^2/\sigma_i^2]$ is the (multiplicative) correction to be applied to the variance estimate already given. (If $c \sim 1$, this value should also be ~ 1.) This is a recent result and the corresponding modification has not yet been made to the algorithms.]

A new series of 2-dimensional runs was done, with 30 users. An initial set of fixed users (reporters) was specified. From then on, units became reporters as soon as they had been fixed for ten location—operations, and ceased to be reporters as soon as they began to move. Average times moving and fixed were set at 15 min., so that about 15 (half the units) were stopped at any one time. A 2½-hour (300 cycle) run was done, and resulted in an overall average error of 3.2m, and a maximum error of 35m; in the last set of 5 cycles, the average was 3.5m, the maximum 11m. Next, a run using longer cycles (90 sec. instead of 30) produced errors of 3.8 and 50.5 overall, 5.7 and 33.8 for the last 5 cycles. Next, the average times fixed and moving were set at 45 min. and this run repeated (because with 15 min. times and 1½ min. cycles, many units didn't stay stopped long enough to be much use as locators). The error figures then came back down toward those of the first run: 3.4 and 33.3 overall, 5.9 and 22.5 for the last 5 cycles.

At this point, the cycle time was returned to 30 sec., and a relocation run was tried, with fraction 0.2. It did poorly: errors 5.8 average and

35 maximum overall, with 8.4 and 17.5 for the last 5 cycles. All errors were in one quadrant, indicating that some instability had been introduced. Next, a run with fraction .001 was done. Since this fraction is so small, the magnitudes of the actual relocations will be essentially negligible; thus the effect of relocation in this run is simply to allow more iterations for the algorithms. (Remember that the number of iterations has been kept small to keep the running time down, with the thought that range errors of the order of 6m don't justify too much precision in locating the true solution of the least squares problem.) Only 100 cycles were done, with results equal (up to roundoff effects) to those without relocation. This supports the conclusion that generally, it is not worth doing too many iterations. (Of course, how many is "too many" depends very much on how good the algorithm is!)

The algorithm was modified to treat aircraft with a 3-dimensional method, while continuing to treat ground units 2-dimensionally. (This is possible since only ground units were being used as locators.) Ten aircraft were added, at (random) altitudes ranging from 128 to 2220m. That low-flying aircraft presented a problem became very obvious, since the algorithm located them where they belonged at some times, near the ground at others, and <u>below</u> ground (mirror image) at still other times. One more attempt was made in this series, using only ranges that were at most 10 times height, to ensure that not all lines of sight were flat. This helped for the second-lowest aircraft, at 339m: it had an average (x,y) error of 19m, with a maximum of 113m. However, the lowest aircraft generally did not have enough acceptable(short enough) ranges to estimate its location. It was decided, therefore, to do a separate systematic

study of aircraft location errors. This will be discussed below. The conclusion is already clear, however, that some independent height information will be required to locate low-flying aircraft, unless other aircraft are serving as locators: however, the use of aircraft as locators will probably require very stable aircraft and good tracking techniques.

A functional representation of terrain - i.e., $z = f(x,y)$, with $f(x,y)$ chosen to give a reasonable surface-was implemented at this stage, to test the use of independent pressure references (p.r.'s) as a source of height-difference information. After consultation with AMCA personnel, these were modeled as giving height with an error that had zero mean and $\sigma = 14m$ (so that the difference of 2 heights will have $\sigma = 20m$). 40 units were used, on a terrain that varied 200m in elevation, with 15 units stopped initially; again only units that had been stopped for 10 location-operations were used as locators. The x,y coordinates were obtained by reducing slant ranges to horizontal distances and applying 2d algorithms. In 2½ hours, the overall average error was 3.5m, the maximum 29.2 ; for the last 5 cycles, the average was 4.8, the maximum 13.7 . Next the p.r. σ was set to 0 , with results identical to another run where the terrain was flat and no heights were considered. For extraneous reasons, it only went 260 cycles instead of 300; it had errors of 3.2 (avg), 28.2 (max) overall; 3.3, 14.6 for the last 5 cycles. Next, the height estimates were set at 0 : i.e., the hills and valleys were ignored by the algorithm. The errors were slightly <u>smaller</u> than the ordinary run using p.r.'s: 3.35 and 28.2 overall, 4.4 and 14.3 for the last 5 cycles. This is not surprising, since the errors introduced by the

p.r.'s are comparable to the terrain differences. If p.r.'s don't perform any better, perhaps they should be ignored when only ground units are concerned. However, in particular situations, the terrain heights may introduce systematic errors if ignored, which may then propagate; the p.r. errors at least have the advantage of being random. More investigation of this possibility should be undertaken when more is known about p.r.'s . For the next run, terrain roughness was tripled: with the p.r.'s the errors were essentially the same as on the flatter terrain.

Only two sets of runs are yet to be described: the aircraft error study and the runs on the Boston terrain. The Boston scenario had no aircraft, so it was treated using slant-range reduction with p.r.'s. Two runs using the 1st Brigade constitute the final trials: one with only the fixed users reporting, and one with all units reporting. In preliminary trials, the effect of dropping those reporters which are in another brigade was seen to be minimal, except for a few front-line users which had severe intervisibility problems. The errors for the two primary runs are summarized in Table 1 .

Several points deserve to be mentioned with regard to these results. It will be noted that for 30 of the 44 units, the average error is less when all units serve as reporters. Furthermore, in 10 of the remaining 14, the higher average using all units seems to be due to poor estimates where otherwise (with only the fixed units reporting) there would be no estimates. For most of the units with maximum error above 20m, those maxima occurred with less than 2 fixed locators (i.e., where no estimate would be available using only fixed locators). The greatest error occurred for unit 28, and was 108m ; it

(as well as the two unsuccessful location-operations for this unit) involved 21 locators, including 2 fixed locators. (It is interesting to note that relatively large errors will occasionally occur even with a system " in control.") Using only the fixed locators, that unit never got lost and never had an error greater than 18m . This anomalous behavior suggests that perhaps one should use only the ranges from fixed units, if there are enough, and should use at most a small number of ranges from moving units in any case. Also worthy of note is the fact that apart from this unit, every other maximum error greater than 31m occurred with 4 or fewer locators.

As mentioned earlier, it became obvious very quickly that low-flying aircraft could not be tracked from the ground. To study the effects on accuracy of such variables as number of locators and height, a separate study was done on aircraft. 20 reporters were randomly positioned on a 10 X 10km piece of terrain, and kept fixed. (See Fig. 1.) Ten(simulated) aircraft were started at the southwest corner, at altituted of 100, 250, 400, 550, 700, 1000, 1500, 2000, 2500, and 3500m. Each flew a level course, with constant x-velocity; the course reflected several times off both north and south boundaries, terminating at the eastern boundary, systematically covering the entire terrain. (Shown on Fig. 1.) Each aircraft went through either 1230 or 2790 location-operations. The average error and average squared error were tabulated, both in the x,y-directions and in z, by number of locators, for each aircraft (i.e., each height). These tabulations were then studied, and regression techniques applied.

The first run counted locators for which the range was less than 8 X height, 12 X height, 16 X height, etc., up to 80 X height, stopping as soon as 4 or more locators were found. It then used LSS (3d) . Table 2 shows the numbers of trials for which the algorithm was and was not successful by aircraft height and number of locators. There are few cases where an aircraft at 700m or higher was not located, but things are not too good below that height. The error statistics bore this out; the maximum z-errors for the 10 aircraft were 1880, 192, 97, 83, 39, 22, 14, 15, 16, 19m (in order of increasing height), while the maximum x,y-errors were 26339, 2373, 337, 600, 72, 36, 29, 23, 27, 18m respectively. Since 3d obviously does not work with

low-flying aircraft, an examination was made of the dependence of average error on number of locators and height, for aircraft heights of 550m and above. Regression analyses showed that the average x,y-error could be represented adequately by the expression $\overline{E}_{x,y} = 16 - .9n - \frac{2500}{\text{alt.}} + .017n^2$, and average z-error by $\overline{E}_z = 31 - 3n + .08\ n^2$, where n is the number of locators used. (Note that these expressions were developed for aircraft higher than about 500m.) The values given by these expressions agree with the values obtained in the experiment within an average deviation of 0.5m for x,y-error, and 2m for z-error. (The experiment values for x,y-error ranged from 3 to 10m; for z-error, from 4 to 29m.)

Finally, a similar run was done using slant range reduction, with aircraft at 100, 250, 400, and 550m altitudes, using p.r.'s, and only those reporters with elevation angles $< 20°$. The average x,y-errors were 3.2, 3.3, 3.4, and 3.7m respectively; the maxima were 15, 15, 15, and 17m respectively. This suggests that one should use slant range reduction up to at least 600m, and use LSS 3-d or slant range reduction above that value depending on whether there are many good ranges at large elevation angles or many at small elevation angles, and on the quality of altitude information available at different heights. (It is understood that p.r.'s of the accuracy contemplated herein are limited in altitute, but will cover at least 0 to 600m.)

Table 1

Error summaries, Boston runs.

Unit no.	All reporting			Fixed units reporting		
	Errors (m) Average	Max.	No. of times not found	Errors (m) Average	Max.	No. of times not found
7	3.8	16.8	49	5.0	21.7	
8	5.9	39.8		7.5	33.4	105
9	8.6	62.1		7.0	39.6	77
10	3.7	12.0		5.9	20.2	
11	3.4	9.3	20	5.1	22.2	
12	4.2	16.8		4.7	14.2	
13	3.7	11.2	104	5.7	21.1	104
14	3.5	10.0		6.8	32.4	
15	3.7	18.7		4.8	18.5	
16	3.7	23.5	1	4.8	16.6	
17	5.1	27.0		5.1	25.4	
18	4.9	20.4		5.3	29.5	
19	5.0	18.0		5.1	23.3	
20	3.9	15.4		5.8	30.7	
24	3.6	24.1		4.3	25.8	
25	4.3	13.3		5.0	16.3	73
26	2.9	10.0		4.1	11.8	
27	3.4	10.1		4.6	21.9	
28	6.3	108.1	2	4.5	17.7	
29	3.4	10.9		3.4	15.0	
30	5.9	51.6		4.3	12.9	79
31	7.7	79.9	8	4.2	12.3	61
32	3.7	14.1		5.1	33.9	
33	3.3	12.2		3.6	12.5	
34	3.7	12.5		3.5	20.4	
35	3.2	11.4	1	3.4	10.5	
36	3.6	11.1	1	3.1	12.8	
37	3.6	11.6		3.3	13.4	
38	2.7	10.1		3.4	10.7	
42	3.9	12.8		3.4	8.5	102
43	3.6	10.9	67	4.5	16.8	67
44	5.7	54.8	3	4.3	12.9	57
45	4.9	23.6		4.7	12.5	69
46	3.5	13.8		3.8	11.7	67
47	9.4	101.3		4.0	11.9	55
48	4.8	27.7	3	4.4	14.2	77
49	3.9	12.8	2	4.2	13.4	2
50	8.3	88.3		5.9	26.7	30
51	5.0	57.5		4.4	19.2	20
52	3.7	14.4		5.2	17.5	
53	3.8	13.5		4.7	23.1	77
54	3.8	13.1		4.6	25.9	105
55	3.9	17.2		5.4	25.5	
56	4.2	16.4		5.7	22.0	
All	4.5	108.1		4.7	39.6	

Table. 2.

Lost and found summary aircraft study

Aircraft height

Number of Locators	100	250	400	550	700	1000	1500	2000	2500	3500
3	4/0									
4	136/640	84/396	44/175	18/80	7/52					
5	40/255	48/233	28/189	18/92	3/20					
6	18/98	18/163	8/131	9/103	0/11					
7	7/24	14/98	22/192	16/128	4/32	0/1				
8	1/5	6/57	4/122	2/119	0/94	0/2				
9	0/2	6/83	3/189	5/159	1/57	0/1				
10		1/22	2/95	1/115	1/70	0/3				
11		0/1	0/16	0/85	1/98	0/0				
12			0/10	0/94	1/130	1/28				
13				0/90	0/44	0/13				
14				0/70	0/65	0/36				
15				0/16	2/228	0/184				
16				0/10	0/133	0/154				
17					0/44	0/212				
18					0/33	0/231				
19						0/232	0/67			
20						0/132	1/1162	2/1228	0/1230	0/1230
Total Lost	206	177	111	69	19	1	1	2	0	0

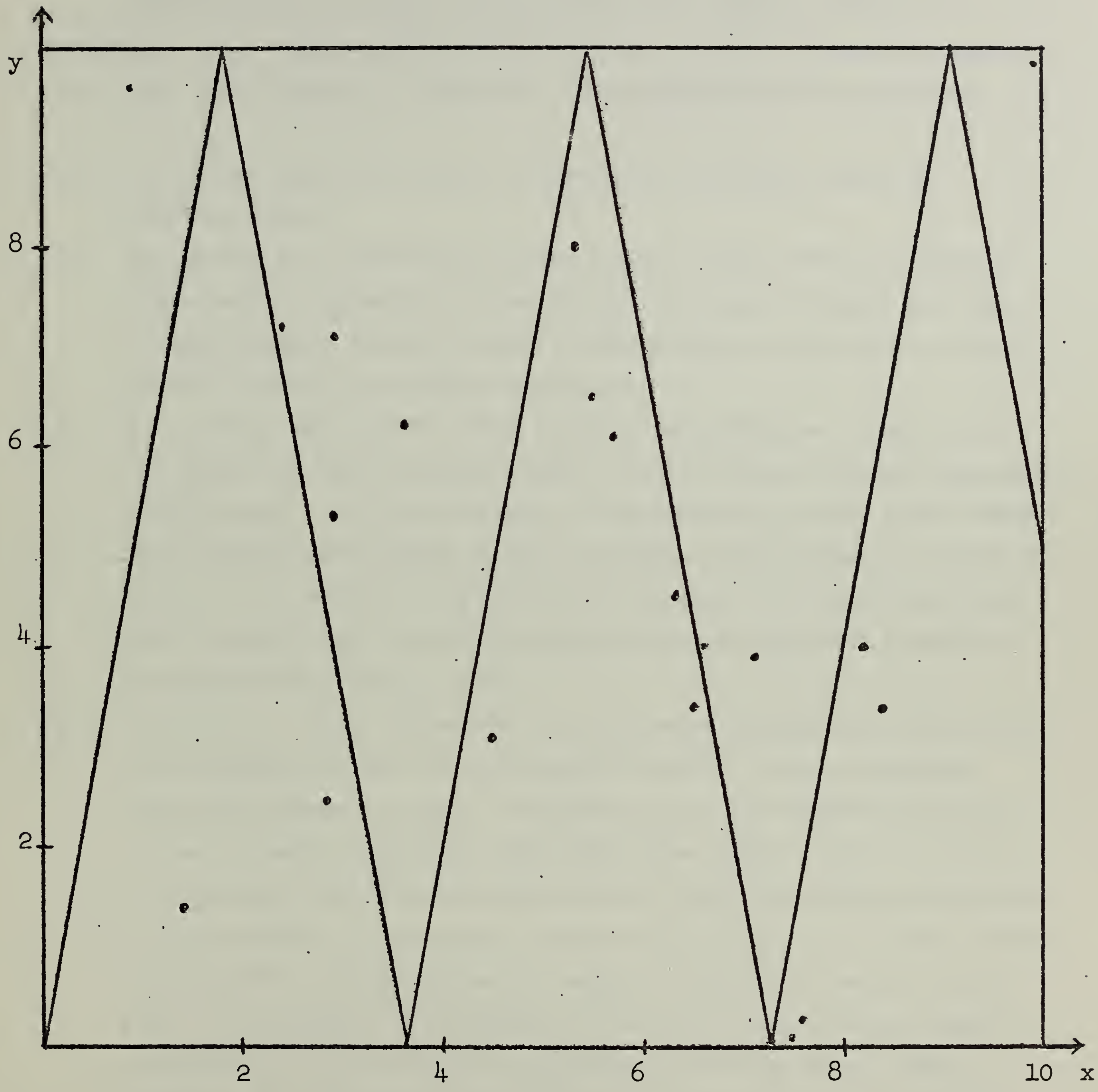


Fig. 1

Locators' positions and aircraft path, Aircraft error study.

4. Conclusions

The following represent conclusions with varying degrees of certainty; some only say what might be true, or what should be checked out. They are organized to parallel the earlier list of questions.

(1) The system will work under reasonable conditions, such as outlined below.

(2) The system will generally locate ground units within reasonable error bounds whenever such units have at least 4 fixed locators or well-located moving locators (which may be airplanes, as long as they are not too nearly overhead).

(3) x, y-errors will be the same order of magnitude as range errors, for about 4-6 good locators; the accuracy (error) should improve as $1/\sqrt{n-2}$ where n is the number of locators, up to the point where the computer word length begins to limit the algorithm's accuracy. The z-errors will be comparable for genuine 3-d situations; they will depend solely on the accuracy of the independent pressure reference when that is used.

(4) Use LSS; to find a starting value when extrapolation from previous estimates is not to be trusted, use LSL. Use an outlier-rejection scheme in each, in addition to a preliminary screening of the incoming ranges to eliminate gross (10%) errors. Use the 2-dimensional version on ground units, with slant-range reduction if the terrain is not smooth (unless all but 1 or 2 of the locators are aircraft at high elevation angles, say 60° or more, in which case 3-d treatment is required); also use 2-d with slant-range reduction for aircraft with altitudes less than about 1000m, possibly in conjunction with 3d for aircraft between (say) 600 and 1000m. Use 3d for aircraft over 1000m. (Note: These numbers depend on the spread of ground units; the values given are appropriate for the Boston deployment.)

(5) Fixed units should constitute a large fraction (at least 1/2) of the set of reporters. The system will work with fewer, but the possibility exists that the set of users' estimated positions will "break its moorings" and wander away, as one entity, from the true position.

(6) It is not necessary that the same units be fixed throughout the period of time that the system is in use. (In which case it follows that the positions in which they stop will not be "known", only estimated.)

(7) This question, concerning the number of reporters needed, is answered under (3) and (4) above.

(8) It is understood that maps will not be generally available, but this is fortunate in a way, since their use extracts a very heavy penalty in computation time. Altimeters (or more accurately, "pressure references", or p.r.'s) will be necessary when aircraft at low elevation angles are working with ground units, either as locatee or as locators, and will be helpful whenever the terrain is rough enough that altitude differences among ground units are as large as the error σ of the p.r.

(9) An accuracy of 14m (σ) in p.r. values (that is, σ = 20m for the difference between the heights of two units) will enable the system to maintain position errors comparable to range-measurement errors. Better p.r.'s will of course permit more accurate position estimates.

(10) Reporting once per 30sec. seems reasonable, except: a) Moving locators should report more frequently, especially aircraft; and b) if the system is to be used for aircraft guidance, close support, or fire control, then all aircraft should report more often, say once every second or two.

(11) To use genuine 3d, a minimum of 3 locators is required; only one need be an aircraft. However, at least 5 (including 2 aircraft) is advisable, so that bad measurements may be detected. Note that aircraft at low elevation angles count as ground units here.

(12) and (13) - See answer to (4).

(14) Use as many locators as possible, but properly weighted.

(15) Weight fixed, well-known locators considerably higher than others. (See discussion of weights earlier in this section.)

(16) Continual relocation is not advisable. Periodic relocation may be worthwhile; compare with (19) below.

(17) Do not use only fixed units as reporters, except in situations with unusually good intervisibility. (There are too many times when a unit won't have enough fixed locators.)

(18) There are indications that relatively large errors will occur when a locator begins to move, if the system continues to treat his position as fixed. Also, whatever time it takes to establish that a unit has stopped becomes a delay in "fixing" that unit's position so that he can become a good locator. Therefore a definite penalty is attached to not having a moving/stopped indicator. The size of this penalty can only be established after the appropriate filtering techniques are implemented. Thus a complete answer awaits a development study.

(19) Indications are that it is possible to resurvey the set of fixed units. In the course of system development, attention should be paid to developing, implementing, and testing a multivariable least-squares approach, allowing the adjustment of all but a few of the fixed reporters' position estimates simultaneously. This technique would be applied periodically during long battles, when the set of currently-fixed reporters becomes considerably different from the set of initially-fixed reporters. (This technique might be either a supplement to, or a replacement for, the periodic use of relocation; see (16) above.)

Appendix

PROJECT "WHERE" : WORKING PAPER NO. 6

ERROR PROPAGATION FOR ONE-DIMENSIONAL LOCATION OF TWO UNITS

M.H. Pearl

March , 1972

ABSTRACT

The analysis of error propagation for position location in a one-dimensional geometry, assuming Gaussian error distributions and maximum-likelihood estimation (i.e., location by minimizing a particular quadratic penalty function), is mathematically tractable. Although this scenario is unrealistic, "solving" it in closed form may yield insights applicable to more general situations. Working Paper No. 4 developed formulas for the law of error propagation through a single location-operation in such a situation. The present note proceeds to treat a <u>series</u> of location-operations for the (simplest) case where <u>only two units</u> are involved, and shows that in this case the precision of location stabilizes over time. It is planned next to seek to extend this encouraging result to more than two units.

Note: Project working papers are informal documents prepared to facilitate discussion and communication; they may contain tentative or relatively unchecked material.

0. INTRODUCTION

Following the terminology of other papers in this series, we shall use the term location-operation to refer to the process of imputing a position to one unit (the locatee), on the basis of its measured distances from a set of other units (the observers) whose estimated positions are at hand. Both the distance measurements, and the estimates of the observers' locations, are subject to error. The purpose of Project "WHERE" can be roughly described, from a technical viewpoint, as that of determining (for a variety of scenarios) the propagation of location-error over a series of such operations, in which the role of "locatee" varies cyclically among a large set of units, many of which may be in motion during part or all of the series. As a principal tool for this purpose, project staff are developing the various components of a computerized simulation ("WHERESIM").

While such a simulation is needed for the examination of error-propagation in "general" or "unpatterned" situations, it is plausible that valuable insights can be obtained from the study of special situations which are simple enough to admit direct mathematical analysis. Working Paper No. 4[(1)] initiated one such study, whose key simplifying assumptions are those of

(a) one-dimensional geometry, and

(b) Gaussian error distributions.

Specifically, WP4 derived the error-propagation formula under those assumptions for a single location-operation, leaving for a next phase of work the recursive application of this formula to determine error build-up through a sequence of location-operations, with various units serving successively as locatee. The present paper carries out this further analysis for an especially simple situation, that in which only two units are involved. The units thus alternate between the roles of locatee and observer.

---

(1) Project "WHERE" Working Paper No. 4, Error Propagation for Position Location in One Dimension, M.H. Pearl, December, 1971. To be referred to in what follows as "WP4."

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

The analysis in WP4 was divided into two cases, according as the single location-operation considered there did or did not make use of a prior estimate of the locatee's position. From the results in WP4 (see its last page) and a variety of additional evidence developed in other parts of Project "WHERE" , it now seems clear that such prior estimates should indeed be employed, and so their use is assumed in what follows. Specifically, each such prior estimate is obtained from (i) the unit's estimated position at the end of the last previous location-operation and (ii) its velocity of motion(assumed known exactly) in the interim.

With the decision between "prior-using" and "prior-ignoring" thus made in favor of the former, the analysis of the present paper is split into two cases along a different line of cleavage. In Working Paper No.2[(2)], it was suggested that the measurements and calculations taking place during a location-operation might be exploited not only to locate the locatee, but also to revise the currently assumed positions of the observers, so that one would in effect have a "location-and-revision" operation. Accordingly, both "with revision" and "without revision" scenarios will be analyzed below. The most significant finding is that (in both cases) the precision of location stabilizes over time. (The natural next step is to investigate whether this encouraging result combines to hold for more than two units.)

Although familiarity with WP4 is not essential, it will be quite helpful in reading the present text, which is organized as follows. Section 1, appearing next, describes the underlying mathematical models involved and sets up most of the necessary notation. Section 2 presents the error-propagation analysis for a sequence of "location-with-revision" operations, while Section 3 does the same for the "without revision" case.

---

(2) Project "WHERE" Working Paper No. 2, An Iterative Approach to the Location-Operation, A.J. Goldman, October, 1971.

1. MATHEMATICAL FORMULATION

The analysis deals with 2 units located on a line (i.e., the geometry is one-dimensional). The relevant time period begins at time $t_o = 0$, and the successive location-operations occur at times t_j $(j = 1,2, \ldots)$ where

$$0 = t_o < t_1 < t_2 < \ldots .$$

For $i = 1,2$ and $j = 0,1,2,\ldots,$ we set

$$U_{ij} = \text{true position of unit } i \text{ at time } t_j ,$$

so that

$$D_j = U_{2j} - U_{1j} \tag{1.1}$$

$$= \text{true (signed) distance between the units at time } t .$$

The U_{ij}'s are unknown to the location-system, which is however "given" at time t_j quantities,

$$u_{ij} = \text{prior estimate of } U_{ij} \text{ at time } t_j ,$$

to be discussed further below. The information on which the j-th location-operation is based consists of u_{1j}, u_{2j}, and

$$d_j = \text{measured value of } D_j .$$

This measurement is assumed unbiased, i.e. $E(D_j) = d_j$, and also normally distributed, so that the random measurement error

$$\delta_j = d_j - D_j \tag{1.2}$$

is Gaussian and satisfies (for some $\sigma > 0$)

$$E(\delta_j) = 0 \quad , \quad E(\delta_j^2) = \sigma^2 . \tag{1.3}$$

Note that our notation involves σ and not σ_j ; i.e., we assume no systematic change over time in the quality of the distance measurements.

The initial position-estimates, u_{10} and u_{20} , are assumed to be unbiased, i.e. $E(u_{io}) = U_{io}$, and also to be normally distributed; in terms of the vector u_o^T defined by

$$u_o^T = (u_{10}, u_{20}) \quad ,$$

we set

$$Q_o = \mathrm{var}(u_o) = \begin{bmatrix} \mathrm{var}(u_{10}) & \mathrm{cov}(u_{10},u_{20}) \\ \mathrm{cov}(u_{10},u_{20}) & \mathrm{var}(u_{20}) \end{bmatrix} ; \tag{1.4}$$

the matrix Q_o (describing the quality of our starting position-data), as well as σ (which describes the quality of distance measurements), are the basic inputs to our analysis of error propagation.

It is assumed that between t_{j-1} and t_j , unit i moves with known velocity v_{ij} ; it therefore covers a known distance

$$\Delta_{ij} = v_{ij}(t_j - t_{j-1}) . \tag{1.5}$$

Thus the prior estimates of the two units' positions, just before the first location-operation, are

$$u_{i1} = u_{io} + \Delta_{io} \qquad (i = 1,2) ;$$

it follows from (1.4) that

$$Q_o = \text{var}(u_1) \quad , \tag{1.6}$$

where we have used for $j = 1$ the notation

$$u_j^T = (u_{1j}, u_{2j}) \; .$$

Let us define the vector U_j by

$$U_j^T = (U_{1j}, U_{2j}) \; .$$

The calculations of the j-th location-operation, in our present model, yield

$$U_j^* = \text{maximum-likelihood estimate of } U_j \text{ based on the data } u_j \text{ and } d_j \;;$$

this estimate is known (cf. WP4) to be unbiased, and we put

$$Q_j^* = \text{var}(U_j^*) \quad . \tag{1.7}$$

It is at this point that the "with revision" and "without revision" cases diverge. In the <u>first</u> case, U_j^* is accepted as the vector of location estimates after the j-th location-operation; thus Q_j^* represent the quality of our position locations at time j, so that our aim is to determine these matrices Q_j^* (and in particular, their behavior as j increases). Moreover, in this case the "next" set of prior estimates is given by

$$u_{j+1} = U_j^* + \Delta_j \quad , \tag{1.8}$$

where we have defined Δ_j by

$$\Delta_j^T = (\Delta_{1j} \; ; \Delta_{2j}) \; .$$

In the <u>second</u> ("without revision") case, the vector of location estimates after the j-th operation, denoted U_j^{**} with components U_{1j}^{**} and U_{2j}^{**}, is taken as

$$U_{1j}^{**} = U_{1j}^{*} , \quad U_{2j}^{**} = u_{2j} \qquad \text{if } j \text{ odd (unit 1 the locatee)} ,$$
$$U_{1j}^{**} = u_{1j} , \quad U_{2j}^{**} = U_{2j}^{*} \qquad \text{if } j \text{ even (unit 2 the locatee)} , \tag{1.9}$$

so that the prior estimate of the observer's position indeed goes unrevised. The "updating" formula (1.8) is of course replaced by

$$u_{j+1} = U_j^{**} + \Delta_j \quad . \tag{1.10}$$

And now the matrices

$$Q_j^{**} = \mathrm{var}(U_j^{**}) \tag{1.11}$$

are the matrices which are to be investigated, and whose limiting behavior (for large j) is to be investigated.

The analyses of the two sequences $\{Q_j^{*}\}$ and $\{Q_j^{**}\}$ of matrices, are the topics of Section 2 and 3 respectively.

2. ANALYSIS ASSUMING REVISION

It is convenient to define the random error vectors

$$\epsilon_j = u_j - U_j \quad . \tag{2.1}$$

Then the likelihood function for the <u>first</u> location-operation is

$$L = K \exp\{-\tfrac{1}{2}(\epsilon_1^T Q_0^{-1} \epsilon_1 + \sigma^{-2} \delta_1^2)\}$$

where K is a positive constant, so that maximizing L is equivalent to minimizing

$$L^* = \tfrac{1}{2}(\epsilon_1^T Q_0^{-1}\epsilon_1 + \sigma^{-2} \delta_1^2) \quad . \tag{2.2}$$

From (1.1) and (1.2) ,

$$\delta_1 = d_1 + [1, -1]U_1 \quad ,$$

and from this, (2.1), and (2.2), it follows that the maximum-likelihood estimate U_1^* is the value of U_1 which minimizes

$$\begin{aligned} L^* = \tfrac{1}{2}\{&(u_1 - U_1)^T Q_0^{-1}(u_1 - U_1) \\ &+ \sigma^{-2}(d_1 + [1,-1]\, U_1)^T(d_1 + [1,-1]\, U_1)\} \quad . \end{aligned} \tag{2.2a}$$

We next apply the (straightforwardly verified) formula

$$\partial[(AX + B)^T S(AX + B)]/\partial X = 2A^T S(AX + B) \tag{2.3}$$

where X is a column vector and S is a symmetric matrix. Taking $A = I$ (the 2×2 identity matrix), $B = u_1$ and $S = Q_0^{-1}$, we obtain

$$\partial\{u_1-U_1)^T Q_0^{-1}(u_1-U_1)\}/\partial U_1 = 2Q_0^{-1}(U_1-u_1) \; ; \tag{2.4}$$

taking $A = [1,-1]$, $B = d_1$, and $S = I$, we obtain

$$\partial\{(d_1+ [1,-1]U_1)^T(d_1+[1,-1]U_1)\}/\partial U_1 = 2[1,-1]^T(d_1+[1,-1]U_1)$$

$$= 2\{[1,-1]^T d_1+ NU_1\} \; , \tag{2.5}$$

where we have put

$$N = [1,-1]^T [1,-1] = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \; . \tag{2.6}$$

Thus the equation $\partial L^*/\partial U_1 = 0$, to be satisfied by $U_1 = U_1^*$, reads

$$Q_0^{-1}(U_1-u_1) + \sigma^{-2}[1,-1]\, d_1 + \sigma^{-2} NU_1 = 0 \; ;$$

setting

$$H_o = \sigma^{-2} Q_o N \; , \tag{2.7}$$

we see that this is equivalent to

$$(I + H_o)U_1^* = u_1 + \sigma^{-2} Q_o[1,-1]\, d_1 \; ,$$

so that

$$U_1^* = (I + H_o)^{-1} (u_1 + \sigma^{-2} Q_o[1,-1]\, d_1) \; . \tag{2.8}$$

Since $\operatorname{cov}(u_1,d_1) = 0$, we have by (1.6) and (1.8)

$$\operatorname{var}(u_1 + \sigma^{-2} Q_o[1,-1]\, d_1)$$

$$= Q_o + \sigma^{-4} Q_o[1,-1]\sigma^2 [1,-1]^T Q_o$$

$$\partial\{u_1-U_1)^T Q_0^{-1}(u_1-U_1)\}/\partial U_1 = 2Q_0^{-1}(U_1-u_1) \; ; \tag{2.4}$$

taking $A = [1,-1]$, $B = d_1$, and $S = I$, we obtain

$$\begin{aligned}\partial\{(d_1+ [1,-1]U_1)^T(d_1+[1,-1]U_1)\}/\partial U_1 &= 2[1,-1]^T(d_1+[1,-1]U_1) \\ &= 2\{[1,-1]^T d_1+ NU_1\} \; ,\end{aligned} \tag{2.5}$$

where we have put

$$N = [1,-1]^T [1,-1] = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \; . \tag{2.6}$$

Thus the equation $\partial L^*/\partial U_1 = 0$, to be satisfied by $U_1 = U_1^*$, reads

$$Q_0^{-1}(U_1-u_1) + \sigma^{-2}[1,-1]\, d_1 + \sigma^{-2} NU_1 = 0 \; ;$$

setting

$$H_o = \sigma^{-2} Q_o N \; , \tag{2.7}$$

we see that this is equivalent to

$$(I + H_o)U_1^* = u_1 + \sigma^{-2} Q_o[1,-1]\, d_1 \; ,$$

so that

$$U_1^* = (I + H_o)^{-1} (u_1 + \sigma^{-2} Q_o[1,-1]\, d_1) \; . \tag{2.8}$$

Since $\mathrm{cov}(u_1,d_1) = 0$, we have by (1.6) and (1.8)

$$\begin{aligned}&\mathrm{var}(u_1 + \sigma^{-2} Q_o[1,-1]\, d_1) \\ &= Q_o + \sigma^{-4} Q_o[1,-1]\sigma^2 [1,-1]^T Q_o\end{aligned}$$

$$= Q_o + \sigma^{-2} Q_o N Q_o = (I + H_o)Q_o \ . \tag{2.9}$$

Since (2.8) yields

$$\begin{aligned} Q_1^* &= \operatorname{var}(U_1^*) \\ &= (I + H_o)^{-1} \operatorname{var}(u_1 + \sigma^{-2} Q_o[1,-1]d_1)(I + H_o)^{-T} \ , \end{aligned}$$

it follows from (2.9) that

$$Q_1^* = Q_o(I + H_o)^{-T} \ , \tag{2.10}$$

which is equivalent to

$$Q_1^* = (I + H_o)^{-1}Q_o \ . \tag{2.11}$$

We turn now to the <u>second</u> location-operation. By (1.8),

$$\operatorname{var}(u_2) = \operatorname{var}(U_1^*) = Q_1^* \ ,$$

i.e. Q_1^* plays the same role now that was played by Q_o previously. Thus, if by analogy with (2.7) we put

$$H_1^* = \sigma^{-2} Q_1^* N \ ,$$

then by analogy with (2.10) and (2.11) we will obtain

$$Q_2^* = Q_1^*(I + H_1^*)^{-T} = (I + H_1^*)^{-1} Q_1^* \ .$$

More generally, the transition from Q_j^* to Q_{j+1}^{**} is given by the two equations

$$H_j^* = \sigma^{-2} Q_j^* N \tag{2.12}$$

$$Q_{j+1}^* = Q_j^*(I + H_j^*)^{-T} = (I + H_j^*)^{-1} Q_j^* \quad , \tag{2.13}$$

with initial conditions $Q_o^* = Q_o$ and $H_o^* = H_o$.

Using these results, we will prove by induction on j that

$$Q_j^* = (I + jH_o)^{-1} Q_o = Q_o(I + jH_o)^{-T} \quad . \tag{2.14}$$

By (2.11), it is true for $j = 1$. Suppose it is true for some particular value of j ; we will prove it correct for $j + 1$. First, by (2.12) and (2.14),

$$H_j^* = \sigma^{-2}(I + jH_o)^{-1} Q_o N = (I + jH_o)^{-1} H_o = H_o(I + jH_o)^{-1} \quad . \tag{2.15}$$

If we set

$$Z_j = (I + H_j^*)^{-1} \quad ,$$

then

$$\begin{aligned} I &= Z_j(I + H_j^*) \quad , \\ &= Z_j + Z_j H_o (I + jH_o)^{-1} \quad , \end{aligned}$$

so that

$$(I + jH_o) = Z_j(I + jH_o) + Z_j H_o = Z_j[I + (j+1)H_o]$$

and thus

$$Z_j = (I + jH_o)[I + (j+1)H_o]^{-1} = [I + (j+1)H_o]^{-1}(I + jH_o);$$

it follows from (2.13 - 15) that

$$Q^*_{j+1} = Z_j Q^*_j$$

$$= [I+(j+1)H_o]^{-1}[I + jH_o][I + jH_o]^{-1} Q_o.$$

$$= [I+(j+1)H_o]^{-1} Q_o \quad ,$$

so that (2.14) also applies for $j+1$. The induction proof is complete.

Note that with (2.6), (2.7) and (2.14), we have succeeded in determining the matrices Q^*_j explicitly in terms of the basic data Q_o and σ . We now prove that the limit

$$Q^*_\infty = \lim_{j \to \infty} Q^*_j \tag{2.16}$$

exists, i.e. that <u>the precision of position-location stabilizes over time</u>, and moreover we shall evaluate Q^*_∞ explicitly.

Let

$$Q_o = \begin{bmatrix} q_1 & q \\ q & q_2 \end{bmatrix} ;$$

then by (2.6 - 7)

$$I + jH_o = \begin{bmatrix} 1 + \sigma^{-2}j(q_1-q) & \sigma^{-2}j(q - q_1) \\ \sigma^{-2}j(q - q_2) & 1 + \sigma^{-2}j(q_2 - q) \end{bmatrix} ,$$

$$= \sigma^{-2}j \begin{bmatrix} (\sigma^2/j) + (q_1-q) & q - q_1 \\ (q - q_2) & (\sigma^2/j) + (q_2-q) \end{bmatrix} . \tag{2.17}$$

The coefficient of $\sigma^{-2}j$ is a matrix with determinant

$$(\sigma^4/j^2) + (\sigma^2/j)(q_1 + q_2 - 2q) = (\sigma^2/j)[(\sigma^2/j) + (q_1 + q_2 - 2q)] ,$$

and thus with inverse matrix

$$(\sigma^2/j)^{-1} [(\sigma^2/j) + (q_1 + q_2 - 2q)]^{-1} \begin{bmatrix} (\sigma^2/j) + (q_2 - q) & q_1 - q \\ q_2 - q & (\sigma^2/j) + (q_1 - q) \end{bmatrix} .$$

It follows from (2.17) that

$$(I + jH_o)^{-1} = [(\sigma^2/j) + (q_1 + q_2 - 2q)]^{-1} \begin{bmatrix} (\sigma^2/j) + (q_2 - q) & q_1 - q \\ q_2 - q & (\sigma^2/j) + (q_1 - q) \end{bmatrix} . \tag{2.18}$$

In particular,

$$\lim_{j \to \infty} (I + jH_o)^{-1} = H^* \tag{2.19}$$

exists and is given by

$$H^* = (q_1 + q_2 - 2q)^{-1} \begin{bmatrix} q_2 - q & q_1 - q \\ q_2 - q & q_1 - q \end{bmatrix} ; \tag{2.20}$$

thus by (2.14) the limit Q_∞^* exists and is given by

$$Q_\infty^* = H^* Q_o \quad .$$

$$= (q_1+q_2-2q)^{-1} (q_1 q_2 - q^2) J \ , \qquad (2.21)$$

where J is a 2×2 matrix of ones. It is somewhat surprising that the limit is independent of σ.

A natural question is whether the position-locations grow increasingly precise over time, i.e. whether the diagonal entries of $Q_j^* = \mathrm{var}(U_j^*)$ are decreasing functions of j. By (2.14) and (2.18), the (1,1) entry of Q_j^* is

$$\mathrm{var}(U_{1j}^*) = q_1[(\sigma^2/j) + q_1^{-1}(q_1q_2 - q^2)]/[(\sigma^2/j) + (q_1+q_2-2q)] \quad . \qquad (2.22)$$

Now it is readily verified that the function

$$f(x) = (x+a)/(x+b) = 1 - (b-a)/(x+b)$$

is increasing or decreasing according as (b-a) is positive or negative. Applying this with

$$x = \sigma^2/j \quad , \quad a = q_1^{-1}(q_1q_2 - q^2) \ , \quad b = q_1 + q_2 - 2q \quad ,$$

we see that $\mathrm{var}(U_{1j}^*)$ is a non-increasing function of j if

$$q_1 + q_2 - 2q \geq q_1^{-1}(q_1q_2 - q^2) \quad , \qquad (2.23)$$

which indeed holds since it is equivalent to $(q_1 - q)^2 \geq 0$. Analogously, $\mathrm{var}(U_{2j}^*)$ is a non-increasing function of j.

3. ANALYSIS FOR LOCATION WITHOUT REVISION

We now turn to the matrices

$$Q_j^{**} = \mathrm{var}(U_j^{**}) \qquad (3.1)$$

corresponding to the location-without-revision scenario. The analysis is more complicated here, since the two units no longer figure symmetrically in each location-operation (recall that unit 1 is the locatee in odd-numbered operations, unit 2 the locatee in even-numbered ones).

It is convenient to repeat from Section 2 the definitions

$$N = [1,-1]^T [1,-1] = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} , \qquad (3.3)$$

$$H_o = \sigma^{-2} Q_o N , \qquad (3.3)$$

and to introduce the matrices E_{ik} ; E_{ik} has entry 1 in the (row i , column k) position, and all other entries zero. Thus, for example, the consequence

$$(U_1^{**})^T = [U_{11}^{*}, 0]^T + [0,u_{21}]^T$$

of (1.9) can be written

$$U_1^{**} = E_{11}U_1^{*} + E_{22}u_1 \quad . \qquad (3.4)$$

From this and (3.1) it follows that

$$Q_1^{**} = E_{11} \operatorname{var}(U_1^*) E_{11} + E_{22} \operatorname{var}(u_1) E_{22}$$

$$+ E_{11} \operatorname{cov}(U_1^*, u_1) E_{22} + E_{22} \operatorname{cov}(u_1, U_1^*) E_{11} . \tag{3.5}$$

By (2.8) and (2.11)

$$\operatorname{cov}(U_1^*, u_1) = (I + H_o)^{-1} Q_o = Q_1^* ,$$

so that (3.5) yields

$$\begin{aligned} Q_1^{**} &= E_{11} Q_1^* E_{11} + E_{22} Q_o E_{22} \\ &\quad + E_{11} Q_1^* E_{22} + E_{22} Q_1^* E_{11} \\ &= Q_1^* + E_{22}(Q_o - Q_1^*) E_{22} . \end{aligned}$$

By use of (2.11), this reads

$$Q_1^{**} = (I + H_o)^{-1} Q_o + E_{22}[I - (I + H_o)^{-1}] Q_o E_{22} . \tag{3.6}$$

The second location-operation follows the same logic as the first, except that Q_1^{**} plays the part of Q_o and that the roles of the two units are interchanged. Thus, by analogy with (3.3) and (3.6), we have

$$H_1^{**} = \sigma^{-2} Q_1^{**} N , \tag{3.7}$$

$$Q_2^{**} = (I + H_1^{**})^{-1} Q_1^{**} + E_{11}[I - (I + H_1^{**})^{-1}] Q_1^{**} E_{11} . \tag{3.8}$$

More generally, the appropriate recursion equations are

$$H_j^{**} = \sigma^{-2} Q_j^{**} N \qquad (3.9)$$

$$Q_{2j+1}^{**} = (I + H_{2j}^{**})^{-1} Q_{2j}^{**} + E_{22}[I - (I + H_{2j}^{**})^{-1}]Q_{2j}^{**} E_{22} \;, \qquad (3.10)$$

$$Q_{2j+2}^{**} = (I + H_{2j+1}^{**})^{-1} Q_{2j+1}^{**} + E_{11}[I - (I + H_{2j+1}^{**})^{-1}]\, Q_{2j+1}^{**} E_{11}; \qquad (3.11)$$

these are accompanied by the initial conditions $Q_o^{**} = Q_o$ and $H_o^{**} = H_o$.

In view of the way in which the analysis "alternates" between the two units, it would be unreasonable to expect the type of convergence given by the existence of a single limiting matrix

$$Q_\infty^{**} = \lim_{j \to \infty} Q_j^{**} \quad .$$

Rather, one might hope for "convergence with period 2" , i.e. for the existence of a pair of matrices

$$Q_\infty^{o} = \lim_{j \to \infty} Q_{2j+1}^{**} \;, \quad Q_\infty^{e} = \lim_{j \to \infty} Q_{2j}^{*} \;, \qquad (3.12)$$

where the superscripts "o" and "e" stand for "odd" and "even" respectively. Furthermore, in view of the results obtained in Section 2, one would not be surprised if these two limiting matrices turned out to be independent of σ .

At present we have <u>not</u> succeeded in analyzing this situation with anything like the completeness achieved in Section 2 for the "location with revision" case. However, we have succeeded in proving that in the present

case, like that of Section 2, the precision of position-location stabilizes over time. To make this precise, let

$$Q_j^{**} = \begin{bmatrix} q_1^{(j)} & q^{(j)} \\ q^{(j)} & q_2^{(j)} \end{bmatrix} \quad ;$$

then $q^{(o)} = q$ and $q_i^{(o)} = q_i$ $(i = 1,2)$. The precision with which the j-th location-operation estimates the position of the locatee is given by

$$q_2^{(j)} = \mathrm{var}(U_{2j}^*) \quad \text{if } j \text{ is even} ,$$

$$q_1^{(j)} = \mathrm{var}(U_{1j}^*) \quad \text{if } j \text{ is odd} .$$

It will be shown below that

$$\text{sequences } \{q_2^{(2j)}\} \text{ and } \{q_1^{(2j+1)}\} \text{ are convergent,} \tag{3.13}$$

thus proving the "stabilization" mentioned above.

The logic of the proof will run as follows. It will be shown that

$$\text{sequences } \{q_1^{(2j)}\} \text{ and } \{q_2^{(2j)}\} \text{ are convergent .} \tag{3.14}$$

This is a property of the sequence $\{Q_{2j}^*\}$ of even-index matrices. Since the sequence $\{Q_{2j+1}^*\}$ of odd-index matrices obeys the same type of recursion, and since (3.14) is symmetric as between the two units, it follows by analogy

that

$$\text{sequences } \{q_1^{(2j+1)}\} \text{ and } \{q_2^{(2j+1)}\} \text{ are convergent.} \tag{3.15}$$

The desired result (3.13) then follows from (3.14) and (3.15).

Thus it suffices to prove (3.14). Since the variance terms $q_1^{(2j)}$ and $q_2^{(2j)}$ are nonnegative, i.e. bounded below by 0, it suffices to prove that the sequences in (3.14) are non-increasing. These statements,

$$q_1^{(2j+2)} \leq q_1^{(2j)} \quad , \quad q_2^{(2j+2)} \leq q_2^{(2j)} \; ,$$

are to be proved by induction on j, but it suffices to establish the prototype induction step, namely

$$q_1^{(2)} \leq q_1 \quad , \quad q_2^{(2)} \leq q_2 \quad . \tag{3.16}$$

The proof of the desired result (3.13) has now been reduced to the demonstration of the two statements (3.16). For proving them, it is convenient to introduce the auxiliary quantities

$$S = (q_1-q) + (q_2-q) \quad , \quad \Delta = q_1q_2-q^2 \quad . \tag{3.17}$$

Since Q_o is a variance-covariance matrix, its determinant $\Delta \geq 0$, and it follows by (2.23) that $S \geq 0$. The following calculations will make use of the easily-checked identities

$$\Delta - qS = (q_1-q)(q_2-q) \quad , \tag{3.18}$$

$$\Delta - q_iS = -(q_i-q)^2 \quad . \tag{3.19}$$

Direct calculation from (3.6) yields

$$Q_1^{**} = (\sigma^2 + S)^{-1} \begin{bmatrix} \sigma^2 q_1 + \Delta & \sigma^2 q + \Delta \\ \sigma^2 q + \Delta & (\sigma^2 + S) q_2 \end{bmatrix} . \tag{3.20}$$

From (3.7), we obtain

$$(I + H_1^{**})^{-1} = [\sigma^4 + 2S\sigma^2 + (q_2 - q)^2]^{-1} \begin{bmatrix} \sigma^4 + (S + q_2 - q)\sigma^2 + (q_2-q)^2 & \sigma^2(q_1 - q) \\ \sigma^2(q_2 - q) + (q_2 - q)^2 & \sigma^4 + (S + q_1 - q)\sigma^2 \end{bmatrix} .$$

Application of (3.8) and (3.20) yields

$$q_1^{(2)} = q_1^{(1)} = (\sigma^2 + S)^{-1}(\sigma^2 q_1 + \Delta) ,$$

so that the first of the desired statements (3.16) is equivalent to

$$\sigma^2 q_1 + \Delta \leq (\sigma^2 + S) q_1$$

and hence to $\Delta - q_1 S \leq 0$, which is true by (3.19). Next, substitution into (3.8) yields

$$q_2^{(2)} = (\sigma^2 + S)^{-1} [\sigma^4 + 2S\sigma^2 + (q_2-q)^2]^{-1} \{q_2\sigma^6 + [\Delta + 2q_2 S]\sigma^4 + [2q_2 S^2 - (q_2-q)^2(q_2-2q)]\sigma^2 + \Delta(q_2-q)^2\} ,$$

so that the second of the desired statements (3.16) is equivalent to

$$q_2\sigma^6 + [\Delta + 2q_2S]\sigma^4 + [2q_2S^2 - (q_2-q)^2(q_2-2q)]\sigma^2 + \Delta(q_2-q)^2$$

$$\leq q_2(\sigma^2 + S)[\sigma^4 + 2S\sigma^2 + (q_2-q)^2] ,$$

which is true since it is in turn equivalent to

$$(q_2-q)^2 [\sigma^2 + (q_2-q)]^2 \geq 0 .$$

This completes the proof of (3.13), so that the existence of the "variance terms" in (3.12), i.e. entries q_1^o and q_2^o in Q_∞^o and entries q_1^e and q_2^e in Q_∞^e , is established. We have not as yet succeeded in proving the existence of the covariance limits

$$q^o = \lim_{j\to\infty} q^{(2j+1)} , \quad q^e = \lim_{j\to\infty} q^{(2j)} , \tag{3.21}$$

i.e. the off-diagonal terms in (3.12), except under the additional hypotheses

$$q_1 \geq q , \quad q_2 \geq q . \tag{3.22}$$

This assumption is probably not very restrictive for the situations of interest here; it asserts that the errors in the initial estimates (of the two units' positions), if positively correlated, have covariance less than either of their individual variances. Because $\Delta \geq 0$, (3.22) will automatically be satisfied if $q_1 = q_2$.

To prove that (3.22) implies (3.21), we first prove that it implies the generalization

$$q_1^{(j)} \geq q^{(j)} , \quad q_2^{(j)} \geq q^{(j)} . \tag{3.23}$$

of itself. The proof is by induction on j , and since (3.23) is symmetric as between the two units, it suffices to treat a prototype induction step, i.e. to show that

$$q_1^{(1)} \geq q^{(1)} \quad , \quad q_2^{(1)} \geq q^{(1)} \tag{3.24}$$

follows from (3.22). From (3.20), we see that the first part of (3.24) is an immediate consequence of the first part of (3.22). Also from (3.20), we see that the second part of (3.24) is equivalent to

$$(\sigma^2 + S)q_2 \geq \sigma^2 q + \Delta \quad ,$$

hence to

$$\sigma^2(q_2 - q) \geq \Delta - q_2 S - (q_2 - q)^2 \ ,$$

which follows from the second part of (3.22) .

We will show below that (3.22) implies

$$\text{sequence } \{q^{(2j)}\} \text{ is non-decreasing.} \tag{3.25}$$

Because the variance-covariance matrix Q_{2j}^{**} must have a non-negative determinant,

$$[q^{(2j)}]^2 \leq q_1^{(2j)} q_2^{(2j)} \quad ,$$

it follows that the sequence in (3.25) is ultimately bounded above by a quantity $[q_1^e q_2^e]^{\frac{1}{2}} + \epsilon$ where $\epsilon > 0$, and so this sequence must have a limit q^e . This pertains to the sequence $\{Q_{2j}^{**}\}$; since the sequence $\{Q_{2j+1}^{**}\}$ is generated by the same type of recursion, and since by (3.24) its initial

matrix Q_1 satisfies the analog of (3.22), the same logic proves the existence of the limiting covariance q^o .

It only remains to prove that (3.22) implies (3.25). Since (3.22) implies (3.23), it suffices to establish the prototype induction step, namely that (3.22) implies

$$q^{(2)} \geq q \ . \tag{3.26}$$

Application of (3.8) yields

$$\begin{aligned} q^{(2)} &= (\sigma^2 + S)^{-1}[\sigma^4 + 2S\sigma^2 + (q_2 - q)^2]^{-1}\{q\sigma^6 \\ &+ [(q_2-q)q_1 + \Delta + q(S + q_1 - q)]\sigma^4 + [(q_2-q)\Delta + q_1(q_2-q)^2 + (S + q_1 - q)\Delta]\sigma^2 \\ &+ (q_2-q)^2 \Delta\} \\ &= (\sigma^2 + S)^{-1}[\sigma^4 + 2S\sigma^2 + (q_2-q)^2]^{-1}\{q\sigma^6 \\ &+ [2\Delta + qS]\sigma^4 + [2S\Delta + q_1(q_2-q)^2]\sigma^2 + (q_2-q)^2\Delta\} . \end{aligned}$$

Thus (3.26) is equivalent to

$$\begin{aligned} &q\sigma^6 + [2\Delta + qS]\sigma^4 + [2S\Delta + q_1(q_2-q)^2]\sigma^2 + (q_2-q)^2\Delta \\ &\quad \geq q(\sigma^2 + S)[\sigma^4 + 2S\sigma^2 + (q_2-q)^2] , \end{aligned}$$

or in turn to

$$(\Delta - qS)\{2\sigma^4 + [2S + (q_2-q)]\sigma^2 + (q_2-q)^2\} \geq 0 \ ,$$

which is true by virtue of (3.18) and (3.22). This completes the proof that the limiting covariances (3.21) exist.

matrix Q_1 satisfies the analog of (3.22), the same logic proves the existence of the limiting covariance q^o.

It only remains to prove that (3.22) implies (3.25). Since (3.22) implies (3.23), it suffices to establish the prototype induction step, namely that (3.22) implies

$$q^{(2)} \geq q . \tag{3.26}$$

Application of (3.8) yields

$$\begin{aligned} q^{(2)} &= (\sigma^2 + S)^{-1}[\sigma^4 + 2S\sigma^2 + (q_2-q)^2]^{-1} \{q\sigma^6 \\ &+ [(q_2-q)q_1 + \Delta + q(S + q_1-q)]\sigma^4 + [(q_2-q)\Delta + q_1(q_2-q)^2 + (S + q_1-q)\Delta]\sigma^2 \\ &+ (q_2-q)^2 \Delta\} \\ &= (\sigma^2 + S)^{-1} [\sigma^4 + 2S\sigma^2 + (q_2-q)^2]^{-1} \{q\sigma^6 \\ &+ [2\Delta + qS]\sigma^4 + [2S\Delta + q_1(q_2-q)^2]\sigma^2 + (q_2-q)^2\Delta\} . \end{aligned}$$

Thus (3.26) is equivalent to

$$\begin{aligned} &q\sigma^6 + [2\Delta + qS]\sigma^4 + [2S\Delta + q_1(q_2-q)^2]\sigma^2 + (q_2-q)^2\Delta \\ &\qquad \geq q(\sigma^2 + S)[\sigma^4 + 2S\sigma^2 + (q_2-q)^2] , \end{aligned}$$

or in turn to

$$(\Delta - qS)\{2\sigma^4 + [2S + (q_2-q)]\sigma^2 + (q_2-q)^2\} \geq 0 ,$$

which is true by virtue of (3.18) and (3.22). This completes the proof that the limiting covariances (3.21) exist.

Although the main objectives in this Section have been achieved, we should note several items of "unfinished business" which need to be settled to obtain a thorough analysis:

(a) Determine whether the limiting covariances q^o and q^e (and hence, the limiting matrices Q_∞^o and Q_∞^e) exist in general, and not only under the extra hypotheses (3.22).

(b) Evaluate the various limits in closed form.

(c) If (b) is too ambitious, at least determine whether the limits are independent of σ .

PROJECT "WHERE" : WORKING PAPER NO. 7

Algorithms

D. J. Sookne

1. Description of the Algorithms used by WHERSM

The algorithm used to locate each unit is the least squares squared (LSS) algorithm. Both two- and three-dimensional subroutines are included in the WHERSM system. The three-dimensional program is described in detail below.

For each locator i, $i = 1,2,\ldots,n$, let (x_i, y_i, z_i) be its estimated position, r_i its reported distance to the locatee, and w_i the reciprocal of its variance, the calculation of which is described below. The w_i's act as weights; the smaller the variance of a locator, the more accurately we know its position, and the higher its weight will be. Then LSS minimizes

$$E = \sum_{i=1}^{n} w_i E_i \qquad \text{where} \tag{1}$$

$$E_i = \frac{(d_i^2 - r_i^2)^2}{4r_i^2} \quad ; \tag{2}$$

here d_i is the calculated distance between (x_i, y_i, z_i) and (x, y, z), the estimated position of the locatee, so

$$d_i^2 = (x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 \ . \tag{3}$$

Note that

$$E_i = \frac{(d_i^2 - r_i^2)^2}{4r_i^2} = \frac{(d_i + r_i)^2}{4r_i^2} \cdot (d_i - r_i)^2 \ .$$

When the estimated positions of locators and locatee are near their true positions, r_i is very near d_i, so the factor $(d_i + r_i)^2/4r_i^2$ is very near 1, and E_i behaves very much like $(d_i - r_i)^2$. Thus we expect this algorithm to produce answers nearly identical to those produced by the least squares (LS) algorithm, for which $E_i = (d_i - r_i)^2$. Indeed, computer tests indicate that there is no significant difference between the results of these two algorithms. Thus LSS is used because it is more economical, since LS requires the calculation of d_i, which in turn requires a square root to be taken (see (3)). LSS avoids the square root, since only d_i^2 is needed.

Where E is a minimum, its partial derivatives with respect to x, y, and z will be zero. Thus the program calculates

$$
\begin{aligned}
F &= \frac{\partial E}{\partial x} = \sum_{i=1}^{n} \frac{w_i(d_i^2 - r_i^2)}{r_i^2}(x - x_i) \\
G &= \frac{\partial E}{\partial y} = \sum_{i=1}^{n} \frac{w_i(d_i^2 - r_i^2)}{r_i^2}(y - y_i) \qquad (4) \\
H &= \frac{\partial E}{\partial z} = \sum_{i=1}^{n} \frac{w_i(d_i^2 - r_i^2)}{r_i^2}(z - z_i) .
\end{aligned}
$$

Since d_i depends on x, y, and z (see (3)), F, G, and H depend non-linearly on these variables. Then the system (4) cannot be solved directly for x, y, and z. Instead, LSS does successive Newton-Raphson iterations ([1], pp. 201-223) , beginning with a starting point $p = (x,y,z)$. The first iteration yields a new point $p' = (x + \Delta x, y + \Delta y, z + \Delta z)$; this point is used as the starting point for the next iteration, and so on. The process begins with the first order approximation

$$F_{p'} = F_p + \frac{\partial F}{\partial x}\Delta x + \frac{\partial F}{\partial y}\Delta y + \frac{\partial F}{\partial z}\Delta z$$

$$G_{p'} = G_p + \frac{\partial G}{\partial x}\Delta x + \frac{\partial G}{\partial y}\Delta y + \frac{\partial G}{\partial z}\Delta z$$

$$H_{p'} = H_p + \frac{\partial H}{\partial x}\Delta x + \frac{\partial H}{\partial y}\Delta y + \frac{\partial H}{\partial z}\Delta z \quad .$$

Here F_p denotes the value of F at point p, and all partial derivatives are evaluated at p. Thus this system represents the expansions of the functions F, G, and H about p. Now we want $F_{p'}$, $G_{p'}$ and $H_{p'}$ all to be zero, so the problem reduces to that of finding $\Delta x, \Delta y, \Delta z$ which satisfy the three simultaneous equations

$$\begin{aligned} \frac{\partial F}{\partial x}\Delta x + \frac{\partial F}{\partial y}\Delta y + \frac{\partial F}{\partial z}\Delta z + F_p &= 0 \\ \frac{\partial G}{\partial x}\Delta x + \frac{\partial G}{\partial y}\Delta y + \frac{\partial G}{\partial z}\Delta z + G_p &= 0 \\ \frac{\partial H}{\partial x}\Delta x + \frac{\partial H}{\partial y}\Delta y + \frac{\partial H}{\partial z}\Delta z + H_p &= 0 \quad . \end{aligned} \tag{5}$$

The program calculates F, G, and H (see (4)), and

$$\left.\begin{aligned} \frac{\partial F}{\partial x} &= \sum_{i=1}^{n} \frac{w_i(d_i^2 - r_i^2 + 2(x - x_i)^2)}{r_i^2} \\ \frac{\partial G}{\partial y} &= \sum_{i=1}^{n} \frac{w_i(d_i^2 - r_i^2 + 2(y - y_i)^2)}{r_i^2} \\ \frac{\partial H}{\partial z} &= \sum_{i=1}^{n} \frac{w_i(d_i^2 - r_i^2 + 2(z - z_i)^2)}{r_i^2} \end{aligned}\right\} \tag{6}$$

$$\left.\begin{aligned}
\frac{\partial G}{\partial x} &= \frac{\partial}{\partial x}\frac{\partial E}{\partial y} = \frac{\partial}{\partial y}\frac{\partial E}{\partial x} = \frac{\partial F}{\partial y} = \sum_{i=1}^{n} \frac{2w_i(x - x_i)(y - y_i)}{r_i^2} \\
\frac{\partial H}{\partial x} &= \frac{\partial}{\partial x}\frac{\partial E}{\partial z} = \frac{\partial}{\partial z}\frac{\partial E}{\partial x} = \frac{\partial F}{\partial z} = \sum_{i=1}^{n} \frac{2w_i(x - x_i)(z - z_i)}{r_i^2} \\
\frac{\partial H}{\partial y} &= \frac{\partial}{\partial y}\frac{\partial E}{\partial z} = \frac{\partial}{\partial z}\frac{\partial E}{\partial y} = \frac{\partial G}{\partial z} = \sum_{i=1}^{n} \frac{2w_i(y - y_i)(z - z_i)}{r_i^2} ,
\end{aligned}\right\} \tag{6}$$

then solves (5) for $\Delta x, \Delta y, \Delta z$. This determines $p' = (x + \Delta x, y + \Delta y, z + \Delta z)$, which is used as the starting point for the next iteration. For a given set of locators, the program will do up to 4 iterations, stopping if an iteration converges, namely

$$\max(|F|, |G|, |H|) \leq \sum_{i=1}^{n} w_i ,$$

or if an iteration diverges (which see below). I.e. convergence occurs when F, G, and H are all small.

Computer tests indicated that for both the two- and three-dimensional versions of LSS, 2 to 4 Newton-Raphson iterations are generally sufficient for convergence, even when the starting point is 100 meters from the true position.

The LSS algorithm allows for easy elimination of outliers. These are locators whose reported distance (r_i) to the locatee contains a greater error than do the reported distances of the other locators. Let

$$\overline{E} = \left(\sum_{i=1}^{n} w_i E_i\right) / \sum_{i=1}^{n} w_i . \tag{7}$$

Now any locator i is eliminated for which

$$E_i > 4\overline{E} \quad . \tag{8}$$

Several other constants greater than 1 were tried in place of 4, but computer tests showed that 4 was best.

The algorithm iterates on all locators until convergence occurs, up to 4 iterations (for divergence, see below). Then $\overline{E}$ is calculated and some locators may be eliminated via (8) . (If no locators are eliminated, we're through). Then more iterations are performed on the reduced set of locators. After convergence, (7) and (8) are again applied to eliminate outliers . This iteration-elimination process continues until either (1) no locators are eliminated in a step, (2) 1/4 of the locators have been eliminated, or (3) the algorithm diverges (see below) . In cases (1) and (2) the algorithm terminates.

There are two types of divergence; an increase in the objective function, E , from one iteration to the next and a badly scaled determinant less than 10^{-8} in absolute value which makes equations (5) impossible to solve accurately. In each case, we halve the step size of the previous iteration, and try again. This can be done unless the iteration at which divergence occurs is the first iteration for this locatee. Then we must quit without finding a solution.

For any locatee and set of locators, the size of successive iteration steps should decrease rapidly. If not, then the problem is ill-behaved, and perhaps dozens of iterations will be necessary to achieve convergence. To avoid this, the algorithm attempts to " home in" on the solution faster. If the step size in the x-direction (or y or z) is Δx_1 for one iteration and Δx_2 for the next, and if $|\Delta x_2| > \frac{1}{2} |\Delta x_1|$, then Δx_2 is replaced by

$$2\min(|\Delta x_1|, |\Delta x_2|) \cdot \mathrm{sgn}\,\Delta x_2$$

if Δx_1 and Δx_2 have the same sign, and if they have opposite signs, Δx_2 is replaced by

$$\frac{\Delta x_2 \cdot \Delta x_1}{\Delta x_1 - \Delta x_2} .$$

A further refinement of the algorithm was tried; this involved relocating the locators. More concerning this may be found in sec. 3 and Working Paper No. 8. Once the Newton iterations have converged, there is an error between the reported distance r_i and the calculated distance d_i, for each locator i in the set. By changing the estimated position of each locator, i.e. by moving it along the straight line connecting it with the locatee, this error $d_i - r_i$ may be reduced, even eliminated. After relocating, more Newton-Raphson iterations may be done. Several runs were made, using different values for the fraction, f, of the error eliminated by relocation, with $0 < f < 1$. The best such f was the smallest, and that was not quite as good as no relocation $(f = 0)$. Relocation remains an option in the LSS routines, but one which is not now used.

The 2-dimensional LSS program is analogous to the 3-dimensional one. It solves the equations

$$\frac{\partial F}{\partial x} \Delta x + \frac{\partial F}{\partial y} \Delta y + F = 0$$

$$\frac{\partial G}{\partial x} \Delta x + \frac{\partial G}{\partial y} \Delta y + G = 0 .$$

Here, F and G and their partial derivatives are as described above, except that all z-components are zero. The 2-dimensional program is less likely to result in divergence of an iteration, so the special steps to handle divergence (see previous page) are omitted.

The weights w_i (see (1)) are again taken as the reciprocals of the variances σ_i^2 . When a unit has been located, the variance of its location is calculated as

$$\frac{\left(\sum_{i=1}^{n} w_i E_i\right)}{(n-m) \sum_{i=1}^{n} w_i)} \tag{9}$$

when n is the number of locators, m the dimensionality (2 or 3) of the algorithm.

When the locatee is lost, so that no starting point exists for the LSS routine, the least squares linear (LSL) routine is called to provide a starting point. First, a slant-range reduction is done to convert 3-D reported distances to 2-D . Then the 2-D LSL algorithm is called. This is the same basic algorithm which is described in sec. 2 of [2] . The modifications are these: all pairs of circles are used, and outliers are eliminated. Consider the M radical axis lines each formed by a pair of circles; the equation for such a line is found by subtracting the equation of one circle from the equation of the other as in [2] , p. 10. For each such line k, $k = 1,2,\ldots M$, let d_k be the calculated distance from the locatee to the line. Then LSL minimizes

$$D = \sum_{k=1}^{M} d_k^2 .$$

After a locatee has been found using all M lines, the routine calculates

$$\overline{d^2} = D/M ,$$

the average squared distance to the lines. Then any line k is eliminated for which

$$d_k^2 > 8 \cdot \overline{d^2} .$$

The reduced set of locators is used to calculate a position, $\overline{d^2}$ is recalculated, and a line k is eliminated for which

$$d_k^2 > 4 \cdot \overline{d^2} \quad .$$

The process is repeated twice more, once with the elimination condition

$$d_k^2 > 2 \cdot \overline{d^2} \quad , \qquad \text{and finally with}$$

$$d_k^2 > \overline{d^2} \quad .$$

The LSL algorithm will fail only when the equations at the bottom of p. 13 of [2] are ill-conditioned. This will happen when the M lines are nearly parallel, i.e. the locators all lie near one straight line. In this case, no starting point can be found for LSS .

The routine FINDIT calls LSS (2-D or 3-D) and LSL when necessary. If the locatee is not lost, two previous locations are used to extrapolate in the x- and y- directions to get a starting point for LSS. The last z-value is used, with no extrapolation. Then LSS is called; the 2-dimensional routine for ground units, 3-dimensional for airplanes.

If the unit is lost, LSL is called after doing a slant-range reduction (using the last known z-value) to convert 3-dimensional distances to planar distances. Then using the planar distances, the 2-dimensional LSS is called. If the unit is an airplane, the x- and y-coordinates returned by LSS(2-D) plus the last known z-value are used as a starting point for LSS(3-D).

Control is returned to FINDIT with the variance defined by (9) . This represents the variance of the position estimate, based on the reported

distances. The total variance is computed by adding RVAR, the variance of the distances. If the locatee is known to be moving, its weight is calculated by inverting the total variance.

If the unit is fixed, the current position estimate is taken to be a weighted average of the old estimate and the just-calculated estimate. Then the weight of the locatee is computed as a weighted average of the old weight and the just-calculated weight. A further description of these calculations may be found in Section three of this report.

2. Description of Algorithms Programmed for WHERSM But Not Now Used

The linear method (LM) and smallest tangent circle (STC) method are described in [2]. They were both found to be less accurate than the LSS and LSL algorithms.

The least squares method mentioned in §1 is the same as LSS, except equations (2), (4), and (6) are replaced by

$$E_i = (d_i - r_i)^2 \tag{2'}$$

$$\left.\begin{aligned} F &= \frac{1}{2}\frac{\partial E}{\partial x} = \sum_{i=1}^{n} \frac{(d_i-r_i)(x-x_i)}{d_i} \\ G &= \frac{1}{2}\frac{\partial E}{\partial y} = \sum_{i=1}^{n} \frac{(d_i-r_i)(y-y_i)}{d_i} \\ H &= \frac{1}{2}\frac{\partial E}{\partial z} = \sum_{i=1}^{n} \frac{(d_i-r_i)(z-z_i)}{d_i}\quad, \end{aligned}\right\} \tag{4'}$$

and

$$\frac{\partial F}{\partial x} = \sum_{i=1}^{n} \left(\frac{d_i - r_i}{d_i} + \frac{r_i (x-x_i)^2}{d_i^3} \right)$$

$$\frac{\partial G}{\partial y} = \sum_{i=1}^{n} \left(\frac{d_i - r_i}{d_i} + \frac{r_i (y-y_i)^2}{d_i^3} \right)$$

$$\frac{\partial H}{\partial z} = \sum_{i=1}^{n} \left(\frac{d_i - r_i}{d_i} + \frac{r_i (z-z_i)^2}{d_i^3} \right)$$

$$\frac{\partial G}{\partial x} = \frac{\partial F}{\partial y} = \sum_{i=1}^{n} \frac{r_i (x-x_i)(y-y_i)}{d_i^3} \tag{6'}$$

$$\frac{\partial H}{\partial x} = \frac{\partial F}{\partial z} = \sum_{i=1}^{n} \frac{r_i (x-x_i)(z-z_i)}{d_i^3}$$

$$\frac{\partial H}{\partial y} = \frac{\partial G}{\partial z} = \sum_{i=1}^{n} \frac{r_i (y-y_i)(z-z_i)}{d_i^3} .$$

The other algorithm programmed was MINMAX which minimizes the maximum of the errors $d_i - r_i$. This algorithm proved to be less accurate than STC and much less accurate than LSS. Further, it uses 10 to 100 times the computer time the other algorithms use, since (in the two-dimensional case) it considers all pairs and triples of locator circles.

References

1. Scarborough, James B., Numerical Mathematical Analysis, John Hopkins Press, Baltimore, 1930.
2. First Interim Report on NBS Project WHERE.

LISTING OF POSITION LOCATION PROGRAMS

@A RUN SOOKNE,01563,1,30 189 09

```
@IT FOR FINDIT,FINDIT                                                              05 MAY 72      11:41:42.871
UNIVAC 1108 FORTRAN V LEVEL    2206 0018 F5018P
THIS COMPILATION WAS DONE ON 05 MAY 72 AT 11:41:42

   SUBROUTINE FINDIT     ENTRY POINT 001120

   STORAGE USED (BLOCK, NAME, LENGTH)

         0001    *CODE    001135
         0000    *DATA    001121
         0002    *BLANK   000000
         0003    CB1      013000
         0004    MXYZ     000673

   EXTERNAL REFERENCES (BLOCK, NAME)

         0005    LOCLSL
         0006    LOCLSS
         0007    HEIGHT
         0010    LSS3D
         0011    SQRT
         0012    NERR3$

   STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

    0001     000150 1L        0001     000376 12L       0001     000027 127G      0001     000374 13L       0001     000601 14L
    0001     000604 15L       0001     000627 16L       0001     000367 17L       0001     000660 19L       0001     000152 2L
    0001     000715 20L       0001     000257 204G      0001     000212 22L       0001     000337 222G      0001     000403 241G
    0001     000412 245G      0001     001074 25L       0001     000473 273G      0001     000253 3L        0001     000554 322G
    0001     000524 34L       0001     000465 35L       0001     001015 36L       0001     000313 4L        0001     000751 401G
    0001     001056 435G      0001     000442 5L        0001     000332 6L        0000     001056 900F      0003 R 011531 CLK
    0000 R 001053 D           0003 R 000000 EP          0000 R 001046 EPX         0000 R 001047 EPY         0000 R 001044 F
    0000 R 001036 FVAR        0003 L 011535 FXD         0000 R 001034 HMAX        0000 I 001042 I           0000 I 001032 IS
    0000 I 001055 J           0003 I 011530 JFIND       0000 I 001043 K           0004 I 000001 L           0003 I 006116 LAM
    0000 I 000341 LAMM        0000 I 001037 LFXD        0004 I 000000 M           0000 I 001054 MFXD        0000 I 001045 MM
    0000 I 001050 N           0000 I 000471 NFXD        0000 I 001041 NPLANE      0003 I 012437 NTYPE       0000 I 001052 NWORST
    0004 R 000672 P           0004 R 000542 R           0000 R 001033 RMAX        0003 L 012076 RPT         0003 R 006460 RR
    0000 R 001035 RVAR        0003 L 007022 S           0000 L 000000 S1          0003 R 007363 TLOC        0003 R 011167 W
    0000 R 001040 WOLD        0000 R 001051 WORST       0004 R 000412 WW          0003 R 011532 XNEW        0004 R 000002 XX
    0003 R 011533 YNEW        0004 R 000132 YY          0003 R 011534 ZNEW        0004 R 000262 ZZ

 00100      1*    C       THIS ROUTINE ATTEMPTS TO LOCATE THE LOCATEE BY CALLING LSL OR THE
 00100      2*    C       TWO- OR 3-DIMENSIONAL VERSIONS OR LSS.
 00101      3*            SUBROUTINE FINDIT
 00103      4*            PARAMETER L1=225,L4=L1+1
 00104      5*            LOGICAL S,S1(L1),RPT,FXD
 00105      6*            COMMON/CB1/EP(L1,14),LAM(L4),RR(L4),S(L1),TLOC(L1,4),W,L1),
 00105      7*           1 JFIND,CLK,XNEW,YNEW,ZNEW,FXD(L1),RPT(L1),NTYPE(L1)
 00106      8*            COMMON/MXYZ/M,L,XX(88),YY(88),ZZ(88),WW(88),R(88),P
 00107      9*            DIMENSION LAMM(88),NFXD(L1)
```

```
      DATA S1,RMAX,HMAX,RVAR,FVAR,LFXD,NFXD/
     1 L1*.TRUE.,5.E6,200.,36.,40.,L4*10/
      WOLD=1.E9
      IF(FXD(JFIND).AND.W(JFIND).GT.0) WOLD=1./W(JFIND)-RVAR
      S1(JFIND)=S(JFIND)
      M=0
      NPLANE=0
C     COUNT THE LOCATORS, KEEPING TRACK OF THEIR POSITIONS, WEIGHTS, AND
C     REPORTED DISTANCES TO THE LOCATEE.  EXTRAPOLATE ON TWO PREVIOUS
C     POSITIONS TO FIND PRESENT POSITION OF EACH LOCATOR.
      DO 1 I=1,L1
      K=LAM(I)
      LAMM(I)=K
      IF(RR(I).LT.0) GO TO 2
      IF(RR(I).GT.RMAX.OR..NOT.(S(K).AND.S1(K))) GO TO 1
      M=M+1
      LAM(M)=K
      RR(M)=RR(I)
      R(M)=RR(I)
      WA(M)=W(K)
      F=0.0
      IF(TLOC(K,3).GT.TLOC(K,1).AND..NOT.FXD(K)) F=(CLK-TLOC(K,3))/
     1 (TLOC(K,3)-TLOC(K,1))
      XX(M)=EP(K,7)+F*(EP(K,7)-EP(K,1))
      YY(M)=EP(K,8)+F*(EP(K,8)-EP(K,2))
      ZZ(M)=EP(K,9)
      IF(NTYPE(K).LT.3) GO TO 1
      ZZ(M)=EP(K,9)+F*(EP(K,9)-EP(K,3))
      NPLANE=NPLANE+1
    1 CONTINUE
    2 F=0.
      ZNEW=EP(JFIND,9)
      IF(RPT(JFIND).AND.FXD(JFIND)) GO TO 22
      IF(M.LT.2) GO TO 5
      MM=M-1
      IF(.NOT.S(JFIND)) GO TO 3
C     LAST ATTEMPT TO FIND THE LOCATEE WAS SUCCESSFUL, SO AN ESTIMATE
C     OF THE PRESENT POSITION MAY BE FOUND BY EXTRAPOLATION.
      IF(TLOC(JFIND,3).GT.TLOC(JFIND,1)) F=(CLK-TLOC(JFIND,3))/
     1 (TLOC(JFIND,3)-TLOC(JFIND,1))
   22 XNEW=EP(JFIND,7)+F*(EP(JFIND,7)-EP(JFIND,1))
      YNEW=EP(JFIND,8)+F*(EP(JFIND,8)-EP(JFIND,2))
      W(JFIND)=FVAR
      IF(RPT(JFIND).AND.FXD(JFIND)) GO TO 19
      IF(NPLANE.GT.0.OR.NTYPE(JFIND).GT.2) GO TO 12
C     THE LOCATEE AND ALL LOCATORS ARE GROUND UNITS, SO LSS-2D IS USED.
      GO TO 6
C     THE LAST ATTEMPT TO LOCATE THE LOCATEE WAS NOT SUCCESSFUL, SO AN
C     ESTIMATE OF THE PRESENT POSITION MUST BE FOUND BY CALLING LSL.
C     FIRST, A SLANT-HEIGHT REDUCTION MUST BE DONE (3-D DISTANCES ARE
C     CONVERTED TO PLANAR DISTANCES).
    3 DO 4 I=1,M
      RR(I)= R(I)-.5*(ZNEW-ZZ(I))**2/ R(I)
      IF(3.*(ZNEW-ZZ(I)).LT.R(I)) GO TO 4
      RR(I)=SQRT(ABS(R(I)**2-(ZNEW-ZZ(I))**2))
    4 CONTINUE
      CALL LOCLSL
      IF(.NOT.S(JFIND)) GO TO 5
```

```
00217              IF(NTYPE(JFIND).GT.2) GO TO 13
00221            6 DO 7 I=1,M
00224            7 RR(I)=R(I)-.5*(ZNEW-ZZ(I))**2/R(I)
00226              S(JFIND)=.FALSE.
00226       C      LSS IS CALLED TO HOME IN ON THE POSITION OF A GROUND UNIT.
00227              CALL LOCLSS
00227       C      WRITE(6,900) JFIND,S(JFIND),XNEW,YNEW,ZNEW,M
00230          900 FORMAT(I5,L3,3E14.8,I5)
00231              IF(.NOT.S(JFIND)) GO TO 5
00231       C      THE ALTITUDE OF THE GROUND UNIT IS FOUND.
00233              CALL HEIGHT(EPX,EPY,XNEW,YNEW,ZNEW,N)
00234           17 IF (N.EQ.-1) GO TO 17
00236              GO TO 15
00236       C      THE LOCATEE OR A LOCATOR IS AN AIRPLANE, AND LSS2D IS USED TO FIND
00236       C      THE X- AND Y-COORDINATES MORE EXACTLY BEFORE CALLING LSS3D.
00237           13 CALL LOCLSS
00237       C      THE PROGRAM SEARCHES FARTHER AND FARTHER FROM THE LOCATEE UNTIL
00237       C      AT LEAST 5 LOCATORS ARE FOUND.
00240           12 DO 33 K=4,10,2
00243              L=0
00244              DO 32 I=1,M
00247              IF(K*(ZNEW-ZZ(I)).GT.R(I)) L=L+1
00251           32 CONTINUE
00253              IF(L.GE.5) GO TO 35
00255           33 CONTINUE
00257              IF(L.GE.4) GO TO 35
00257       C      THE LOCATEE CANNOT BE FOUND.
00261            5 S(JFIND)=.FALSE.
00262              IF(.NOT.FXD(JFIND)) W(JFIND)=0.
00264              IF (.NOT.RPT(JFIND)) RETURN
00266              WORST=0.
00267              NWORST=JFIND
00270              RETURN
00270       C      FIVE LOCATORS HAVE BEEN FOUND, SO LSS3D IS CALLED.
00271           35 L=0
00272              DO 34 I=1,M
00275              IF(K*(ZNEW-ZZ(I)).LT.R(I))  GO TO 34
00277              L=L+1
00300              RR(L)=R(I)
00301              WW(L)=WW(I)
00302              XX(L)=XX(I)
00303              YY(L)=YY(I)
00304              ZZ(L)=ZZ(I)
00305              LAM(L)=LAM(I)
00306           34 CONTINUE
00310              M=L
00311              S(JFIND)=.FALSE.
00312              CALL LSS3D
00313              IF(S(JFIND)) GO TO 15
00315              IF(NTYPE(JFIND).GT.2) GO TO 5
00315       C      THE LOCATEE IS A GROUND UNIT, AND SOME OF THE LOCATORS (AT LEAST
00315       C      ONE) ARE AIRPLANES. LSS3D COULD NOT FIND THE LOCATEE, SO SOME OF
00315       C      THE LOCATORS ARE ELIMINATED, AND LSS2D IS USED.
00317              L=M
00320              M=0
00321              DO 14 I=1,L
00324              IF(4.*(ZNEW-ZZ(I)).GT.R(I)) GO TO 14
00326              M=M+1
```

```
00327            XX(M)=XX(I)
00330            YY(M)=YY(I)
00331            ZZ(M)=ZZ(I)
00332            WW(M)=WW(I)
00333            R(M)=RR(I)
00334         14 CONTINUE
00336            GO TO 6
00336      C     THE LOCATEE HAS BEEN FOUND.
00337         15 IF(FXD(JFIND)) GO TO 16
00337      C     THE LOCATEE IS NOT A FIXED UNIT.
00341            EP(JFIND,1)=EP(JFIND,10)
00342            EP(JFIND,2)=EP(JFIND,11)
00343            EP(JFIND,3)=EP(JFIND,12)
00344            NFXD(JFIND)=0
00345            W(JFIND)=1./(W(JFIND)+RVAR)
00346            GO TO 20
00346      C     THE LOCATEE IS A FIXED UNIT, SO A WEIGHTED AVERAGE OF ITS OLD AND
00346      C     NEW POSITIONS IS CALCULATED.  A WEIGHTED AVERAGE OF ITS OLD AND
00346      C     NEW WEIGHTS IS ALSO CALCULATED.
00347         16 D=WOLD+W(JFIND)
00350            XNEW=(EP(JFIND,4)*W(JFIND)+XNEW*WOLD)/D
00351            YNEW=(EP(JFIND,5)*W(JFIND)+YNEW*WOLD)/D
00352            ZNEW=(EP(JFIND,6)*W(JFIND)+ZNEW*WOLD)/D
00353         19 WOLD=WOLD*NFXD(JFIND)**2
00354            NFXD(JFIND)=NFXD(JFIND)+1
00355            MFXD=NFXD(JFIND)**2
00356            W(JFIND)=MFXD/(RVAR*MFXD+WOLD+W(JFIND))
00357            EP(JFIND,1)=XNEW
00360            EP(JFIND,2)=YNEW
00361            EP(JFIND,3)=ZNEW
00361      C     UPDATE THE ESTIMATED POSITION AND TIME OF LOCATION ARRAYS.
00362         20 EP(JFIND,4)=XNEW
00363            EP(JFIND,5)=YNEW
00364            EP(JFIND,6)=ZNEW
00365            EP(JFIND,7)=XNEW
00366            EP(JFIND,8)=YNEW
00367            EP(JFIND,9)=ZNEW
00370            EP(JFIND,10)=XNEW
00371            EP(JFIND,11)=YNEW
00372            EP(JFIND,12)=ZNEW
00373            EP(JFIND,13)=0.
00374            TLOC(JFIND,1)=TLOC(JFIND,4)
00375            TLOC(JFIND,2)=CLK
00376            TLOC(JFIND,3)=CLK
00377            TLOC(JFIND,4)=CLK
00377      C     RELOCATE THE LOCATORS (THIS OPTION IS NOT NOW USED--SEE LSS3D).
00400            DO 36 I=1,L
00403            J=LAM(I)
00404            EP(J,7)=XX(I)
00405            EP(J,8)=YY(I)
00406            EP(J,9)=ZZ(I)
00407            EP(J,13)=EP(J,13)+1.0
00410            IF (EP(J,13).LT.EP(J,14)) GO TO 36
00412            EP(J,1)=EP(J,4)
00413            EP(J,2)=EP(J,5)
00414            EP(J,3)=EP(J,6)
00415            EP(J,4)=EP(J,7)
00416            EP(J,5)=EP(J,8)
```

```
00417                EP(J,6)=EP(J,9)
00420                EP(J,13)=0.0
00421                TLOC(J,1)=TLOC(J,2)
00422                TLOC(J,2)=TLOC(J,3)
00423             36 TLOC(J,3)=CLK
00425                RETURN
00425          C     IF YOU WANT THE REPORTING UNITS TO BE PRECISELY THOSE THAT HAVE
00425          C     BEEN FIXED FOR TEN LOCATION OPERATIONS, REMOVE THE ABOVE RETURN
00425          C     STATEMENT, AND REMOVE THE C FROM COLUMN 1 ON THE CARD BELOW.
00425          C     RPT(JFIND)=NFXD(JFIND).GE.LFXD
00426                IF(RPT(JFIND).OR.W(JFIND).LE.WORST)  RETURN
00430                RPT(NWORST)=.FALSE.
00431                RPT(JFIND)=.TRUE.
00432                WORST=W(JFIND)
00433                NWORST=JFIND
00434                DO 25 I=1,MM
00437                L=LAMM(I)
00440                IF(W(L).GT.WORST) GO TO 25
00442                WORST=W(L)
00443                NWORST=L
00444             25 CONTINUE
00446                RETURN
00447                END

       END OF UNIVAC 1108 FORTRAN V COMPILATION.      0 *DIAGNOSTIC* MESSAGE(S)
PHASE 1 TIME =   1 SEC.
PHASE 2 TIME =   0 SEC.
PHASE 3 TIME =   1 SEC.
PHASE 4 TIME =   0 SEC.
PHASE 5 TIME =   1 SEC.
PHASE 6 TIME =   0 SEC.

TOTAL COMPILATION TIME =    3  SEC
```

```
@IT FOR LOCLSL,LOCLSL                                                                    05 MAY 72          11:41:45.737
UNIVAC 1108 FORTRAN V LEVEL   2206 0018 F5018P
THIS COMPILATION WAS DONE ON 05 MAY 72 AT 11:41:45

    SUBROUTINE LOCLSL    ENTRY POINT 000464

    STORAGE USED (BLOCK, NAME, LENGTH)

        0001    *CODE    000501
        0000    *DATA    003176
        0002    *BLANK   000000
        0003    CB1      013000
        0004    MXYZ     000673

    EXTERNAL REFERENCES (BLOCK, NAME)

        0005    NWDU$
        0006    N102$
        0007    NEXP1$
        0010    NERR3$

    STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

    0001    000035 135G      0001    000044 141G      0001    000247 200G      0001    000333 216G      0001    000376 241G
    0001    000173 3L        0001    000177 4L        0001    000352 8L        0000    003126 99F       0000 R 000000 A
    0000 R 000620 B          0000 R 001440 C          0003 R 011531 CLK        0000 R 003122 CUTOFF     0000 R 003116 D
    0000 R 003117 E          0000 R 003124 ELIM       0003 R 000000 EP         0000 R 003120 F          0003 R 011535 FXD
    0003 I 011530 JFIND      0000 I 003115 K          0004 I 000001 L          0003 I 006116 LAM        0004 I 000000 M
    0000 I 003102 MELIM      0000 I 003123 MLINE      0000 I 003125 N          0000 I 003114 NBEGIN     0000 I 003101 NELIM
    0000 I 003104 NLINE      0000 I 003103 NN         0000 I 003113 NOFF       0003 I 012437 NTYPE      0004 L 000672 P
    0004 R 000542 R          0003 R 012076 RPT        0003 R 006460 RR         0003 L 007022 S          0003 R 007343 TLOC
    0000 R 003100 TOL        0000 R 003105 U1         0000 R 003106 U2         0000 R 003121 VAR        0000 R 002260 VRSQ
    0000 R 003107 V1         0000 R 003110 V2         0003 R 011167 W          0004 R 000412 WW         0000 R 003111 W1
    0000 R 003112 W2         0003 R 011532 XNEW       0004 R 000002 XX         0003 R 011533 YNEW       0004 R 000132 YY
    0003 R 011534 ZNEW       0004 R 000262 ZZ

00101      1*          SUBROUTINE LOCLSL
00101      2*    C
00101      3*    C     LEAST SQUARES LINEAR (LSL).
00101      4*    C     THIS ROUTINE MINIMIZES THE SUM OF THE SQUARED DISTANCES FROM THE
00101      5*    C     LOCATEE TO THE RADICAL AXIS LINES FORMED BY PAIRS OF CIRCLES.
00101      6*    C     EACH CIRCLE HAS A LOCATOR FOR ITS CENTER, AND THE REPORTED
00101      7*    C     DISTANCE TO THE LOCATEE FOR ITS RADIUS.  SEE FIRST INTERIM REPORT
00101      8*    C     ON PROJECT WHERSM, PP. 10-13.
00101      9*    C     THIS IS THE TWO-DIMENSIONAL ROUTINE.
00101     10*    C
00103     11*          PARAMETER L1=225,L4=L1+1
00104     12*          DIMENSION                  A(400),B(400),C(400),VRSQ(400)
00105     13*          COMMON/CB1/EP(L1,14),LAM(L4),RR(L4),S(L1),TLOC(L1,4),W(L1),
00105     14*         *          JFIND,CLK,XNEW,YNEW,ZNEW,FXD(L1),RPT(L1),NTYPE(L1)
```

```
      COMMON/MXYZ/M,L,XX(88),YY(88),ZZ(88),WW(88),R(88),P
      LOGICAL S,P
      DATA TOL/50./
      P=.TRUE.
      IF (P) WRITE(6,99) JFIND,CLK
   99 FORMAT(' * * LSL2D; UNIT =',I5,' CLOCK =',F10.4)
      S(JFIND)=.FALSE.
      NELIM=4
      MELIM=0
      NN=M-1
    2 NLINE=0
      U1=0.
      U2=0.
      V1=0.
      V2=0.
      W1=0.
      W2=0.
      DO 3 NOFF=1,NN
      NBEGIN=NOFF+1
C     FOR EACH PAIR OF CIRCLES, CALCULATE THE COEFFICIENTS OF THE
C     RADICAL AXIS LINE, A*X+B*Y-C=0, WHERE A*A+B*B=1.  CALCULATE
C     COEFFICIENTS OF THE TWO EQUATIONS IN X AND Y, ON BOTTOM OF P. 13
C     OF FIRST INTERIM REPORT.
      DO 3 L=NBEGIN,M
      IF(NLINE.EQ.400) GO TO 4
      K=L-NOFF
      IF ((XX(K)-XX(L))**2+(YY(K)-YY(L))**2.LT.1.) GO TO 3
      NLINE=NLINE+1
      A(NLINE)=2.*(XX(K)-XX(L))
      B(NLINE)=2.*(YY(K)-YY(L))
      C(NLINE)=RR(L)**2-RR(K)**2+XX(K)**2-XX(L)**2+YY(K)**2-YY(L)**2
      D=A(NLINE)*A(NLINE)+B(NLINE)*B(NLINE)
      E=A(NLINE)/D
      F=B(NLINE)/D
      U1=U1+A(NLINE)*E
      V1=V1+B(NLINE)*E
      W1=W1+C(NLINE)*E
      U2=U2+A(NLINE)*F
      V2=V2+B(NLINE)*F
      W2=W2+C(NLINE)*F
    3 CONTINUE
    4 D=U1*V2-V1*U2
      IF(1.E6*ABS(D).LE.ABS(U1*V2)) RETURN
C     THE SYSTEM OF TWO EQUATIONS IS NON-SINGULAR.  SOLVE FOR XNEW AND
C     CALCULATE SQUARED DISTANCES TO EACH RADICAL AXIS, AND AVERAGE.
      S(JFIND)=.TRUE.
      XNEW=(W1*V2-V1*W2)/D
      YNEW=(U1*W2-W1*U2)/D
      VAR=0.
      DO 6 M=1,NLINE
      VRSQ(M)=(A(M)*XNEW+B(M)*YNEW-C(M))**2/(A(M)*A(M)+B(M)*B(M))
    6 VAR=VAR+VRSQ(M)
      VAR=VAR/FLOAT(NLINE)
      MELIM=MELIM+1
      IF(MELIM.GT.NELIM) RETURN
      CUTOFF=2**(4-MELIM)
      MLINE=NLINE
      NLINE=0
```

```
00214               ELIM=CUTOFF*VAR
00214          C    ELIMINATE RADICAL AXES WHOSE SQUARED DISTANCE, VRSQ, IS GREATER
00214          C    THAN CUTOFF TIMES THE AVERAGE, WHERE CUTOFF IS, SUCCESSIVELY,
00214          C    8, 4, 2, AND 1.
00215               DO 8 N=1,MLINE
00220               IF(VRSQ(N).GT.ELIM) GO TO 8
00222               NLINE=NLINE+1
00223               A(NLINE)=A(N)
00224               B(NLINE)=B(N)
00225               C(NLINE)=C(N)
00226             8 CONTINUE
00230               IF(NLINE.LE.2) RETURN
00232               U1=0.
00233               U2=0.
00234               V1=0.
00235               V2=0.
00236               W1=0.
00237               W2=0.
00237          C    RECALCULATE COEFFICIENTS OF THE TWO LINEAR EQUATIONS.
00240               DO 9 N=1,NLINE
00243               D=A(N)*A(N)+B(N)*B(N)
00244               E=A(N)/D
00245               F=B(N)/D
00246               U1=U1+A(N)*E
00247               V1=V1+B(N)*E
00250               W1=W1+C(N)*E
00251               U2=U2+A(N)*F
00252               V2=V2+B(N)*F
00253             9 W2=W2+C(N)*F
00255               GO TO 4
00256               END

       END OF UNIVAC 1108 FORTRAN V COMPILATION.      0 *DIAGNOSTIC* MESSAGE(S)
PHASE 1 TIME =   1 SEC.
PHASE 2 TIME =   0 SEC.
PHASE 3 TIME =   0 SEC.
PHASE 4 TIME =   0 SEC.
PHASE 5 TIME =   1 SEC.
PHASE 6 TIME =   0 SEC.

TOTAL COMPILATION TIME =   2 SEC
```

```
@IT FOR LOCLSS,LOCLSS
UNIVAC 1108 FORTRAN V LEVEL   2206 0018 F5018P
THIS COMPILATION WAS DONE ON 05 MAY 72 AT 11:41:47

    SUBROUTINE LOCLSS      ENTRY POINT 000276

    STORAGE USED (BLOCK, NAME, LENGTH)

          0001    *CODE    000636
          0000    *DATA    000300
          0002    *BLANK   000000
          0003    CB1      013000
          0004    CB3      002261
          0005    MXYZ     000673


    EXTERNAL REFERENCES (BLOCK, NAME)

          0006    NEXP1$
          0007    NERR3$


    STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

      0001   000010 1L         0001   000016 132G       0001   000073 151G       0001   000512 2I         0001   000226 230G
      0001   000220 25L        0001   000324 254G       0001   000551 322G       0001   000032 4L         0001   000155 5L
      0001   000203 8L         0000   000170 80F        0000   000173 81F        0000   000175 82F        0003 R 011531 CLK
      0000 R 000166 D          0000 R 000157 DELXSQ     0000 R 000160 DELYSQ     0000 R 000153 DFDX       0000 R 000155 DFDY
      0000 R 000162 DGDX       0000 R 000154 DGDY       0000 R 000161 DSQ        0000 R 000165 DSQ        0000 R 000163 DX
      0000 R 000164 DY         0000 R 000146 E          0000 R 000143 EAVE       0000 R 000000 EE         0000 R 000142 ELIM
      0003 R 000000 EP         0000 R 000151 F          0000 R 000167 F          0000 R 000132 FRAC       0003 R 011535 FXD
      0000 R 000152 G          0000 I 000144 I          0000 I 000141 I1         0004 I 000000 IN         0004 I 002260 I9
      0000 I 000145 J          0003 I 011530 JFIND      0000 I 000135 K          0005 I 000001 L          0003 I 006116 LAM
      0000 I 000134 LR         0005 I 000000 M          0000 I 000140 MELIM      0000 I 000136 MRELOC     0000 I 000130 NELIM
      0000 I 000131 NRELOC     0003 I 012437 NTYPE      0000 R 000137 OLDEAV     0005 L 000672 P          0005 R 000542 R
      0003 R 012076 RPT        0003 R 006460 RR         0000 R 000150 RSQ        0003 L 007022 S          0000 L 000133 STOP
      0003 R 007363 TLOC       0000 R 000156 TW         0000 R 000147 V          0003 R 011167 W          0005 R 000412 WW
      0003 R 011532 XNEW       0005 R 000002 XX         0003 R 011533 YNEW       0005 R 000132 YY         0003 R 011534 ZNEW
      0005 R 000262 ZZ


00101       1*          SUBROUTINE LOCLSS
00101       2*    C     THIS IS THE TWO-DIMENSIONAL LEAST SQUARES SQUARED ROUTINE.  IT
00101       3*    C     ATTEMPTS TO MINIMIZE THE FUNCTION EAVE BY FINDING A POINT WHERE
00101       4*    C     ITS DERIVATIVES F AND G, WITH RESPECT TO X AND Y ARE ZERO.  TO DO
00101       5*    C     THIS, IT USES NEWTON-RAPHSON ITERATION.
00101       6*    C
00103       7*          PARAMETER L1=225,L4=L1+1
00104       8*          COMMON     /CB1/EP(L1,14),LAM(L4),RR(L4),S(L1),TLOC(L1,4),W(L1),
00104       9*         *           JFIND,CLK,XNEW,YNEW,ZNEW,FXD(L1),RPT(L1),NTYPE(L1)
00105      10*          COMMON /CB3/IN(600,2),I9
00106      11*          COMMON/MXYZ/M,L,XX(88),YY(88),ZZ(88),WW(88),R(88),P
00107      12*          DIMENSION EE(88)
```

```
00[illegible]          DATA NELIM,MRELOC,[illegible]/1,0,.1/
00114          LOGICAL S,STOP
00115          LOGICAL   P
00116          DATA P/.FALSE./
00120          IN(I9,1)=0
00121          IN(I9,2)=0
00122          LR=0
00123          L=M
00124          K=0
00125          MRELOC=0
00126        1 STOP=.FALSE.
00127          OLDEAV=1.E30
00130          MELIM=0
00130    C     DO UP TO FOUR NEWTON-RAPHSON ITERATIONS, STOPPING IF CONVERGENCE
00130    C     OR DIVERGENCE OCCURS.
00131          DO 2 II=0,3
00134          IF(STOP) GO TO 4
00136          CALL NEWTON
00137          IF(S(JFIND)) GO TO 4
00141        2 CONTINUE
00143        4 MELIM=MELIM+1
00144          K=0
00145          IF(MELIM.GT.NELIM) GO TO 8
00145    C     ELIMINATE ANY LOCATOR I SUCH THAT EE(I) IS TOO BIG.  PUT
00145    C     ELIMINATED LOCATORS AT THE BOTTOM OF THE LIST.
00147          ELIM=EAVE*2**(3-MELIM)
00150          DO 5 I=L,1,-1
00153          IF(EE(I).LE.ELIM) GO TO 5
00155          K=K+1
00156          J=L+1-K
00157          IF(I.EQ.J) GO TO 5
00161          E=WW(J)
00162          WW(J)=WW(I)
00163          WW(I)=E
00164          E=XX(J)
00165          XX(J)=XX(I)
00166          XX(I)=E
00167          E=YY(J)
00170          YY(J)=YY(I)
00171          YY(I)=E
00172          E=ZZ(J)
00173          ZZ(J)=ZZ(I)
00174          ZZ(I)=E
00175          E=R(J)
00176          R(J)=R(I)
00177          R(I)=E
00200          E=RR(J)
00201          RR(J)=RR(I)
00202          RR(I)=E
00203          II=LAM(J)
00204          LAM(J)=LAM(I)
00205          LAM(I)=II
00206        5 CONTINUE
00210          IN(I9,MRELOC+1)=IN(I9,MRELOC+1)+K
00210    C     DO NOT ELIMINATE MORE THAN ONE FOURTH OF THE LOCATORS.
00211          IF(4*(L-K).LT.3*M) GO TO 25
00213          L=L-K
00214          IF(K) 4,4,1
```

```
00217         8 MRELOC=MRELOC+1
00220           IF(MRELOC.GT.NRELOC) GO TO 25
00220     C     RELOCATE THE LOCATORS.  IF NRELOC=0 (SEE DATA STATEMENTS), THIS
00220     C     OPTION IS NOT USED.
00222           LR=L
00223           CALL RELOC
00224           MELIM=0
00225           GO TO 1
00225     C     CALCULATE VARIANCE OF THE POSITION LOCATION ESTIMATE.
00226        25 V=0.
00227           DO 26 I=1,L
00232           RSQ=RR(I)**2
00233        26 V=V+(RSQ-((XNEW-XX(I))**2+(YNEW-YY(I))**2))**2/RSQ
00235           W(JFIND)=V/(4*L-8)
00236           L=LR
00237           RETURN
00237     C     NEWTON-RAPHSON ITERATION.  CALCULATE F, G, AND THEIR DERIVATIVES.
00237     C     SOLVE FOR DX AND DY, AND INCREMENT XNEW AND YNEW.
00240           SUBROUTINE NEWTON
00243           F=0.
00244           G=0.
00245           DFDX=0.
00246           DGDY=0.
00247           DFDY=0.0
00250           EAVE=0.
00250     C     IF (P) WRITE(6,80)
00251        80 FORMAT(' R,R2,D2,E')
00252           TW=0.
00253           DO 1 I=1,L
00256           RSQ=RR(I)**2
00257           DELXSQ=(XNEW-XX(I))**2/RSQ
00260           DELYSQ=(YNEW-YY(I))**2/RSQ
00261           DSQ=(DELXSQ+DELYSQ)*RSQ
00262           E=DELXSQ-1.+DELYSQ
00263           EE(I)=E*E*RSQ
00264           EAVE=EAVE+EE(I)*WW(I)
00264     CC    IF (P) WRITE(6,81) RR(I),RSQ,DSQ,EE(I)
00265        81 FORMAT(9E14.8)
00266           TW=TW+WW(I)
00267           F=F+E*(XNEW-XX(I))*WW(I)
00270           G=G+E*(YNEW-YY(I))*WW(I)
00271           DFDX=DFDX+(E+2.*DELXSQ)*WW(I)
00272           DGDY=DGDY+(E+2.*DELYSQ)*WW(I)
00273         1 DFDY=DFDY+2.*(XNEW-XX(I))*(YNEW-YY(I))*WW(I)/RSQ
00275           EAVE=EAVE/TW
00276           IF(EAVE.GT.OLDEAV) GO TO 2
00300           OLDEAV=EAVE
00301           DGDX=DFDY
00302           E=DFDX*DGDY-DFDY*DGDX
00303           IF(1.E6*ABS(E).LE.ABS(DFDX*DGDY)) GO TO 2
00305           DX=(DFDY*G-DGDY*F)/E
00306           DY=(DGDX*F-DFDX*G)/E
00307           XNEW=XNEW+DX
00310           YNEW=YNEW+DY
00311           S(JFIND)=AMAX1(ABS(F),ABS(G)).LT.TW
00311     C     IF (P) WRITE(6,82) DX,DY,XNEW,YNEW,F,G,EAVE
00312        82 FORMAT(' ΔX,ΔY,XNEW,YNEW,DE/DX,DE/DY,EAVE'/8E14.8)
00313           RETURN
```

```
00313     C     THE ITERATION DIVERGED.  DO NOT CHANGE XNEW AND YNEW.
00314         2 STOP=.TRUE.
00315           RETURN
00315     C     RELOCATE THE LOCATORS.
00316           SUBROUTINE RELOC
00321           DO 1 I=1,L
00324           DSQ=(XNEW-XX(I))**2+(YNEW-YY(I))**2
00325           D=RR(I)
00326           D=.5*(D+DSQ/D)
00327           D=.5*(D+DSQ/D)
00330           F=FRAC*(D-RR(I))/D
00331           XX(I)=XX(I)+F*(XNEW-XX(I))
00332           YY(I)=YY(I)+F*(YNEW-YY(I))
00333         1 ZZ(I)=ZZ(I)+F*(ZNEW-ZZ(I))
00335           RETURN
00336           END

      END OF UNIVAC 1108 FORTRAN V COMPILATION.      0 *DIAGNOSTIC* MESSAGE(S)
PHASE 1 TIME =   0 SEC.
PHASE 2 TIME =   1 SEC.
PHASE 3 TIME =   0 SEC.
PHASE 4 TIME =   0 SEC.
PHASE 5 TIME =   1 SEC.
PHASE 6 TIME =   0 SEC.

TOTAL COMPILATION TIME =   2  SEC
```

```
UNIVAC 1108 FORTRAN V LEVEL   2206 0018 F5018P                                    05 MAY 72          11:41:49.54
THIS COMPILATION WAS DONE ON 05 MAY 72 AT 11:41:49

     SUBROUTINE LSS3D      ENTRY POINT 000310

     STORAGE USED (BLOCK, NAME, LENGTH)

          0001    *CODE    001216
          0000    *DATA    000322
          0002    *BLANK   000000
          0003    CB1      013000
          0004    CB3      002261
          0005    MXYZ     000673

     EXTERNAL REFERENCES (BLOCK, NAME)

          0006    NEXP1$
          0007    NERR3$
          0010    NWDU$
          0011    NI02$

     STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

     0001   001046 1L        0001   000010 1L        0001   000022 133G      0001   000102 163G      0001   000723 2L
     0001   000234 231G      0001   000226 25L       0001   000341 260G      0001   001120 375G      0001   000040 4L
     0001   000164 5L        0001   001062 5L        0001   000212 8L        0000   000214 9F        0000   000204 99F
     0000 R 000200 ALPHA     0003 R 011531 CLK       0000 R 000145 CUTOFF    0000 R 000202 D         0000 R 000170 DELXSQ
     0000 R 000171 DELYSQ    0000 R 000172 DELZSQ    0000 R 000173 DET       0000 R 000174 DETINV    0000 R 000160 DFDX
     0000 R 000161 DFDY      0000 R 000162 DFDZ      0000 R 000163 DGDY      0000 R 000164 DGDZ      0000 R 000165 DHDZ
     0000 R 000201 DSQ       0000 R 000175 DX        0000 R 000176 DY        0000 R 000177 DZ        0000 R 000152 E
     0000 R 000147 EAVE      0000 R 000000 EE        0000 R 000146 ELIM      0003 R 000000 EP        0000 R 000141 EX
     0000 R 000142 EY        0000 R 000143 EZ        0000 R 000155 F         0000 R 000203 F         0000 R 000133 FRAC
     0003 R 011535 FXD       0000 R 000156 G         0000 R 000157 H         0000 I 000150 I         0000 I 000144 II
     0004 I 000000 IN        0004 I 002260 I9        0000 I 000151 J         0003 I 011530 JFIND     0000 I 000135 K
     0005 I 000001 L         0003 I 006116 LAM       0000 I 000134 LR        0005 I 000000 M         0000 I 000137 MELIM
     0000 I 000136 MRELOC    0000 I 000131 NELIM     0000 I 000132 NRELOC    0003 I 012437 NTYPE     0000 R 000140 OLDEAV
     0005 L 000672 P         0005 R 000542 R         0003 R 012076 RPT       0003 R 006460 RR        0000 R 000154 RSQ
     0000 R 000167 RSQINV    0003 L 007022 S         0000 L 000130 STOP      0003 R 007363 TLOC      0000 R 000166 TW
     0000 R 000153 V         0003 R 011167 W         0005 R 000412 WW        0003 R 011532 XNEW      0005 R 000002 XX
     0003 R 011533 YNEW      0005 R 000132 YY        0003 R 011534 ZNEW      0005 R 000262 ZZ

00101       1*          SUBROUTINE LSS3D
00101       2*     C    THIS SUBROUTINE IS THE 3-DIMENSIONAL LEAST SQUARES SQUARED METHOD.
00101       3*     C    IT MINIMIZES A FUNCTION, EAVE, BY TRYING TO GET ITS 3 DERIVATIVES
00101       4*     C    (F,G,H RESPECTIVELY) WITH RESPECT TO X, Y, AND Z TO BE ZERO.
00101       5*     C    NEWTON-RAPHSON ITERATION IS USED TO APPROACH THE ZEROES.
00101       6*     C
00103       7*          PARAMETER L1=225,L4=L1+1
00104       8*          COMMON/CB1/EP(L1,14),LAM(L4),RR(L4),S(L1),TLOC(L1,4),W(L1),
```

```
     JFIND,CLK,XNEW,YNEW,ZNEW,FXD(L1),RPT(L1),NTYPE(L1)
      COMMON /CB3/IN(600,2),I9
      COMMON/MXYZ/M,L,XX(88),YY(88),ZZ(88),WW(88),R(88),P
      DIMENSION EE(88)
      LOGICAL S,STOP,P
      DATA NELIM,NRELOC,FRAC/1,0,.2/
C     IF (P) WRITE(6,99) JFIND,CLK
   99 FORMAT(' * * LSS3D: UNIT =' I5,' CLOCK =' F10.4)
      IN(I9,1)=0
      IN(I9,2)=0
      LR=0
      L=M
      K=0
      MRELOC=0
    1 STOP=.FALSE.
      MELIM=0
      OLDEAV=1.E30
      EX=10000.
      EY=10000.
      EZ=10000.
C     DO UP TO FOUR ITERATIONS, STOPPING IF CONVERGENCE OCCURS, OR IF
C     THE FIRST ITERATION DIVERGES.
      DO 2 II=0,3
      IF(STOP) GO TO 4
      CALL NEWTON(II,K)
      IF(S(JFIND)) GO TO 4
    2 CONTINUE
    4 MELIM=MELIM+1
      K=0
      IF(MELIM.GT.NELIM) GO TO 8
      CUTOFF=2**(3-MELIM)
      ELIM=CUTOFF*EAVE
C     ELIMINATE ANY LOCATOR I SUCH THAT EE(I) IS TOO BIG.  PUT
C     ELIMINATED LOCATORS AT THE BOTTOM OF THE LIST.
      DO 5 I=L,1,-1
      IF(EE(I).LE.ELIM) GO TO 5
      K=K+1
      J=L+1-K
      IF(I.EQ.J) GO TO 5
      E=WW(J)
      WW(J)=WW(I)
      WW(I)=E
      E=XX(J)
      XX(J)=XX(I)
      XX(I)=E
      E=YY(J)
      YY(J)=YY(I)
      YY(I)=E
      E=ZZ(J)
      ZZ(J)=ZZ(I)
      ZZ(I)=E
      E=R(J)
      R(J)=R(I)
      R(I)=E
      E=RR(J)
      RR(J)=RR(I)
      RR(I)=E
      II=LAM(J)
```

```
00206          LAM(J)=LAM(I)
00207          LAM(I)=II
00210        5 CONTINUE
00212          IN(I9,MRELOC+1)=IN(I9,MRELOC+1)+K
00212    C     DO NOT ELIMINATE MORE THAN ONE FOURTH OF THE LOCATORS.
00213          IF(4*(L-K).LT.3*M) GO TO 25
00215          L=L-K
00216          IF(K) 4,4,1
00221        8 MRELOC=MRELOC+1
00222          IF(MRELOC.GT.NRELOC) GO TO 25
00222    C     RELOCATE THE LOCATORS.  IF NRELOC=0 (SEE DATA STATEMENTS), THIS
00222    C     OPTION IS NOT NOW USED.
00224          LR=L
00225          CALL RELOC
00226          GO TO 1
00226    C     CALCULATE VARIANCE OF THE POSITION ESTIMATE.
00227       25 V=0.
00230          DO 26 I=1,L
00233          RSQ=RR(I)**2
00234       26 V=V+(RSQ-((XNEW-XX(I))**2+(YNEW-YY(I))**2+(ZNEW-ZZ(I))**2))**2/RSQ
00236          W(JFIND)=V/(4*L-12)
00237          L=LR
00240          RETURN
00240    C     NEWTON RAPHSON ITERATION.  CALCULATE F, G, H AND THEIR DERIVATIVES
00240    C     AND FIND DX, DY, DZ BY SOLVING EQUATIONS (5) OF WORKING PAPER NO.
00240    C     7 OF THE FINAL REPORT (MAY 1972).
00241          SUBROUTINE NEWTON(II,K)
00244          F=0.
00245          G=0.
00246          H=0.
00247          DFDX=0.
00250          DFDY=0.
00251          DFDZ=0.
00252          DGDY=0.
00253          DGDZ=0.
00254          DHDZ=0.
00255          EAVE=0.
00256          TW=0.
00257          DO 1 I=1,L
00262          RSQ=RR(I)**2
00263          RSQINV=1./RSQ
00264          DELXSQ=RSQINV*(XNEW-XX(I))**2
00265          DELYSQ=RSQINV*(YNEW-YY(I))**2
00266          DELZSQ=RSQINV*(ZNEW-ZZ(I))**2
00267          E=DELXSQ+DELYSQ-1.+DELZSQ
00270          EE(I)=E*E*RSQ
00271          EAVE=EAVE+EE(I)*WW(I)
00272          TW=TW+WW(I)
00273          F=F+E*(XNEW-XX(I))*WW(I)
00274          G=G+E*(YNEW-YY(I))*WW(I)
00275          H=H+E*(ZNEW-ZZ(I))*WW(I)
00276          DFDX=DFDX+(E+2.*DELXSQ)*WW(I)
00277          DFDY=DFDY+2.*(XNEW-XX(I))*(YNEW-YY(I))*RSQINV*WW(I)
00300          DFDZ=DFDZ+2.*(XNEW-XX(I))*(ZNEW-ZZ(I))*RSQINV*WW(I)
00301          DGDY=DGDY+(E+2.*DELYSQ)*WW(I)
00302          DGDZ=DGDZ+2.*(YNEW-YY(I))*(ZNEW-ZZ(I))*RSQINV*WW(I)
00303        1 DHDZ=DHDZ+(E+2.*DELZSQ)*WW(I)
00305          EAVE=EAVE/TW
```

```
00306              IF(EAVE.GT.OLDEAV) GO TO 2
00310              DET=DFDX*(DGDY*DHDZ-DGDZ**2)+DFDY*(DGDZ*DFDZ-DFDY*DHDZ)
00310             1 +DFDZ*(DFDY*DGDZ-DGDY*DFDZ)
00311              IF(ABS(DET).LT.1.E-8) GO TO 2
00313              DETINV=1./DET
00314              DX=DETINV*(F*(DGDZ**2-DGDY*DHDZ)+G*(DFDY*DHDZ-DFDZ*DGDZ)+
00314             1 H*(DFDZ*DGDY-DFDY*DGDZ))
00315              DY=DETINV*(F*(DFDY*DHDZ-DGDZ*DFDZ)+G*(DFDZ**2-DFDX*DHDZ)+
00315             1 H*(DFDX*DGDZ-DFDZ*DFDY))
00316              DZ=DETINV*(F*(DGDY*DFDZ-DFDY*DGDZ)+G*(DFDX*DGDZ-DFDY*DFDZ)+
00316             1 H*(DFDY**2-DFDX*DGDY))
00317              OLDEAV=EAVE
00317        C     IF(P)WRITE(6,9)F,G,H,DFDX,DFDY,DFDZ,DGDY,DGDZ,DHDZ,XNEW,YNEW,ZNEW,
00317        C    1 DX,DY,DZ,(XX(I),I=1,M),(YY(I),I=1,M),(ZZ(I),I=1,M),(R(I),I=1,M),
00317        C    2(WW(I),I=1,M),(EE(I),I=1,M)
00320           9 FORMAT(/(1X,15E8.3))
00320        C     IF THIS STEP SIZE IS NOT MUCH SMALLER THAN THAT OF THE PREVIOUS
00320        C     ITERATION, CORRECT THE SIZE OF THIS ITERATION.
00321              CALL CORECT(EX,DX)
00322              CALL CORECT(EY,DY)
00323              CALL CORECT(EZ,DZ)
00324              XNEW=XNEW+DX
00325              YNEW=YNEW+DY
00326              ZNEW=ZNEW+DZ
00327              S(JFIND)=AMAX1(ABS(F),ABS(G),ABS(H)).LT.TW
00330              RETURN
00330        C     THE ITERATION HAS DIVERGED.  IF THIS IS THE FIRST ITERATION FOR
00330        C     THIS LOCATEE, THE ROUTINE CONKS OUT.
00331           2 STOP=.TRUE.
00332              WRITE(6,9)  EAVE,OLDEAV,DET
00337              IF(II.EQ.0.AND.K.EQ.0) RETURN
00337        C     DIVERGENCE FOR SOME OTHER ITERATION (NOT THE FIRST).  HALVE THE
00337        C     STEP SIZE OF THE PREVIOUS ITERATION, AND RETURN TO TRY ANOTHER
00337        C     ITERATION.
00341              STOP=.FALSE.
00342              DX=DX/2.
00343              DY=DY/2.
00344              DZ=DZ/2.
00345              XNEW=XNEW-DX
00346              YNEW=YNEW-DY
00347              ZNEW=ZNEW-DZ
00350              RETURN
00351              SUBROUTINE CORECT(A,B)
00354              IF(ABS(B/A).LT..5) GO TO 5
00356              ALPHA=B/A
00357              IF(ALPHA.GT.0.)GO TO 1
00361              B=B/(1.-ALPHA)
00362              GO TO 5
00363           1 A=2.
00364              IF(ALPHA.LT.1.)  A=2.*ALPHA
00366              B=B*A
00367           5 A=B
00370              RETURN
00370        C     RELOCATE THE LOCATORS.
00371              SUBROUTINE RELOC
00374              DO 1 I=1,L
00377              DSQ=(XNEW-XX(I))**2+(YNEW-YY(I))**2+(ZNEW-ZZ(I))**2
00400              D=RR(I)
```

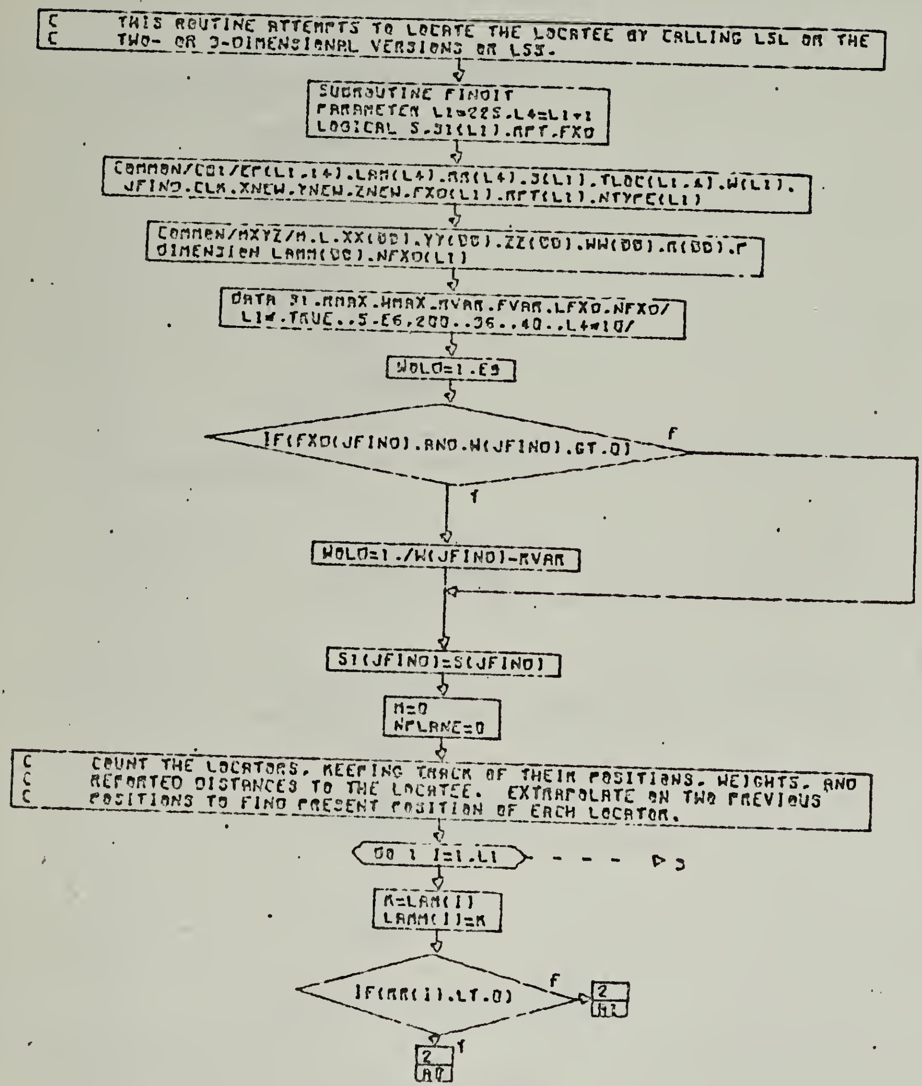
```
00401    183*          D=.5*(D+DSQ/D)
00402    184*          D=.5*(D+DSQ/D)
00403    185*          D=.5*(D+DSQ/D)
00404    186*          F=FRAC*(D-RR(I))/D
00405    187*          XX(I)=XX(I)+F*(XNEW-XX(I))
00406    188*          YY(I)=YY(I)+F*(YNEW-YY(I))
00407    189*        1 ZZ(I)=ZZ(I)+F*(ZNEW-ZZ(I))
00411    190*          RETURN
00412    191*          END

       END OF UNIVAC 1108 FORTRAN V COMPILATION.      0 *DIAGNOSTIC* MESSAGE(S)
 PHASE 1 TIME =   1 SEC.
 PHASE 2 TIME =   0 SEC.
 PHASE 3 TIME =   0 SEC.
 PHASE 4 TIME =   0 SEC.
 PHASE 5 TIME =   1 SEC.
 PHASE 6 TIME =   0 SEC.

 TOTAL COMPILATION TIME =    2  SEC
```

```
 ** CONSOLE TYPEOUTS **
 TIME       TYPE  MESSAGE
11:41:42     E    ### 205 RUN 11:41:42    ***************************************
11:41:42     E    A-09
11:41:42     E    SOOKNE,01563,1,30
11:41:48     E    114147 GLADNE  130 MNT      @@6007,
11:41:52     E    I:  649 C:     OP:    2;LP:    818 E:0  0 10 L: 189
11:41:52     E     418 COUNT IS  9
11:41:54     E     ZFILE COUNT IS     1


 TAPE READS       0,RETRYS    0
 TAPE WRITES      0,RETRYS    0
```

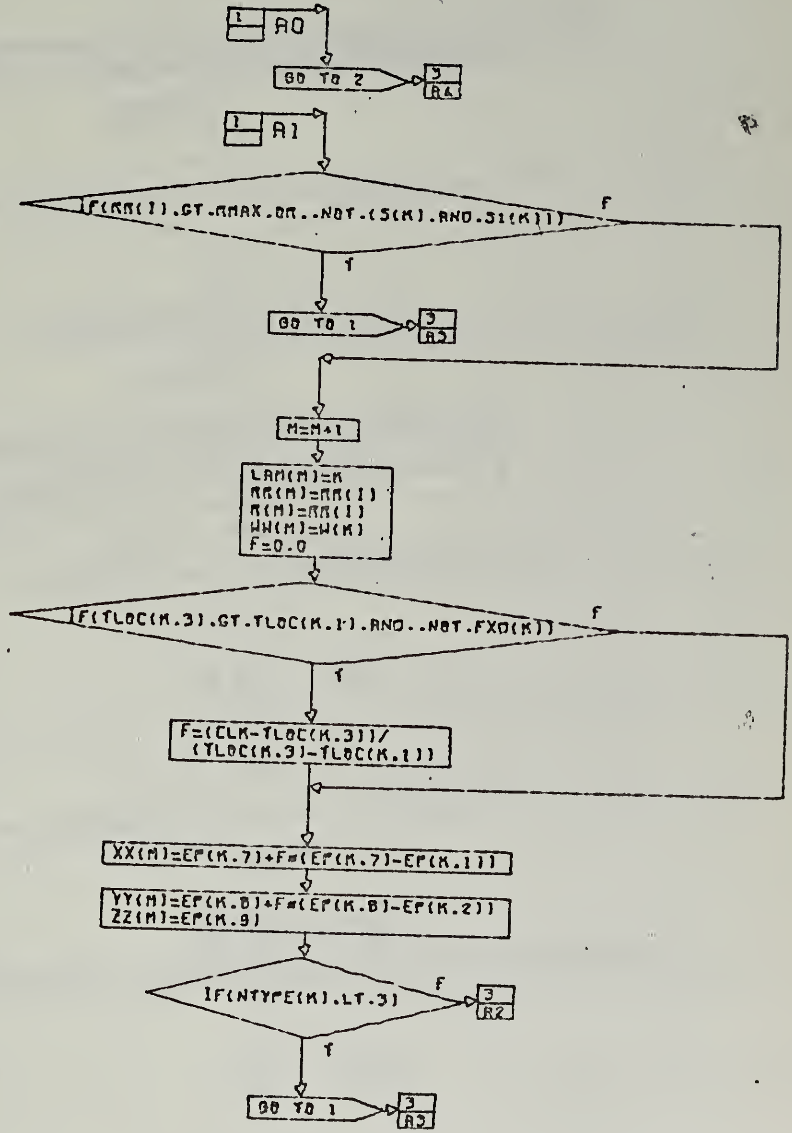
```
05 MAY 72 11:41:42 TO 11:41:52 IDENT:SOOKNE ACCOUNT:01563   CARDS IN:  649  CARDS OUT:     0 PAGES:    2? ELAPSED:  0 10  LOG 189
RUN ENTERED SYSTEM AT 11:41:37 PRIORITY: A PRINT CHANNEL: 14 PUNCH CHANNEL: 13 LINES PRINTED:   818  IO: 09  SCRATCH: 0  USER  0




**    **    **    **    **  NBS  EXEC II LEVEL 6.0A MOD 5.0  REEL 7368 ***   MAY 2  , 1972 **       **    **    **    **    **



************************* FORTRAN COMPILER *************** 49 **************

FIXES HAVE BEEN INSERTED IN THE FORTRAN COMPILER LEVEL 25.  TESTING CONTINUES
ON THIS REVISED COMPILER.  THE OPERATIONAL DATE OF LEVEL 25 WILL BE ANNOUNCED.

**************************** MAY 4, 1972 *****************************************
```

```
**    **    **    **    **   NBS  EXEC II LEVEL 6.0A MOD 5.0  REEL 7368 ***   MAY 2  , 1972 **        **    **    **    **    **

1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012
```

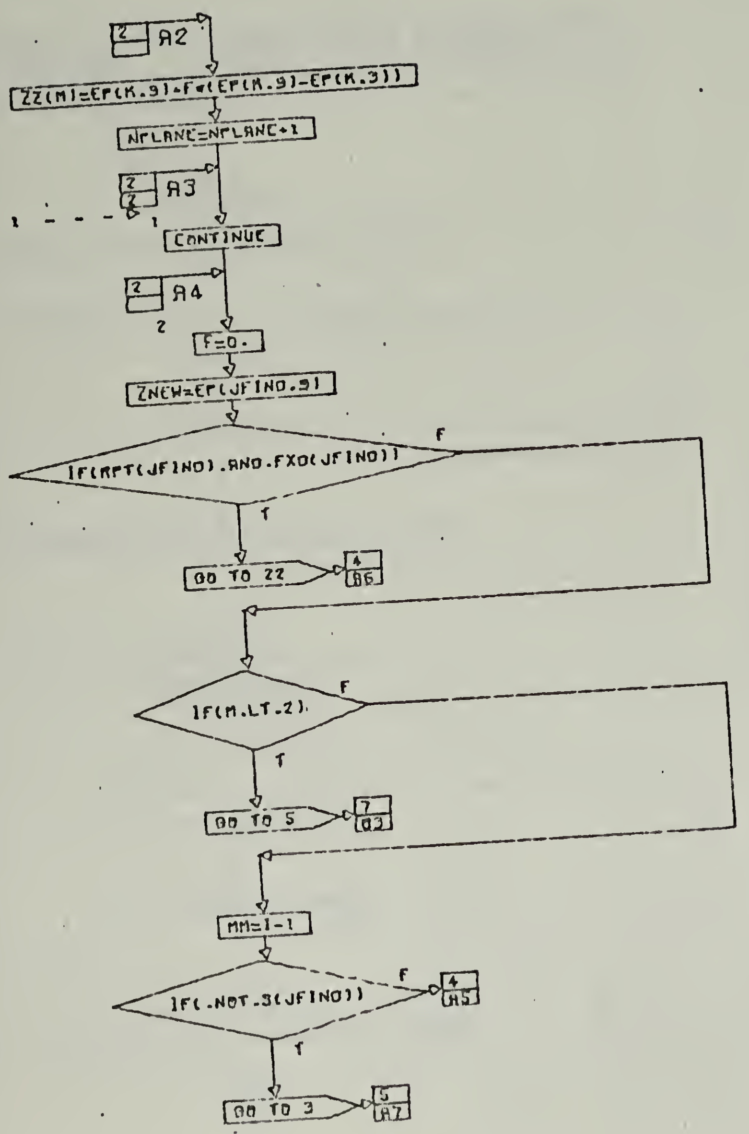
FLOW CHARTS FOR POSITION LOCATION PROGRAMS



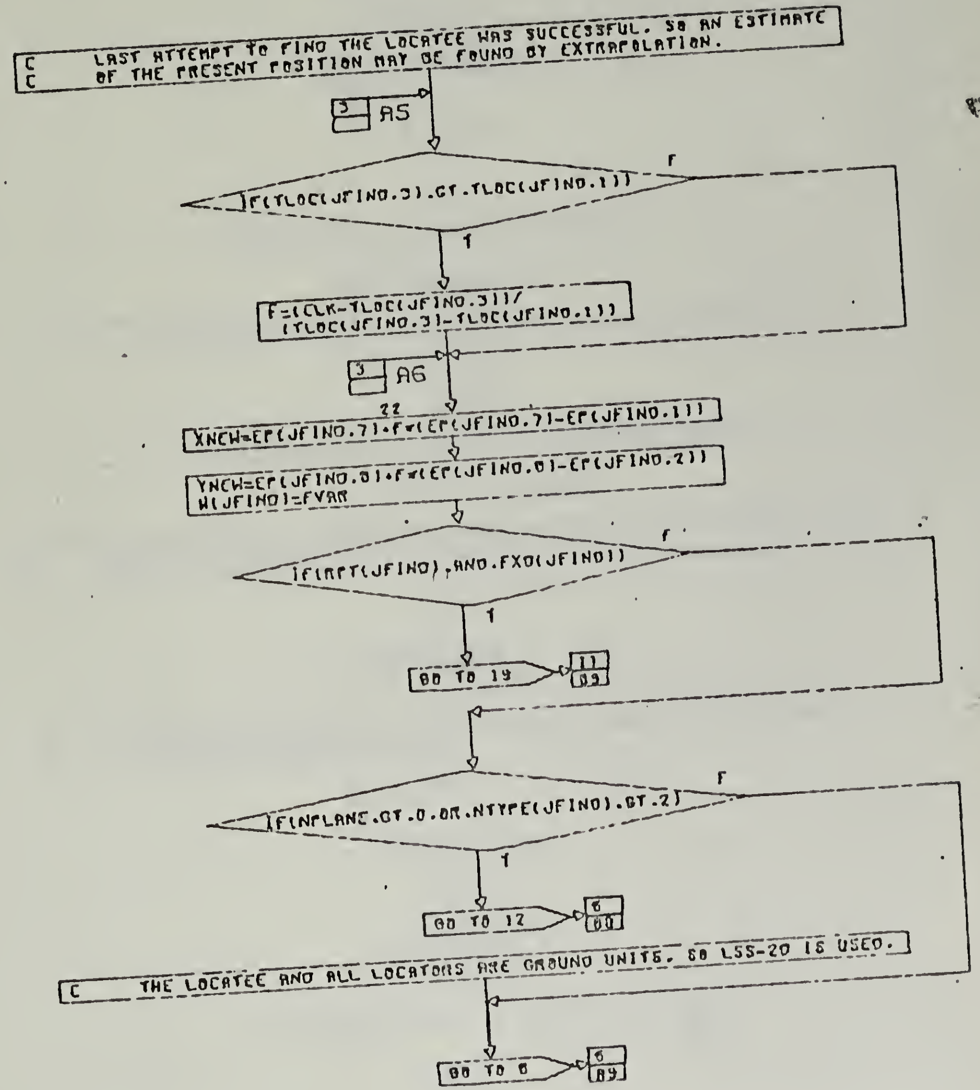
CONT. ON PG 2



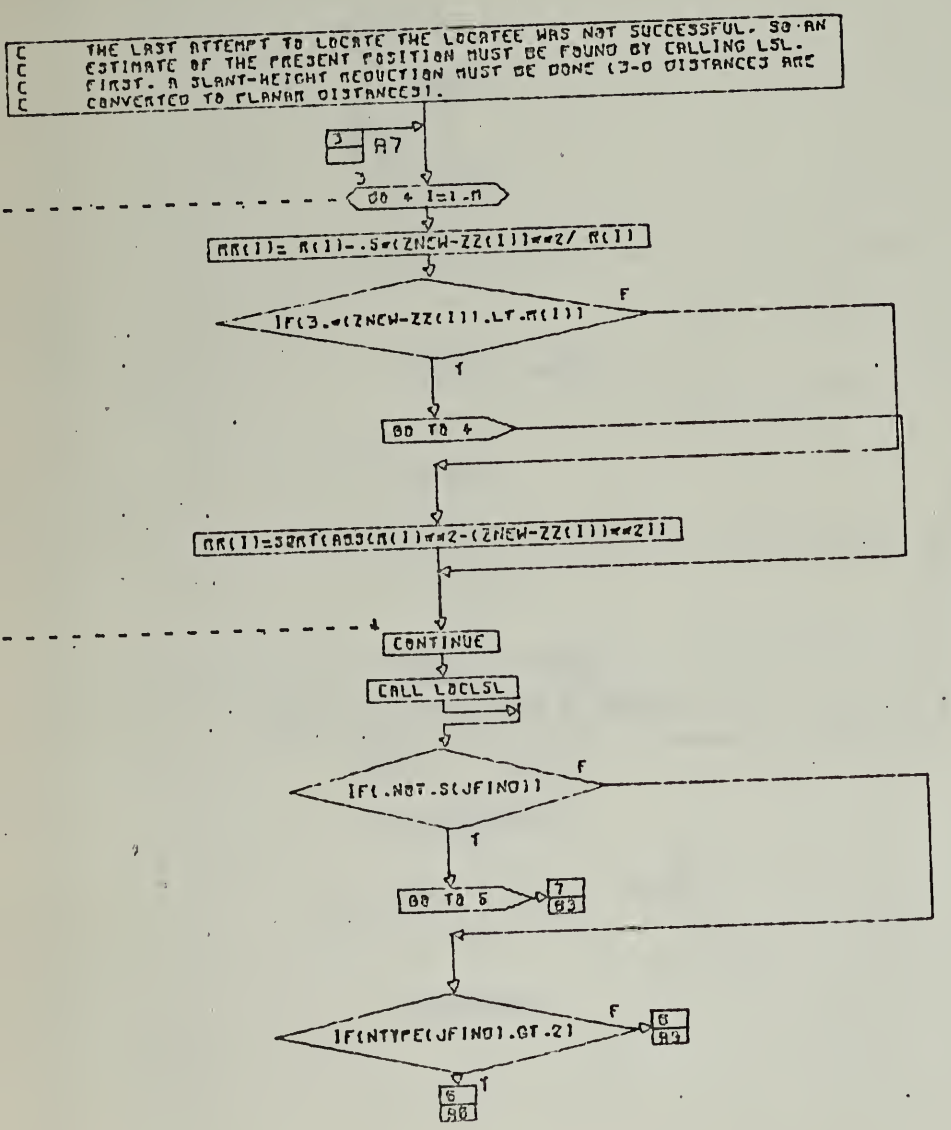
CONT. ON PG 3



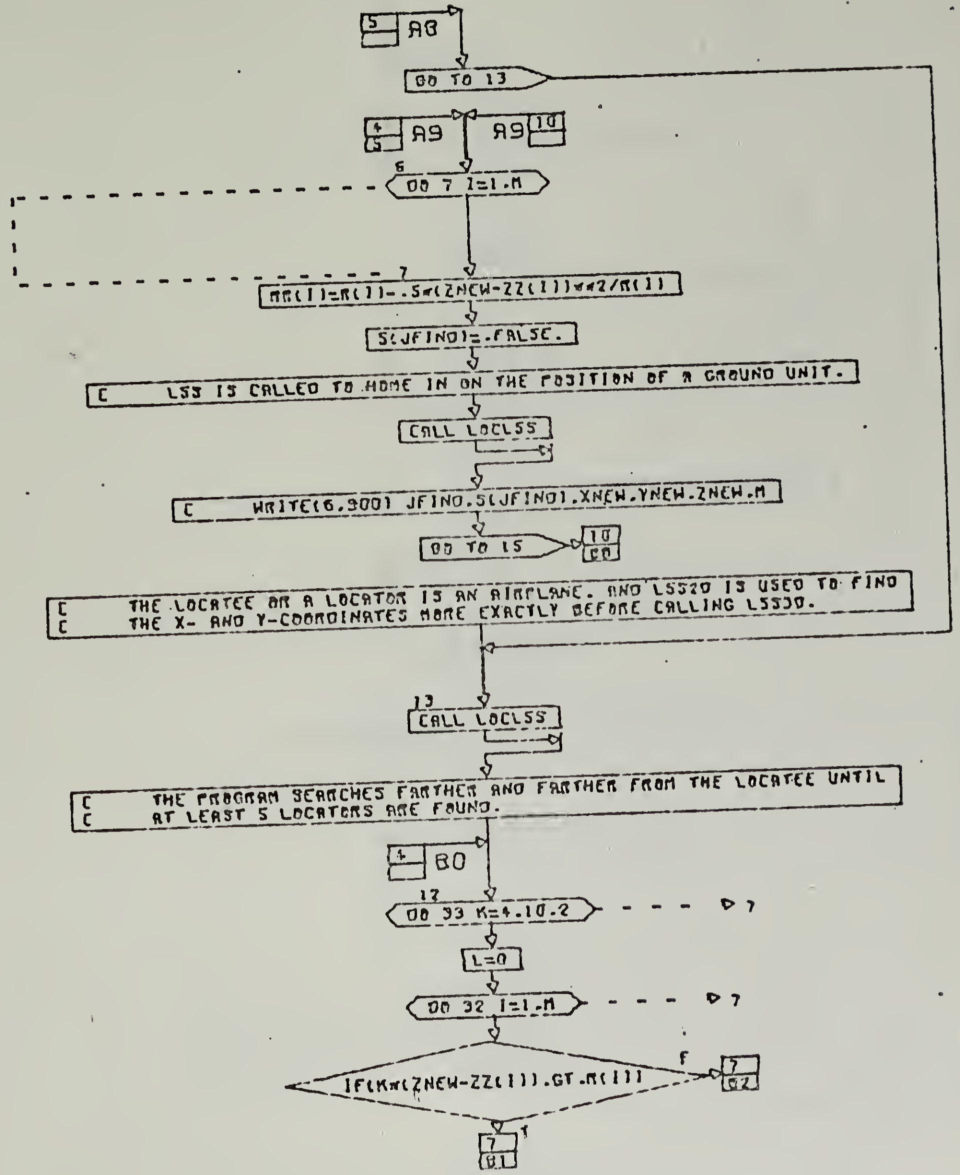
CONT. ON PG 4



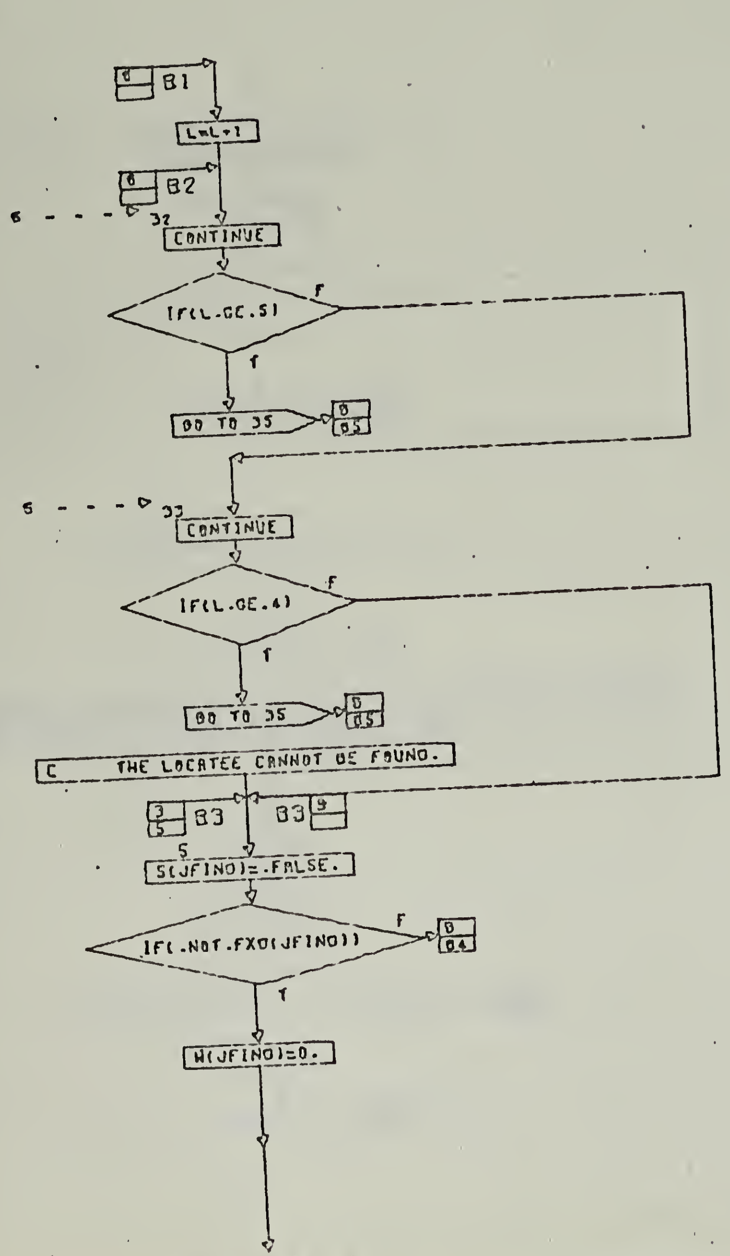
CONT. ON PG 5



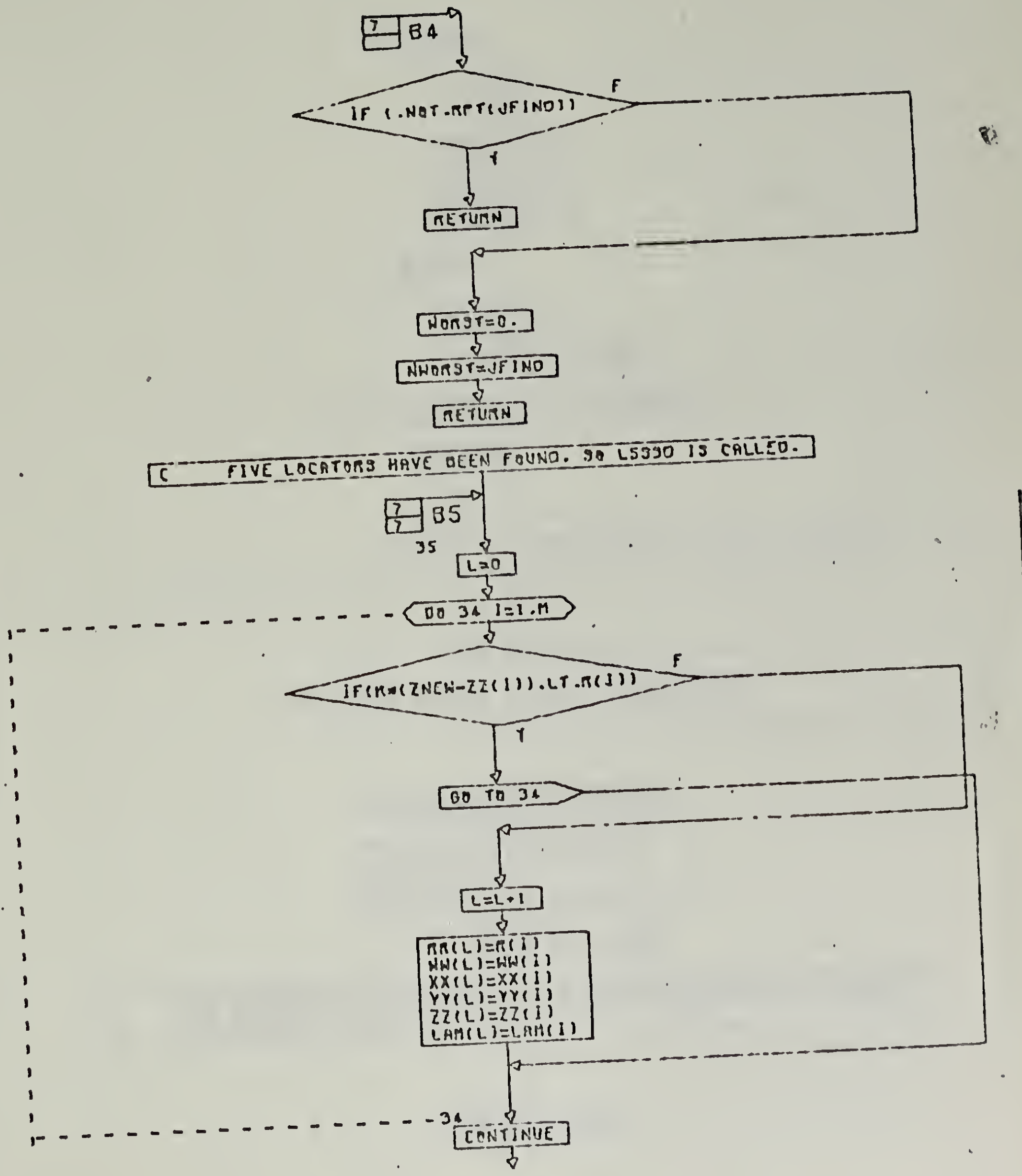
CONT. ON PG 6



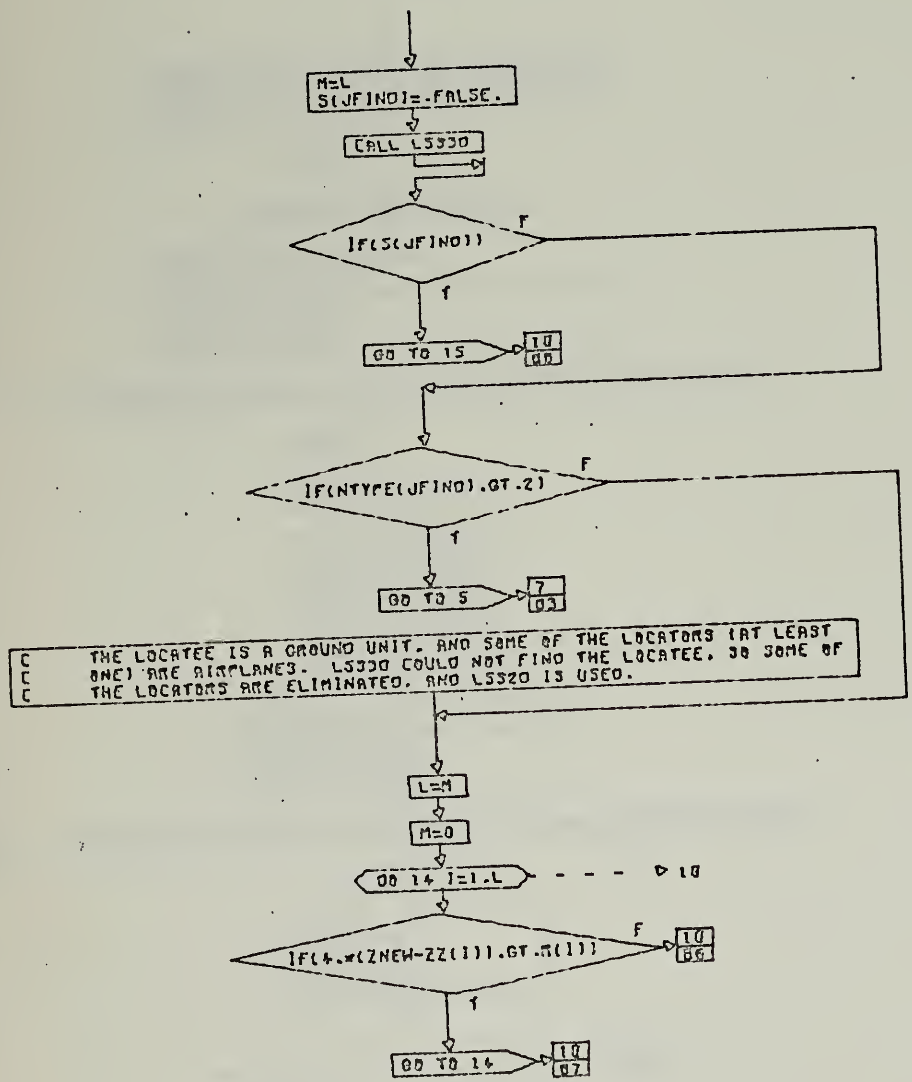
CONT. ON PG 7



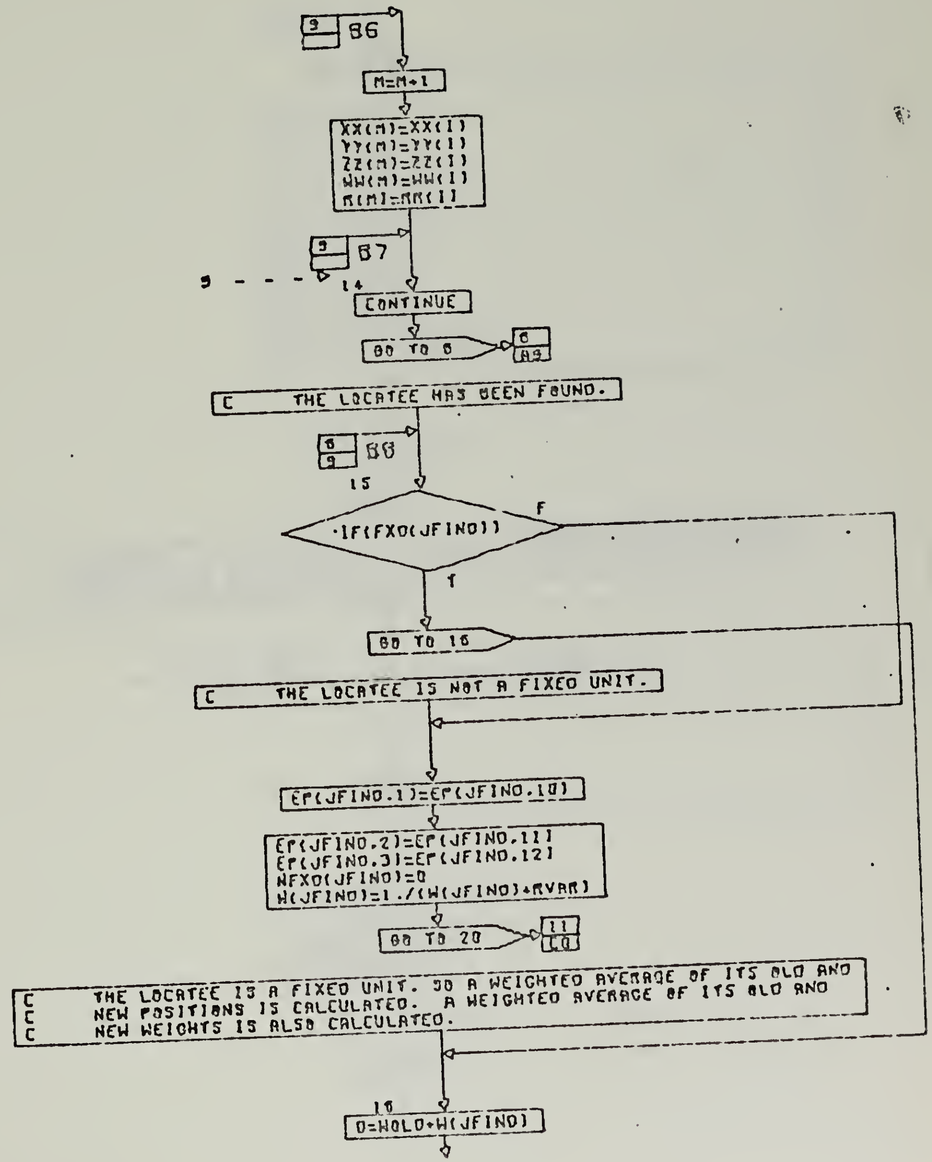
CONT. ON PG 8



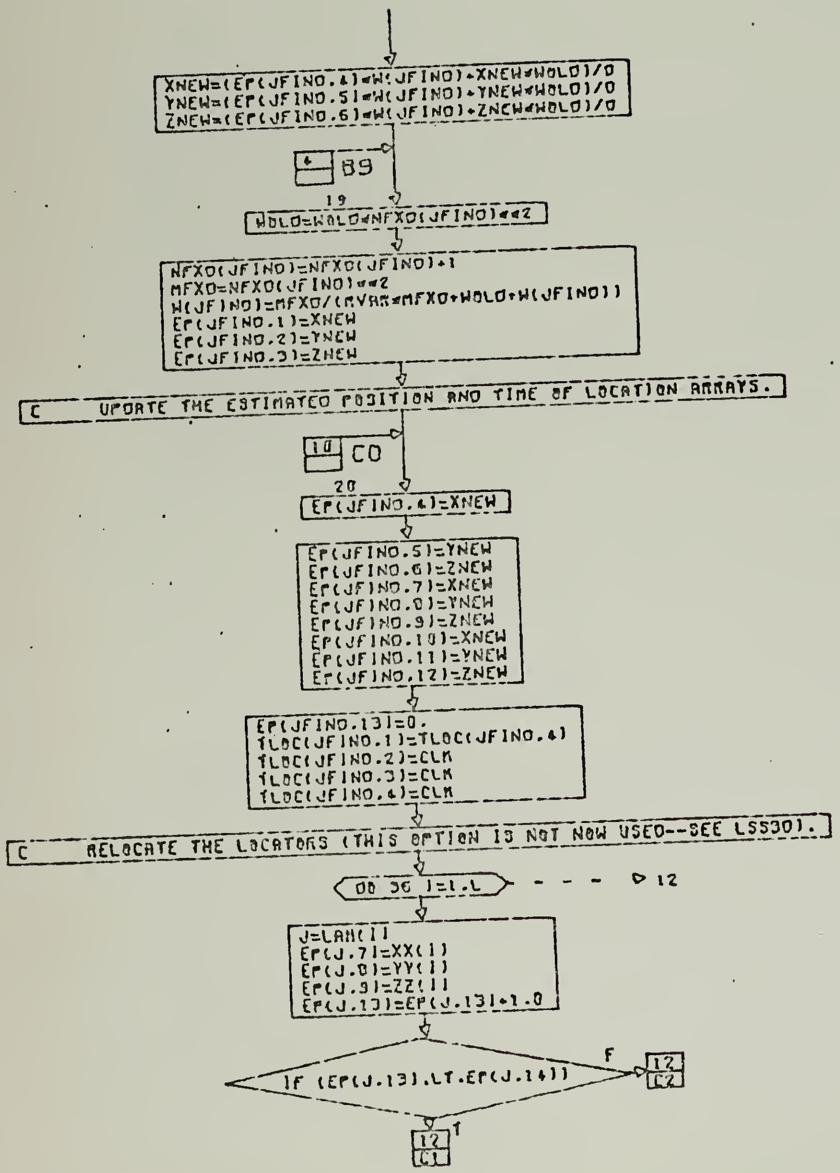
CONT. ON PG 9



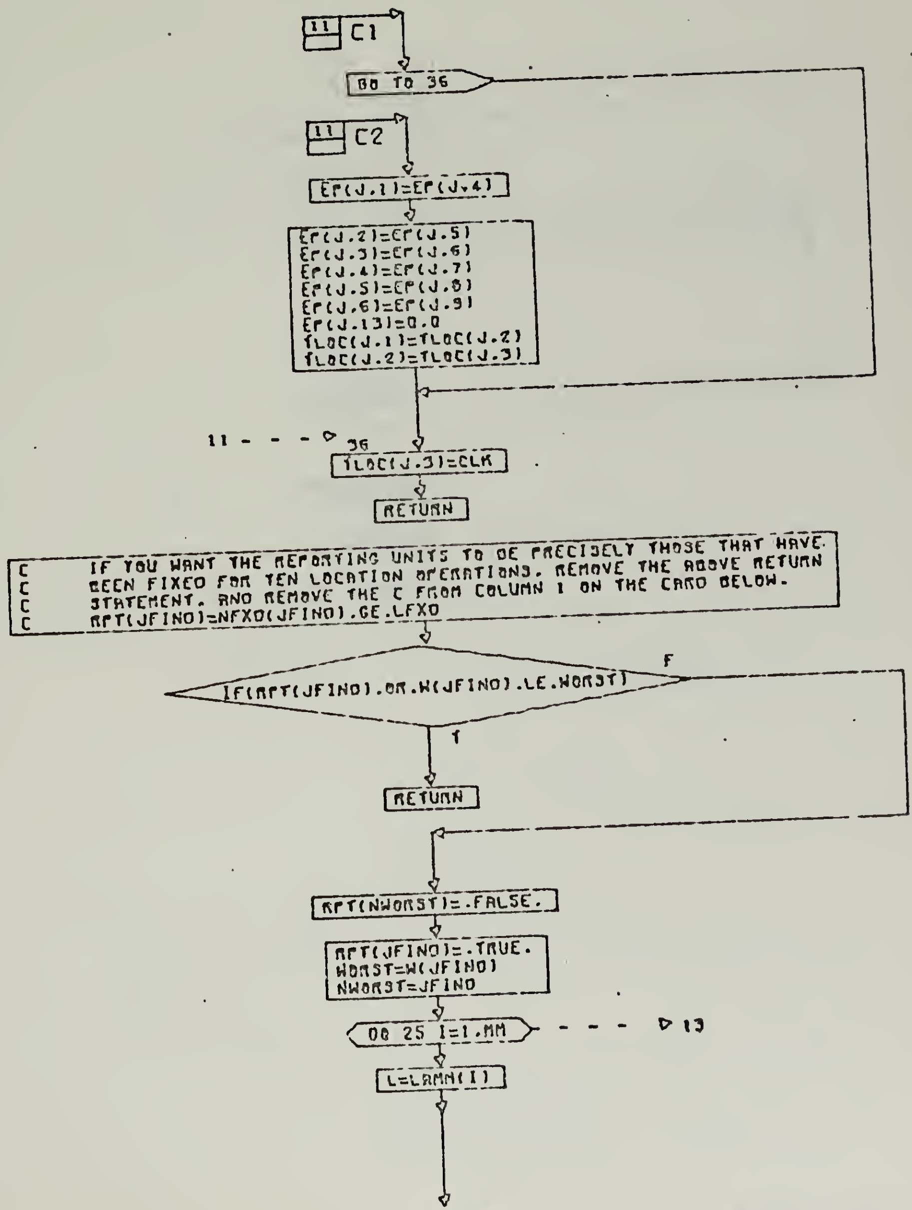
CONT. ON PG 10



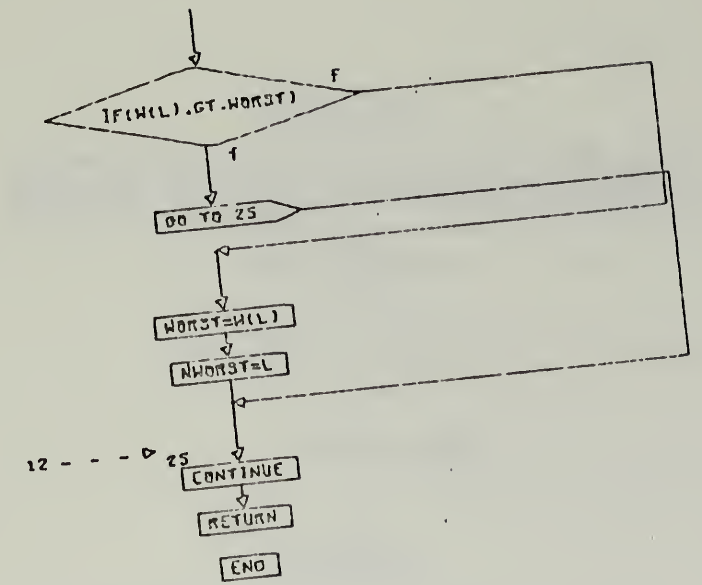
CONT. ON PG 11



CONT. ON PG 12



CONT. ON PG 13

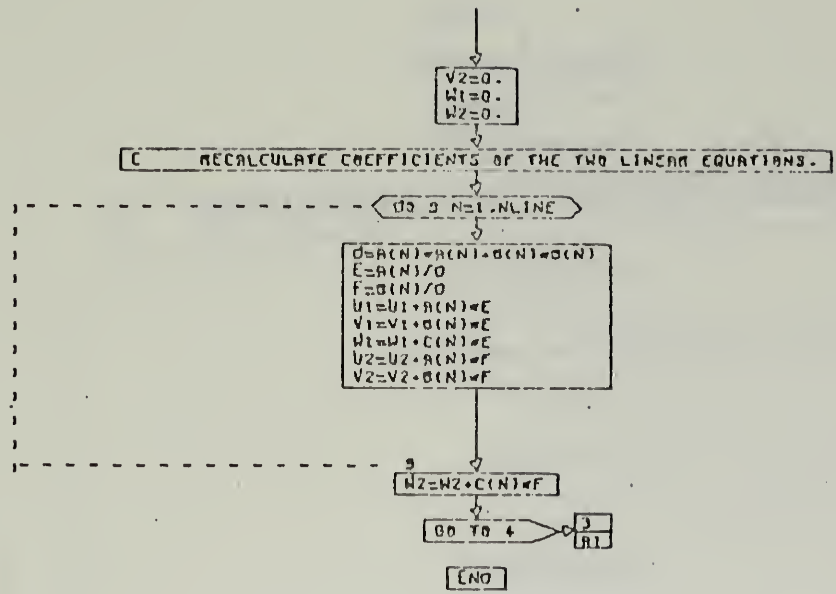

IF(H(L).GT.WORST)
f
f
DO TO 25
WORST=H(L)
NWORST=L
12
25
CONTINUE
RETURN
END

```
SUBROUTINE LOCLSL

C     LEAST SQUARES LINEAR (LSLI.
C     THIS ROUTINE MINIMIZES THE SUM OF THE SQUARED DISTANCES FROM THE
C     LOCATEE TO THE RADICAL AXIS LINES FORMED BY PAIRS OF CIRCLES.
C     EACH CIRCLE HAS A LOCATOR FOR ITS CENTER. AND THE REPORTED
C     DISTANCE TO THE LOCATEE FOR ITS RADIUS.  SEE FIRST INTERIM REPORT
C     ON PROJECT WHERSM. PP. 10-13.
C     THIS IS THE TWO-DIMENSIONAL ROUTINE.

PARAMETER LI=225.L4=LI+1
DIMENSION                 A(400).B(400).C(400).VRSQ(400)

COMMON/CGI/EP(LI.14).LAM(L4).RR(L4).S(LI).TLOC(LI.4).W(LI).
      JFIND.CLK.XNEW.YNEW.ZNEW.FXD(LI).RPT(LI).NTYPE(LI)

COMMON/MXYZ/M.L.XX(80).YY(80).ZZ(80).WW(80).R(80).P
LOGICAL S.P
DATA TOL/50./
P=.FALSE.

IF (P)  --F-->
   T
WRITE(6.99) JFIND.CLK

99
FORMAT(' * * LSL20: UNIT ='I5.' CLOCK ='F10.4)

S(JFIND)=.FALSE.
NCLIM=4
MELIM=0
NN=M-1

2
NLINE=0

U1=0.
U2=0.
V1=0.
V2=0.
W1=0.
W2=0.

DO 3 NOFF=1.NN  - - -  > 3
```

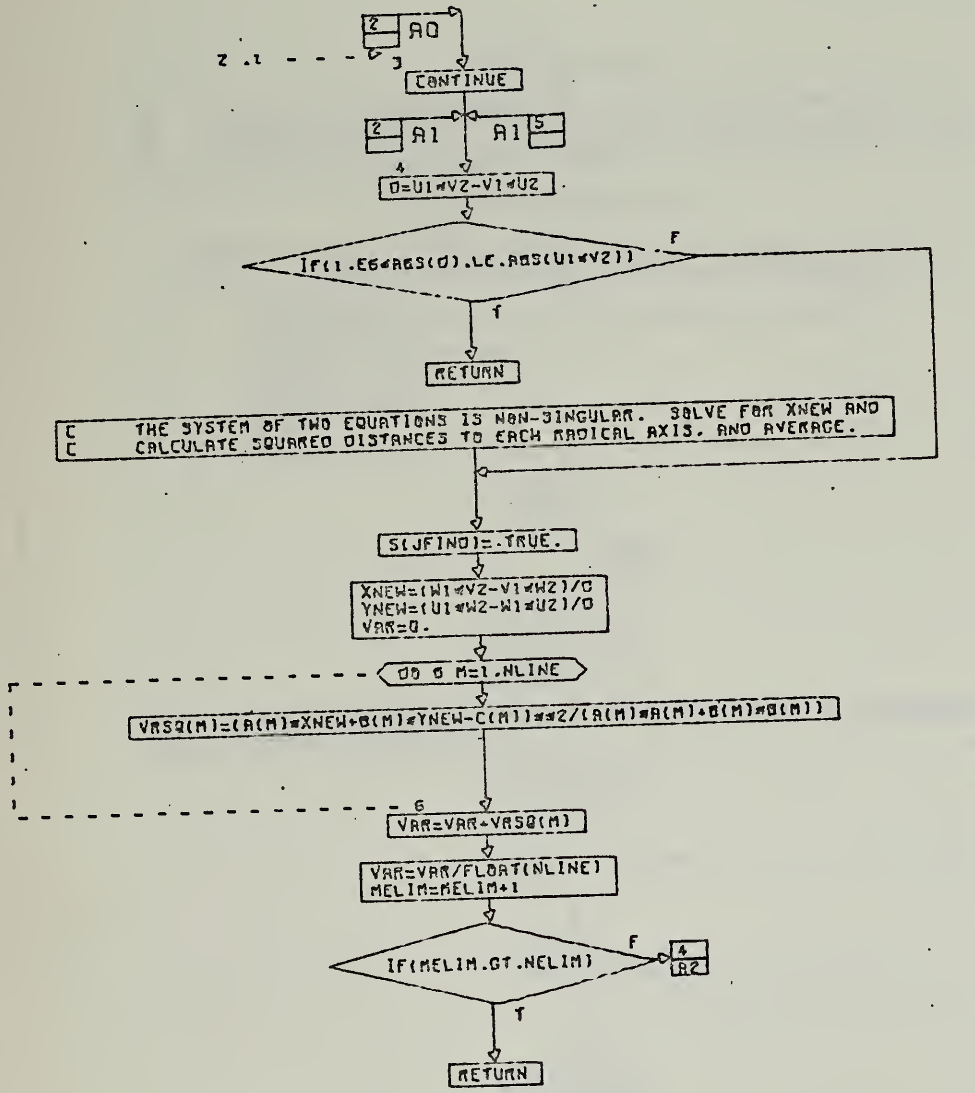
CONT. ON PG 2


```
NBEGIN=NOFF+1

C     FOR EACH PAIR OF CIRCLES. CALCULATE THE COEFFICIENTS OF THE
C     RADICAL AXIS LINE. A*X+B*Y-C=0. WHERE A*A+B*B=1.  CALCULATE
C     COEFFICIENTS OF THE TWO EQUATIONS IN X AND Y. ON BOTTOM OF P. 13
C     OF FIRST INTERIM REPORT.

DO 3 L=NBEGIN.M  - - -  > 3

IF(NLINE.EQ.400)  --F-->
   T
GO TO 4  > 3/A1

K=L-NOFF

IF ((XX(K)-XX(L))**2+(YY(K)-YY(L))**2.LT.1.)  --F-->
   T
GO TO 3  > 3/A2

NLINE=NLINE+1

A(NLINE)=2.*(XX(K)-XX(L))
B(NLINE)=2.*(YY(K)-YY(L))
C(NLINE)=RR(L)**2-RR(K)**2+XX(K)**2-XX(L)**2+YY(K)**2-YY(L)**2
D=A(NLINE)*A(NLINE)+B(NLINE)*B(NLINE)
E=A(NLINE)/D
F=B(NLINE)/D
U1=U1+A(NLINE)*E
V1=V1+B(NLINE)*E

W1=W1+C(NLINE)*E
U2=U2+A(NLINE)*F
V2=V2+B(NLINE)*F
W2=W2+C(NLINE)*F
```

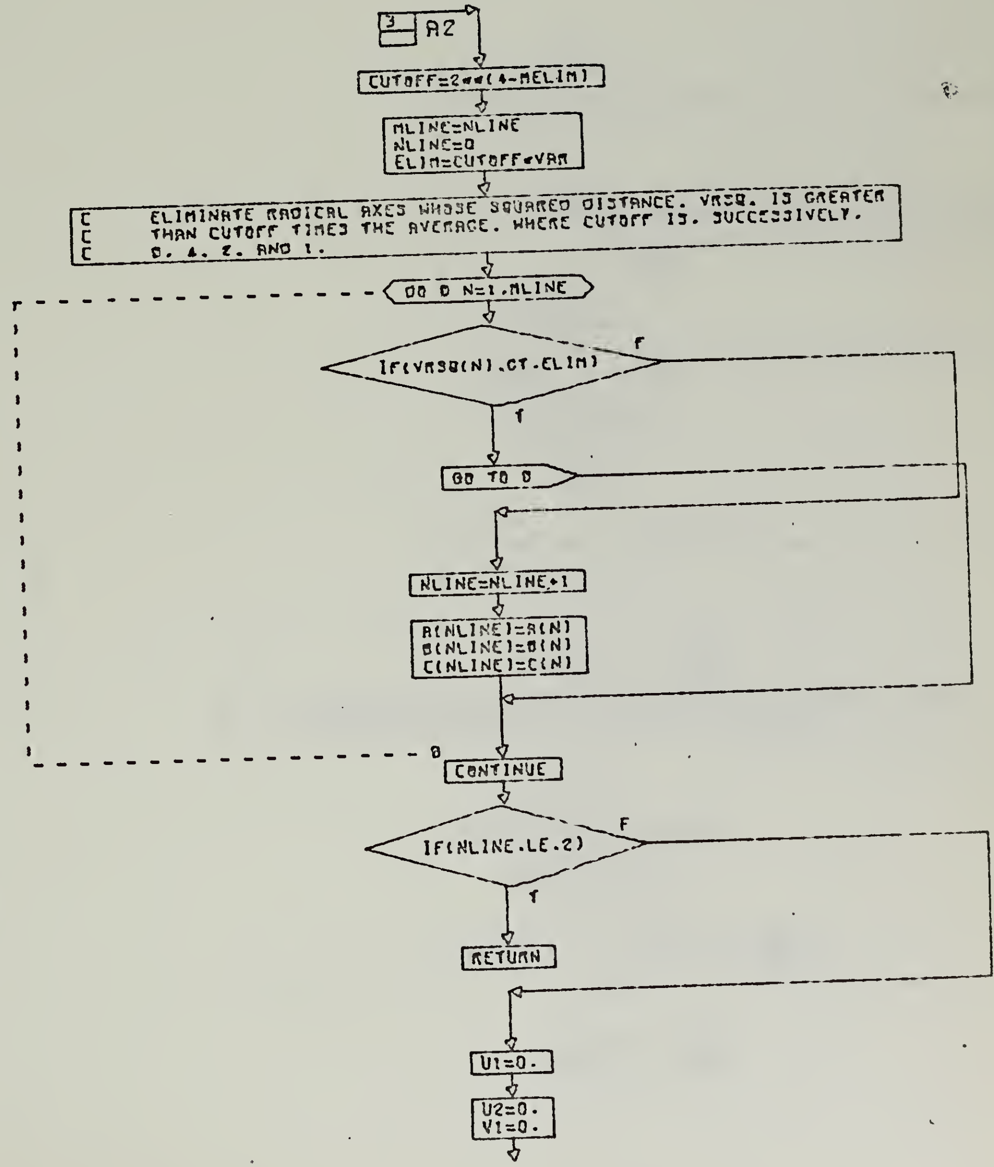
CONT. ON PG 3



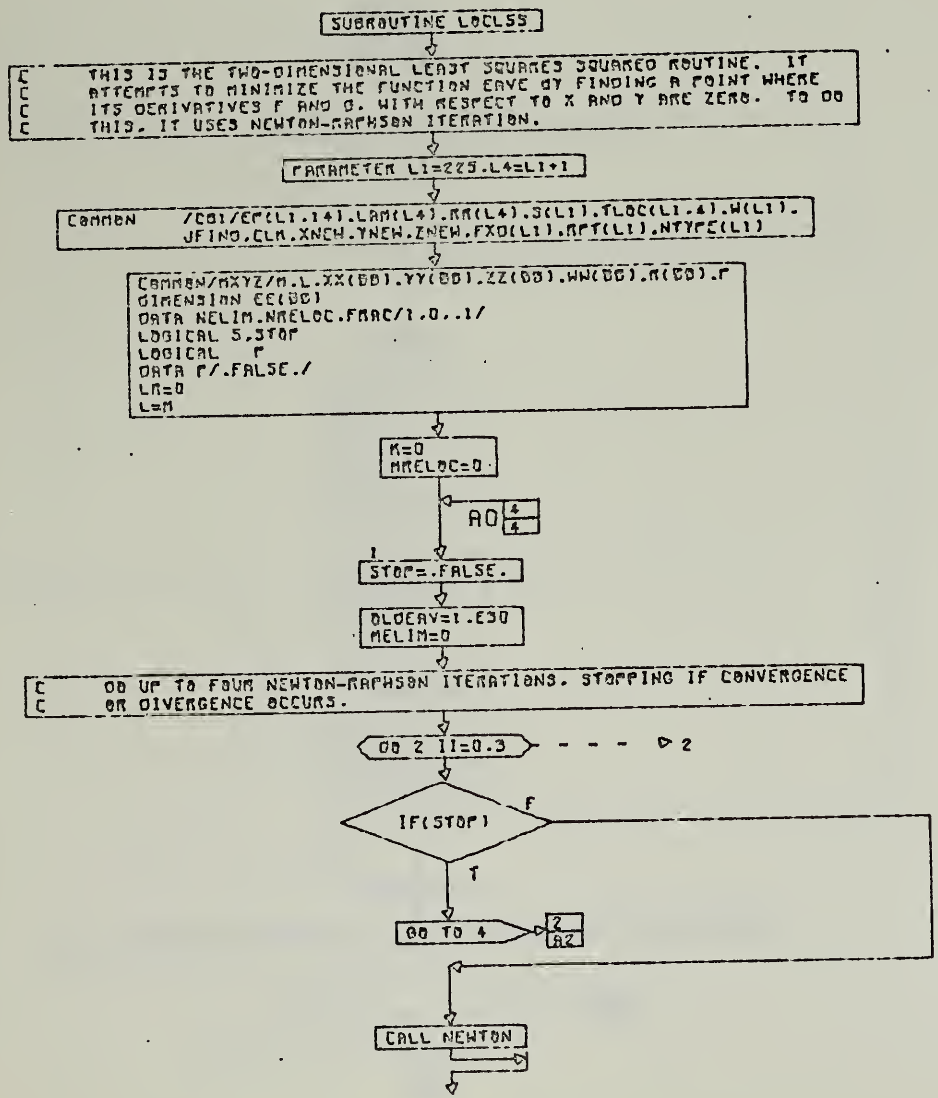
V2=0.
W1=0.
W2=0.
C RECALCULATE COEFFICIENTS OF THE TWO LINEAR EQUATIONS.
DO 9 N=1,NLINE
D=A(N)*A(N)+B(N)*B(N)
E=A(N)/D
F=B(N)/D
U1=U1+A(N)*E
V1=V1+B(N)*E
W1=W1+C(N)*E
U2=U2+A(N)*F
V2=V2+B(N)*F
9
W2=W2+C(N)*F
GO TO 4
3
A1
END



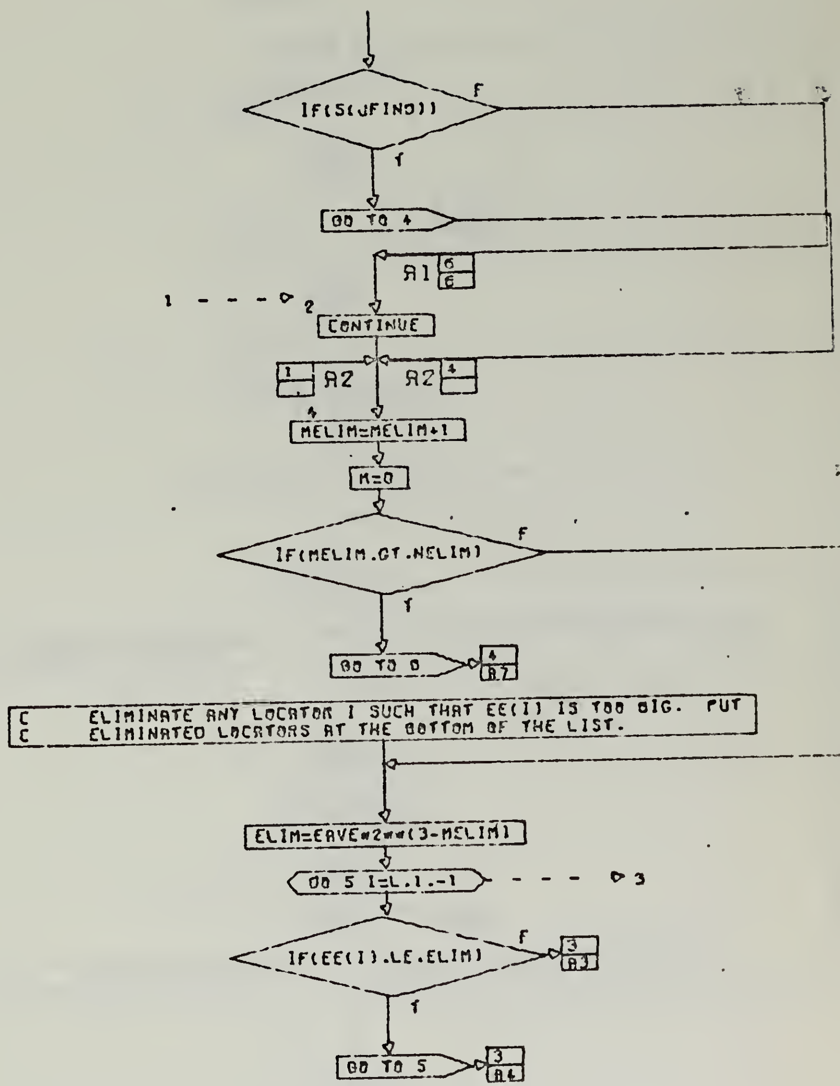
CONT. ON PG 4



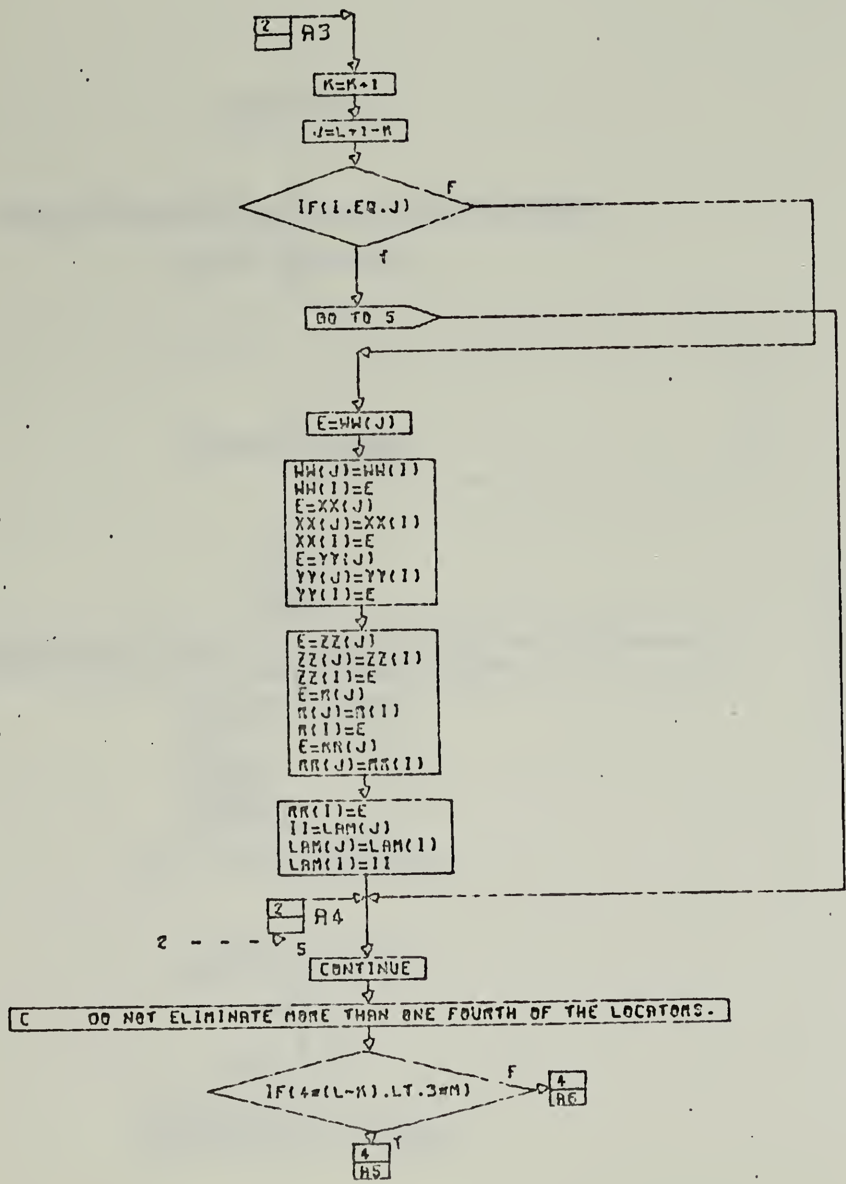
CONT. ON PG 5



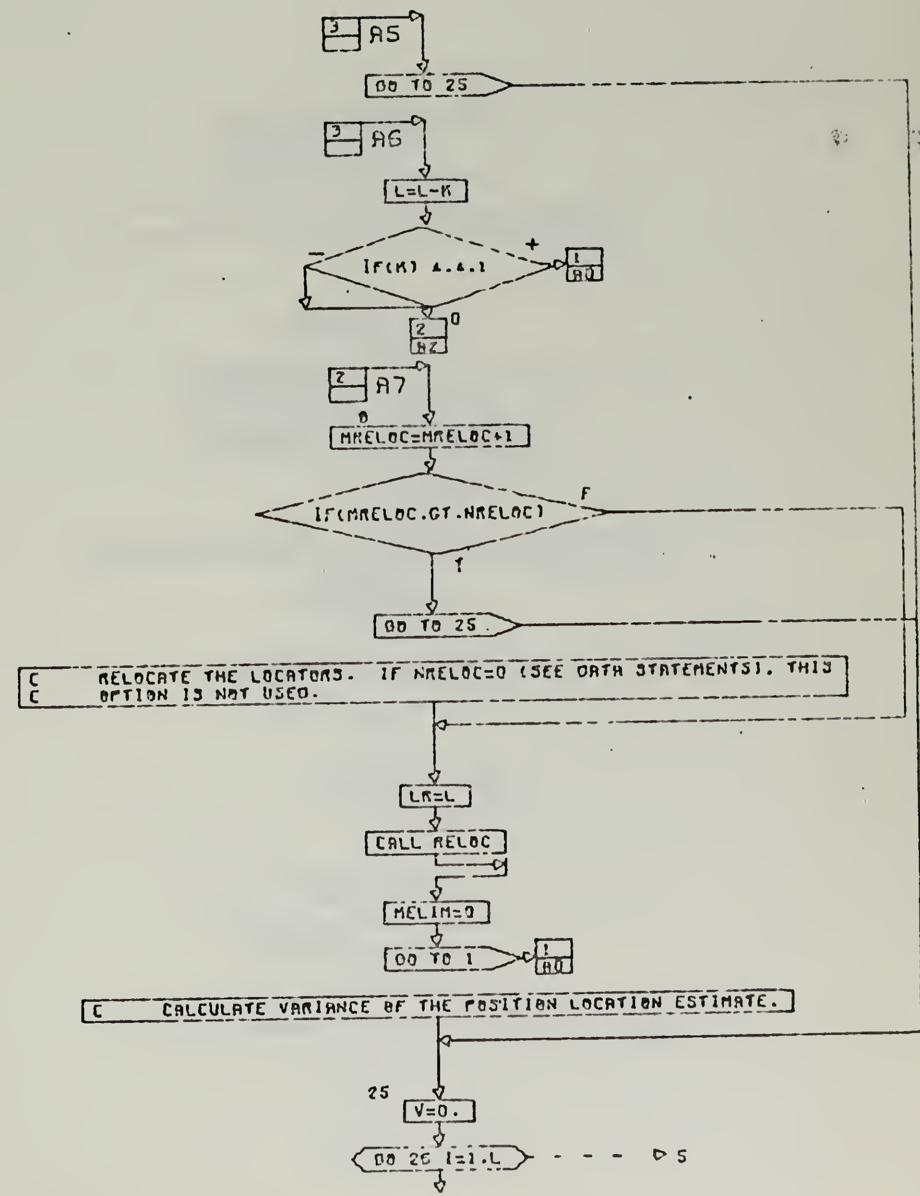
CONT. ON PG 2



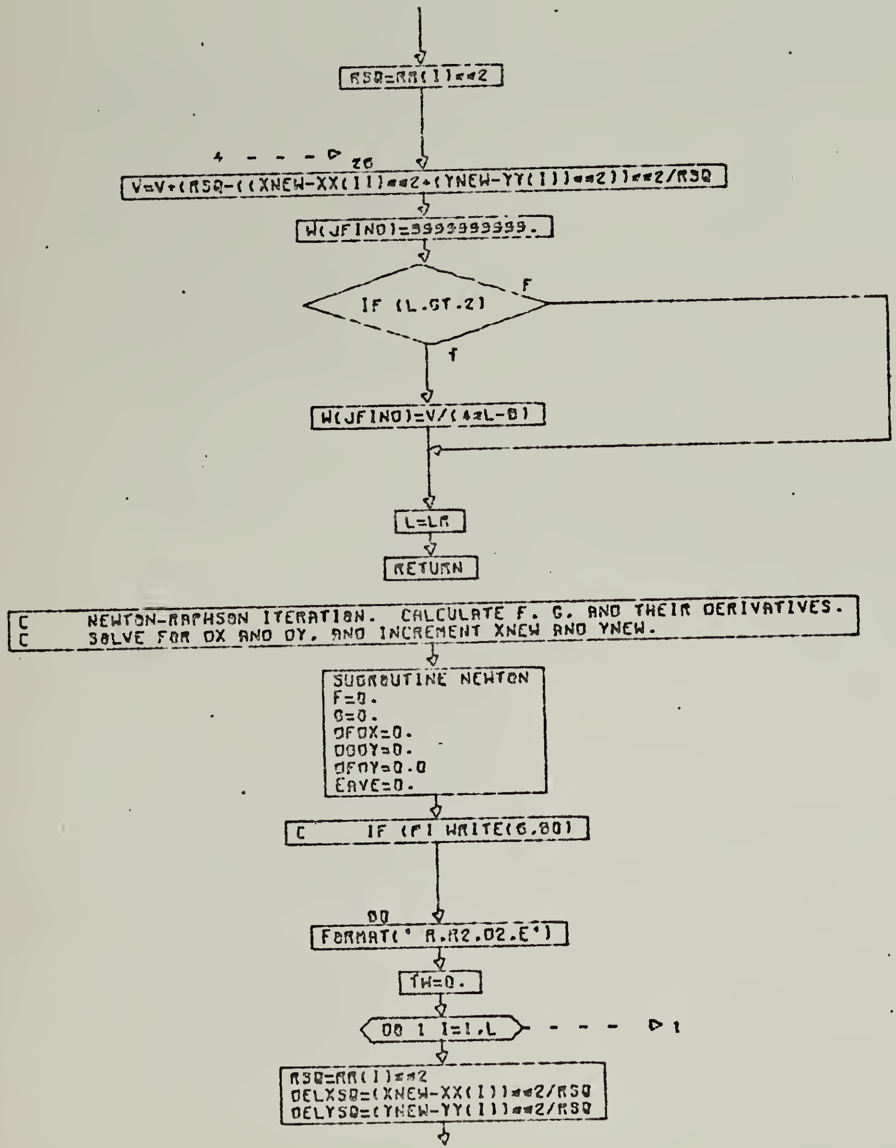
CONT. ON PG 3



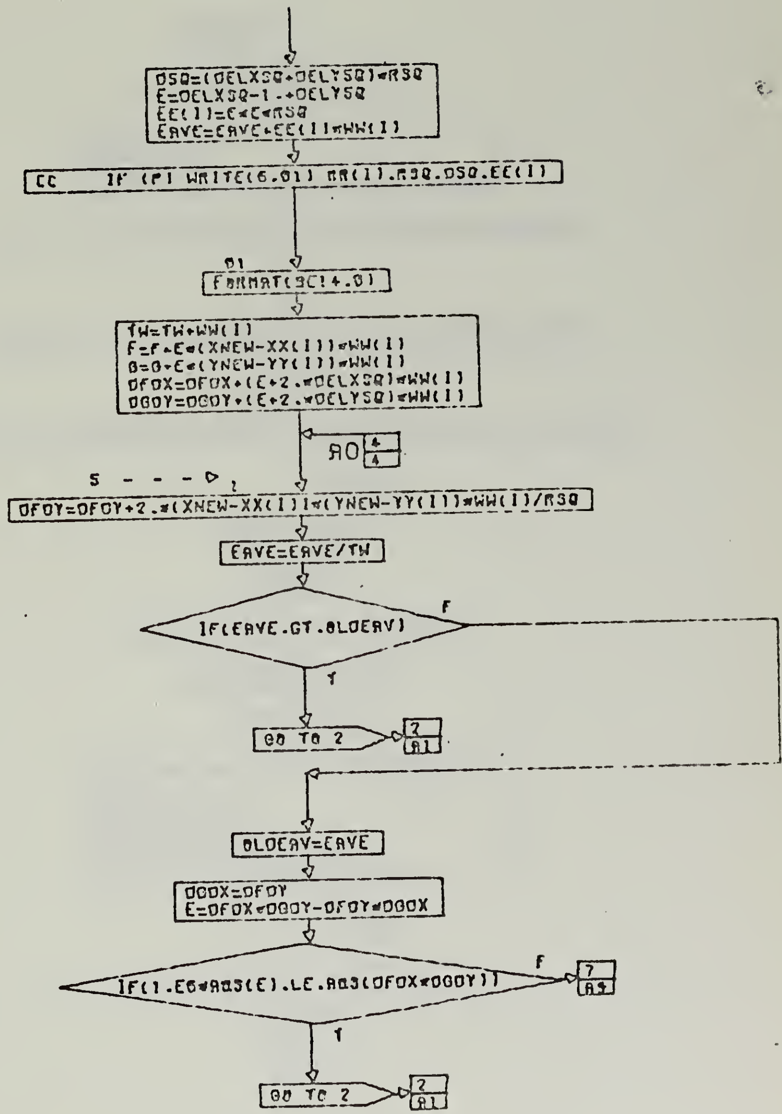
CONT. ON PG 4



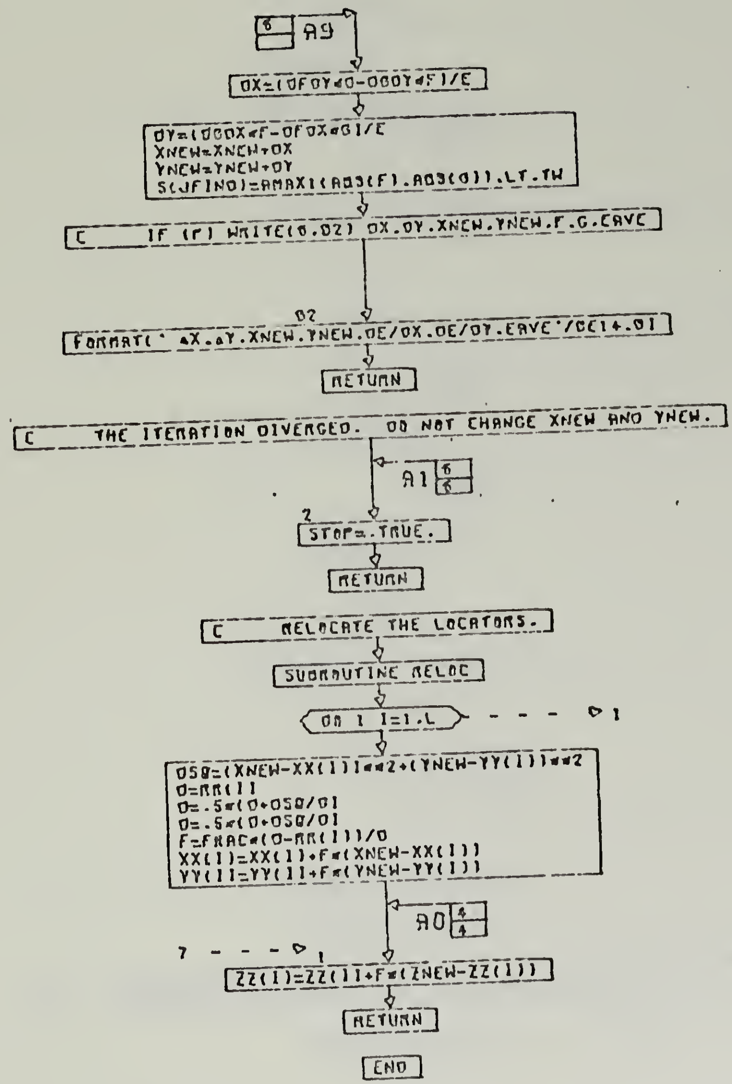
CONT. ON PG 5



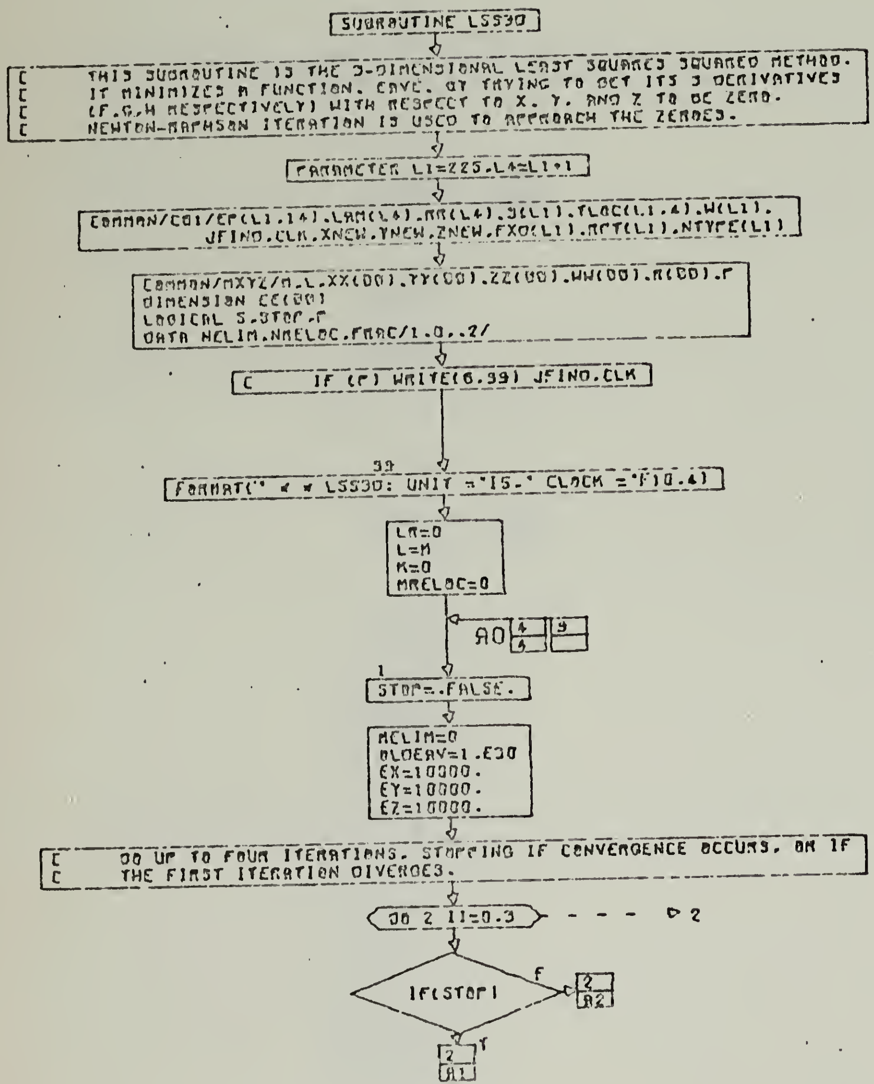
CONT. ON PG 6



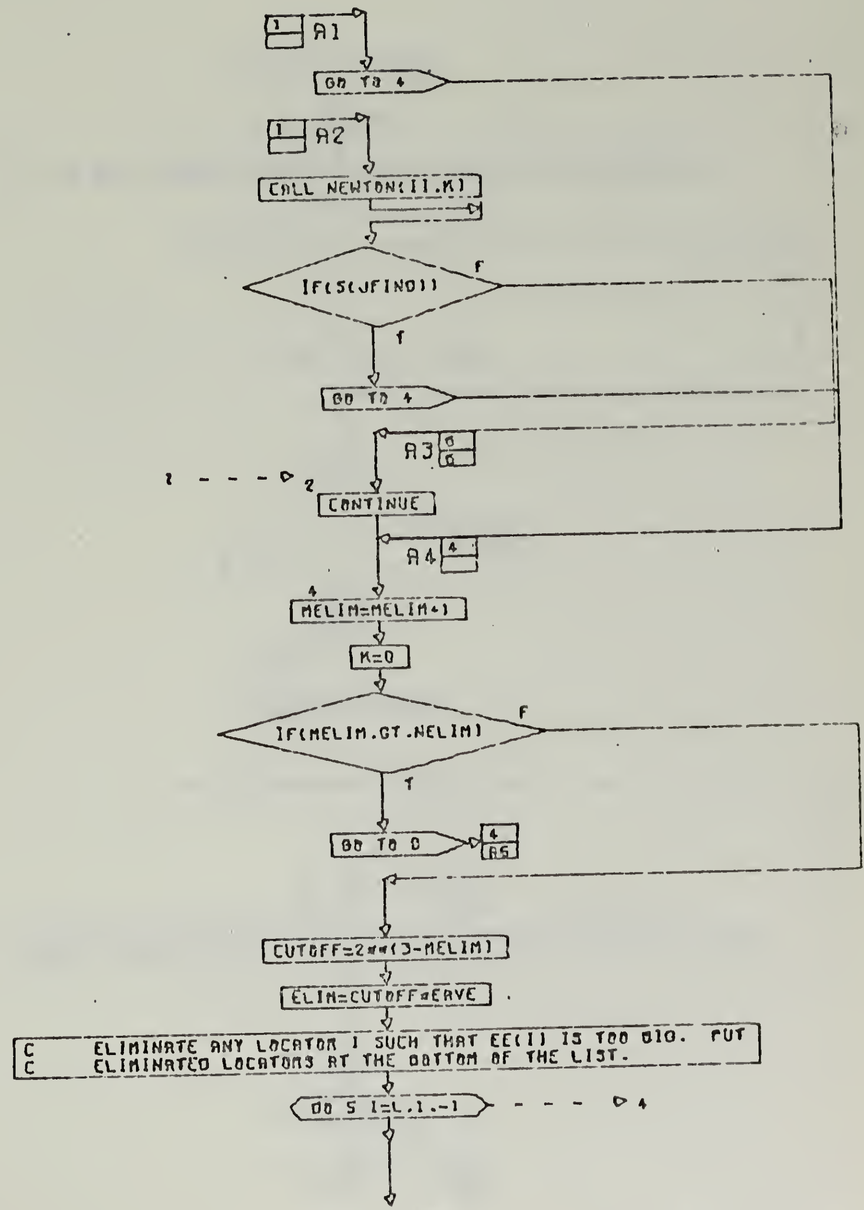
CONT. ON PG 7


A9
DX=(DFDY*D-DGDY*F)/E
DY=(DGDX*F-DFDX*G)/E
XNEW=XNEW+DX
YNEW=YNEW+DY
S(JFIND)=AMAX1(ABS(F),ABS(G)).LT.TH
C IF (F) WRITE(6,D2) DX,DY,XNEW,YNEW,F,G,CAVE
D2
FORMAT(' ∆X,∆Y,XNEW,YNEW,DE/DX,DE/DY,CAVE'/GE14.8)
RETURN
C THE ITERATION DIVERGED. DO NOT CHANGE XNEW AND YNEW.
A1
2
STOP=.TRUE.
RETURN
C RELOCATE THE LOCATORS.
SUBROUTINE RELOC
DO 1 I=1,L
1
DSQ=(XNEW-XX(I))**2+(YNEW-YY(I))**2
D=RR(I)
D=.5*(D+DSQ/D)
D=.5*(D+DSQ/D)
F=FRAC*(D-RR(I))/D
XX(I)=XX(I)+F*(XNEW-XX(I))
YY(I)=YY(I)+F*(YNEW-YY(I))
A0
7
1
ZZ(I)=ZZ(I)+F*(ZNEW-ZZ(I))
RETURN
END

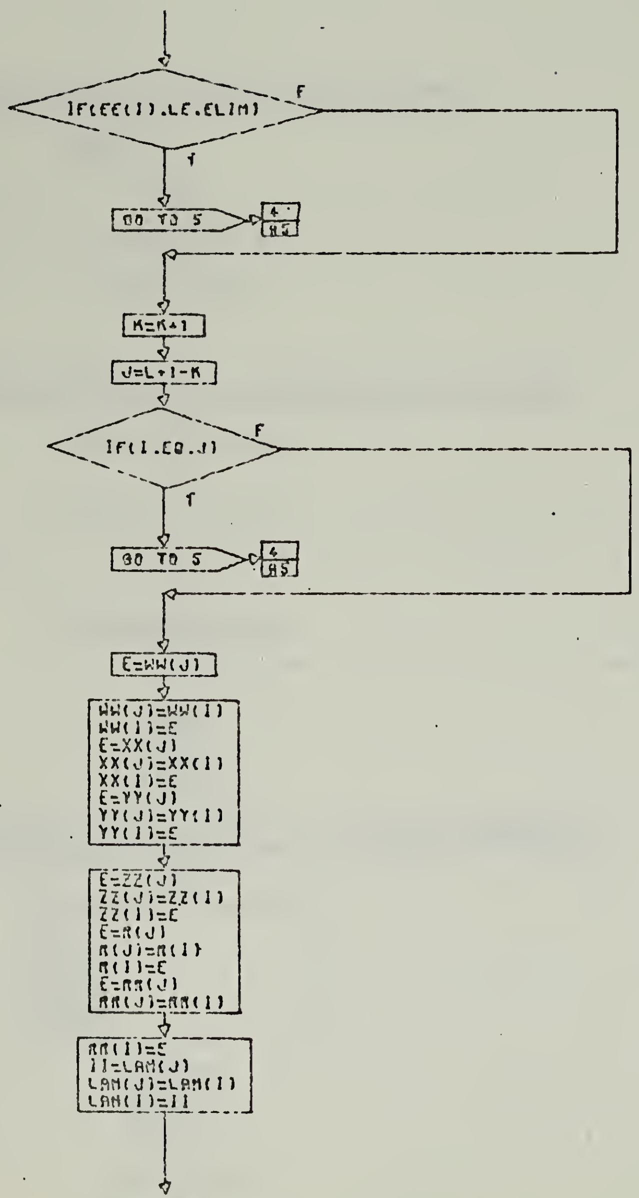
ED 7 FINAL



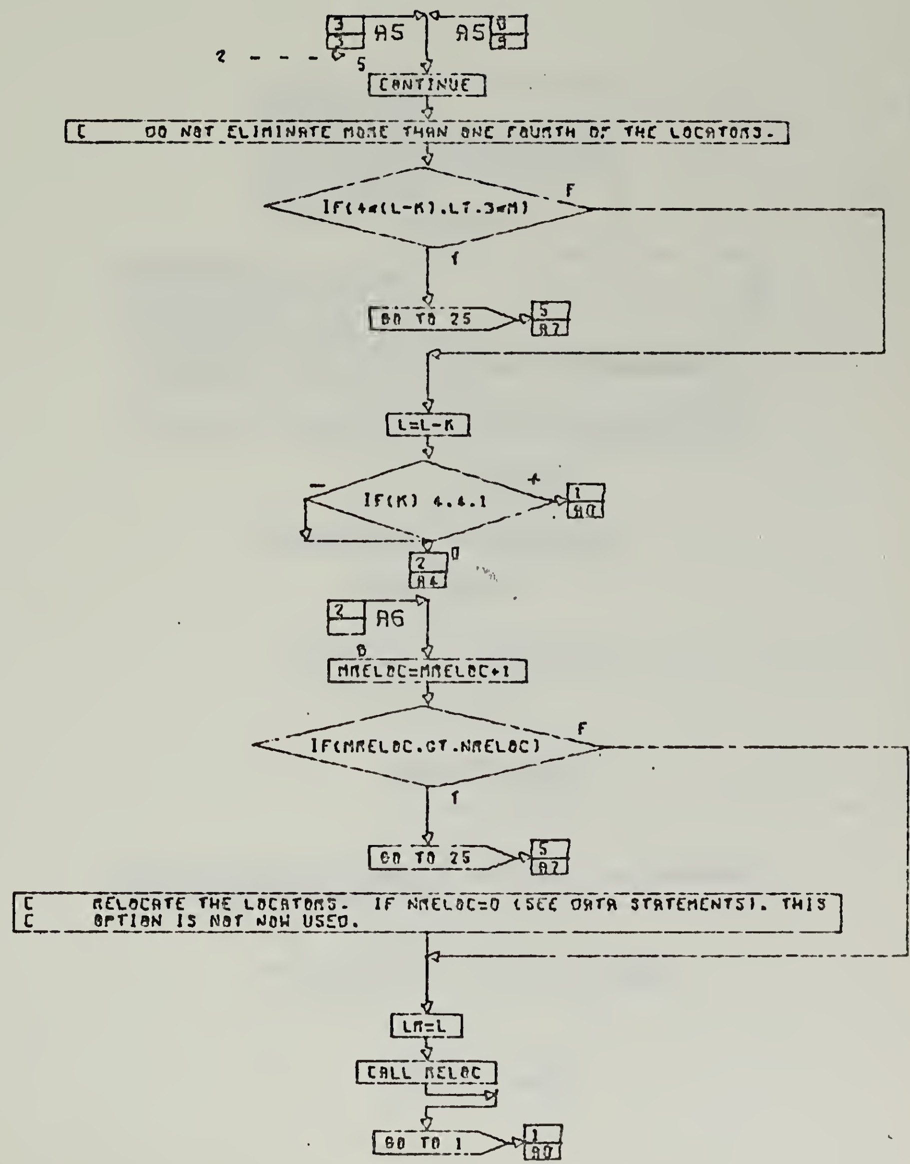
CONT. ON PG 2



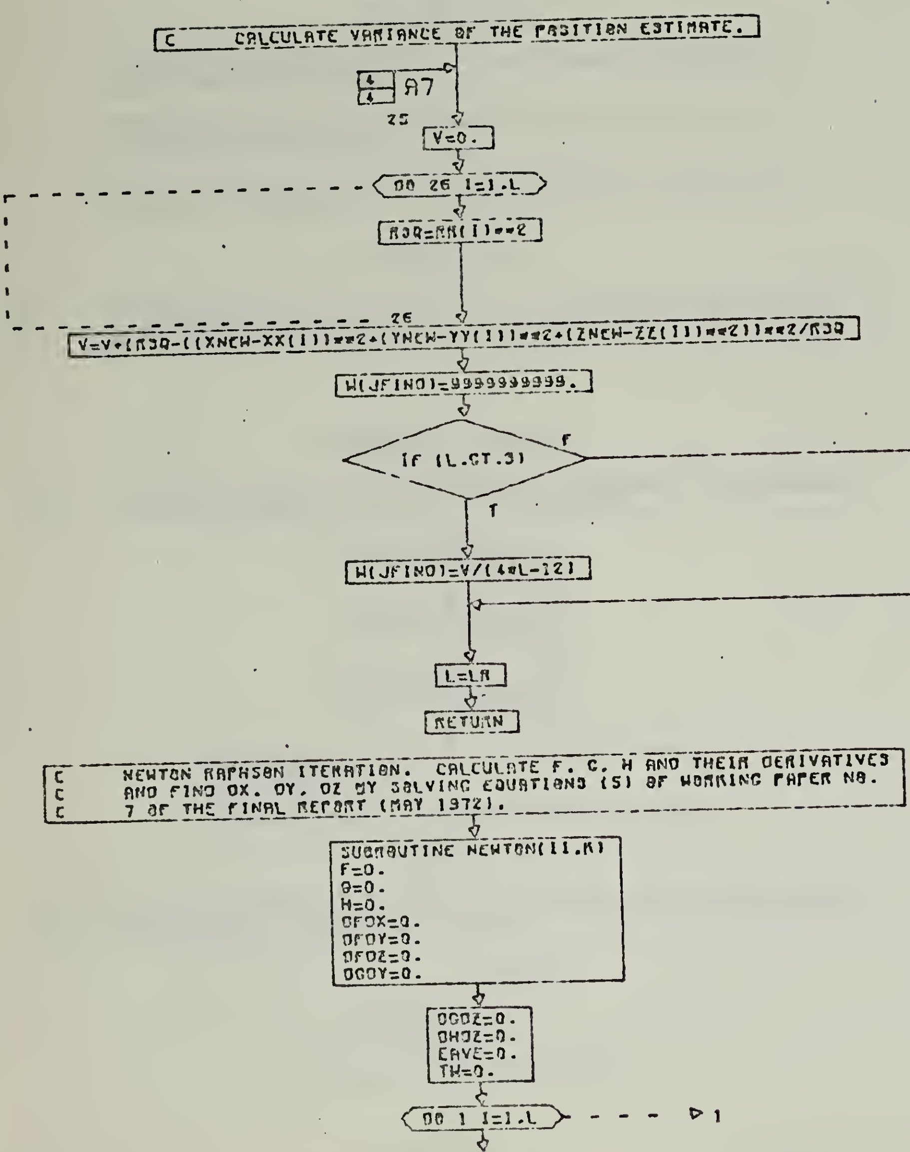
CONT. ON PG 3



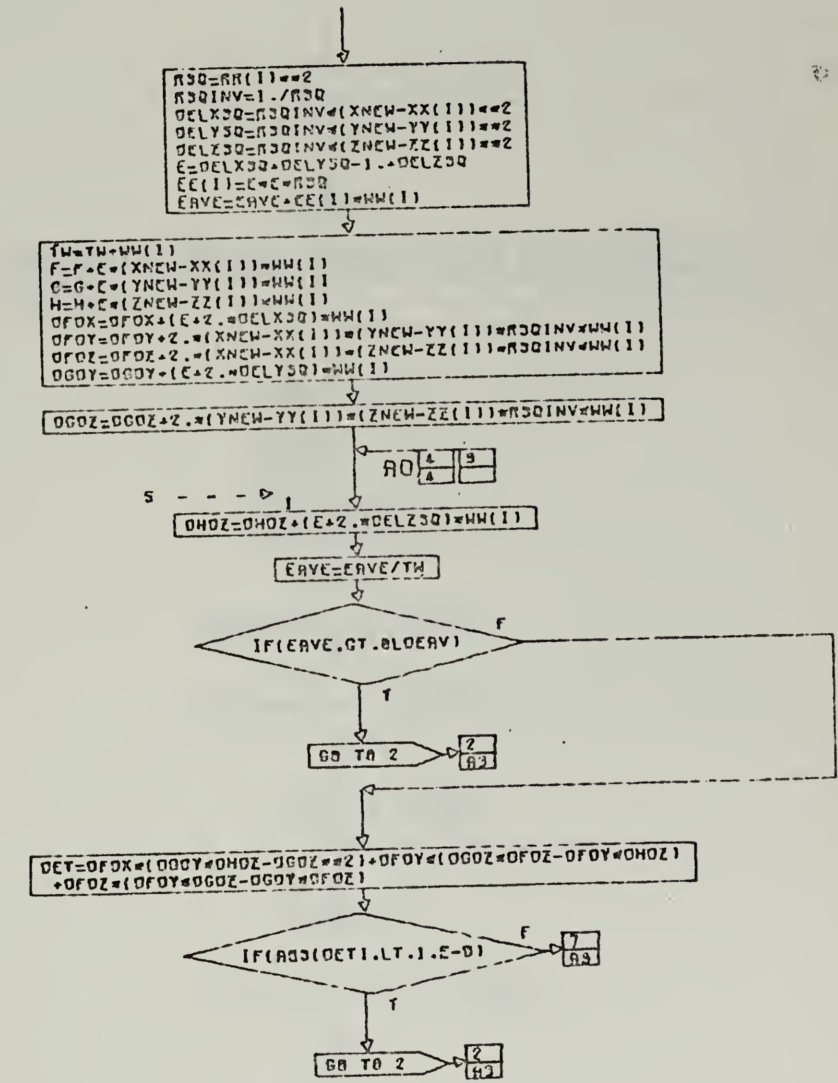
CONT. ON PG 4



CONT. ON PG 5



CONT. ON PG 6



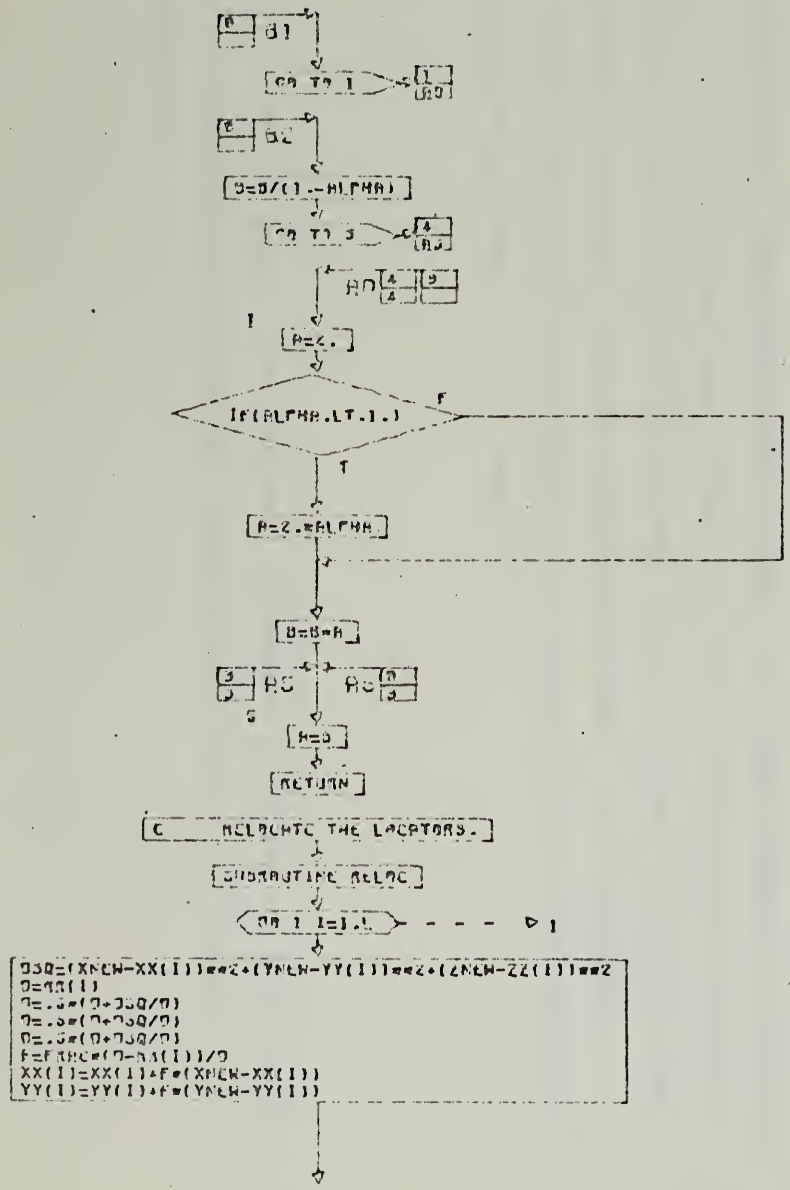
CONT. ON PG 7

```
A9
DETINV=1./DET
DX=DETINV*(F*(DGDZ**2-DGDY*DHDZ)+G*(DFDY*DHDZ-DFDZ*DGDZ)+
 H*(DFDZ*DGDY-DFDY*DGDZ))
DY=DETINV*(F*(DFDY*DHDZ-DGDZ*DFDZ)+G*(DFDZ**2-DFDX*DHDZ)+
 H*(DFDX*DGDZ-DFDZ*DFDY))
DZ=DETINV*(F*(DGDY*DFDZ-DFDY*DGDZ)+G*(DFDX*DGDZ-DFDY*DFDZ)+
 H*(DFDY**2-DFDX*DGDY))
ALDEAV=EAVE
C     IF(P)WRITE(6,9)F,G,H,DFDX,DFDY,DFDZ,DGDY,DGDZ,DHDZ,XNEW,YNEW,ZNEW,
C    1 DX,DY,DZ,(XX(I),I=1,M),(YY(I),I=1,M),(ZZ(I),I=1,M),(R(I),I=1,M),
C    2(WW(I),I=1,M),(EE(I),I=1,M)
9 FORMAT(/(1X,13E8.3))
C     IF THIS STEP SIZE IS NOT MUCH SMALLER THAN THAT OF THE PREVIOUS
C     ITERATION, CORRECT THE SIZE OF THIS ITERATION.
CALL CORECT(EX,DX)
CALL CORECT(EY,DY)
CALL CORECT(EZ,DZ)
XNEW=XNEW+DX
YNEW=YNEW+DY
ZNEW=ZNEW+DZ
S(JFIND)=AMAX1(ABS(F),ABS(G),ABS(H)).LT.TH
RETURN
C     THE ITERATION HAS DIVERGED.  IF THIS IS THE FIRST ITERATION FOR
C     THIS LOCATEE, THE ROUTINE CONKS OUT.
A3 6/6
2 STOP=.TRUE.
WRITE(6,3) EAVE,ALDEAV,DET
```

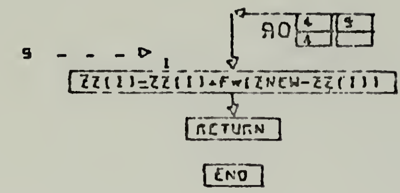
CONT. ON PG 8


```
IF(I1.EQ.0.AND.K.EQ.0)   T: RETURN   F:
C     DIVERGENCE FOR SOME OTHER ITERATION (NOT THE FIRST).  HALVE THE
C     STEP SIZE OF THE PREVIOUS ITERATION, AND RETURN TO TRY ANOTHER
C     ITERATION.
STOP=.FALSE.
DX=DX/2.
DY=DY/2.
DZ=DZ/2.
XNEW=XNEW-DX
YNEW=YNEW-DY
ZNEW=ZNEW-DZ
RETURN
SUBROUTINE CORECT(A,B)
IF(ABS(B/A).LT..5)   T: GO TO 5 (4/A5)   F:
ALPHA=B/A
IF(ALPHA.GT.0.)   T: 9/B1   F: 3/B2
```

CONT. ON PG 9


B1
GO TO 1
B2
D=D/(1.-ALPHA)
GO TO 3
AO
A=2.
IF(ALPHA.LT.1.)
F
T
A=2.*ALPHA
B=B*A
AO
A=0
RETURN
C RELOCATE THE LOCATORS.
SUBROUTINE RELOC
DO 1 I=1,L
DSQ=(XNEW-XX(I))**2+(YNEW-YY(I))**2+(ZNEW-ZZ(I))**2
XX(I)=XX(I)+F*(XNEW-XX(I))
YY(I)=YY(I)+F*(YNEW-YY(I))

CONT. ON PG 10


AO
ZZ(I)=ZZ(I)+F*(ZNEW-ZZ(I))
RETURN
END

PROJECT " WHERE " : WORKING PAPER NO. 8

LS, LSS, AND RELOCATION

James A. Lechner

April, 1972

ABSTRACT

In Working Paper No. 2 it was suggested that since the locators' positions are not known exactly, it might prove profitable to re-estimate these positions as an adjunct to estimating the position of the "locatee" unit. This paper presents an iterative method, along the lines proposed in Working Paper No. 2, for handling the computations of such a "relocation" process. The discussion is carried out in the context of the LS(least squares) and LSS(least squares, squared) position-location methods. As background, evidence is given which shows that these two criteria behave very similarly so long as range measurement errors remain small.

Note: Project working papers are informal documents prepared to facilitate discussion and communication; they may contain tentative or relatively unchecked material.

CONTENTS

0. INTRODUCTION

In this paper the term location-operation will be used for a process of attributing a position to one unit (the locatee) on the basis of its recorded distances from n other units (the locators). These recorded distances are of course subject to measurement error.

Roughly speaking, the situation studied in Project "WHERE" is one in which the role of "locatee" varies over some totality of units, with many of these units in motion at any time. From this it is apparent that for any one location-operation, the positions of the locators (which may have been the locatees of previous operations) are not in general known exactly. This point was emphasized in Section 5 of Working Paper No. 2 [" An Iterative Approach to the Location-Operation", A.J. Goldman, 10/71], where it was suggested that the location-operation be expanded to include re-estimation (i.e., relocation) of the locators' positions, based on the latest set of range measurements.

The present paper has as its main purpose the computational specification of such a relocation procedure. This is given in Section 2, which also discusses the respective merits of the LS (least squares) and LSS (least squares, squared) position-location methods in this context. Section 1 presents some relevant background information concerning the relationship between these two algorithms.

We conclude this Introduction by introducing some necessary mathematical notation:

O = (arbitrary) origin of coordinate system,

X = position to be atrributed to locatee,

x = vector OX

A_i = position to be attributed to i-th observer $(i = 1, 2, \ldots, n)$,

a_i = vector OA_i ,

α_i = initial estimate of a_i ,

$r_i = \|x - a_i\|$ = estimated distance between locatee and i-th observer,

ρ_i = measured value of this distance .

If the distinction between a_i and α_i is dropped (i.e., the initial estimates of the locators' position are accepted without revision), then we have the "pure" location-operation, which can be expressed as the problem of making a "good" choice of x , given the ρ_i and a_i . The LS method expresses this problem as that of minimizing the quadratic "penalty function"

$$f(x) = \sum_i w_i'(r_i - \rho_i)^2 \qquad (0.1)$$

where the w_i' are positive "weights" and x enters via the r_i's ; the LSS approach involves the minimization of

$$g(x) = \sum_i w_i(r_i^2 - \rho_i^2)^2 , \qquad (0.2)$$

with "weights" w_i . The modification of these formulations, to encompass relocation, will be taken up in Section 2 .

1. RELATION BETWEEN LS AND LSS APPROACHES

In this section, as background for the discussion of relocation in Section 2, we discuss the relation between the LS and LSS approaches to the "pure" location-operation. Three pieces of evidence will be given to show that if the "weights" in these two criteria --- the w_i' of (0.1) and the w_i of (0.2) --- are properly related, then the two methods will not differ significantly in accuracy so long as the tracking process is <u>under control</u>. (The last phrase means that the units' positions remain well enough known that the discrepancies between measured ranges ρ_i and calculated ranges r_i are small relative to the magnitudes of the ranges themselves.)

The <u>first</u> piece of evidence begins with the observation that the "under control" assumption, $r_i/\rho_i \approx 1$, yields

$$
\begin{aligned}
(r_i^2 - \rho_i^2)^2 &= (r_i + \rho_i)^2 (r_i - \rho_i)^2 \\
&= (1 + r_i/\rho_i)^2 \rho_i^2 (r_i - \rho_i)^2 \\
&\approx 4\rho_i^2 (r_i - \rho_i)^2 ,
\end{aligned}
$$

or equivalently

$$(1/4\rho_i^2)(r_i^2 - \rho_i^2)^2 \approx (r_i - \rho_i)^2 .$$

Thus, if we set $w_i = (1/4\rho_i^2)\, w_i'$, then comparison of (0.1) with (0.2) shows that $g(x) \approx f(x)$; equivalently, if we set

$$w_i = w_i'/\rho_i^2 \tag{1.1}$$

then comparison of (0.1) yields $g(x) \approx 4f(x)$, so that minimization of $f(x)$ --- i.e., the LS approach --- is "approximately equivalent" to the LSS approach (minimization of $g(x)$).

Equation (1.1) gives the appropriate relation between the two sets of weights. As yet, no rationale has been found for assigning to the w_i' any value other than unity. In fact, since the distribution of errors in (unrejected) ranges seems (by best engineering judgement at this time) to be Gaussian, with range-independent variance and zero mean (except for a small percentage of ranges biased by about 8m), maximum likelihood estimation would strongly recommend[1] use of LS with $w_i' = 1$ (together with some technique for removing "outliers"), which by the above is approximately equivalent to use of LSS with

$$w_i = 1/\rho_i^2 \quad . \tag{1.2}$$

This choice of w_i' and w_i will be used throughout the rest of this paper.

The preceding analysis shows that the penalty functions for LS and LSS are approximately equal (when measured and calculated ranges are close together), but one would like some direct assurance that the position-estimates obtained by minimizing these two functions are also nearly equal. This is provided by our <u>second</u> piece of evidence, an analysis for a simple 1-dimensional configuration in which the locators are situated at the points with coordinates R and $(-R)$ with R <u>large</u>, while the locatee is actually situated at the origin. The LS approach calls for the minimization of

$$\begin{aligned} f(x) &= [(R-x) - \rho_1]^2 + [(R+x) - \rho_2]^2 \\ &= 2\{x^2 + (\rho_1 - \rho_2)x\} + \text{const.}, \end{aligned}$$

---

(1) See Section 1 of Working Paper No. 2 .

which is readily found to be given by

$$x = (\rho_2 - \rho_1)/2 \quad . \qquad \text{(LS solution)} \qquad (1.3)$$

The LSS approach calls for the minimization of

$$\begin{aligned} g(x) &= \rho_1^{-2}[(R-x)^2 - \rho_1^2]^2 + \rho_2^{-2}[(R+x)^2 - \rho_2^2]^2 \\ &= \rho_1^{-2}(R-x)^4 + \rho_2^{-2}(R+x)^4 - 4x^2 + \text{const.} \; ; \end{aligned}$$

setting $dg/dx = 0$ leads to the cubic equation

$$(\rho_1^2 + \rho_2^2)x^3 + 3R(\rho_1^2 - \rho_2^2)x^2 + [3R^2(\rho_1^2 + \rho_2^2) - 2\rho_1^2\rho_2^2]\,x + R^3(\rho_1^2 - \rho_2^2) = 0 \, . \qquad (1.4)$$

Let E denote a measure of maximum range-error. In the worst-case scenario described by $\rho_1 = R-E$ and $\rho_2 = R + E$, the LS and LSS approaches yield exactly the same answer, both (1.3) and (1.4) giving $x = E$. In the alternative scenario described by $\rho_1 = R$ and $\rho_2 = R + E$, the respective solutions are

$$x_{LS} = \tfrac{1}{2}E \; , \quad x_{LSS} = \tfrac{1}{2}E\,[1 - \tfrac{3}{8}(E/R)^2] \qquad (1.5)$$

where the expression for x_{LSS} omits higher-order terms in the small quantity E/R; if for example R is at least 1 km and E is at most 20m, then the two position-estimates differ by at most 0.0015 m.

Our <u>third</u> piece of evidence is empirical in nature. For each of two distributions of range-measurement errors ("small" and "large", respectively), 72 trial 2-dimensional location-operations were carried out using both LS and LSS . Each of 18 units served as locatee 4 times (for each error distribution and each method), with the other 17 units serving as locators. For the "small" case, the locations were as much as 3.32m in error, but the LS and LSS results differed from each other by no more than 0.0004 m ; for the "large" case, despite location errors up to 31.9m, the two methods yielded locations no more than 0.045m apart.

2. THE RELOCATION PROCESS

When the location-operation is enlarged to include relocation (of the locators), it can be formulated as making a "good" choice of x and the a_i, given the ρ_i and α_i. The approach given in Section 4 of Working Paper No. 2 is based on two notions, one of which we have found it is useful to modify in a manner described below.

The <u>first</u> notion is that the choices of x and a_i's should be made so as to minimize a penalty function which generalizes those given earlier [see (0.1) and (0.2)] for "pure" location. Specifically, this function is

$$f(x; a_1, \ldots, a_n) = \Sigma_i w_i'(r_i - \rho_i)^2 + \Sigma_i W_i \| a_i - \alpha_i \|^2 \qquad (4.1a)$$

for the LS approach, and

$$g(x; a_1, \ldots, a_n) = \Sigma_i w_i(r_i^2 - \rho_i^2)^2 + \Sigma_i W_i \| a_i - \alpha_i \|^2 \qquad (4.1b)$$

for the LSS approach ; the W_i are positive numerical weights.

The <u>second</u> notion is to accomplish this minimization by a process which <u>alternates</u> between <u>location steps</u> (in which the a_i are held fixed and an appropriate x is chosen) and <u>relocation steps</u> (in which x is held fixed and appropriate a_i's are chosen. This process can be formalized as follows:

INITIALIZATION: Set $k = 1$, and $a_i^{(1)} = \alpha_i$ (all i) .

STEP 1 (location, k-th pass) : Given the current estimates $a_i^{(k)}$ of the locator positions, choose $x = x^{(k)}$ to minimize the first sum in (4.1), i.e. the part of (4.1) which depends on x .

STEP 2 (relocation, k-th pass) : For each i, keep x fixed at $x^{(k)}$ and choose a_i to minimize the sum of those terms in (4.1) which depend on a_i. Increase k by 1, and set each $a_i^{(k)}$ equal to the a_i just found. Return to Step 1.

For Step 1, the material in Section 1 shows that there is little reason to prefer one of LS or LSS over the other so far as location-errors are concerned. We therefore select LSS, since it seems likely to converge more quickly and requires fewer time-consuming calculations per iteration. Thus Step 1 employs (4.1b) rather than (4.1a), and so according to the description above, the treatment of the i-th locator in the k-th relocation step would require choosing a_i to minimize

$$w_i\{\|x^{(k)} - a_i\|^2 - \rho_i^2\}^2 + W_i\|a_i - \alpha_i\|^2 . \tag{4.2}$$

For Step 2, however, it is more convenient to minimize instead the sum of corresponding summands from (4.1a), namely

$$h(a_i) = w_i'\{\|x^{(k)} - a_i\| - \rho_i\}^2 + W_i\|a_i - \alpha_i\|^2 . \tag{4.3}$$

The over-all method becomes a kind of LS-LSS hybrid ; Steps 1 and 2 in effect involve different penalty functions, so that the "first notion" described above has indeed been modified.

In Working Paper No. 2 it was envisiged that the alternating process would be iterated until it had "settled down" satisfactorily. This idea, too, has been modified after further reflection. The relocation of the i-th locator is based mainly upon a single new datum, namely ρ_i ; it seems

unwise to put too much reliance or to base too much calculation on this one datum (which is subject to measurement-error). Besides, from the viewpoint of reducing total computation time it is desirable to limit the number of iterations. Thus a sensible compromise seems to be: locate the locatee (Step 1); relocate the locators (Step 2) ; then locate the locatee again (second pass through Step 1), and stop. This truncated version of the full process is assumed below.

We turn now to the execution of Step 2. With superscripts (k) and subscripts (i) dropped for simplicity, the function (4.3) to be minimized can be written

$$h(a) = w'\{\|x-a\| - \rho\}^2 + W\|a-\alpha\|^2 . \tag{4.4}$$

Consider any trial position a' of a . It lies on a sphere of radius $\|x-a'\|$ centered at x . Suppose a' is replaced by the point of that same sphere which is closest to α ; then the first summand in (4.4) is unchanged, while the second summand is reduced. Hence the minimum can only occur for such a "closest point". Simple geometric considerations show that such a point lies either on the line segment $[x, \alpha]$ or on its extension through α . Thus, for some scalar $s \geq 0$, we have

$$a = x + s(\alpha - x) ,$$

leading to

$$x - a = s(x-\alpha) , \quad a - \alpha = (1-s)(x-\alpha) .$$

Substitution of these results into (4.4) converts $h(a)$ into a function of s ; with the abbreviation

$$r = \|x - \alpha\| , \tag{4.5}$$

this function reads

$$H(s) = w'\{sr-\rho)^2 + Wr^2(1-s)^2$$
$$= (w'+W)r^2s^2 - 2r(w'\rho + Wr)s + (w'\rho^2 + Wr^2), \tag{4.6}$$

and is minimized by setting

$$s = (w'\rho + Wr)/(w' + W)r . \tag{4.7}$$

(The availability of such a simple explicit solution to the minimization problem justifies our earlier assertion that use of (4.4) rather than (4.2) is more convenient.) If $\rho > r$ then $s > 1$, signifying that a lies on the extension of segment $[x, \alpha]$ through α (i.e., relocation has moved the locator farther from the locatee) ; if $\rho < r$, then $s < 1$ and a lies on the segment (so that the locator has been moved closer to the locatee). These results, which do not depend on w and W, seem quite consistent with what one would expect intuitively.

The only remaining question is the selection of numerical values for the weights W_i. We have at present no definitive answer to this problem. There are two distinct considerations to be taken into account:

(a) If we consider the current location (and relocation) operation in isolation, then the weight W_i should depend primarily on our confidence in the accuracy of the initial estimate α_i of the i-th locator's position (relative to the accuracies of the other α_j's, and also of the range measurements). In an idealized Gaussian-error case, for example, maximum-likelihood estimation would dictate weights inversely proportional to the respective variances of the associated random errors. For our practical situation, how can we quantify the degree of confidence merited by the initial estimate α_i ?

It may well prove possible to concoct an appropriate measure based on the sizes of the discrepancies $|r_j - \rho_j|$ obtained during the last previous location-operation in which the i-th of the current locators figured as locatee---probably modified by some "decay factor" to take account of how much time has elapsed since this earlier operation.

(b) However, one needs to consider the "current" operation in the light of the need to maintain stability of the entire location process; the selection (or least the effect) of the W_i must reflect not only the one-shot independent-measurement case as in (a), but also the sequential correlated-error many-locator nature of the real problem. If a relocation(based mainly on a single and possible "bad" range measurement ρ_i) is allowed to be too large, its influence would persist through a number of operations until the unit in question again took its turn as locatee, and so such instabilities in estimation of positions could result in large errors.

The following tactic may prove an acceptable compromise. Weights W_i will be selected according to the considerations in (a) above. The resultant solutions (4.7) of the n minimization problems for relocation (1 problem for each of the n locators) will dictate moving the estimated position of the i-th locator a certain distance (from α_i) in a certain direction along the line through $x^{(k)}$ and α_i . To "temper" this solution in accordance with the considerations in (b), we will "damp" that distance by a multiplicative factor (say, between 0.1 and 0.6) before actually revising the estimate of the locator's position. Numerical experimentation will be required to test the performance of this approach and to determine what values of the adjustment factor are preferable.

PROJECT "WHERE": WORKING PAPER NO. 9

THE MNPLS SIMULATION ("WHERESIM")

R.H.F. Jackson

May, 1972

Abstract: This paper documents the present version of the MNPLS simulation. The documentation is presented at three levels: the "executive" level (an overview), the "user" level (description of input and output), and the "analyst" level (detailed discussion of the logic flow). It focuses on the concepts indigenous to such a simulation study, namely: movement of units, range computation (including the occurrence of measurement error), and position estimation. A dictionary of variables used in the program, as well as a listing and a detailed flowchart of the program, are also included.

Note: Project working papers are informal documents prepared to facilitate discussion and communication; they may contain tentative or relatively unchecked material.

TABLE OF CONTENTS

1. INTRODUCTION

WHERSM(*) is a computer code that simulates the operation of a Micro-Navigation Position Location System (MNPLS). It provides a "real life" framework within which the propagation over time of position location errors, resulting from the operation of such a system, can be determined. It further provides a facility for monitoring the operation of such a system under a variety of movement scenarios, and using a variety of algorithms for accomplishing the location operation.

This paper documents the present version of the WHERSM code at three levels: the "executive" level, the "user's" level, and the "analyst's" level. This introductory section, which serves as the executive level of documentation, is aimed at the person who is interested only in an overview of what WHERSM is and what it does. The user's level of documentation is meant for one who is concerned with making a computer run with the code as it now stands; i.e., with no major program modifications. Sections 10 and 11 provide documentation of all the inputs and outputs -- the information needed to make a run. The remainder of the paper, Sections 2 through 9 and appendices A, B, and C, is designed with the analyst in mind. By an "analyst" is meant one who needs not only to run the program, but also perhaps to modify it to handle somewhat different situations. This paper should include enough information to start him on his way.

Due to the nature of the MNPLS under study, WHERSM is a "fixed time increment" simulation; i.e., simulated time is advanced by a constant increment, Δt, rather than by variable increments dictated by the occurrence of events, as would be the case in the type of simulation classified as "next event".

(*) Full name: WHERESIM. Compressed because of computer restrictions to 6 characters per name.

The program is written in FORTRAN V, but with very little conversion it can be compiled with a FORTRAN IV compiler. It is liberally annotated with comments, and although the code documented here employs program overlays, parallel processing, and drum usage, every attempt has been made to keep machine dependence minimized. Furthermore, the WHERSM code is modular in nature, so that the logic flow is kept simple and easy to understand, and so that different concepts in deployment, movement, intervisibility, position location method, etc., can with a minimum of reprogramming be implemented and tested. This modularity also allows extreme flexibility in the type and quantity of inputs required to make a run.

The basic ideas involved in an implementation of the MNPLS are that there are n units in the field, any of which at each moment might (in the most general case) be either moving or fixed. Every Δt seconds, a master computer receives information in the form of ranges, subject to measurement error, from one of these units to some or all of the others. The master computer must then, using position location algorithms, estimate the position of that unit and also prepare to receive the set of ranges for the next unit in sequence. The elements of this description that are relevant to a simulation study of an MNPLS are: movement, range computation (including the occurrence of measurement error), and position estimation.

In particular, the simulation must have some facility for determining the true positions of all the units at all relevant times. It is easy to provide a movement scenario in terms of movements in the xy-plane (longitude-latitude). A starting x and y, an azimuth of movement, and a speed will suffice for this(*). It is the z coordinate, the height above sea level, that causes difficulty.

(*) As will be explained in Section 2, the movements are assumed to be piecewise linear, so that in fact a <u>set</u> of azimuths, speeds, <u>and</u> change times are required.

There are at least two methods of providing the z coordinate in a simulation. One involves a continuous function which yields a value of z for <u>any</u> values of x and y; the other is by means of a discrete function which specifies values of z for <u>particular</u> values of x and y. In the discrete case, values of x and y other than the prescribed ones are assumed to coincide with the nearest available grid point, whose specified height is then used(*). Both of these techniques are explained in detail in section 2.

The next item with which the simulation of the MNPLS must be concerned is the computation of ranges. Of course, given the ability to compute the true positions of all units at any instant of time, the true ranges can easily be calculated, since they are simply Euclidean distances. But, in the "real life" situation, <u>true</u> ranges are never reported to the master computer due to measurement errors intrinsic to the particular method of measuring those ranges. This phenomenon is easily simulated once a probability distribution of measurement errors is specified, since the simulation program can simply compute pseudo-random numbers which satisfy such a distribution law.

Another point relevant to range computation is that a simulation must be capable of determining whether a range <u>should</u> be computed; i.e., whether, for a specified pair of units at a specific moment, a radio line of sight (LOS) exists. There are many ways(**) of providing a simulation program with the ability to determine the existence or non-existence of these intervisibilities. The method used in WHERSM is to maintain a set(***) of "intervisibility snapshot matrices", I_{ijt} ($i = 1,2,\ldots,n$; $j = 1,2,\ldots,n$;

(*) Four-point interpolation could have been used here, but is very time consuming. And since it was determined that no significant further loss of accuracy occurs with the "nearest point" method, that one was implemented.

(**) See p.28 of "First Interim Progress Report on NBS Project WHERE in Support of USAAMCA Project MNPLS", NBS Report 10663, December 1971.

(***) Obtained from the Electromagnetic Compatibility Analysis Center, Annapolis, Md.

$t = 0, 20, 40, \ldots, 120$) where $I_{ijt} = 1$ if unit j can receive <u>and report</u> a ranging signal from unit i at time t minutes into the simulation, and $I_{ijt} = 0$ if it cannot. Furthermore, in order to avoid drastic changes in the intervisibility pattern at the discrete times indicated by values of the subscript t, each row of a particular intervisibility matrix defined by a specific value of t, say t', is overlaid with the corresponding row of the next-in-sequence matrix (defined by $t' + 20$) at some randomly chosen time between t' and $t' + 20$. (These times are chosen independently for each row and for each value of t.) More detail regarding each aspect of range computation will be found in sections 3 and 5.

The last item of concern for the MNPLS simulation is that of position estimation. Although that item is an integral part of any simulation study of this nature (indeed, the most important part), no discussion of the various algorithms used in WHERSM is provided here, since the project staff members responsible for the development and implementation of position location algorithms have presented such a discussion in Working Paper Number 7.

In order to provide the reader with a better understanding of the flow of events within WHERSM, and to facilitate further discussion of the WHERSM program, a flowchart is given in figure 1 which indicates the basic logic of the simulation. With one exception (see section 9), each box in the flowchart corresponds to an easily recognizable module within the WHERSM code listed in appendix B. In every case, entry to box 2 is made just after the simulation's current time is incremented by a constant value. That constant, Δt, is referred to as the <u>subcycle</u> time and is the time during which all the operations required to locate a given unit (one completion of the loop formed by boxes 2 through 7 of the flowchart) must be performed. <u>Cycle time</u> is the time required to locate each unit <u>at least once</u>. This may be larger than $n \cdot \Delta t$, where n is the total number of units, since certain faster moving units may need to be located more often

than do slow moving units. Locatee refers to that unit which, at any instant of simulated time, is in the process of being located, and locator refers to a unit whose range to the locatee is used to estimate the position of the locatee.

The remainder of this report discusses the workings of WHERSM at a more detailed level. Section 2 explains Boxes 2 and 3 of the flowchart in figure 1, while Section 3 discusses Box 4. Section 4 explains Boxes 5, 6, and 7; Section 5, Boxes 8 and 9; Section 6, Boxes 10 and 11; Section 7, Box 12, Section 8, Boxes 13 and 14; and Section 9, Box 15. Section 10 contains a discussion of the input needed to make a run with the present version of the code; and Section 11 contains a discussion, with examples, of the outputs that will result from such a run. Appendix A is a dictionary of variables used in the program; Appendix B, a listing of the program; and Appendix C, a very detailed flowchart of the program.

Although the remaining section headings refer directly to the flowchart in figure 1, the content of those sections can be better understood by referring to Appendices B and C.

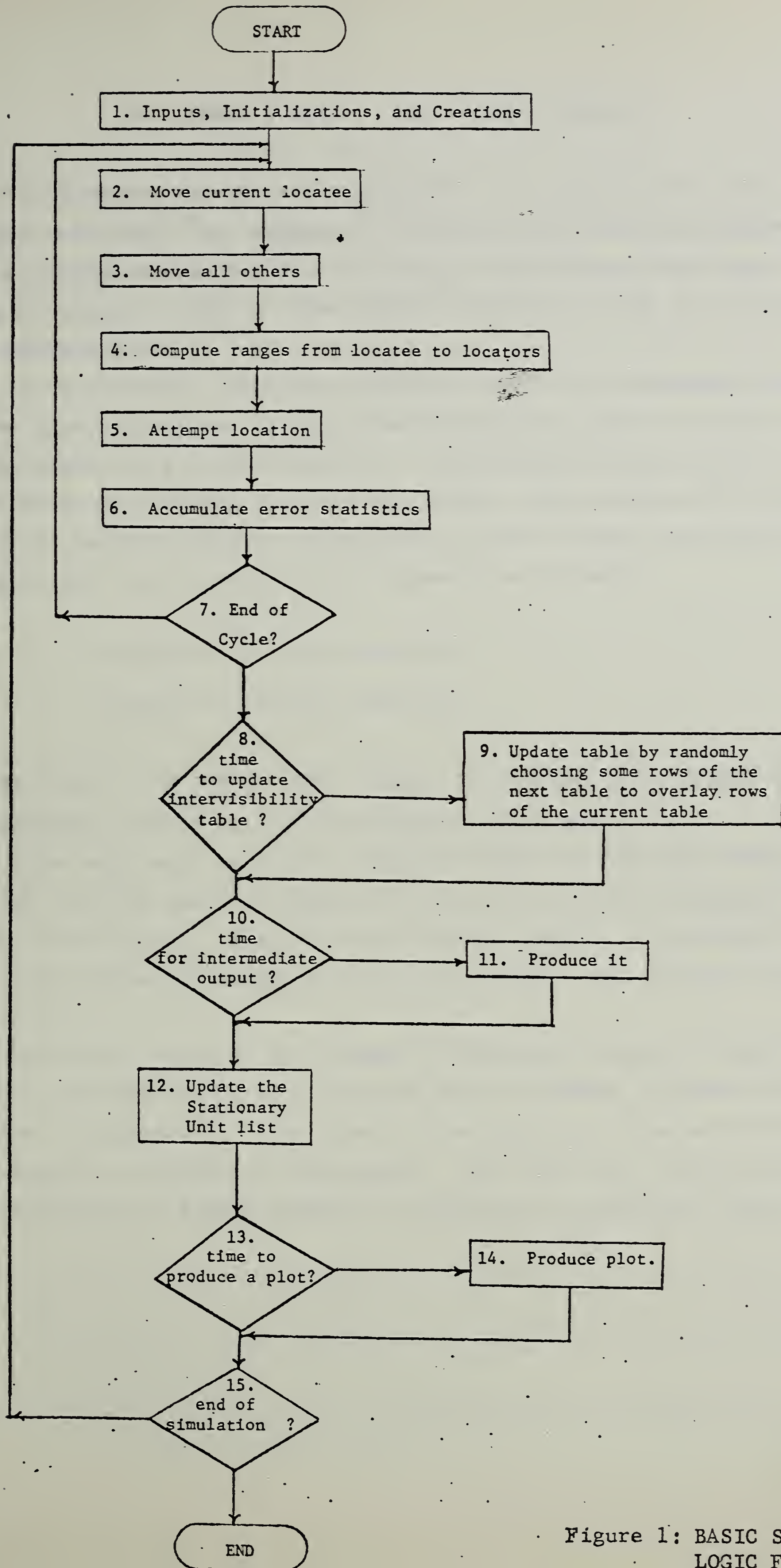


Figure 1: BASIC SIMULATION LOGIC FLOWCHART

2. BOXES 2 and 3: THE MOVEMENT MODULES

The movement modules are responsible for computing the true positions of each unit every Δt seconds. In the current version of WHERSM, this is a two-stage process; i.e., x and y are computed, and then z is computed using x and y. The method by which x and y are computed is explained first.

In the xy-plane, units are assumed to move in a piecewise linear manner (the projections of the three-dimensional curves of motion onto the xy-plane are piecewise linear). The piecewise linear motion is effected by maintaining a matrix of current true positions (TP) and a matrix of current velocity vectors (VV), both of which are initialized by card input, and by updating TP linearly as follows:

```
TP(I,1) = TP(I,1) + VV(I,2)

TP(I,2) = TP(I,2) + VV(I,2)
```

for each unit I = 1,2,..., NU, where NU is the total number of units. See Appendix A for a detailed explanation of TP and VV.

After each unit is moved, a check is made to determine whether it is time for that unit to change its direction of motion (change times are stored in VV(I,3)). If so, the new velocity vector is retrieved from the matrix VV1 where <u>all</u> velocity vectors are stored, and replaces the existing one in VV.

The second stage of the movement computation, that of computing z, has been implemented in two different ways in WHERSM, although only one appears in appendix B. (The other is too specific to include here and, furthermore, is trivial to implement.) The first one, the one not included here, is simply a matter of coding up a continuous function

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

representation(*) for the z coordinate and computing a value of z once x and y have been determined.

The method of determining z that is included in Appendix B is somewhat complicated, in that it involves a preprocessor, overlays, a mass storage retrieval routine, and parallel processing. The preprocessor is a routine called BUILD, whose function is to read a tape containing topographical data in the format designed by the Electromagnetic Compatibility Analysis Center(**), decode and unpack the individual altitudes above sea level that are given on that tape, write the individual elevations out on the drum, create an index (NDX) to the information on the drum, and initiate the overlaying of itself by the rest of the WHERSM code.

The subroutine that actually provides the z coordinate, HEIGHT, is mainly a drum retrieval routine. HEIGHT takes the x and y that are input to it, computes the nearest available grid point to that x and y, determines the position on the drum of the altitude associated with that grid point, and initiates the retrieval of that altitude. This last point is important -- HEIGHT only initiates the read. It does not wait for the completion of that operation, but instead immediately returns control to the calling routine, which has the option of waiting for the completion of the retrieval or of performing other tasks that are not dependent on knowledge of the new z coordinate. In view of the fact that the time required to retrieve one elevation from the drum is approximately 4.254 milliseconds, the above parallel processing capability provided by the HEIGHT routine allows considerable time savings to be

---

(*) Note that obtaining such a function for "fictitious" terrains is rather easy, but that surface fitting for "real" terrains may not be trivial. See, e.g., Pavilidis, T., "Piecewise Approximations of Functions of Two Variables and its Application in Topographic Data Reduction", Princeton University, Tech. Report No. 86, Sept. 1970, and "A Program Implementing Piecewise Linear Approximations of Functions of Two Variables", Princeton University Tech. Memo. No. 3, March 1971.

(**) Encoded altitudes above sea level at points that are three seconds of latitude and longitude apart.

achieved through the thoughtful organization of the main routine relative to height retrieval.

The previous comment regarding organization serves to introduce the last point to be discussed in this section -- the reason for including <u>two</u> movement modules -- one (Box 2) for the current locatee, and one (Box 3) for all other units in the field. Although all units move every subcycle, a new height is determined only when a unit serves as locatee. (In between, heights are treated as remaining constant.) Hence, while the height for the locatee is being retrieved from the drum, the program can continue on with the moving of all other units, thus taking advantage of the parallel processing provided by HEIGHT. There is very little loss of "realism" with this approach, since due to the design criteria no unit moves farther than 44 meters between times when it is a locatee, and further, since the topographical grid points define a rectangle with dimensions approximately 68m x 92m. Hence, only a small distortion of an already grossly simplified terrain structure occurs.

There is yet another reason for separating movement into two modules. Namely, it reduces the effort involved in implementing a very specific type of movement scenario. For example, the code in Appendix B includes a technique for keeping units from spreading out too much, by disallowing movement past a boundary that moves at a speed determined by input. (This is convenient when a randomly generated scenario that includes foot soldiers and ground vehicles is used. By letting the boundary move at foot soldier speed, the ground vehicles can be kept "close by".) In this case, it is wasteful to monitor a unit's movements <u>all</u> the time. To check only when a unit serves as locatee is sufficient. And this of course is easier and faster when there is a separate movement module for the locatee.

3. BOX 4: THE RANGE COMPUTATION

The range computation module is responsible for computing reported ranges by computing true ranges and perturbing them, and for determining whether each range should be reported. The computation of true ranges involves only Euclidean distance calculations, and the perturbation values are obtained by a call to subroutine RG. Subroutine RG generates pseudo-random numbers having a Gaussian distribution with mean zero and variance $36m^2$, and then randomly chooses 1% of them to which it imputes a positive bias by adding 7 meters to each. Except for the biasing procedure, the method used to generate these numbers is the same as the one proposed by A. Rotenburg(*). RG calls RGU which provides uniformly distributed pseudo-random numbers, generated by the multiplicative congruential method. Other distributions, or other parameter values in these, could easily be substituted in these routines. In order to save time,since RGU is called so frequently during the simulation, Rit is written in SLEUTH, the machine language for the UNIVAC 1108. However, it is very short and can easily be rewritten in FORTRAN.

There are two items that determine whether a particular range should be reported. The first of these is the logical matrix RPT, which is initialized at input time, and which, for each unit, has the logical value "TRUE" if that unit is one that reports received ranges, and "FALSE" if it is not. Note that the maintenance of such a matrix allows the testing of scenarios in which the set of reporting units changes during a run, and where such changes are either exogenous <u>or</u> endogenous events.

The second item that determines whether a particular range should be reported is the matrix of intervisibilities (IT), which records whether or not a radio line of sight (LOS) exists between each pair of units

---

(*) Rotenburg, A., "A New Pseudo-Random Number Generator", J. ACM <u>7</u>. (1960), pp. 75-77.

identified by the subscripts for IT. Also, each row of IT is condensed so that 36 row entries fit into one computer word, thereby greatly reducing storage use. Therefore, the only concern of the range computation module regarding intervisibilities is that of locating and retrieving a particular entry in IT from a block of 36 entries, which informs it whether the range between the defining units should be calculated.

In summary, then, for each subcycle the range computation module steps through a list of the units (excluding, of course, the one currently being located) and checks RPT to see first if a particular unit is a reporter. If so, IT is checked to determine whether that unit can "see" the locatee. If that also is true, then the true distance is calculated, RG is called to obtain a perturbation value, and that value is added to the true distance yielding a reported range. Reported ranges are stored in the matrix RANGE, with the unit numbers of the units that reported each range in matrix LOCATR.

4. BOXES 5, 6, and 7; THE LOCATION ATTEMPT, STATISTICS ACCUMULATION, AND "END OF CYCLE" TEST

The discussions of Boxes 5, 6, and 7 have been combined into this one section simply because there is very little to say about any of them.

The location attempt consists of a call to subroutine FINDIT, which is a preprocessor for the actual location algorithms. As was mentioned in Section 1, no discussion of the algorithms is provided here; nevertheless, some discussion of the input to the algorithms from WHERSM should be given. All input to the position location algorithms is achieved via the labeled common block CB1, which consists of the following variables: EP, LOCATR, RANGE, S, TLOC, W, IU, CLK, XNEW, YNEW, ZNEW, FXD, RPT, and NTYPE. For a description of each of these variables, see Appendix A.

The error statistics that are accumulated are: the current coordinate errors (coordinate-wise subtraction of true from estimated positions), the locatee's position location error (Euclidean distance between true and estimated positions in xy-plane), the error in the z coordinate, the max error in the xy-plane over all cycles and over the current block of KIC cycles (KIC is an input parameter), the maximum error in z over all cycles and over the current KIC cycles, and the time to perform the location operation (obtained by interrogating the computer's internal clock). The statistics accumulated here are used to produce the intermediate output explained in Section 11.

The "end of cycle" test (Box 7 of the flowchart in figure 1) is nothing more than the end of a DO loop which steps the simulation program through each one of the subcycles in a cycle.

5. BOXES 8 and 9: THE "INTERVISIBILITY TABLE UPDATE"

The "intervisibility update" module is concerned with maintaining an LOS pattern that is checked by the WHERSM program during range computation (see Section 3). The "current" intervisibility pattern is kept in the matrix IT, which is automatically updated in a manner determined by parameters that are input to the simulation program. This section explains the manner in which IT is updated.

The intervisibility snapshot matrices that have been discussed in Sections 1 and 3 are read in from tape at input time and stored in the matrix ITB. (The format of the tape is explained in Section 10.) At the start of a simulation run, IT is initialized to the first of the snapshot matrices--the matrix that corresponds to a line of sight snapshot taken at time zero. Every K2C cycles thereafter, K3C randomly chosen rows of IT are overlaid with the corresponding rows of the snapshot matrix that is next in sequence. (K2C and K3C are input parameters; see Section 10.) This overlaying process continues until simulated time has been advanced to the time at which that next snapshot was taken, at which point the intervisibility update module completes the overlaying of all rows not previously updated, and prepares to start over again with the new matrix and the one after that.

The choice of a row to be updated is achieved by choosing a pseudorandom number, call it I, between 1 and NU, where NU is the number of units in the field and also the number of rows and columns in IT. If the corresponding row of IT has not previously been updated (determined by a check of the logical matrix ITIND), it is then updated. If that row <u>has</u> been updated, try row I+1. If that row has been updated, try row I+2. This search process continues until either an "updatable" row is found or until the checking process circles back to row I. (This last comment implies, of course, that row NU+1 is considered to be row 1.) If no updatable row is found, nothing changes and the "intervisibility update" module has completed its task.

6. BOXES 10 and 11: THE PRODUCTION OF INTERMEDIATE OUTPUT

This section of code controls the production of intermediate error output as portrayed in figure 3, and which is discussed in Section 11.

Basically, this module's function is to convert the statistics accumulated in Box 6 to a form that is more meaningful for output. For example, sums of previous errors are divided by the appropriate number to obtain averages, coordinate error counts are made to produce quadrant counts, etc.

The frequency with which the intermediate error output is produced is controlled by an input parameter called K1C. The output summary is provided at the end of every K1C cycles.

7. BOX 12: UPDATE THE STATIONARY UNIT LIST

The stationary unit list (logical matrix FXD) is a list that contains "known" information regarding which units are moving and which units are not moving throughout the simulation. By "known" is meant that this information is available to the position location algorithms.

There are two types of real world scenarios that the matrix FXD has been instrumental in simulating. The first of these is the situation where units stop and start throughout the "battle", and send a signal back to the master computer indicating that they have just stopped or that they have just started moving. The second scenario is one in which certain units are "surveyed in" and never move, whereas no other units are ever treated as fixed. Knowledge of these facts is very helpful to the position location algorithms, but the "how" and the "why" of that helpfulness will not be discussed here. For that information, the reader is referred to Section 3 of the parent document to which this Working Paper is an appendix.

In the current version of WHERSM, the starting and stopping of units mentioned above is achieved by maintaining the matrix FXD, and by updating it in a random manner that is controlled by parameters which are input to the simulation. For each unit, FXD is "TRUE" if that unit is currently fixed, and "FALSE" if that unit is currently moving. The input parameter PSS(2) is the average time a unit remains fixed once it stops moving, while PSS(1) is the average time it keeps moving once it starts moving. These times are converted to probabilities, stored back into PSS(1) and PSS(2), and those probabilities in turn are used to determine, at each cycle, whether a unit should have its state ("fixed" or "moving") changed.

The determination of state changes is precisely the purpose of the "stationary list update" module. It is achieved by obtaining a uniformly distributed (0,1) pseudo-random number from the internal subroutine RU, and, if that number is less than the appropriate PSS probability value, changing the state of that particular unit.

It should be noted that in the second scenario mentioned above, one of course does not want the stationary unit list to be changed at all during the simulation. This can be guaranteed simply by letting the values of PSS(1) and PSS(2) at input time be very large numbers. It should also be noted that FXD is initialized at input time.

8. BOXES 13 and 14: PRODUCE THE X-Y PLOTS

This module is very small, consisting mainly of two calls to subroutine GRAPH. The first call produces x-y plots of the true positions of all the units, and the second call produces x-y plots of the estimated positions of all the units.

There are eight parameters in the call statement for subroutine GRAPH: N, X, Y, IWRD, XMAX, XMIN, YMAX, and YMIN. N is the number of points to be plotted, X and Y are vectors of the x and y coordinates of the points to be plotted, IWRD contains either 'TRUE' or 'EST.' depending on whether the points to be plotted are true or estimated positions, and XMAX, XMIN, YMAX, and YMIN are the upper and lower limits of the axes. Those last (XMAX through YMIN) are part of the input to the simulation program.

For a description of the plots that are produced, see Section 11.

The frequency with which the plots are produced is controlled by the input parameter K4C; the plots are produced after every K4C cycles.

9. BOX 15; THE END OF SIMULATION

Box 15 of the flowchart in figure 1 is the only one that does not correspond to an easily recognizable module in the WHERSM code. Rather, its function is performed by a set of IF statements scattered throughout the other modules. The IF statements are all identical; they test whether the number of cycles required to run the simulation for the length of time indicated by the input parameter SIMTIM have been completed. If so, then a final set of "intermediate" output is produced before termination.

10. THE INPUT TO THE SIMULATION

There are three types of input used in the WHERSM program: input from cards, tape input, and "hardwired" input in the form of DATA and PARAMETER statements inserted in the program deck. This section will discuss each type of input with regard to input values and formats. Card input is explained first.

The "Catch-All" Cards: These cards contain the various parameters and constants needed by the program; there are two of these cards, and the table below explains their contents. For a definition of the variables mentioned, see Appendix A.

	Columns	Format	Variable Name
Card No. 1:	1-5	I5	NU
	6-10	I5	NS
	11-15	I5	K1C
	16-20	I5	K2C
	21-25	I5	K3C
	26-30	I5	K4C
	31-35	I5	INR
	36-40	I5	NSP
	41-50	F10.5	PSS(1)
	51-60	F10.5	PSS(2)
	61-70	F10.5	BND
	71-80	F10.5	BNDSP
Card No. 2:	1-10	F10.5	CYCTM
	11-20	F10.5	SIMTIM
	21-30	F10.5	XMAX
	31-40	F10.5	XMIN
	41-50	F10.5	YMAX
	51-60	F10.5	YMIN

[illegible]

[illegible]

[illegible]

[illegible]	[illegible]	[illegible]	
[illegible]	[illegible]	[illegible]	[illegible]

The "Reporting Units" Cards: These cards contain the unit numbers of the units that will be used at the start of the simulation as reporters. More specifically, the elements of the matrix RPT that correspond to the numbers that appear on the Reporting Units Cards are the only elements that are initialized to the logical value "TRUE". All others are initialized to "FALSE". There are 20 numbers per card, each with a format of I4. The number of unit-numbers that appear on these cards is the value of INR (see Catch-all cards above). Consequently, there will be [(INR-1)/20] + 1 Reporting Units cards.

The "Fixed Units" Cards: These cards contain the unit numbers of the units that will be used at the start of the simulation as fixed units. The elements of the matrix FXD are initialized in the same manner as are the elements of the matrix RPT above. The format for each card again is 20I4, and there will be [(NSP-1)/20] + 1 of these cards.

The "Type" Cards: The type cards contain the type number for each unit in the simulation. There currently are four unit types:

Type Number	Type	Speed
1	Foot Soldier	1 1/2 - 2 km/h
2	Ground Vehicle	8-10 km/h
3	Low Performance Aircraft	161 km/h
4	High Performance Aircraft	322 km/h

There will be exactly NU of these type numbers each with a format of I4, and therefore [(NU - 1)/20] + 1 Type cards.

The "Movement" Cards: These cards contain all the information needed by the program so that it can provide for the movement of units. There will be one of these cards for each unit in the scenario, hence NU cards in all. Their contents are as follows:

[illegible]

[illegible]

[illegible]

[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	

[illegible]

Card Columns	Format	Meaning
1-3	I3	Unit number
4-21	3A6	Unit ID (anything)
		Starting latitude:
22-24	I3	Degrees
25-26	I2	Minutes
27-28	I2	Seconds
		Starting longitude:
29-31	I3	Degrees
32-33	I2	Minutes
34-35	I2	Seconds
37-40	F4.0	Time(*) (hrs.)
41-44	F4.0	Azimuth (deg.)
45-48	F4.0	Speed (km/h)
50-53	F4.0	Time(*) (hrs.)
54-57	F4.0	Azimuth (deg.)
58-61	F4.0	Speed (km/h)
63-66	F4.0	Time(*) (hrs.)
67-70	F4.0	Azimuth (deg.)
71-74	F4.0	Speed (km/h)

There are two kinds of tape input used in the WHERSM program. The first of these is the tape that contains the height information in the form of a digitized terrain map. The format of this tape is as appears in figure 2, but no further discussion of format will be given here since the terrain tape was obtained directly from the Electromagnetic Compatability

(*) By "time" is meant the length of time during which the immediately following azimuth and speed are used. Control is passed to the next set when "time runs out" for this set.

WORD	S1	S2	S3	S4	S5	S6	S1	S2	S3	S4	S5	S6	S1	S2	S3	S4	S5	S6
1	INT			INT			Latitude θ (min)			Longitude ϕ (min)								
2	INT			INT			Vertical Spacing S_v (sec)			Horizontal Spacing S_h (sec)								
3	INT	FLD	FLD	INT			Elevations per word	Security Class.	Source	Quantization Factor (QF)								
4	INT						Initial Bias ΔE (Feet)											
5	INT						Minimum Elevation (Feet)											
6	INT						Maximum Elevation (Feet)											
7	INT						Number of missing elevations			Blank								
8							Blank											
9							Blank											
10							Blank											
11							Blank											
12	All elevations are integer						$h_{1,1}$;-- $h_{2,1}$;-- $h_{3,1}$-- $h_{121,1}$;-- $h_{1,2}$;-- $h_{2,2}$;-- $h_{121,121}$											
							The length of the record depends on the number of elevations stored per word. This can be found in word 3. The maximum record size is 4892 words.											
	S1	S2	S3	S4	S5	S6	S1	S2	S3	S4	S5	S6	S1	S2	S3	S4	S5	S6

TOPOGRAPHIC DATA RECORD FORMAT

Figure 2

SOURCE: Clemmitt, M, op.cit.

Analysis Center which has published a report(*) that documents such tapes in detail. The terrain tape must be mounted on logical unit number 8.

The other tape used as input to WHERSM is one that is mounted on logical unit number 7 and contains the intervisibility snapshot matrices discussed in Sections 1 and 3. It is a binary tape, created as follows: Let ITB be a three-dimensional matrix that contains the set of "packed" intervisibility snapshot matrices;

$$\text{ITB}(I,J,1), \quad I = 1,2,\ldots,NU; \quad J = 1,2,\ldots, [(NU - 1)/36] + 1$$

is the snapshot matrix taken at the first discrete time value; ITB(I,J,2) is the matrix taken at the second discrete time value; etc. Then the tape should have been created using a write statement of the form:

```
WRITE (7)   (((ITB(I,J,K), J = 1, NITCOL), I = 1,NU), K = 1,K1)
```

where NITCOL = [(NU - 1)/36] + 1 (implying of course that 36 entries of each row of the 2 dimensional matrices are packed into one computer word), and K1 equals the number of snapshot matrices that exist.

The third and last type of input required to make a simulation run is the type that must physically be inserted in the program deck - the PARAMETER and DATA statement cards.

The table below lists these. The reader is referred to Appendix A for their meanings.

(*) Clemmitt, M. and Stone, R., "The Topographic Information System", Tech. Note No. ECAC-TN-007-222, June 1969, Electromagnetic Compatibility Analysis Center, Annapolis, Md.

Variable Name	Type of Statement Contained In	Routine Contained In
MNLAT	DATA	BUILD
MXLON	DATA	BUILD
TPOUT	DATA	WHERSM
IFREQ	DATA	WHERSM
CNX	DATA	HEIGHT
CNY	DATA	HEIGHT
W	DATA	WHERSM
FXDINI	DATA	WHERSM
L1	PARAMETER	WHERSM
L2	PARAMETER	WHERSM
L6	PARAMETER	BUILD
		WHERSM
		HEIGHT
L7	PARAMETER	BUILD
		WHERSM
		HEIGHT

11. THE OUTPUT FROM THE SIMULATION

Figure 3 contains a sample of the intermediate output produced by WHERSM. This section explains the contents of that output sample.

The first line of output contains the cycle number of that cycle after which the summary was produced, the simulation clock time, the average real clock time the program took to provide for the processing of one subcycle, and the average time taken by the algorithms to perform one location operation.

Each row of the table contains statistics for the unit whose unit number appears in column 1. Columns 2-7 contain statistics accumulated over <u>all</u> cycles previous to the current one. They are, in order:

the average error in the xy-plane,
the average error in the z coordinate,
the maximum error in the xy-plane,
the cycle during which that maximum error occurred,
the maximum z error that occurred, and
the cycle during which that maximum z error occurred.

Columns 8-11 contain statistics accumulated over the previous K1C cycles. Those statistics are: the average x-y error, the average z error, the maximum x-y error that occurred, and the cycle during which that error occurred.

Columns 12-14 contain the individual coordinate errors (estimated minus true) that occurred the last time each unit was located prior to the printing of the intermediate output. Columns 15, 16, and 17 list the current values of the S, RPT, and FXD vectors respectively; and column 18 gives the current value of the W vector, the weight.

The last row of the table, which begins with the word "ALL", gives for a column of averages the average of the numbers appearing above it; and for a column of maximums, the maximum of the maximums in the column.

CYCLE NUMBER 140. TIME = 4199.759 S. AVERAGE SIMULATION SUBCYCLE TIME = .0114 S. AVERAGE ALGORITHM SUBCYCLE TIME = .0063 S.

UNIT	ERROR STATS. OVER ALL 140 CYCLES XY AVG	Z AVG	XY MAX	CYC	Z MAX	CYC	ERROR STATS. OVER LAST 10 CYCLES XY AVG	Z AVG	XY MAX	CYC
1	.24414+01	.91294+00	.31908+01	25	.16242+01	34	.46277+01	.50339+00	.78275+01	135
2	.35464+01	.85762-01	.13054+02	25	.15279+00	38	.48220+01	.38292-01	.67530+01	137
3	.20537+01	.91748+00	.10124+02	105	.13243+01	111	.25380+01	.13243+01	.25380+01	131
4	.24157+01	.11534+02	.34763+01	137	.13297+02	116	.42995+01	.10892+02	.94783+01	137
5	.24718+01	.10172+01	.12040+02	99	.11822+01	28	.39787+01	.11395+01	.73695+01	137
6	.53024+00	.27290+01	.88017+01	122	.28855+01	1	.38415+01	.22898+01	.71701+01	132
7	.22493+01	.19100+00	.10557+02	118	.41177+00	09	.12492+01	.14232+00	.12492+01	131
8	.21109+01	.99590-01	.86535+01	58	.21060+00	04	.16679+01	.21060+00	.16679+01	131
9	.22773+01	.38277+01	.95356+01	108	.41295+01	85	.30588+01	.32576+01	.47633+01	133
10	.21455+01	.16734+01	.11402+02	48	.19608+01	67	.12905+01	.19608+01	.12905+01	131
11	.35134+01	.15303+01	.94926+01	123	.30024+01	136	.34411+01	.29655+01	.74786+01	131
12	.35254+01	.37031+01	.10452+02	6	.54271+01	140	.38779+01	.52943+01	.41314+01	131
13	.27175+01	.26127+01	.99780+01	99	.29795+01	118	.24089+01	.29795+01	.24089+01	131
14	.34774+01	.79410+01	.13946+02	30	.85953+01	98	.37038+01	.67913+01	.79043+01	139
15	.32230+01	.85750+00	.10452+02	121	.16539+01	1	.40113+01	.13546+01	.82947+01	136
16	.26355+01	.69727+00	.18338+02	104	.15900+01	1	.46227+01	.76429-01	.71994+01	136
17	.44461+01	.55555+00	.10144+02	129	.26535+01	140	.43296+01	.23610+01	.87303+01	136
18	.15675+01	.21635+01	.98921+01	136	.26927+01	140	.19873+01	.26083+01	.88921+01	136
19	.25402+01	.13782+01	.11559+02	33	.86025+01	1	.54770+01	.52087+01	.10985+02	133
20	.14833+01	.45086+00	.13031+02	101	.43675+01	135	.30819+01	.39596+01	.90679+01	132
21	.37867+01	.42597+01	.12970+02	76	.79752+01	71	.42395+01	.65663+01	.42395+01	131
22	.30509+01	.42597+01	.15526+02	53	.15755+02	13	.12305+01	.33493+01	.12305+01	131
23	.26150+01	.32765+01	.12393+02	74	.10188+02	77	.35549+01	.34079+01	.78397+01	137
24	.38342+01	.27763+01	.14701+02	131	.13077+02	73	.51618+01	.58671+00	.14701+02	131
25	.28212+01	.17163+01	.11560+02	113	.74766+01	62	.43317+01	.55639+00	.43318+01	131
26	.20503+01	.15245+01	.14317+02	139	.21374+01	20	.63410+01	.10169-01	.14317+02	138
27	.32551+01	.35603+01	.14652+02	105	.91286+01	14	.39969+01	.64362-01	.79851+01	137
28	.33505+01	.23907+01	.13544+02	33	.10227+02	25	.29465+01	.16676+01	.29465+01	131
29	.19225+01	.53512+01	.14575+02	73	.14732+02	140	.38722+01	.13182+02	.82939+01	140
30	[illegible]	.11586+01	.13351+02	130	.44768+01	135	.50031+01	.36662+01	.12265+02	135
31	.24528+01	.49557+01	.12022+02	101	.11916+02	103	.35341+01	.11557+02	.10252+02	138
32	.22023+01	.70161+01	.13918+02	35	.11451+02	39	.38284+01	.81999+01	.38284+01	131
33	[illegible]	.30115+01	.17199+02	70	.18947+02	77	.16223+01	.11483+02	.10166+02	140
34	.31122+01	.52590+01	.15504+02	112	.22580+02	118	.15980+01	.19463+02	.18385+01	131
35	.27434+01	.70917+00	.15032+02	139	.33704+01	54	.53590+01	.55589-01	.15939+02	138
36	.26132+01	.19790+01	.15338+02	15	.95032+01	140	.38359+01	.81762+01	.94795+01	135
37	.35031+01	.55150+01	.16954+02	24	.27189+02	140	.53173+01	.20046+02	.12370+02	139
38	.43751+01	.47546+01	.24436+02	71	.20980+02	67	.61602+01	.86331-01	.16866+02	139
39	.33527+01	.35762+01	.15018+02	95	.66419+01	86	.54237+00	.66419+01	.54237+00	131
40	.22145+01	.31754+00	.14432+02	134	.21461+01	136	.47178+01	.14505+01	.14432+02	134
ALL	.28795+01	.29773+01	.24436+02	71	.27189+02	140	.36377+01	.43893+01	.16866+02	139

UNIT	CURRENT COORDINATE ERRORS EX-TX	EY-TY	EZ-TZ	SRF	WEIGHT
1	.284+01	-.124+01	-.435+00	TFF	.14161-01
2	.264+01	-.363+01	-.561-01	TFF	.11573-01
3	.194+01	-.164+01	-.132+01	TTT	.25947-01
4	-.899+00	-.295+00	-.104+02	TFF	.16989-01
5	-.926+00	-.203+01	-.116+01	TFF	.24685-01
6	.170+01	.231+00	-.229+01	TFT	.25712-01
7	.138+00	-.124+01	-.142+00	TTT	.25724-01
8	.111+01	-.125+01	-.211+00	TTT	.27179-01
9	.288+01	-.123+01	-.325+01	TFT	.23854-01
10	.129+01	.101+00	-.196+01	TTT	.27391-01
11	.384+00	-.217+01	-.300+01	TFT	.20566-01
12	.353+01	-.694+00	-.543+01	TFF	.10270-01
13	.152+01	-.187+01	-.299+01	TTT	.26687-01
14	.420+01	-.394+01	-.702+01	TFF	.13082-01
15	.298+01	-.170+01	-.155+01	TFF	.10463-01
16	.409+01	-.107+01	-.691-01	TFF	.13824-01
17	.299+01	-.841+00	-.265+01	TFF	.17387-01
18	-.103+01	.164+01	-.269+01	TFF	.14294-01
19	.273+01	-.234+01	-.748+01	TFF	.16934-01
20	-.153+00	-.161+01	-.437+01	TTT	.25841-01
21	.421+01	.465+00	-.657+01	TTT	.27685-01
22	.114+01	-.467+00	-.335+01	TTT	.27451-01
23	.264+00	-.185+01	-.189+01	TFF	.83285-02
24	.464+01	-.882-01	-.379+00	TFF	.12657-01
25	.429+01	-.631+00	-.557+00	TTT	.27585-01
26	.392+01	-.625+01	-.465-01	TFF	.21695-01
27	-.125+01	-.356+01	-.504-01	TFF	.19767-01
28	.285+01	-.733+00	-.157+01	TTT	.27650-01
29	.178+00	-.392+01	-.147+02	TFF	.20970-01
30	.375+01	-.687+01	-.410+01	TFF	.11522-01
31	-.105+01	-.396+00	-.105+02	TFF	.16591-01
32	.354+01	-.146+01	-.820+01	TTT	.27611-01
33	.903+01	.467+01	-.109+02	TFF	.19610-01
34	.205+00	-.159+01	-.195+02	TTT	.27351-01
35	.245+01	-.301+01	-.220+00	TFF	.14553-01
36	.271+01	-.275+01	-.950+01	TFF	.16609-01
37	-.105+01	-.222+01	-.255+02	TFF	.11397-01
38	-.618+00	-.200+01	-.450+00	TFF	.11193-01
39	-.387+00	-.380+00	-.664+01	TTT	.27683-01
40	.274+01	-.218+01	-.882+00	TFF	.22936-01

QUADRANT COUNTS OF ERRORS

1 4
0
8 27

The next item of output is the quadrant counts of errors. This is a display of the directions of errors. It is obtained from columns 12 and 13 of the table by counting the number of entries in those columns (by their algebraic signs) that fall in each quadrant.

The last items of output are the xy-plots, an example of which is given in figure 4. The two lines at the top of the graph state whether the graph is of true or estimated positions, list the unit numbers of the units that appear on that graph, and list the plotting symbols used to represent those units. The letters of the alphabet are used as plotting symbols, so that if more than 26 units are in the simulation, a separate graph will be produced for every 26 (or fraction thereof) units. If more than one unit occupies the same graph position, a number will appear in that graph position indicating the number of units in that position. The axis ranges are controlled by XMAX, XMIN, YMAX, and YMIN, which are part of the input to the simulation, and if points fall beyond those boundaries an annotation to the graph will appear indicating the occurrence of that situation.

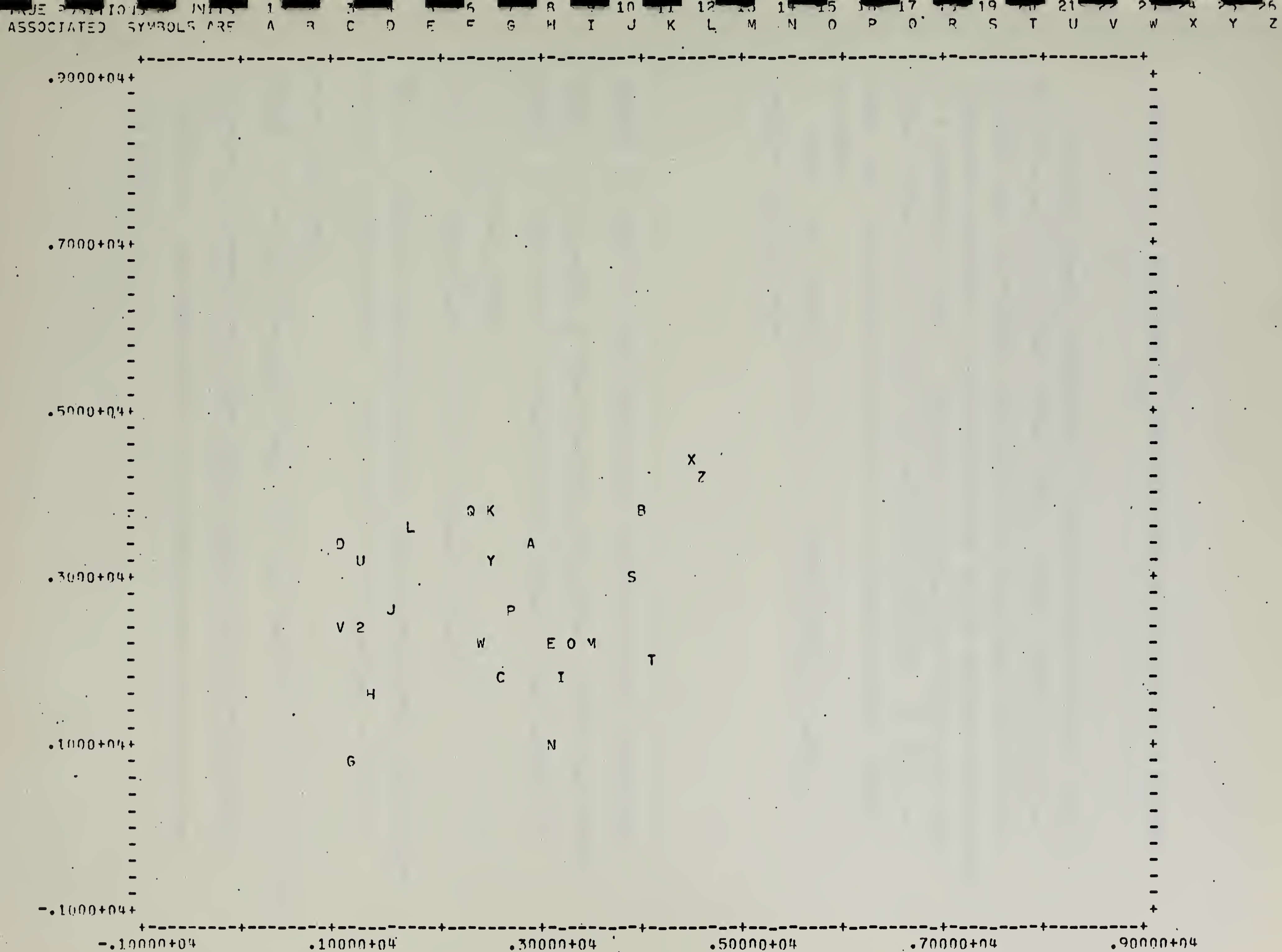


Figure 4: Example of XY-Plots

APPENDIX A: A DICTIONARY OF VARIABLES USED IN THE SIMULATION PROGRAMS

In this appendix, definitions of all non-trivial variables used in the three main simulation routines (BUILD, WHERSM, HEIGHT) are given. In each case, the variable being defined is underlined, and if it is a dimensioned variable, it is written exactly as that variable should appear in a DIMENSION statement. Immediately following the variable name will appear either a B,W, or H (or any combination thereof). By this is meant that the variable being defined is a variable of BUILD, WHERSM or HEIGHT.

Note: All variables conform to the FORTRAN implicit naming conventions regarding mode.

* * * * *

<u>AVASBT</u> W. The sum of real times used to perform position estimations.

<u>AVKTR</u> W. The number of time intervals accumulated in AVASBT and AVSUBT.

<u>AVSUBT</u> W. The sum of real times used to simulate a "real world" subcycle (includes AVASBT).

<u>AZ(4)</u> W. Buffer used in reading in movement azimuths.

<u>BND</u> W. Boundary beyond which units are not allowed to move. Unit of measurement is the kilometer. If this option is not desired, set BND to some large number.

<u>BNDSP</u> W. On input, the speed (km/h) with which BND moves in both directions. Immediately converted to a value that is added to BND at every subcycle.

<u>C(14641)</u> B. An output buffer that contains decoded elevations above sea level.

<u>CLK</u> W. The simulation's clock. Contains simulated time in seconds.

<u>CNX</u> H. The inverse of the average longitudinal distance (in meters) between grid points on the terrain tape.

<u>CNY</u> H. Same as CNX but for latitude.

<u>CT(4)</u> W. Input buffer used to read in the length of time a unit is to move according to a particular azimuth and speed.

<u>CYCTM</u> W. The "real time" length of a cycle in seconds.

<u>D(14641)</u> B. Output buffer that contains decoded altitudes above sea level.

<u>DELTA</u> W. "Real time" length of a subcycle.

<u>DX(L1)</u> W. List used to store the difference in the x-coordinates of the true and estimated positions. Computed immediately after a unit has had its position estimated.

<u>DY(L1)</u> W. Same as DX, but for y-coordinates.

<u>DZ(L1)</u> W. Same as DX, but for z-coordinates.

<u>EP(L1,14)</u> W. Contains estimated positions for each unit, in addition to some data relevant to those positions. EP(I,1-3) is the estimated position of unit I which was determined the next-to-last time it served as locatee; EP(I,4-6), the last time it served as locatee. The rest of the EP matrix is used to manipulate positions when the relocation option is used (see Working Paper No. 8).

<u>ERR</u> W. The error in position estimation in the xy-plane and then in the z-coordinate. The Euclidean distance between true (TP) and estimated (EP) positions is used as the error.

IFY H. Same as IFX, but for the y-coordinate. Units of measurement here are 3 seconds of latitutde.

IGP W. Pointer to the unit with the largest xy-error over all cycles.

IGT W. Pionter to the unit with the largest xy-error over the last K1C cycles.

INR W. Initial number of units allowed to report.

IP W. Pointer used in constructing VV1.

IPX H. The x-coordinate of some other position (for which a height is already available), after it has been converted to the units of the terrain.

IPY H. Same as IPX, but for y-coordinate.

IQ0 W. Number of units whose estimated position is exact.

IQ1 W. The number of units whose estimated position is in error in the direction of the first quadrant.

IQ2 W. Same as IQ1, but for quadrant 2.

IQ3 W. Same as IQ1, but for quadrant 3.

IQ4 W. same as IQ1, but for quadrant 4.

IQUAN B. Quantization factor for the block of terrain grid points currently being decoded. (See ECAC-TN-007-222, op.cit.)

IR W. Counter for the number of ranges computed in a particular subcycle.

IT(L1,L3) W. The intervisibility table currently being used.

ITB(L1,L3,7) W. The set of all intervisibility tables.

ITIND(L1) W. Logical matrix which indicates for each row of IT, whether that row has been updated yet.

IU W. The unit number of the unit currently serving as locatee.

IV W. The (0-1) answer to the question of existence of an LOS between the pair of units in question.

IVMAT W. Pointer to the intervisibility matrix currently being used to update IT.

IZT W. Pointer to the unit with the largest z-error over all cycles.

K1C W. The number of cycles after which intermediate output is to be produced.

K1CC W. Counter used with K1C.

K2C W. The number of cycles after which an intervisibility update is to occur.

K2CC W. Counter used with K2C.

K3C W. The number of rows of IT to be updated when an intervisibility update occurs.

K4C W. The number of cycles after which xy plots are to be produced.

K4CC W. Counter used with K4C.

KUF(L1) W. For I = 1,2,...,NS, KUF(I) is the unit number of the unit which is to serve as locatee during the Ith subcycle.

KZ W. The seed value for the internal uniform psuedo-random number generator, RU.

L B. Status parameter for I/O processor NTRAN. L equals -1 until the completion of the I/O command.

L1 W. Parameter variable used in dimensioning. L1 must be greater than or equal to NU.

L2 W. Parameter variable used in dimensioning. L2 $\geq$ NS.

L3-L5 W. Parameter variables used in dimensioning, but since they are strictly dependent on L1, they are automatically set.

L6 B,W,H. Conceptually organize all the 6' x 6' terrain blocks on the terrain tape according to the latitude-longitude of their southwest corner. L6, then, is the number of these blocks when counting from west to east.

L7 B,W,H. Same as L6, but count from south to north.

LOCATR(L1) W. The unit numbers of the units whose ranges are being reported in a given subcycle.

M W. Status parameter for I/O processor. See L.

M1 B. Status parameter for I/O processor. See L.

M2 B. Status parameter for I/O processor. See L.

MAXCYC W. The length of the simulation in terms of cycles.

MNLAT B. The minimum latitude (in seconds) for which a height is available on the terrain tape.

MXLON B. The maximum longitude (in seconds) for which a height is available on the terrain tape.

NBITS B. The number of bits into which the elevation in question is "packed".

NCYC W. The cycle number of the cycle currently being processed.

NDX(L6,L7) B,H. An index to terrain blocks that are written on the drum by BUILD. NDX(1,1) is the position of the 6' x 6' terrain block whose southwest corner is identified by (MNLAT, MXLON); NDX(1,2) is the position of block (MNLAT, MXLON + 3600); NDX(2,1) is block (MNLAT + 3600, MXLON); etc.

NITCOL W. The number of computer words in a packed row of the intervisibility matrices.

NS W. The number of subcycles in a cycle.

NSP W. The number of units which, at the start of the simulation, will be identified as being stationary.

NTYPE(4) W. The unit type of each unit. If NTYPE(I) is 1,2,3, or 4, then unit I is respectively a foot soldier, a ground vehicle, a low performance aircraft, or a high performance aircraft.

NU W. The number of units in the simulation.

NUMEL B. The number of elevations per word for a given terrain block represented by a logical record on the terrain tape.

PSS(2) W. At input time, PSS(1) is the average time a unit keeps moving once it starts moving, and PSS(2) is the average time it remains fixed once it stops. Subsequently, they are the probabilities of changing state. If no random starting and stopping is desired, set both PSS(1) and PSS(2) to very large numbers on the catch-all-cards.

RADS W. Constant representing the number of degrees in a radian.

RANGE(L1) W. The ranges being reported for a given subcycle.

RCON W. Constant used in converting a speed and azimuth combination into direction cosines.

RPT(L1) W. Logical matrix that indicates for each unit, whether or not that unit is a unit that is allowed to report ranges.

S(L1) W. Logical matrix that indicates for each unit whether or not that unit was "successfully" located the last time a position estimation was attempted. S is set by the position location algorithms.

SIMTIM W. The "real time" length of the simulation, in minutes.

SP(4) W. Input buffer used in reading in speeds for each unit.

STAT1(L1,10) W. For each row: column 1 is the sum of position estimation errors (in xy-plane) over all cycles for the unit associated with that row; column 2, the maximum xy-error that occurred over all cycles for that unit; column 3, the cycle number of the cycle during which that maximum error occurred; 4, the partial sum of xy-errors over the current K1C cycles; 5, the maximum xy-error over the current K1C cycles; 6, the cycle number associated with that error; 7, the sum of the z-errors over all cycles for that unit; 8, the maximum z-error over all cycles; 9, the cycle number associated with that z-error; and 10, the partial sum of z-errors over the current K1C cycles.

STAT2(L1,2) W. Column 1 contains, for each unit, the number of times that unit was "successfully" located since the start of the simulation; and column 2, the number since the start of the current K1C cycles.

TLOC(L1,4) W. The clock time at which the estimated positions in EP were obtained.

TP(L1,3) W. The current true position of each unit. TP(I,1) is the x-coordinate; TP(I,2), the y-coordinate; and TP(I,3), the z-coordinate.

TPOUT W. Logical variable that allows for the writing of all intermediate output on logical unit 9, in addition to the standard output unit.

T1-T4 W. Parameters in calls to the system's clock. Used in determining AVASBT and AVSUBT.

VV(L1,4) W. Currently used velocity vector for each unit. Column 1 is the x component; column 2, the y component; 3, the time at which that vector should no longer be used; and 4, a pointer into VV1 where the next velocity vector to be used resides.

VV1(L5,4) W. The set of all velocity vectors for all units. The format of each vector is the same here as in VV, but the h ($h \leq 3$ currently) vectors for unit number 1 are in rows 1 through h of VV1, followed by the vectors for unit number 2, etc.

W(L1) W. Vector used by the position location algorithms to weight the various estimates (see Working Papers No. 7 and 8). W should be initialized to the weight to be assigned to non-fixed units at the start of the simulation.

XMAX, XMIN, YMAX, YMIN W. The upper and lower bounds of the x-and y-axes (in meters) to be used in producing the xy-plots.

ZPSUM W. The sum of the position estimation errors (in the z-coordinate) over all units over the last K1C cycles.

ZTSUM W. The sum of the position estimation errors (in z-coordinate) over all units over all cycles.

APPENDIX B: LISTING OF THE SIMULATION PROGRAM

```
@ MAP ZAP,ZAP                                         01 JUN 72        10:55:40.867

    1.      CHN 1
    2.      SEG BUILD

    3.      CHN 2
    4.      SEG WHERSM

MAP LCC 1104 0036
```

```
@IT FOR BUILD,BUILD                                                                    01 JUN 72          10:55:41.23
UNIVAC 1108 FORTRAN V LEVEL   2206 0018 F5018P
THIS COMPILATION WAS DONE ON 01 JUN 72 AT 10:55:41


  MAIN PROGRAM

  STORAGE USED (BLOCK, NAME, LENGTH)

       0001    *CODE    000475
       0000    *DATA    114647
       0002    *BLANK   000073


  EXTERNAL REFERENCES (BLOCK, NAME)

       0003    NTRAN
       0004    CHAIN
       0005    NWDU$
       0006    NIO2$
       0007    NSTOP$


  STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

   0000   114604 1F        0001   000021 10L      0001   000334 100L     0001   000035 127G     0001   000123 151G
   0000   114613 2F        0001   000347 200L     0001   000362 210L     0001   000266 213G     0001   000372 220L
   0001   000410 230L      0001   000420 240L     0000   114622 3F       0001   000031 30L      0001   000112 50L
   0001   000161 68L       0001   000171 70L      0001   000255 80L      0001   000442 900L     0001   000452 910L
   0001   000462 920L      0001   000471 930L     0001   000324 98L      0000 R 023420 C        0002 R 000002 CNX
   0002 R 000003 CNY       0000 R 060101 D        0000 I 114566 I        0000 I 000000 IA       0000 I 011610 IB
   0000 I 114574 IBIAS     0000 I 114577 IBIT     0000 I 114601 IEL      0000 I 114573 IQUAN    0000 I 114562 I$
   0000 I 114571 J         0000 I 114564 K        0000 I 114565 L        0000 I 114567 LAT      0000 I 114570 LON
   0000 I 114600 M         0002 I 000000 MNLAT    0002 I 000001 MXLON    0000 I 114602 M1       0000 I 114563 M2
   0000 I 114576 N         0000 I 114575 NBITS    0002 I 000004 NDX      0000 I 114572 NUMEL    0000 R 114603 T


00101     1*          PARAMETER L6=5,L7=11
00103     2*          PARAMETER L8=L6*L7
00104     3*          COMMON MNLAT,MXLON,CNX,CNY,NDX(L6,L7)
00105     4*          DIMENSION IA(5000),IB(5000),C(14641),D(14641)
00106     5*          DATA NDX/L8*-1/
00110     6*          DATA MNLAT,MXLON/2544,4302/
00113     7*          CALL NTRAN(8,22)
00114     8*          CALL NTRAN(35,22)
00115     9*          M2=0
00116    10*          K=0
00116    11*   C            READ FIRST RECORD.
00117    12*          CALL NTRAN(8,2,5000,IA,L)
00117    13*   C            WAIT FOR END OF TAPE READ.
00120    14*        10 IF (L+1) 20,10,30
00120    15*   C            IF ERROR, 900.  IF EOF, 200.
00123    16*        20 IF (L+2) 900,200,30
00126    17*        30 DO 40 I=1,L
```

```
00131           IB(I)=IA(I)
00131      C         INITIATE THE NEXT TAPE READ.
00133            CALL NTRAN(8,2,5000,IA,L)
00133      C         DECODE LATITUDE AND LONGITUDE OF FIRST TERRAIN BLOCK.
00134            LAT=FLD(0,18,IB(3))
00135            LON=FLD(18,18,IB(3))
00135      C         COMPUTE ITS RELATIVE POSITION IN THE TERRAIN AND SET INDEX
00135      C         TO THE DRUM.
00136            I=(LAT-MNLAT)/6+1
00137            J=(MXLON-LON)/6+1
00140            NDX(I,J)=K*14641
00140      C       . DECODE NUMBER OF ELEVATIONS PER WORD, QUANTIZATION FACTOR,
00140      C         BIAS, AND NUMBER OF BITS PER ELEVATION.
00141            NUMEL=FLD(0,6,IB(5))
00142            IQUAN=FLD(18,18,IB(5))
00143            IBIAS=IB(6)
00144            NBITS=36/NUMEL
00145            N=14
00146            I=1
00146      C         DECODE ELEVATIONS IN THE FIRST TERRAIN BLOCK.
00147         50 IBIT=0
00150            DO 60 M=1,NUMEL
00153            IEL=FLD(IBIT,NBITS,IB(N))
00154            C(I)=(IEL*IQUAN+IBIAS)*.3048
00155            I=I+1
00156            IF (I.EQ.14642) GO TO 68
00160         60 IBIT=IBIT+NBITS
00162            N=N+1
00163            GO TO 50
00163      C         WAIT FOR CHANCE TO WRITE OUT THIS TERRAIN BLOCK.
00164         68 IF (M2+1) 69,68,70
00167         69 IF (M2+2) 910,920,70
00167      C         WRITE IT OUT.
00172         70 CALL NTRAN(35,1,14641,C,M1)
00172      C         BEGIN PROCESSING SECOND TERRAIN BLOCK.  (PROCEDURE IS THE
00172      C         SAME AS FOR THE THE FIRST BLOCK.)
00173            K=K+1
00174            N=IB(2)+4
00175            LAT=FLD(0,18,IB(N))
00176            LON=FLD(18,18,IB(N))
00177            I=(LAT-MNLAT)/6+1
00200            J=(MXLON-LON)/6+1
00201            NDX(I,J)=K*14641
00202            N=N+2
00203            NUMEL=FLD(0,6,IB(N))
00204            IQUAN=FLD(18,18,IB(N))
00205            IBIAS=IB(N+1)
00206            NBITS=36/NUMEL
00207            N=N+9
00210            I=1
00211         80 IBIT=0
00212            DO 90 M=1,NUMEL
00215            IEL=FLD(IBIT,NBITS,IB(N))
00216            D(I)=(IEL*IQUAN+IBIAS)*.3048
00217            I=I+1
00220            IF (I.EQ.14642) GO TO 98
00222         90 IBIT=IBIT+NBITS
00224            N=N+1
```

```
00225             GO TO 80
00226          98 IF (M1+1) 99,98,100
00231          99 IF (M1+2) 910,920,100
00234         100 CALL NTRAN(35,1,14641,D,M2)
00235             K=K+1
00236             GO TO 10
00236       C            GOOD RUN SO FAR.  READ THE TWO DECODED RECORDS AT END OF TAPE
00237         200 CALL NTRAN(8,22)
00240             CALL NTRAN(8,2,14641,C,L)
00241         210 IF (L+1) 211,210,220
00244         211 IF (L+2) 900,900,220
00247         220 CALL NTRAN(8,2,14641,D,L)
00250             CALL NTRAN(35,1,14641,C,M1)
00251         230 IF (L+1) 231,230,240
00254         231 IF (L+2) 900,900,240
00257         240 CALL NTRAN(35,1,14641,D,M1)
00260             NDX(3,5)=K*14641
00261             NDX(3,6)=(K+1)*14641
00261       C            GOOD RUN.  GO CHAIN IN THE SIMULATION.
00262             CALL CHAIN(2,T)
00262       C            ERROR HAS OCCURRED.  GIVE MESSAGE AND ABORT.
00263         900 WRITE(6,1) LAT,LON
00267             GO TO 930
00270         910 WRITE(6,2) LAT,LON
00274             GO TO 930
00275         920 WRITE(6,3) LAT,LON
00301         930 STOP
00302           1 FORMAT(/' BAD TAPE READ.  LAST READ WAS'2I10)
00303           2 FORMAT(/' BAD DRUM WRITE.  CURRENT IS'2I10)
00304           3 FORMAT(/' EOF ON DRUM.  CURRENT IS'2I10)
00305             END

     END OF UNIVAC 1108 FORTRAN V COMPILATION.     0 *DIAGNOSTIC* MESSAGE(S)
PHASE 1 TIME =   0 SEC.
PHASE 2 TIME =   0 SEC.
PHASE 3 TIME =   1 SEC.
PHASE 4 TIME =   0 SEC.
PHASE 5 TIME =   0 SEC.
PHASE 6 TIME =   0 SEC.

TOTAL COMPILATION TIME =   1  SEC
```

```
@IT FOR WHERSM,WHERSM                                                     01 JUN 72        10:55:42.867
UNIVAC 1108 FORTRAN V LEVEL   2206 0018 F5018P
THIS COMPILATION WAS DONE ON 01 JUN 72 AT 10:55:42

   MAIN PROGRAM

   STORAGE USED (BLOCK, NAME, LENGTH)

        0001    *CODE    003140
        0000    *DATA    051716
        0002    *BLANK   000004
        0003    CB1      013000

   EXTERNAL REFERENCES (BLOCK, NAME)

        0004    HEIGHT
        0005    GRAPH
        0006    CLOCKS
        0007    HRG
        0010    RG
        0011    FINDIT
        0012    NRDU$
        0013    NIO1$
        0014    NIO2$
        0015    NWDU$
        0016    COS
        0017    NWBU$
        0020    NRBU$
        0021    SQRT
        0022    NWEF$
        0023    NSTOP$
        0024    NERR3$

   STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

    0000    051263 1F        0000    051333 10F       0001    001201 1000L     0001    002140 1015G     0001    001302 1020L
    0001    002255 1063G     0000    051342 11F       0001    001357 1110L     0001    001375 1120L     0001    001415 1130L
    0001    001433 1140L     0001    002530 1177G     0000    051363 12F       0001    001433 1200L     0001    001503 1260L
    0001    001511 1290L     0000    051366 13F       0001    002762 1313G     0001    003000 1326G     0000    051474 14F
    0001    000031 143G      0000    051507 15F       0000    051511 16F       0000    051535 17F       0001    000105 176G
    0000    051566 18F       0000    051275 2F        0000    051573 20F       0001    001514 2010L     0001    001646 2100L
    0001    000162 217G      0001    000201 227G      0001    000207 234G      0001    000316 240L      0001    000230 245G
    0001    000424 250L      0001    000247 255G      0001    000444 260L      0001    000255 262G      0001    000273 272G
    0000    051304 3F        0000    051624 30F       0001    000312 302G      0001    000334 310G      0001    000341 315G
    0001    000401 327G      0001    000467 353G      0001    000504 363G      0001    000524 376G      0001    000636 390L
    0000    051310 4F        0001    001737 4020L     0001    001761 4040L     0001    000663 422G      0001    000655 425L
    0001    000701 430L      0001    000775 454G      0001    000747 460L      0001    001021 467G      0001    000754 470L
    0001    001032 476G      0001    001774 4900L     0000    051311 5F        0001    001040 500L      0001    002042 5020L
    0001    002051 5040L     0001    002103 5050L     0001    002134 5060L     0001    001053 512G      0001    001056 514G
    0001    002127 5200L     0001    002171 5210L     0001    002300 5220L     0001    001100 523G      0001    002553 5236L
    0001    002560 5237L     0001    002565 5238L     0001    002571 5239L     0001    002575 5240L     0001    002601 5241L
    0001    002605 5242L     0001    002610 5250L     0001    001112 527G      0001    001126 531G      0001    001157 540G
    0001    001160 543G      0001    002774 5500L     0001    003031 5550L     0001    001221 556G      0001    003041 5610L
```

```
0000   051315 7F        0001   002022 755G      0001   002037 762G      0001   [illegible]500 [illegible]G   0001   001[illegible] 673G
0000   051324 9F        0001   003072 9000L     0000 R 051233 AVASBT    0001   002064 776G      0000   051323 8F
0000 R 004201 AZ        0000 R 051157 BND       0000 R 051160 BNDSP     0000 R 051230 AVKTR     0000 R 051232 AVSUBT
0002 R 000003 CNY       0000 R 004204 CT        0000 R 051161 CYCTM     0003 R 011531 CLK       0002 R 000002 CNX
0000 R 022252 DY        0000 R 022613 DZ        0003 R 000000 EP        0000 R 051167 DELTA     0000 R 021711 DX
0000 R 051146 FXDINI    0000 R 051237 GPKTR     0000 R 051235 GPSUM     0000 R 051227 ERR       0003 L 011535 FXD
0000 I 051175 I         0000 I 023154 IFREQ     0000 I 051243 IGP       0000 R 051236 GTKTR     0000 R 051234 GTSUM
0000 I 051205 IP        0000 I 051177 IPOS      0000 I 051247 IQ0       0000 I 051242 IGT       0000 I 051155 INR
0000 I 051252 IQ3       0000 I 051253 IQ4       0000 I 051223 IR        0000 I 051250 IQ1       0000 I 051251 IQ2
0000 L 050601 ITIND     0003 I 011530 IU        0000 I 051224 IV        0000 I 000000 IT        0000 I 023160 ITB
0000 I 051143 I$        0000 I 051200 I1        0000 I 051202 I2        0000 I 051173 IVMAT     0000 I 051244 IZT
0000 I 051176 J         0000 I 051206 J1        0000 I 051211 J12       0000 I 051204 I3        0000 I 051217 I9
0000 I 051212 K         0000 I 003047 KUF       0000 I 051144 KZ        0000 I 051207 J2        0000 I 051210 J3
0000 I 051214 K1CC      0000 I 051152 K2C       0000 I 051215 K2CC      0000   051222 K1        0000 I 051151 K1C
0000 I 051216 K4CC      0003 I 006116 LOCATR    0000 I 051174 M         0000 I 051153 K3C       0000 I 051154 K4C
0002 I 000001 MXLON     0000 I 051213 NCYC      0000 I 051172 NITCOL    0000 I 051170 MAXCYC    0002 I 000000 MNLAT
0003 I 012437 NTYPE     0000 I 051147 NU        0000 R 004177 PSS       0000 I 051150 NS        0000 I 051156 NSP
0000 R 051171 RCON      0000 R 051203 REM       0003 L 012076 RPT       0000 R 051145 RADS      0003 R 006460 RANGE
0003 L 007022 S         0000 R 051162 SIMTIM    0000 R 004207 SP        0001 R 003130 RU        0000 R 051254 RU
0003 R 007363 TLOC      0000 R 011426 TP        0000 L 051142 TPOUT     0000 R 005114 STAT1     0000 R 004212 STAT2
0000 R 051225 T3        0000 R 051226 T4        0000 R 012671 VV        0000 R 051220 T1        0000 R 051231 T2
0000 R 051201 X         0000 R 051163 XMAX      0000 R 051164 XMIN      0000 R 014475 VV1       0003 R 011167 W
0000 R 051165 YMAX      0000 R 051166 YMIN      0003 R 011533 YNEW      0003   011532 XNEW      0000 R 051221 Y
0000 R 051240 ZTSUM     0000 R 051245 Z1        0000 R 051246 Z2        0003 R 011534 ZNEW      0000 R 051241 ZPSUM

00101          PARAMETER L1=225,L2=600
00103          PARAMETER L3=L1/36+1,L4=L1+1,L5=3*L1
00104          DIMENSION IT(L1,L3),KUF(L2),PSS(2),AZ(3),CT(3),SP(3),STAT2(L1,2)
00105          DIMENSION STAT1(L1,10),TP(L1,3),VV(L1,4),VV1(L5,4),DX(L1)
00106          DIMENSION DY(L1),DZ(L1),IFREQ(4),ITB(L1,L3,7),ITIND(L1)
00107          LOGICAL S,FXD,RPT,TPOUT,ITIND
00110          COMMON MNLAT,MXLON,CNX,CNY
00111          COMMON /CB1/EP(L1,14),LOCATR(L4),RANGE(L4),S(L1),TLOC(L1,4),W(L1),
00111         *      IU,CLK,XNEW,YNEW,ZNEW,FXD(L1),RPT(L1),NTYPE(L1)
00112          DATA TPOUT/.FALSE./
00114          DATA IFREQ/1,6,30,60/
00116          DATA KZ,RADS/3141,57.295779/
00121          DATA FXD,RPT,S/L1*.FALSE.,L1*.FALSE.,L1*.TRUE./
00125          DATA W/L1*.028/
00127          DATA FXDINI/1.0/
00127    C          INPUT THE CATCH-ALL-CARDS.
00131          READ(5,3) NU,NS,K1C,K2C,K3C,K4C,INR,NSP,PSS,BND,BNDSP,
00131         *        CYCTM,SIMTIM,XMAX,XMIN,YMAX,YMIN
00157          DELTA=CYCTM/FLOAT(NS)
00160          MAXCYC=(60.0*SIMTIM+DELTA)/(FLOAT(NS)*DELTA)
00161          WRITE(6,13) NU,NS,K1C,K2C,K3C,K4C,INR,NSP,SIMTIM,MAXCYC,
00161         *        DELTA,PSS,BND,BNDSP
00204          PSS(1)=CYCTM/(PSS(1)*60.0)
00205          PSS(2)=CYCTM/(PSS(2)*60.0)
00206          RCON=DELTA/3600.0*1000.0
00207          BND=1000.0*BND
00210          BNDSP=BNDSP*RCON
00211          SIMTIM=60.0*SIMTIM
00212          NITCOL=(NU-1)/36+1
```

```
                T=2
00214          M=1
00214   C           INPUT THE INITIAL REPORTING UNITS, AND SET UP RPT.
00215          READ(5,4) (KUF(I),I=1,INR)
00223          WRITE(6,7)
00225          WRITE(6,8)(KUF(I),I=1,INR)
00233          DO 200 I=1,INR
00236          J=KUF(I)
00237      200 RPT(J)=.TRUE.
00237   C           INPUT THE INITIAL FIXED UNITS, AND SET UP FXD.
00241          IF (NSP.EQ.0) GO TO 240
00243          READ(5,4) (KUF(I),I=1,NSP)
00251          WRITE(6,9)
00253          WRITE(6,8)(KUF(I),I=1,NSP)
00261          DO 210 I=1,NSP
00264          J=KUF(I)
00265.         W(J)=FXDINI
00266      210 FXD(J)=.TRUE.
00266   C           INPUT THE UNIT TYPES.
00270          READ(5,4) (NTYPE(I),I=1,NU)
00276          WRITE(6,18)
00300          WRITE(6,8) (NTYPE(I),I=1,NU)
00300   C           MAKE THE UNIT TO SLOT ASSIGNMENTS.
00306      240 IPOS=0
00307          DO 241 I=1,NS
00312      241 KUF(I)=0
00314          DO 270 I=NU,1,-1
00314   C           COMPUTE NUMBER OF SUBCYCLES BETWEEN LOCATION OPERATIONS FOR
00314   C           THIS UNIT.
00317          I1=NTYPE(I)
00320          I1=IFREQ(I1)
00321          X=FLOAT(NS)/FLOAT(I1)
00322          I2=X
00323          X=X-FLOAT(I2)
00324          REM=0.0
00325          EP(I,14)=FLOAT(NS)/(5.*FLOAT(I1))
00326          DO 260 I3=1,I1
00331          REM=REM+X
00331   C           FIND SLOT TO ASSIGN.  IF SLOT FIRST TRIED IS NOT AVAILABLE,
00331   C           TRY ADJACENT SLOTS UNTIL AN AVAILABLE ONE IS FOUND.
00332          IPOS=IPOS+I2+INT(REM)
00333          IF (IPOS.GT.NS) IPOS=IPOS-NS
00335      250 IF (KUF(IPOS).EQ.0) GO TO 260
00337          IPOS=IPOS+1
00340          IF (IPOS.GT.NS) IPOS=IPOS-NS
00342          GO TO 250
00343      260 KUF(IPOS)=I
00345      270 CONTINUE
00347          WRITE(6,10)
00351          WRITE(6,8)(KUF(I),I=1,NS)
00351   C           INPUT MOVEMENT DATA AND STARTING POSITIONS.
00357          WRITE(6,11)
00361          IP=1
00362          DO 500 I=1,NU
00365          READ(5,30) IU,I1,I2,I3,J1,J2,J3,(CT(J),AZ(J),SP(J),J=1,3)
00365   C           CONVERT DEGREES,MINUTES, AND SECONDS TO SECONDS.
00404          I1=(I1*60+I2)*60+I3
00405          J1=(J1*60+J2)*60+J3
```

```
      TP(IU,1)=(MXLON*60-J1)/(3.0*CNX)
      TP(IU,2)=(I1-MNLAT*60)/(3.0*CNY)
      IF (NTYPE(IU).LT.3) GO TO 390
C           AIRCRAFT HAVE A RANDOM ALTITUDE ASSIGNED TO THEM.
      IF (NTYPE(IU).EQ.3) TP(IU,3)=100.+5000.*RU(KZ)
      IF (NTYPE(IU).EQ.4) TP(IU,3)=100.+4000.*RU(KZ)
      GO TO 425
C           INITIATE HEIGHT RETRIEVAL.
  390 CALL HEIGHT(-1.,-1.,TP(IU,1),TP(IU,2),TP(IU,3),M)
C           CONVERT MOVEMENT DATA INTO DIRECTION COSINES AND BUILD
C           THE VELOCITY VECTORS.
  425 VV(IU,4)=IP
      DO 450 J=1,3
      IF (CT(J).LE.0.0) GO TO 460
      CT(J)=3600.*CT(J)
      IF (J.EQ.1) GO TO 430
      CT(J)=CT(J)+CT(J-1)
  430 VV1(IP,1)=COS((AZ(J)-90.0)/RADS)*SP(J)*RCON
      VV1(IP,2)=COS(AZ(J)/RADS)*SP(J)*RCON
      VV1(IP,4)=IP+1
      VV1(IP,3)=CT(J)
      IP=IP+1
      IF (CT(J).GT.SIMTIM) GO TO 460
  450 CONTINUE
      J=3
  460 VV1(IP-1,4)=-5.0
      J12=J
C           WAIT FOR END OF HEIGHT RETRIEVAL.
  470 IF (M.EQ.-1) GO TO 470
      WRITE(6,12) IU,NTYPE(IU),(TP(IU,J),J=1,3),AZ(1),SP(1),CT(1)
      IF (J12.EQ.1) GO TO 500
      WRITE(6,6) (AZ(J),SP(J),CT(J),J=2,J12)
C           INITIALIZE EP, THE ESTIMATED POSITIONS MATRIX.
      DO 480 J=1,3
      EP(IU,J)=TP(IU,J)
      EP(IU,J+3)=TP(IU,J)
      EP(IU,J+6)=TP(IU,J)
  480 EP(IU,J+9)=TP(IU,J)
  500 CONTINUE
      IF (TPOUT) WRITE(9) ((TP(I,J),J=1,3),NTYPE(I),I=1,NU)
C           INPUT THE INTERVISIBILITY TABLES.
      DO 520 K=1,7
  520 READ(7) ((ITB(I,J,K),J=1,NITCOL),I=1,NU)
C           INITIALIZE IT, THE CURRENT INTERVISIBILITY TABLE.
      DO 530 I=1,NU
      DO 530 J=1,NITCOL
  530 IT(I,J)=ITB(I,J,1)
      CALL GRAPH(NU,TP(1,1),TP(1,2),'TRUE',XMAX,XMIN,YMAX,YMIN)
C
C           * * * BEGIN THE PROCESSING OF A CYCLE. * * *
C
 1000 NCYC=NCYC+1
      K1CC=K1CC+1
      K2CC=K2CC+1
      K4CC=K4CC+1
      DO 4900 I9=1,NS
      CALL CLOCKS(T1)
```

```
      C   RT   NDS   METE
00405    TP(IU,1)=(MXLON*60-J1)/(3.0*CNX)
00406    TP(IU,2)=(I1-MNLAT*60)/(3.0*CNY)
00407    IF (NTYPE(IU).LT.3) GO TO 390
00410            AIRCRAFT HAVE A RANDOM ALTITUDE ASSIGNED TO THEM.
00410 C
00410    IF (NTYPE(IU).EQ.3) TP(IU,3)=100.+5000.*RU(KZ)
00412    IF (NTYPE(IU).EQ.4) TP(IU,3)=100.+4000.*RU(KZ)
00414    GO TO 425
00416            INITIATE HEIGHT RETRIEVAL.
00416 C
00416 390 CALL HEIGHT(-1.,-1.,TP(IU,1),TP(IU,2),TP(IU,3),M)
00417            CONVERT MOVEMENT DATA INTO DIRECTION COSINES AND BUILD
00417 C
00417 C          THE VELOCITY VECTORS.
00420 425 VV(IU,4)=IP
00421    DO 450 J=1,3
00424    IF (CT(J).LE.0.0) GO TO 460
00426    CT(J)=3600.*CT(J)
00427    IF (J.EQ.1) GO TO 430
00431    CT(J)=CT(J)+CT(J-1)
00432 430 VV1(IP,1)=COS((AZ(J)-90.0)/RADS)*SP(J)*RCON
00433    VV1(IP,2)=COS(AZ(J)/RADS)*SP(J)*RCON
00434    VV1(IP,4)=IP+1
00435    VV1(IP,3)=CT(J)
00436    IP=IP+1
00437    IF (CT(J).GT.SIMTIM) GO TO 460
00441 450 CONTINUE
00443    J=3
00444 460 VV1(IP-1,4)=-5.0
00445    J12=J
00445            WAIT FOR END OF HEIGHT RETRIEVAL.
00446 C
00446 470 IF (M.EQ.-1) GO TO 470
00450    WRITE(6,12) IU,NTYPE(IU),(TP(IU,J),J=1,3),AZ(1),SP(1),CT(1)
00463    IF (J12.EQ.1) GO TO 500
00465    WRITE(6,6) (AZ(J),SP(J),CT(J),J=2,J12)
00465            INITIALIZE EP, THE ESTIMATED POSITIONS MATRIX.
00465 C
00475    DO 480 J=1,3
00500    EP(IU,J)=TP(IU,J)
00501    EP(IU,J+3)=TP(IU,J)
00502    EP(IU,J+6)=TP(IU,J)
00503 480 EP(IU,J+9)=TP(IU,J)
00505 500 CONTINUE
00507    IF (TPOUT) WRITE(9) ((TP(I,J),J=1,3),NTYPE(I),I=1,NU)
00507            INPUT THE INTERVISIBILITY TABLES.
00507 C
00522    DO 520 K=1,7
00525 520 READ(7) ((ITB(I,J,K),J=1,NITCOL),I=1,NU)
00525            INITIALIZE IT, THE CURRENT INTERVISIBILITY TABLE.
00525 C
00537    DO 530 I=1,NU
00542    DO 530 J=1,NITCOL
00545 530 IT(I,J)=ITB(I,J,1)
00550    CALL GRAPH(NU,TP(1,1),TP(1,2),'TRUE',XMAX,XMIN,YMAX,YMIN)
00550 C
00550 C          * * * BEGIN THE PROCESSING OF A CYCLE. * * *
00550 C
00551 1000 NCYC=NCYC+1
00552    K1CC=K1CC+1
00553    K2CC=K2CC+1
00554    K4CC=K4CC+1
00555    DO 4900 I9=1,NS
00560    CALL CLOCKS(T1)
```

```
      CLK=CLK+DELTA
      IU=KUF(I9)
      BND=BND+BNDSP
C          DETERMINE THE LOCATEE'S NEW POSITION BY INCREMENTING THE
C          OLD POSITION WITH VV.  DO NOT MOVE A FIXED UNIT.
      IF (FXD(IU)) GO TO 1200
      IF (VV(IU,3).GT.CLK) GO TO 1020
C          CURRENT VV IS OUTDATED.  GET A NEW ONE FROM VV1,
C          IF AVAILABLE.
      IF (VV(IU,4).LT.0.0) GO TO 1200
      X=TP(IU,1)
      Y=TP(IU,2)
      K1=VV(IU,4)+.5
      DO 1010 J=1,4
 1010 VV(IU,J)=VV1(K1,J)
 1020 TP(IU,1)=TP(IU,1)+VV(IU,1)
      TP(IU,2)=TP(IU,2)+VV(IU,2)
C          GET THE LOCATEE'S HEIGHT.
      IF (NTYPE(IU).LT.3) CALL HEIGHT(X,Y,TP(IU,1),TP(IU,2),TP(IU,3),M)
C          IF LOCATEE IS BEYOND THE BOUNDARY, CHANGE HIS DIRECTION.
 1050 IF (TP(IU,1).LT.BND) GO TO 1110
      IF (VV(IU,1).GT.0.0) VV(IU,1)=-VV(IU,1)
      GO TO 1120
 1110 IF (TP(IU,1).GT.0.0) GO TO 1120
      IF (VV(IU,1).LT.0.0) VV(IU,1)=-VV(IU,1)
 1120 IF (TP(IU,2).LT.BND) GO TO 1130
      IF (VV(IU,2).GT.0.0) VV(IU,2)=-VV(IU,2)
      GO TO 1140
 1130 IF (TP(IU,2).GT.0.0) GO TO 1140
      IF (VV(IU,2).LT.0.0) VV(IU,2)=-VV(IU,2)
 1140 CONTINUE
C          DETERMINE ALL OTHER UNITS' NEW POSITIONS.  PROCEDURE IS THE
C          SAME HERE  AS FOR THE LOCATEE, BUT NO HEIGHT RETRIEVAL AND
C          NO BOUNDARY CHECKING.
 1200 DO 1290 I=1,NU
      IF (FXD(I)) GO TO 1290
      IF (I.EQ.IU) GO TO 1290
      IF (VV(I,3).GT.CLK) GO TO 1260
      IF (VV(I,4).LT.0.0) GO TO 1290
      K1=VV(I,4)+.5
      DO 1230 J=1,4
 1230 VV(I,J)=VV1(K1,J)
 1260 TP(I,1)=TP(I,1)+VV(I,1)
      TP(I,2)=TP(I,2)+VV(I,2)
 1290 CONTINUE
 2000 IR=0
C          WAIT FOR END OF HEIGHT RETRIEVAL.
 2010 IF (M.EQ.-1) GO TO 2010
C          PERTURB HEIGHT FOR NON-FIXED UNITS, AND STORE IN EP.
      CALL HRG(X)
      IF (FXD(IU)) X=0.0
      X=X+TP(IU,3)
      EP(IU,6)=X
      EP(IU,9)=X
      EP(IU,12)=X
      IF (FXD(IU)) GO TO 4900
C          COMPUTE RANGES FOR THIS SUBCYCLE.
      DO 2100 J=1,NU
```

```
00672   C          IF LOCATOR CANDIDATE IS NOT A REPORTER, SKIP THAT RANGE
00672   C          CALCULATION.
00675         IF (.NOT.RPT(J)) GO TO 2100
00677         IF (J.EQ.IU) GO TO 2100
00677   C          RETRIEVE INTERVISIBILITY FOR LOCATEE-LOCATOR PAIR.
00701         J1=(J-1)/36
00702         IV=FLD(J-J1*36,1,IT(IU,J1+1))
00702   C          IF NO LOS, SKIP RANGE CALCULATION.
00703         IF (IV.EQ.0) GO TO 2100
00705    2035 IR=IR+1
00705   C          COMPUTE RANGE AND PERTURB IT.
00706         RANGE(IR)=SQRT((TP(J,1)-TP(IU,1))**2+(TP(J,2)-TP(IU,2))**2
00706        *          +(TP(J,3)-TP(IU,3))**2)
00707         CALL RG(X)
00710         RANGE(IR)=RANGE(IR)+X
00711         LOCATR(IR)=J
00712    2100 CONTINUE
00714         RANGE(IR+1)=-5.0
00714   C          GO ATTEMPT TO LOCATE THE LOCATEE.
00715    3000 CALL CLOCKS(T3)
00716         CALL FINDIT
00717         CALL CLOCKS(T4)
00717   C          ACCUMULATE ERROR STATISTICS ON THE LOCATEE.
00720    4000 IF (.NOT.S(IU)) GO TO 4900
00722    4005 DX(IU)=EP(IU,4)-TP(IU,1)
00723         DY(IU)=EP(IU,5)-TP(IU,2)
00724         DZ(IU)=EP(IU,6)-TP(IU,3)
00725         ERR=SQRT(DX(IU)**2+DY(IU)**2)
00726         STAT1(IU,4)=STAT1(IU,4)+ERR
00727         STAT2(IU,1)=STAT2(IU,1)+1.0
00730         STAT2(IU,2)=STAT2(IU,2)+1.0
00731         AVKTR=AVKTR+1.0
00732         IF (STAT1(IU,5)-ERR) 4010,4020,4020
00735    4010 STAT1(IU,5)=ERR
00736         STAT1(IU,6)=NCYC
00737    4020 ERR=ABS(DZ(IU))
00740         STAT1(IU,10)=STAT1(IU,10)+ERR
00741         IF (STAT1(IU,8).GE.ERR) GO TO 4040
00743         STAT1(IU,8)=ERR
00744         STAT1(IU,9)=NCYC
00745    4040 CALL CLOCKS(T2)
00746         AVSUBT=AVSUBT+T2-T1
00747         AVASBT=AVASBT+T4-T3
00747   C
00747   C          * * * END OF THE PROCESSING FOR A CYCLE. * * *
00747   C
00750    4900 CONTINUE
00750   C          UPDATE THE INTERVISIBILITY TABLE.
00752    5000 IF (CLK.LT.FLOAT(1200*(IVMAT-1))) GO TO 5040
00752   C          IF THE TIME IS NIGH, COMPLETE THE OVERLAYING OF THE CURRENT
00752   C          INTERVISIBILITY MATRIX, AND ARRANGE FOR FUTURE USE OF THE
00752   C          NEXT INTERVISIBILITY MATRIX IN SEQUENCE.
00754         DO 5020 I=1,NU
00757         IF (ITIND(I)) GO TO 5020
00761         DO 5010 J=1,NITCOL
00764    5010 IT(I,J)=ITB(I,J,IVMAT)
00766    5020 ITIND(I)=.FALSE.
00770         IVMAT=IVMAT+1
```

```
      =0
 5040 IF (K2CC.LT.K2C) GO TO 5200
      K2CC=0
C          UPDATE THE CURRENT INTERVISIBILITY MATRIX BY RANDOMLY
C          FINDING K3C  ROWS THAT HAVE NOT BEEN CHANGED AND
C          CHANGING THEM.
      DO 5090 I=1,K3C
      I1=FLOAT(NU)*RU(KZ)+1.0
      I2=I1
 5050 IF (.NOT.ITIND(I2)) GO TO 5060
      I2=I2+1
      IF (I2.GT.NU) I2=I2-NU
      IF (I2.NE.I1) GO TO 5050
      GO TO 5200
 5060 DO 5070 J=1,NITCOL
 5070 IT(I2,J)=ITB(I2,J,IVMAT)
 5090 ITIND(I2)=.TRUE.
C          PREPARE AND  PRINT INTERMEDIATE ERROR OUTPUT.
 5200 IF (NCYC.EQ.MAXCYC) GO TO 5210
      IF (K1CC.LT.K1C) GO TO 5500
 5210 GTSUM=0.0
      GPSUM=0.0
      GTKTR=0.0
      GPKTR=0.0
      ZTSUM=0.0
      ZPSUM=0.0
      IGT=1
      IGP=1
      IZT=1
      AVSUBT=AVSUBT/AVKTR
      AVASBT=AVASBT/AVKTR
C          WRITE HEADINGS.
      WRITE(6,20) NCYC,CLK,AVSUBT,AVASBT
      WRITE(6,16) NCYC,K1CC
      WRITE(6,17)
      IF (TPOUT) WRITE(9) NCYC,CLK,K1CC
C          PREPARE AND PRODUCE THE BODY OF THE OUTPUT TABLE,
C          ONE LINE AT A TIME.
      DO 5250 I=1,NU
      STAT1(I,1)=STAT1(I,1)+STAT1(I,4)
      STAT1(I,7)=STAT1(I,7)+STAT1(I,10)
      IF (STAT1(I,2).GE.STAT1(I,5)) GO TO 5220
      STAT1(I,2)=STAT1(I,5)
      STAT1(I,3)=STAT1(I,6)
C          ACCUMULATE STATISTICS FOR LAST LINE OF TABLE.
 5220 IF (STAT1(I,2).GT.STAT1(IGT,2)) IGT=I
      IF (STAT1(I,5).GT.STAT1(IGP,5)) IGP=I
      IF (STAT1(I,8).GT.STAT1(IZT,8)) IZT=I
      GTSUM=GTSUM+STAT1(I,1)
      GPSUM=GPSUM+STAT1(I,4)
      ZTSUM=ZTSUM+STAT1(I,7)
      ZPSUM=ZPSUM+STAT1(I,10)
      GTKTR=GTKTR+STAT2(I,1)
      GPKTR=GPKTR+STAT2(I,2)
      J1=STAT1(I,3)+.5
      J2=STAT1(I,6)+.5
      J3=STAT1(I,9)+.5
      X=0.0
```

```
01113         IF (STAT2(I,1).GT.0.0) X=STAT1(I,1)/STAT2(I,1)
01115         Y=0.0
01116         IF (STAT2(I,2).GT.0.0) Y=STAT1(I,4)/STAT2(I,2)
01120         Z1=0.0
01121         IF (STAT2(I,1).GT.0.0) Z1=STAT1(I,7)/STAT2(I,1)
01123         Z2=0.0
01124         IF (STAT2(I,2).GT.0.0) Z2=STAT1(I,10)/STAT2(I,2)
01124   C           WRITE OUT A LINE OF THE TABLE.
01126         WRITE(6,1) I,X,Z1,STAT1(I,2),J1,STAT1(I,8),J3,Y,Z2,STAT1(I,5),J2,
01126        *           DX(I),DY(I),DZ(I),S(I),RPT(I),FXD(I),W(I)
01152         IF (TPOUT) WRITE(9) I,X,Z1,STAT1(I,2),J1,STAT1(I,8),J3,Y,Z2,
01152        *           STAT1(I,5),J2,DX(I),DY(I),DZ(I),S(I),RPT(I),FXD(I),
01152        *           W(I),(TP(I,J),J=1,3)
01152   C           ACCUMULATE QUADRANT COUNTS.
01203         IF (DX(I)) 5235,5236,5237
01206    5235 IF (DY(I)) 5241,5240,5240
01211    5236 IF (DY(I)) 5242,5238,5239
01214    5237 IF (DY(I)) 5242,5239,5239
01217    5238 IQ0=IQ0+1
01220         GO TO 5250
01221    5239 IQ1=IQ1+1
01222         GO TO 5250
01223    5240 IQ2=IQ2+1
01224         GO TO 5250
01225    5241 IQ3=IQ3+1
01226         GO TO 5250
01227    5242 IQ4=IQ4+1
01230    5250 CONTINUE
01230   C           PREPARE AND PRODUCE LAST LINE OF THE TABLE.
01232         X=GTSUM/GTKTR
01233         Y=GPSUM/GPKTR
01234         Z1=ZTSUM/GTKTR
01235         Z2=ZPSUM/GPKTR
01236         J1=STAT1(IGT,3)+.5
01237         J2=STAT1(IGP,6)+.5
01240         J3=STAT1(IZT,9)+.5
01241         WRITE(6,2) X,Z1,STAT1(IGT,2),J1,STAT1(IZT,8),J3,Y,Z2,
01241        *           STAT1(IGP,5),J2
01255         IF (TPOUT) WRITE(9) X,Z1,STAT1(IGT,2),J1,STAT1(IZT,8),J3,Y,Z2,
01255        *           STAT1(IGP,5),J2
01255   C           PRODUCE QUADRANT COUNTS.
01272         WRITE(6,14) IQ2,IQ1,IQ0,IQ3,IQ4
01272   C           REINITIALIZE STATISTICS ACCUMULATORS.
01301         IQ0=0
01302         IQ1=0
01303         IQ2=0
01304         IQ3=0
01305         IQ4=0
01306         K1CC=0
01307         AVSUBT=0.0
01310         AVASBT=0.0
01311         AVKTR=0.0
01312         DO 5260 I=1,NU
01315         STAT2(I,2)=0.0
01316         STAT1(I,4)=0.0
01317         STAT1(I,5)=0.0
01320         STAT1(I,10)=0.0
01321    5260 STAT1(I,6)=0.0
```

```
      IF (NCYC.EQ.MAXCYC) GO TO 5610
C          UPDATE THE STATIONARY UNIT LIST.
 5500 DO 5550 I=1,NU
C          AIRCRAFT CANNOT BE STATIONARY.
      IF (NTYPE(I).GT.2) GO TO 5550
      J=1
      IF(FXD(I)) J=2
      IF (RU(KZ).GT.PSS(J)) GO TO 5550
      FXD(I)=.NOT.FXD(I)
      W(I)=0.0
 5550 CONTINUE
C          PRODUCE THE INTERMEDIATE XY PLOTS.
 5600 IF (K4CC.LT.K4C) GO TO 1000
      K4CC=0
 5610 CALL GRAPH(NU,TP(1,1),TP(1,2),'TRUE',XMAX,XMIN,YMAX,YMIN)
      CALL GRAPH(NU,EP(1,7),EP(1,8),'EST.',XMAX,XMIN,YMAX,YMIN)
      IF (NCYC.EQ.MAXCYC) GO TO 9000
      GO TO 1000
 9000 IF (TPOUT) ENDFILE 9
      STOP
C          FORMAT STATEMENTS AND INTERNAL SUBPROGRAMS FOLLOW.
    1 FORMAT(1XI4,3E10.5,I4,E10.5,I4,2X,3E10.5,I4,2X3E9.3,L2,2L1,E10.5)
    2 FORMAT(/'  ALL'3E10.5,I4,E10.5,I4,2X3E10.5,I4)
    3 FORMAT(6I5,5F10.5/6F10.5)
    4 FORMAT(20I4)
    5 FORMAT(I5,F5.0,I5)
    6 FORMAT(53X3E14.8)
    7 FORMAT(/' INITIAL REPORTING UNITS ARE:')
    8 FORMAT(26I5)
    9 FORMAT(/' INITIAL STATIONARY UNITS ARE:')
   10 FORMAT(/' ASSIGNMENT OF UNITS TO SLOTS IS:')
   11 FORMAT(/' UNIT  TYPE   STARTING X    STARTING Y    STARTING Z'6X'A
     *ZIMUTH'8X'SPEED'6X'CHANGE TIME')
   12 FORMAT(I5,I6,6E14.8)
   13 FORMAT(' NUMBER  OF  UNITS ='I9/' NUMBER  OF  SLOTS ='I9/' OUTPUT
     *   COUNTER ='I9/' IV UPDATE COUNTER ='I9/' ROWS IN IV UPDATE ='I9/
     *' XY  PLOT  COUNTER ='I9/
     *' MAX NUMBER RANGES ='I9/' NUM.  STAT. UNITS ='I9/' SIMULATION   T
     *IME ='F9.2/' MAX NUMBER CYCLES ='I9/' SUBCYCLE  DELTA T ='F9.4/' A
     *VG  TIME  MOVING ='F9.2/' AVG  TIME STOPPED ='F9.2/' UPPER  BOUNDA
     *RIES ='F9.4/' BOUNDARY    SPEED ='F9.2)
   14 FORMAT(/53X'QUADRANT COUNTS OF ERRORS'//60XI3,5XI3/64XI3/60XI3,
     *5XI3)
   15 FORMAT(21(1X5I1))
   16 FORMAT(/14X'ERROR STATS. OVER ALL'I4,' CYCLES'11X'ERROR STATS. OVE
     *R LAST'I3,' CYCLES'4X'CURRENT COORDINATE ERRORS')
   17 FORMAT(' UNIT  XY AVG'5X'Z AVG'4X'XY MAX   CYC   Z MAX   CYC'4X'XY
     * AVG'5X'Z AVG'4X'XY MAX   CYC'5X'EX-TX'4X'EY-TY'4X'EZ-TZ  SRF  WEI
     *GHT'/)
   18 FORMAT(/' THE UNIT TYPES ARE:')
   20 FORMAT(///' CYCLE NUMBER'I5,'.  TIME ='F10.3,' S.  AVERAGE SIMULAT
     *ION SUBCYCLE TIME ='F7.4,' S.  AVERAGE ALGORITHM SUBCYCLE TIME ='F
     *7.4,' S.')
   30 FORMAT(I3,18X2(I3,2I2),3(1X3F4.0))
      FUNCTION RU(KZ)
      KZ=KZ*2045
      KZ=KZ-(KZ/4194304)*4194304
      RU=KZ
```

```
01410    436*         RU=RU*2.3841858E-7
01411    437*         RETURN
01412    438*         END

       END OF UNIVAC 1108 FORTRAN V COMPILATION.     0 *DIAGNOSTIC* MESSAGE(S)
PHASE 1 TIME =   1 SEC.
PHASE 2 TIME =   0 SEC.
PHASE 3 TIME =   2 SEC.
PHASE 4 TIME =   0 SEC.
PHASE 5 TIME =   1 SEC.
PHASE 6 TIME =   1 SEC.

TOTAL COMPILATION TIME =   5  SEC
```

```
JIT FOR HEIGHT,HEIGHT                                                                     01 JUN 72        10:55:48.426
UNIVAC 1108 FORTRAN V LEVEL   2206 0018 F5018P
THIS COMPILATION WAS DONE ON 01 JUN 72 AT 10:55:48

   SUBROUTINE HEIGHT    ENTRY POINT 000270

   STORAGE USED (BLOCK, NAME, LENGTH)

        0001    *CODE    000306
        0000    *DATA    000054
        0002    *BLANK   000073


   EXTERNAL REFERENCES (BLOCK, NAME)

        0003    NTRAN
        0004    NWDU$
        0005    NI02$
        0006    NSTOP$
        0007    NERR3$


   STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

    0000    000011 1F        0001    000057 100L     0001    000144 105L     0001    000161 110L     0001    000164 120L
    0001    000167 130L      0001    000173 200L     0002 R 000002 CNX      0002    000003 CNY      0000 I 000004 I
    0000 I 000002 IFX        0000 I 000003 IFY      0000 I 000000 IPX      0000 I 000001 IPY      0000 I 000006 I1
    0000 I 000005 J          0000 I 000007 J1       0002 I 000000 MNLAT    0002    000001 MXLON    0000 I 000010 N
    0002 I 000004 NDX


00101      1*          SUBROUTINE HEIGHT(PX,PY,FX,FY,FZ,M)
00103      2*          PARAMETER L6=5,L7=11
00104      3*          COMMON MNLAT,MXLON,CNX,CNY,NDX(L6,L7)
00105      4*          DATA CNX,CNY/.0145515310,.0107798475/
00105      5*    C          CONVERT FROM POSITION IN METERS TO POSITION IN TERMS OF
00105      6*    C          THREE SECONDS OF LATITUDE AND LONGITUDE.
00110      7*          IPX=PX*CNX+.5
00111      8*          IPY=PY*CNY+.5
00112      9*          IFX=FX*CNX+.5
00113     10*          IFY=FY*CNY+.5
00114     11*          IF (IPX.NE.IFX) GO TO 100
00116     12*          IF (IPY.NE.IFY) GO TO 100
00116     13*    C          IF FUTURE POSITION IS ON SAME TERRAIN PLATFORM AS PRESENT
00116     14*    C          POSITION, FEED BACK SAME HEIGHT.
00120     15*          M=1
00121     16*          RETURN
00121     17*    C          COMPUTE THE SUBSCRIPTS FOR THE REQUIRED TERRAIN BLOCK.
00122     18*      100 I=IFY/120+1
00123     19*          J=IFX/120+1
00124     20*          IF (NDX(I,J).GE.0) GO TO 200
00124     21*    C          IF THAT TERRAIN BLOCK DOES NOT EXIST, UNIT MAY BE ON A
00124     22*    C          BOUNDARY BETWEEN TWO OR FOUR BLOCKS.  CHECK EXISTENCE OF
```

```
00124     C            THOSE BLOCKS.
00126           I1=(IFY-1)/120+1
00127           IF (NDX(I1,J).GE.0) GO TO 110
00131           J1=(IFX-1)/120+1
00132           IF (NDX(I,J1).GE.0) GO TO 120
00134           IF (NDX(I1,J1).GE.0) GO TO 130
00136       105 WRITE(6,1) FX,FY,I,J,IFX,IFY
00146         1 FORMAT(/' NO HEIGHT FOR'2E14.8,' NEED BLOCK'2I5,' REDUCED COORDINA
00146          *TES ARE:'2I5)
00147           STOP
00150       110 I=I1
00151           GO TO 200
00152       120 J=J1
00153           GO TO 200
00154       130 I=I1
00155           J=J1
00155     C            CHECK VALIDITY OF TERRAIN BLOCK SUBSCRIPTS.
00156       200 IF (I.GT.L6) GO TO 105
00160           IF (I.LT.1) GO TO 105
00162           IF (J.GT.L7) GO TO 105
00164           IF (J.LT.1) GO TO 105
00164     C            COMPUTE SUBSCRIPTS FOR LOCATION OF HEIGHT WITHIN THE BLOCK.
00166           I1=IFY-120*(I-1)+1
00167           J1=IFX-120*(J-1)+1
00167     C            COMPUTE POSITION OF REQUIRED HEIGHT ON THE DRUM.
00170           N=NDX(I,J)+121*(J1-1)+I1-1
00170     C            REWIND THE DRUM, POSITION TO REQUIRED HEIGHT, AND READ IT  N.
00171           CALL NTRAN(35,10,6,N,2,1,FZ,M)
00172           RETURN
00173           END

      END OF UNIVAC 1108 FORTRAN V COMPILATION.     0 *DIAGNOSTIC* MESSAGE(S)
PHASE 1 TIME =   0 SEC.
PHASE 2 TIME =   0 SEC.
PHASE 3 TIME =   0 SEC.
PHASE 4 TIME =   1 SEC.
PHASE 5 TIME =   0 SEC.
PHASE 6 TIME =   0 SEC.

TOTAL COMPILATION TIME =  1  SEC
```

```
@IT FOR HRG,HRG
UNIVAC 1108 FORTRAN V LEVEL   2206 0018 F5018P
THIS COMPILATION WAS DONE ON 01 JUN 72 AT 10:55:49

   SUBROUTINE HRG        ENTRY POINT 000072

   STORAGE USED (BLOCK, NAME, LENGTH)

        0001   *CODE   000077
        0000   *DATA   000021
        0002   *BLANK  000000

   EXTERNAL REFERENCES (BLOCK, NAME)

        0003   HRGU
        0004   ALOG
        0005   SQRT
        0006   COS
        0007   NERR3$

   STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

    0001     000057 50L       0004 R 000000 ALOG      0000 R 000005 A1       0000 R 000004 A1S      0000 R 000006 A2
    0000 I 000001 IFLAG       0000 R 000000 PI        0000 R 000002 U1       0000   000003 U2

00101     1*          SUBROUTINE HRG(X)
00103     2*          DATA PI/3.14159265358979/
00105     3*          DATA IFLAG/0/
00107     4*          IF (IFLAG.GT.0) GO TO 50
00111     5*          CALL HRGU(U1,U2)
00112     6*          A1S=-2.0*ALOG(U1)
00113     7*          A1=SQRT(A1S)
00114     8*          A2=2.0*PI*U2
00115     9*          U1=A1*COS(A2)
00116    10*          U2=SQRT(A1S-U1**2)
00117    11*          IF (A2.GT.PI) U2=-U2
00121    12*          X=U1*14.0
00122    13*          IFLAG=1
00123    14*          RETURN
00124    15*       50 X=U2*14.0
00125    16*          IFLAG=0
00126    17*          RETURN
00127    18*          END

      END OF UNIVAC 1108 FORTRAN V COMPILATION.     0 *DIAGNOSTIC* MESSAGE(S)
PHASE 1 TIME =   0 SEC.
PHASE 2 TIME =   0 SEC.
PHASE 3 TIME =   0 SEC.
PHASE 4 TIME =   0 SEC.
PHASE 5 TIME =   1 SEC.
```

```
PHASE 6 TIME =   0 SEC.

TOTAL COMPILATION TIME =   1  SEC
```

```
@IT ASM HRGU,HRGU                                                                                    01 JUN 72        10:55:50.336
ASSEMBLED BY UNIVAC 1108 EXEC II ASSEMBLER   2404 0008A
THIS ASSEMBLY WAS DONE ON 01 JUN 72 AT 10:55:50
                                                               AXR$
                                                     $(1)
          01    000000  10 00 00 00 0 000006   HRGU*   LA      A0,I2
                000001  30 00 00 00 0 000002           MI      A0,K2
                000002  71 10 00 00 0 000003           DA      A0,K3-1
                000003  42 00 01 00 0 000000           AND     A1,MASK
                000004  30 00 02 00 0 000002           MI      A2,K2
                000005  71 10 02 00 0 000003           DA      A2,K3-1
                000006  42 00 03 00 0 000000           AND     A3,MASK
                000007  01 00 04 00 0 000006           SA      A4,T2
                000010  10 00 00 00 0 000005           LA      A0,I1
                000011  30 00 00 00 0 000001           MI      A0,K1
                000012  14 00 01 00 0 000007           AA      A1,(1)
                000013  42 00 01 00 0 000000           AND     A1,MASK
                000014  01 00 02 00 0 000005           SA      A2,I1
                000015  73 06 02 00 0 000016           LSC     A2,A2
                000016  73 12 02 00 0 000001           LSSL    A2,1
                000017  10 00 01 00 0 000010           LA      A1,(0200)
                000020  15 00 01 00 0 000017           ANA     A1,A3
                000021  73 03 01 00 0 000011           DSL     A1,9
                000022  73 06 04 00 0 000020           LSC     A4,A4
                000023  73 12 04 00 0 000001           LSSL    A4,1
                000024  10 00 03 00 0 000010           LA      A3,(0200)
                000025  15 00 03 00 0 000021           ANA     A3,A5
                000026  73 03 03 00 0 000011           DSL     A3,9
                000027  01 00 02 13 1 000000           SA      A2,*0,X11
                000030  01 00 04 13 1 000001           SA      A4,*1,X11
                000031  74 04 00 13 0 000003           J       3,X11
                                                     $(0)
          00    000000  377777777777           MASK    +       0377777777777
                000001  000000010001           K1      +       4097
                000002  000000000201           K2      +       129
                000003  000000000000                   +       0
                000004  311564570651           K3      +       27075473833
                000005  233362477003           I1      +       20867350019
                000006  212312312323           I2      +       18575103187
                                                       END
                000007  000000000001
                000010  000000000200
```

```
FOR G                                                                 01 J  72   0:55 .639
UNIVAC 1108 FORTRAN V LEVEL   2206 0018 F5018P
THIS COMPILATION WAS DONE ON 01 JUN 72 AT 10:55:51

   SUBROUTINE RG          ENTRY POINT 000153

   STORAGE USED (BLOCK, NAME, LENGTH)

         0001    *CODE    000160
         0000    *DATA    000027
         0002    *BLANK   000000


   EXTERNAL REFERENCES (BLOCK, NAME)

         0003    RGU
         0004    ALOG
         0005    SQRT
         0006    COS
         0007    NERR3$


   STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

    0001     000024 10L        0001     000030 20L       0001     000044 30L       0001     000050 40L       0001     000141 50L
    0004 R 000000 ALOG         0000 R 000007 A1          0000 R 000006 A1S        0000 R 000010 A2         0000 I 000004 IB1
    0000 I 000005 IB2          0000 I 000001 IFLAG       0000 R 000000 PI         0000 R 000002 U1         0000 R 000003 U2


00101      1*           SUBROUTINE RG(X)
00103      2*           DATA PI/3.14159265358979/
00105      3*           DATA IFLAG/0/
00107      4*           IF (IFLAG.GT.0) GO TO 50
00111      5*           CALL RGU(U1,U2)
00112      6*           IF (U1.LT..99) GO TO 10
00114      7*           U1=(U1-.99)*100.0
00115      8*           IB1=1
00116      9*           GO TO 20
00117     10*        10 U1=U1*1.01010101
00120     11*           IB1=0
00121     12*        20 IF (U2.LT..99) GO TO 30
00123     13*           U2=(U2-.99)*100.0
00124     14*           IB2=1
00125     15*           GO TO 40
00126     16*        30 U2=U2*1.01010101
00127     17*           IB2=0
00130     18*        40 A1S=-2.0*ALOG(U1)
00131     19*           A1=SQRT(A1S)
00132     20*           A2=2.0*PI*U2
00133     21*           U1=A1*COS(A2)
00134     22*           U2=SQRT(A1S-U1**2)
00135     23*           IF (A2.GT.PI) U2=-U2
00137     24*           U1=U1*6
```

```
00140     25*           U2=U2*6
00141     26*           IF (IB1.GT.0) U1=U1+7
00143     27*           IF (IB2.GT.0) U2=U2+7
00145     28*           X=U1
00146     29*           IFLAG=1
00147     30*           RETURN
00150     31*        50 X=U2
00151     32*           IFLAG=0
00152     33*           RETURN
00153     34*           END

       END OF UNIVAC 1108 FORTRAN V COMPILATION.      0 *DIAGNOSTIC* MESSAGE(S)
PHASE 1 TIME =   0 SEC.
PHASE 2 TIME =   0 SEC.
PHASE 3 TIME =   1 SEC.
PHASE 4 TIME =   0 SEC.
PHASE 5 TIME =   0 SEC.
PHASE 6 TIME =   0 SEC.


TOTAL COMPILATION TIME =   1  SEC
```

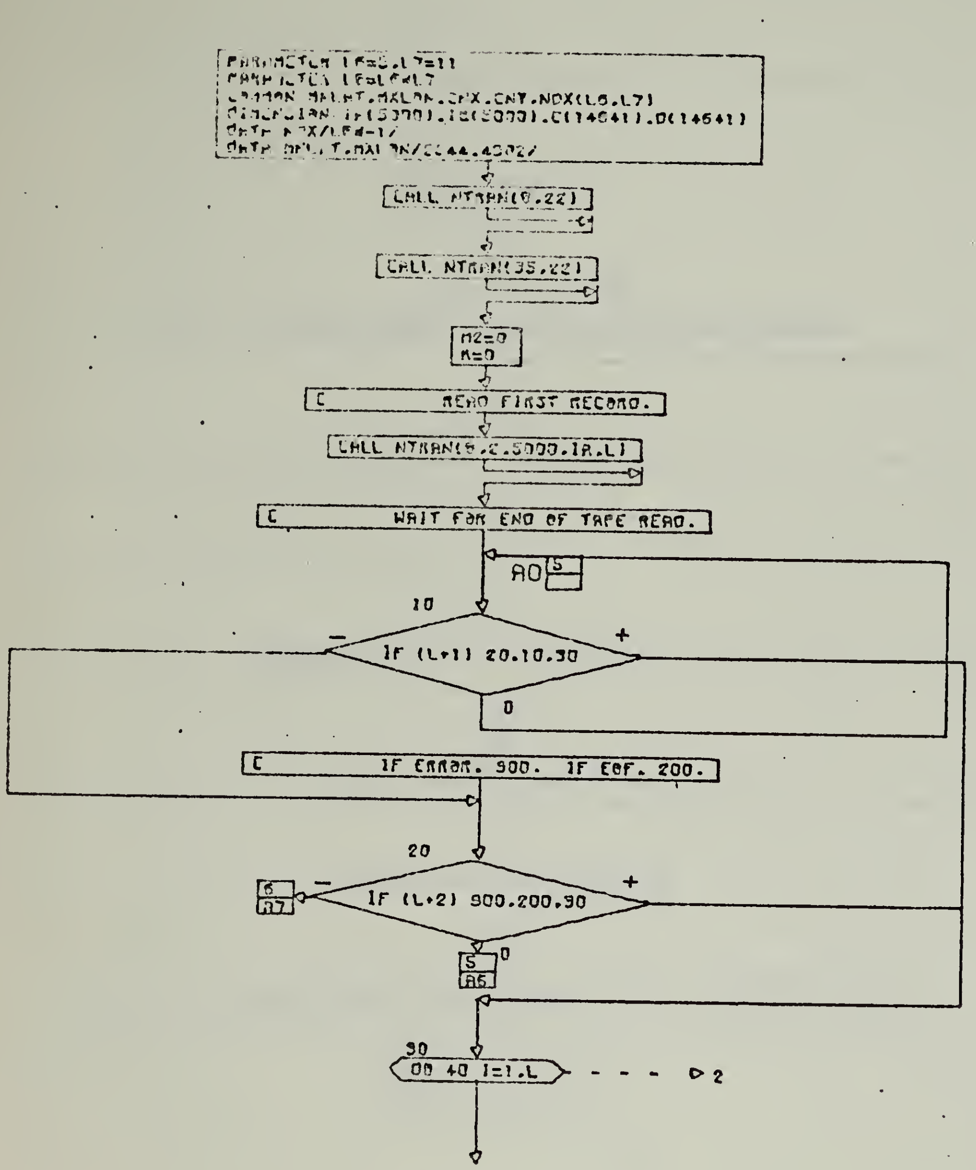
```
@IT ASM RGU,RGU                                                                       01 JUN 72      10:55:52.594
ASSEMBLED BY UNIVAC 1108 EXEC II ASSEMBLER   2404 0008A
THIS ASSEMBLY WAS DONE ON 01 JUN 72 AT 10:55:52
                                                     AXR$
                                             $(1)
      01   000000  10 00 00 00 0 000006  RGU*       LA      A0,I2
           000001  30 00 00 00 0 000002             MI      A0,K2
           000002  71 10 00 00 0 000003             DA      A0,K3-1
           000003  42 00 01 00 0 000000             AND     A1,MASK
           000004  30 00 02 00 0 000002             MI      A2,K2
           000005  71 10 02 00 0 000003             DA      A2,K3-1
           000006  42 00 03 00 0 000000             AND     A3,MASK
           000007  01 00 04 00 0 000006             SA      A4,I2
           000010  10 00 00 00 0 000005             LA      A0,I1
           000011  30 00 00 00 0 000001             MI      A0,K1
           000012  14 00 01 00 0 000007             AA      A1,(1)
           000013  42 00 01 00 0 000000             AND     A1,MASK
           000014  01 00 02 00 0 000005             SA      A2,I1
           000015  73 06 02 00 0 000016             LSC     A2,A2
           000016  73 12 02 00 0 000001             LSSL    A2,1
           000017  10 00 01 00 0 000010             LA      A1,(0200)
           000020  15 00 01 00 0 000017             ANA     A1,A3
           000021  73 03 01 00 0 000011             DSL     A1,9
           000022  73 06 04 00 0 000020             LSC     A4,A4
           000023  73 12 04 00 0 000001             LSSL    A4,1
           000024  10 00 03 00 0 000010             LA      A3,(0200)
           000025  15 00 03 00 0 000021             ANA     A3,A5
           000026  73 03 03 00 0 000011             DSL     A3,9
           000027  01 00 02 13 1 000000             SA      A2,*0,X11
           000030  01 00 04 13 1 000001             SA      A4,*1,X11
           000031  74 04 00 13 0 000003             J       3,X11
                                             $(0)
      00   000000  377777777777              MASK   +       0377777777777
           000001  000000010001              K1     +       4097
           000002  000000000201              K2     +       129
           000003  000000000000                     +       0
           000004  311564570651              K3     +       27075473833
           000005  233362477003              I1     +       20867350019
           000006  212312312323              I2     +       18575103187
                                                   END
           000007  000000000001
           000010  000000000200
```

```
UNIVAC 1108 FORTRAN V LEVEL   2206 0018 F5018P
THIS COMPILATION WAS DONE ON 01 JUN 72 AT 10:55:54

   SUBROUTINE GRAPH      ENTRY POINT 000536

   STORAGE USED (BLOCK, NAME, LENGTH)

        0001    *CODE    000571
        0000    *DATA    000572
        0002    *BLANK   000000

   EXTERNAL REFERENCES (BLOCK, NAME)

        0003    NWDU$
        0004    NIO1$
        0005    NIO2$
        0006    NERR2$
        0007    NERR3$

   STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

    0000     000455 100F        0001     000151 109L        0001     000154 110L        0001     000050 125G        0001     000064 134G
    0001     000130 152G        0001     000205 207G        0001     000215 213G        0001     000220 216G        0001     000232 225G
    0001     000261 240G        0001     000304 240L        0001     000312 250L        0001     000320 260L        0001     000346 272L
    0001     000334 274G        0001     000351 275L        0001     000353 278L        0001     000363 280L        0000     000500 290F
    0001     000407 300L        0000     000504 310F        0001     000402 322G        0001     000415 332G        0001     000434 345G
    0001     000421 350L        0001     000456 357G        0001     000470 365G        0001     000425 400L        0000     000510 420F
    0001     000014 910L        0000     000416 915F        0000     000453 916F        0000     000425 917F        0000     000434 918F
    0000 I 000363 BLANK       0000 I 000364 I           0000 I 000405 IFLAG       0000 I 000365 IO          0000 I 000401 IOUT
    0000 I 000410 ITA         0000 I 000216 ITOTAL      0000 I 000414 I$          0000 I 000371 I4          0000 I 000403 J
    0000 I 000404 K           0000 I 000407 KA          0000 I 000412 KK          0000 I 000402 L           0000 I 000411 LA
    0000 I 000370 LIMA        0000 I 000366 LIML        0000 I 000367 LIMU        0000 I 000205 NUMB        0000 I 000000 PRINT
    0000 I 000153 SYM         0000 R 000373 XDELTA      0000 R 000400 XHIGH       0000 R 000377 XL          0000 R 000145 XP
    0000 R 000415 XR          0000 R 000406 XT          0000 R 000372 YDELTA      0000 R 000374 YL          0000 R 000376 YLOW
    0000 R 000413 YP          0000 R 000375 YT

00101      1*          SUBROUTINE GRAPH(N,X,Y,IWRD,XMAX,XMIN,YMAX,YMIN)
00103      2*          DIMENSION X(1),Y(1),PRINT(101),XP(6),SYM(26)
00104      3*          DIMENSION NUMB(9),ITOTAL(101)
00105      4*          INTEGER PRINT,BLANK,SYM
00106      5*          DATA BLANK/1H  /
00110      6*          DATA (NUMB(I),I=1,9)/1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9/
00112      7*          DATA (SYM(I),I=1,26)/1HA,1HB,1HC,1HD,1HE,1HF,1HG,1HH,1HI,1HJ,1HK,
00112      8*         *    1HL,1HM,1HN,1HO,1HP,1HQ,1HR,1HS,1HT,1HU,1HV,1HW,1HX,1HY,1HZ/
00112      9*    C     IO = NUMBER OF PRINT TAPE
00114     10*          IO = 6
00115     11*          LIML=1
00116     12*          LIMU=26
00117     13*      910 IF (LIMU.GT.N) LIMU=N
```

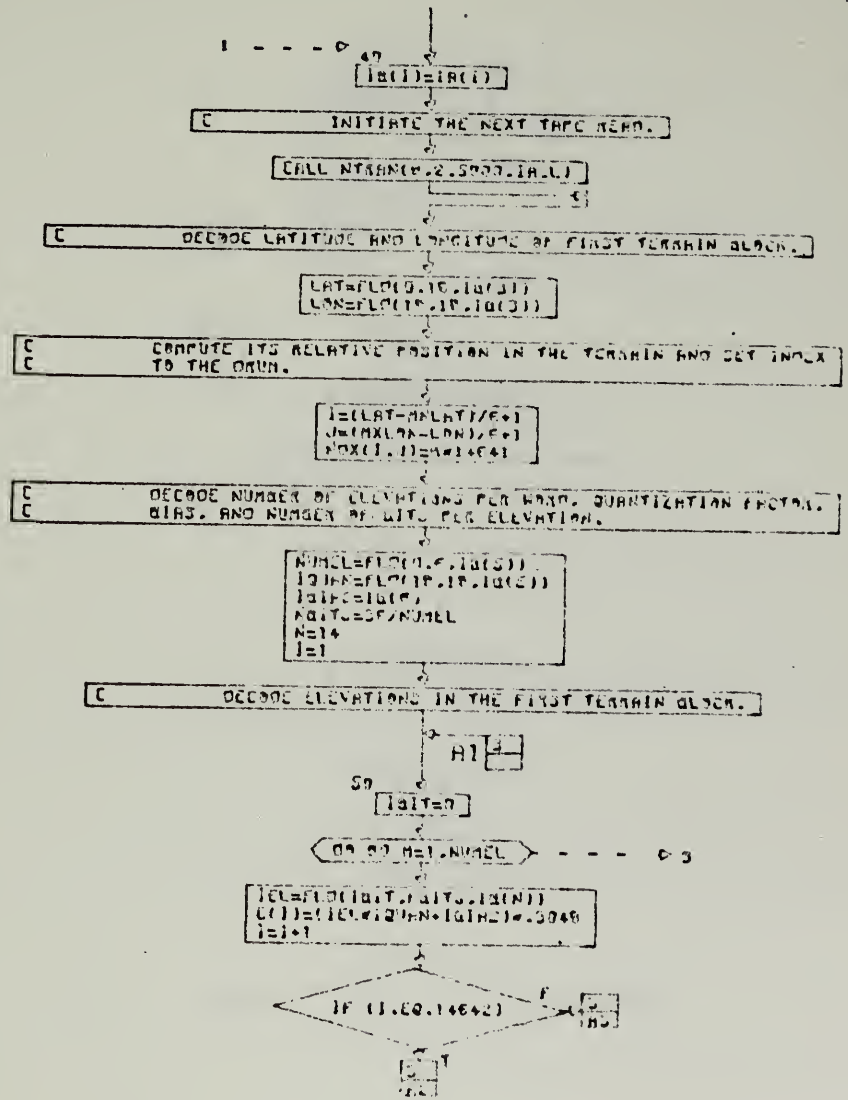
```
      LI  IMU-    +1
      WRITE(6,915) IWRD,(I4,I4=LIML,LIMU)
  915 FORMAT('1'A4,' POSITIONS OF UNITS'26I4)
      WRITE(6,917) (SYM(I4),I4=1,LIMA)
  917 FORMAT(' ASSOCIATED  SYMBOLS ARE'26(3X,A1))
      YDELTA=(YMAX-YMIN)/50.                                            100
      XDELTA=(XMAX-XMIN)/100.                                           110
      YL  =YMAX-YDELTA/2.                                               120
      YT=YMAX+YDELTA/2.
      YLOW=YMIN-YDELTA/2.
      XL=XMIN-XDELTA/2.
      XHIGH=XMAX+XDELTA/2.
      IOUT=0
      DO 110 I=LIML,LIMU
      IF (Y(I)-YT) 101,101,109
  101 IF (Y(I)-YLOW) 109,109,102
  102 IF (X(I)-XL) 109,103,103
  103 IF (X(I)-XHIGH) 110,109,109
  109 IOUT=IOUT+1
  110 CONTINUE
      IF (IOUT.GT.0) WRITE(6,918) IOUT
  918 FORMAT(' * * * NOTE.'I4,' POINTS ARE OUTSIDE THE LIMITS OF THE GRA
     *PH AND HAVE BEEN OMITTED')
      WRITE(6,916)
  916 FORMAT('  ')
      WRITE(IO,100)
  100 FORMAT(14X,101H+---------+---------+---------+---------+---------+
     1---------+---------+---------+---------+---------+ )
      DO 350  I=1,6                                                     140
      L=1                                                               150
      DO 350  J=1,10                                                    160
      DO  200 K=1,101                                                   170
      ITOTAL(K)=1
  200 PRINT(K)=BLANK                                                    175
      IFLAG=0
      DO 260 K=LIML,LIMU
      IF(Y(K)-YT) 205,205,260
  205 IF(Y(K)-YL) 260,260,210
  210 XL=XMIN
      XT=XMIN+XDELTA/2.                                                 210
      DO 255 KA=1,101                                                   220
      IF(X(K)-XL) 250,215,215
  215 IF(X(K)-XT) 220,250,250
  220 IF (PRINT(KA)-BLANK) 240,230,240
  230 ITA=K-(K-1)/26*26
      PRINT(KA)=SYM(ITA)                                                250
      GO TO 260                                                         260
  240 ITOTAL(KA)=ITOTAL(KA)+1
      IFLAG=1
      GO TO 260
  250 XL=XT
  255 XT=XT+XDELTA
  260 CONTINUE
      YT=YL
      YL=YL-YDELTA
      IF (IFLAG) 265,278,265
  265 DO 275 LA=1,101
      IF (ITOTAL(LA)-1) 268,275,268
```

```
00301     268 KK=ITOTAL(LA)
00302         IF (KK-9) 272,272,270
00305     270 KK=9
00306     272 PRINT(LA)=NUMB(KK)
00307     275 CONTINUE
00311     278 GO TO (280,300),L
00312     280 IF(I-5) 285,285,400
00315     285 L=2
00316         YP=YT+YDELTA/2.
00317         WRITE (IO,290) YP,PRINT
00326     290 FORMAT(1X,E12.4,1H+,101A1,1H+)
00327         GO TO 350
00330     300 WRITE (IO,310) PRINT
00336     310 FORMAT (13X,1H-,101A1,1H-)
00337     350 CONTINUE
00342     400 WRITE (IO,290) YMIN,PRINT
00351         WRITE (IO,100)
00353         XP(1)=XMIN
00354         XP(6)=XMAX
00355         XR=20.*XDELTA
00356         DO 410 I=2,5
00361     410 XP(I)=XP(I-1)+XR
00363         WRITE (IO,420) XP
00371     420 FORMAT(6(7X,E13.5))
00372         IF (LIMU.EQ.N) RETURN
00374         LIML=LIML+26
00375         LIMU=LIMU+26
00376         GO TO 910
00377         END

      END OF UNIVAC 1108 FORTRAN V COMPILATION.     0 *DIAGNOSTIC* MESSAGE(S)
PHASE 1 TIME =   0 SEC.
PHASE 2 TIME =   0 SEC.
PHASE 3 TIME =   0 SEC.
PHASE 4 TIME =   0 SEC.
PHASE 5 TIME =   1 SEC.
PHASE 6 TIME =   0 SEC.

TOTAL COMPILATION TIME =  1  SEC
```

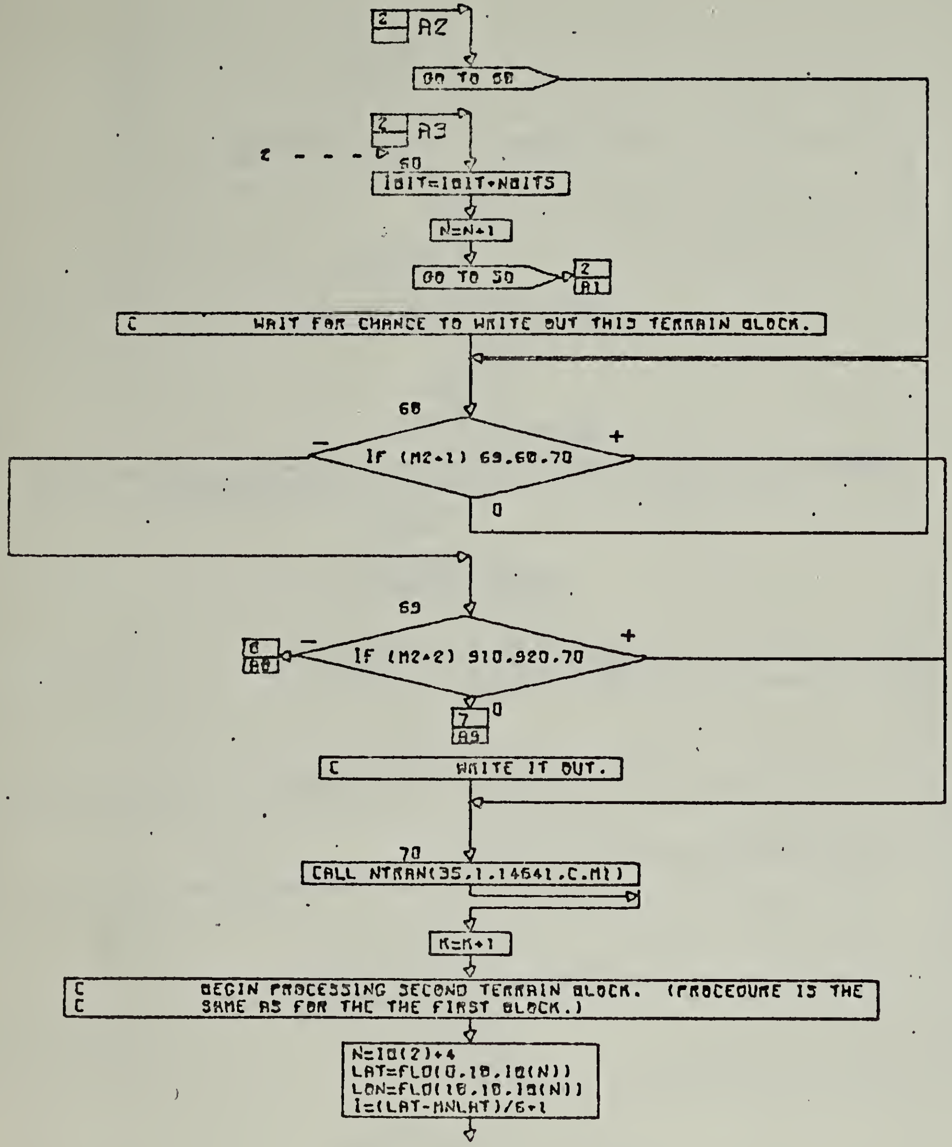
APPENDIX C: FLOWCHARTS OF THE MAIN SIMULATION PROGRAMS



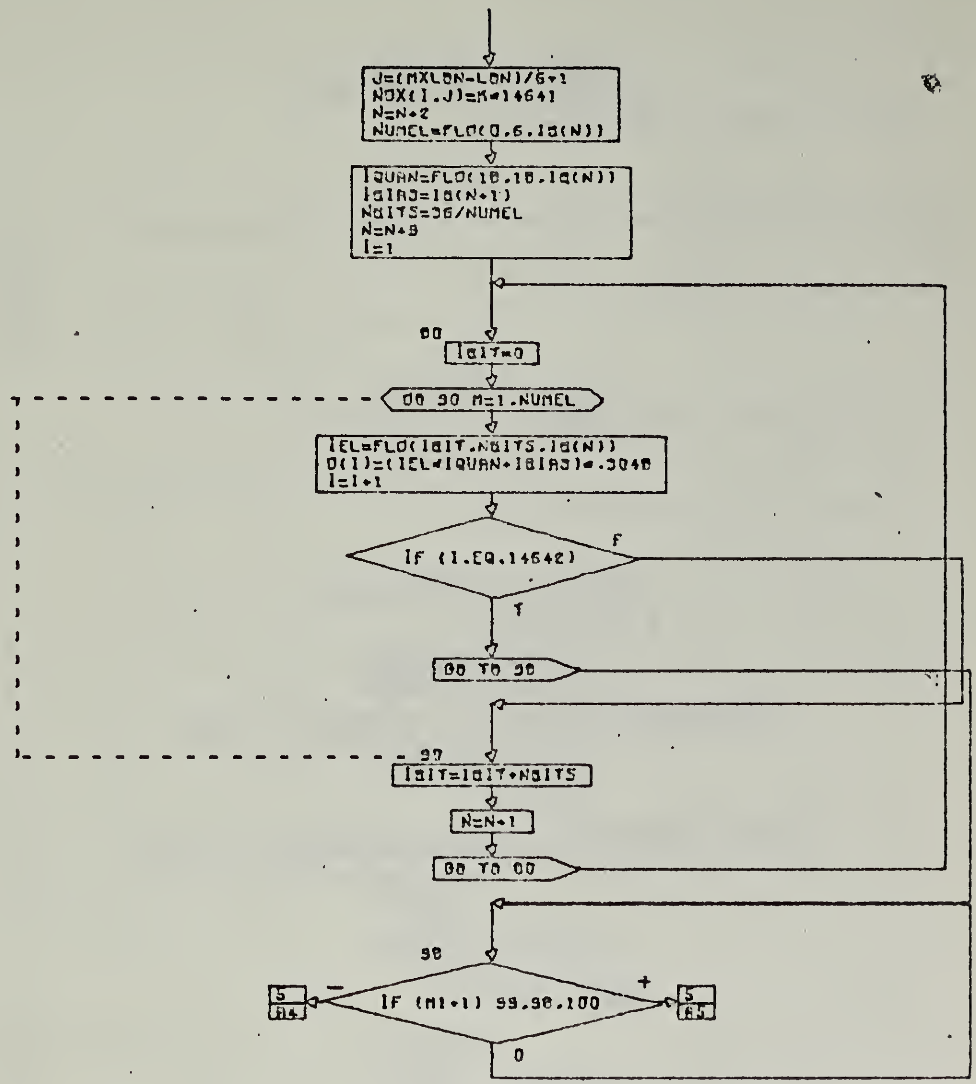
CONT. ON PG 2



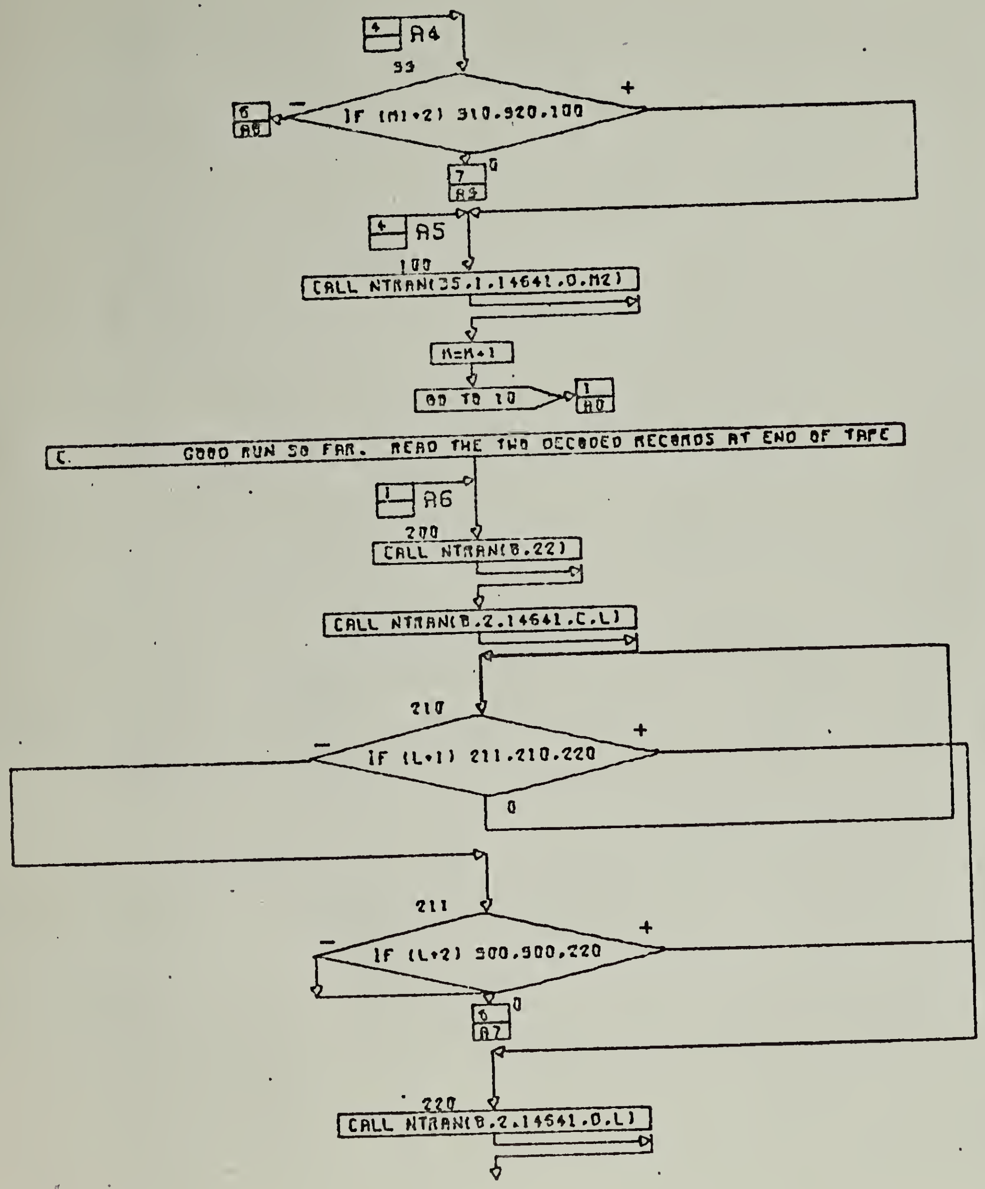
CONT. ON PG 3



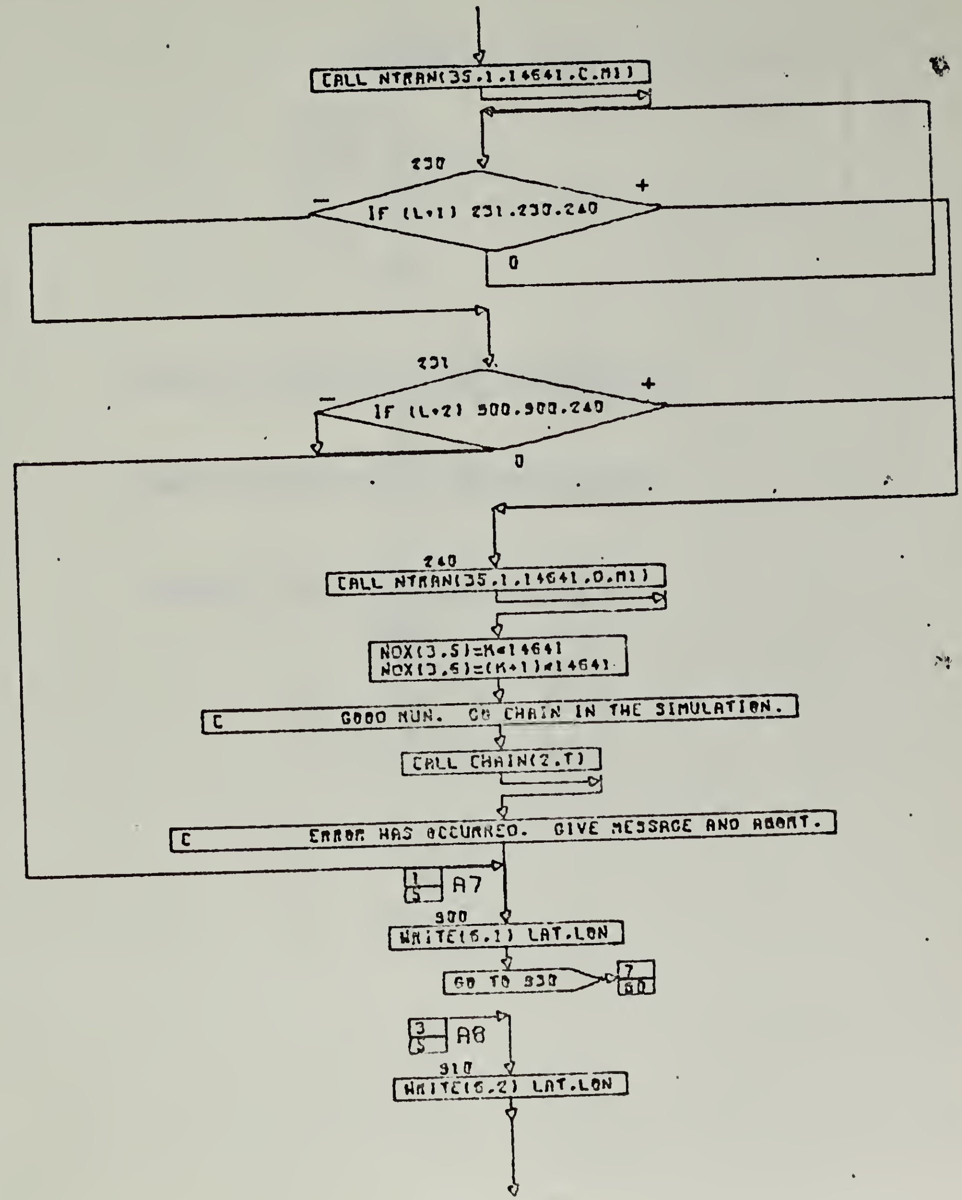
CONT. ON PG 4



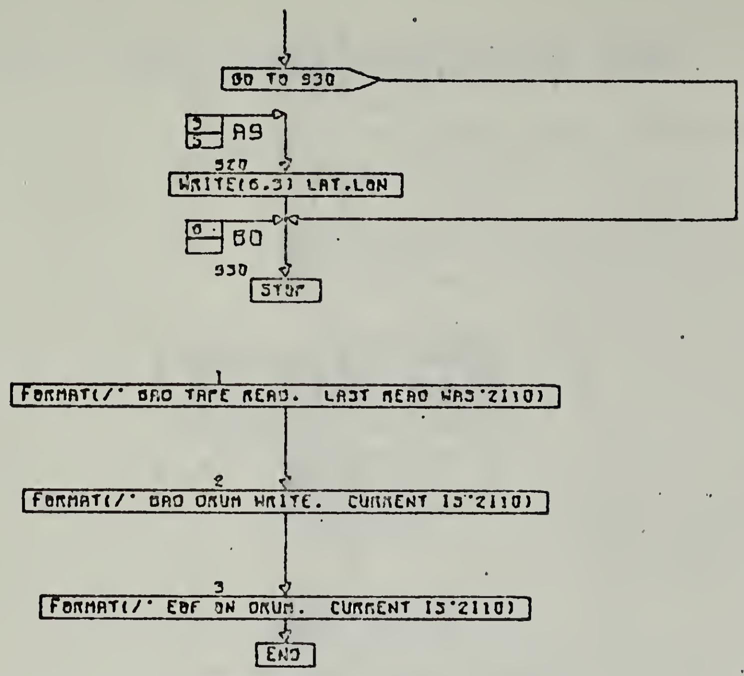
CONT. ON PG 5

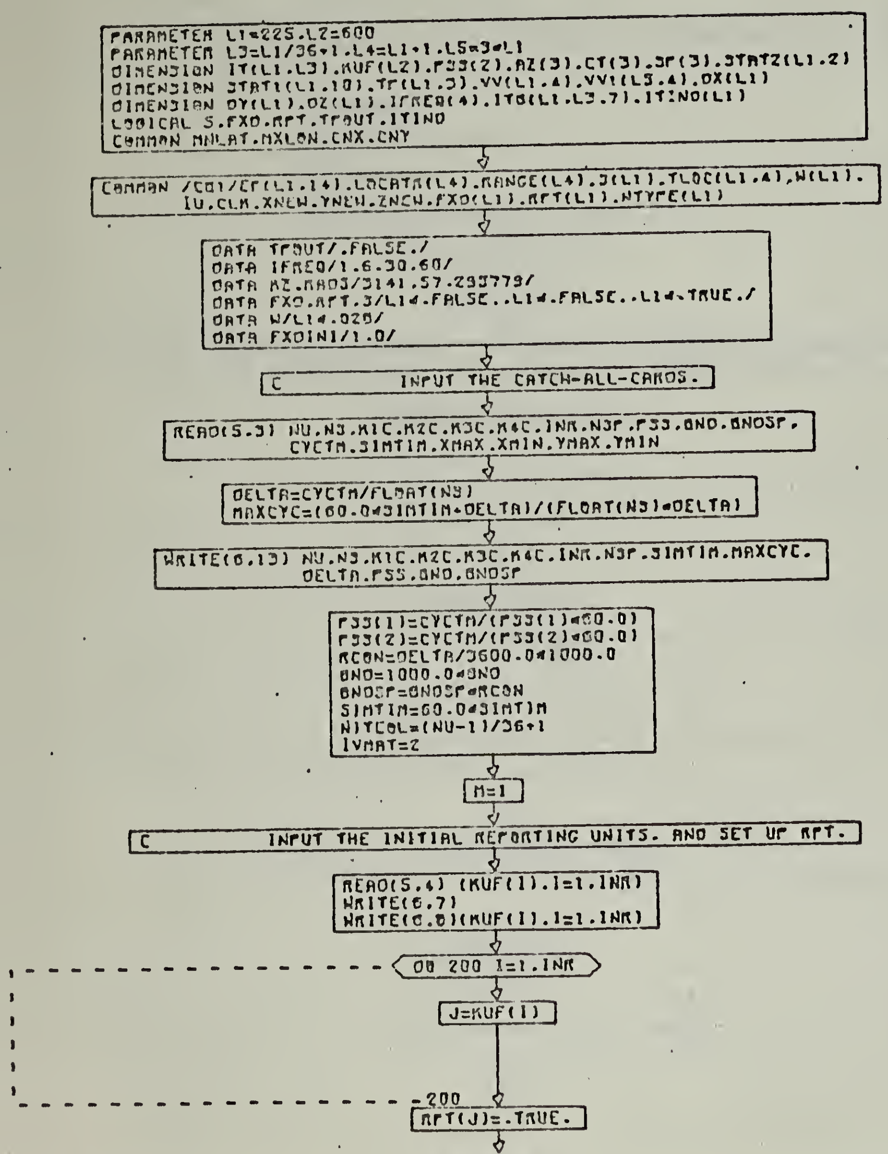


CONT. ON PG 6

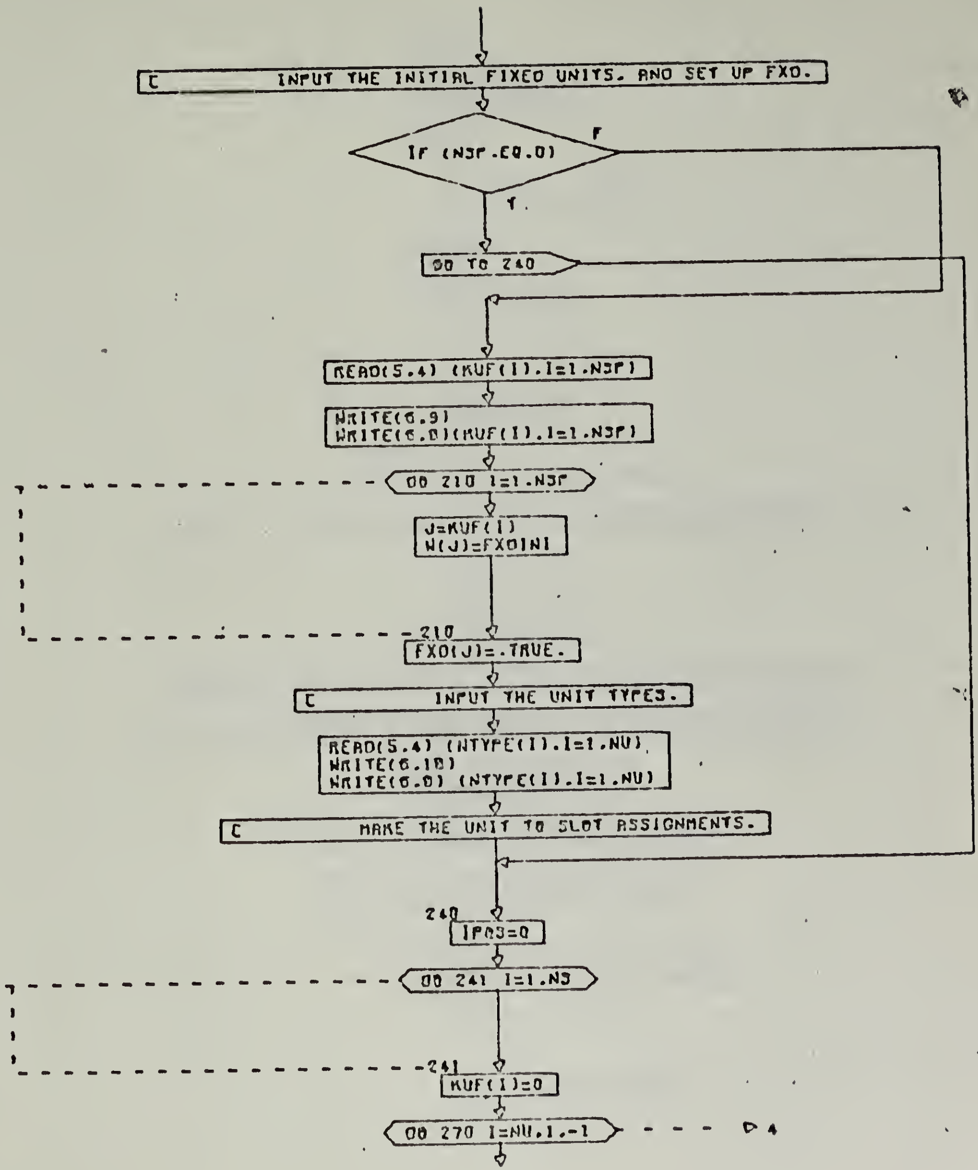


CONT. ON PG 7

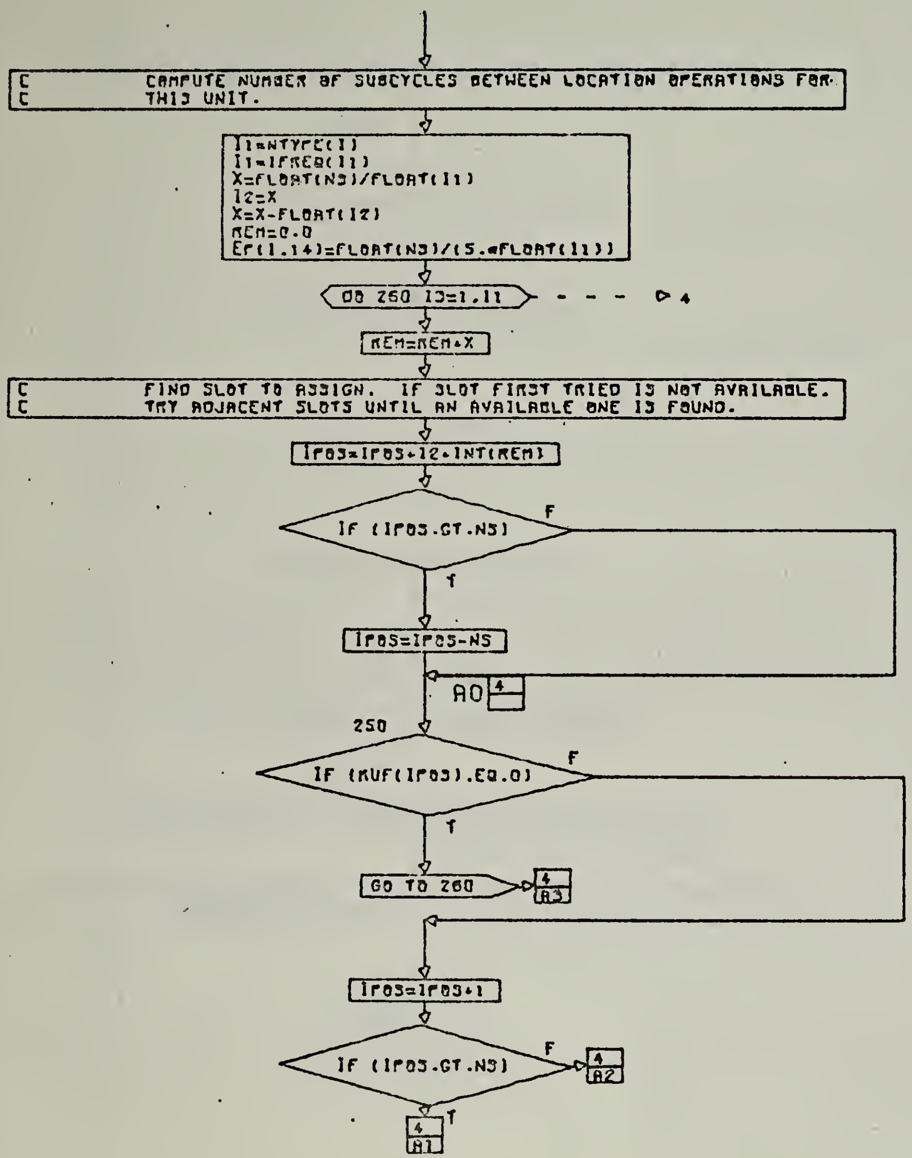

GO TO 930
A9
920
WRITE(6,3) LAT,LON
B0
930
STOP
1
FORMAT(/' BAD TAPE READ. LAST READ WAS'2I10)
2
FORMAT(/' BAD DRUM WRITE. CURRENT IS'2I10)
3
FORMAT(/' EOF ON DRUM. CURRENT IS'2I10)
END



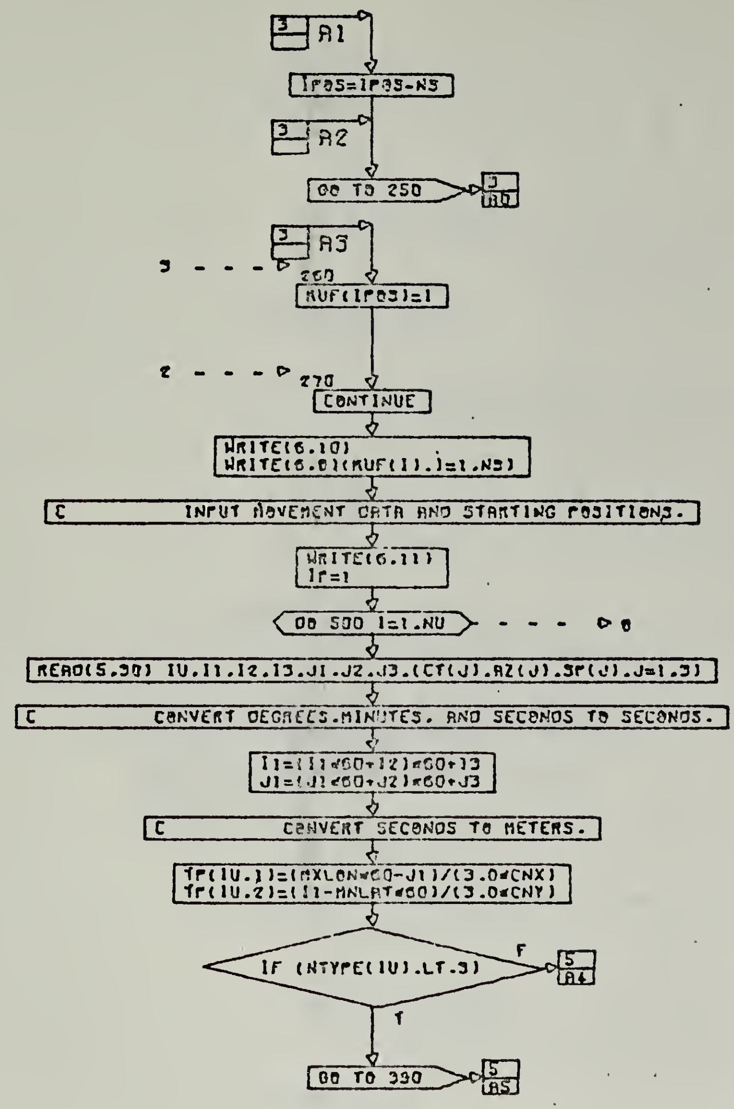
CONT. ON PG 2



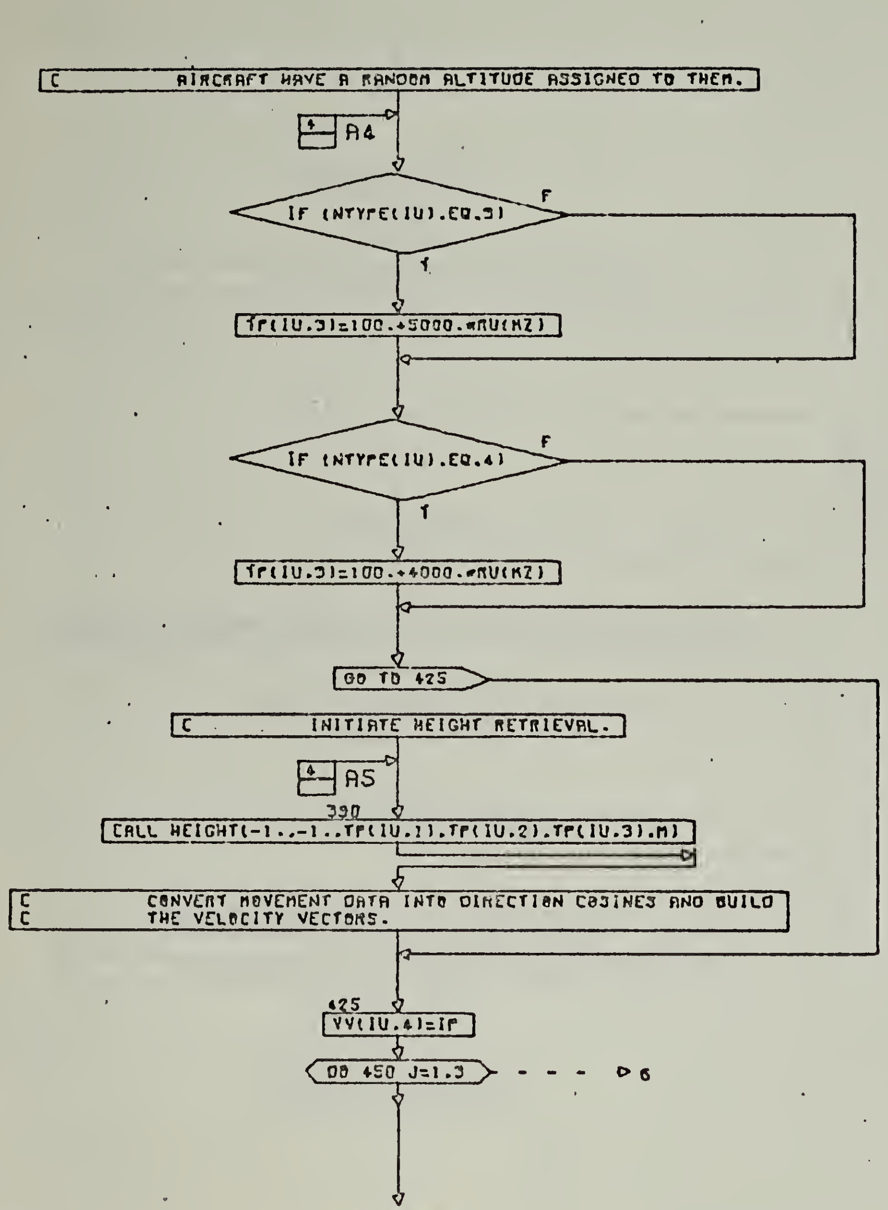
CONT. ON PG 3



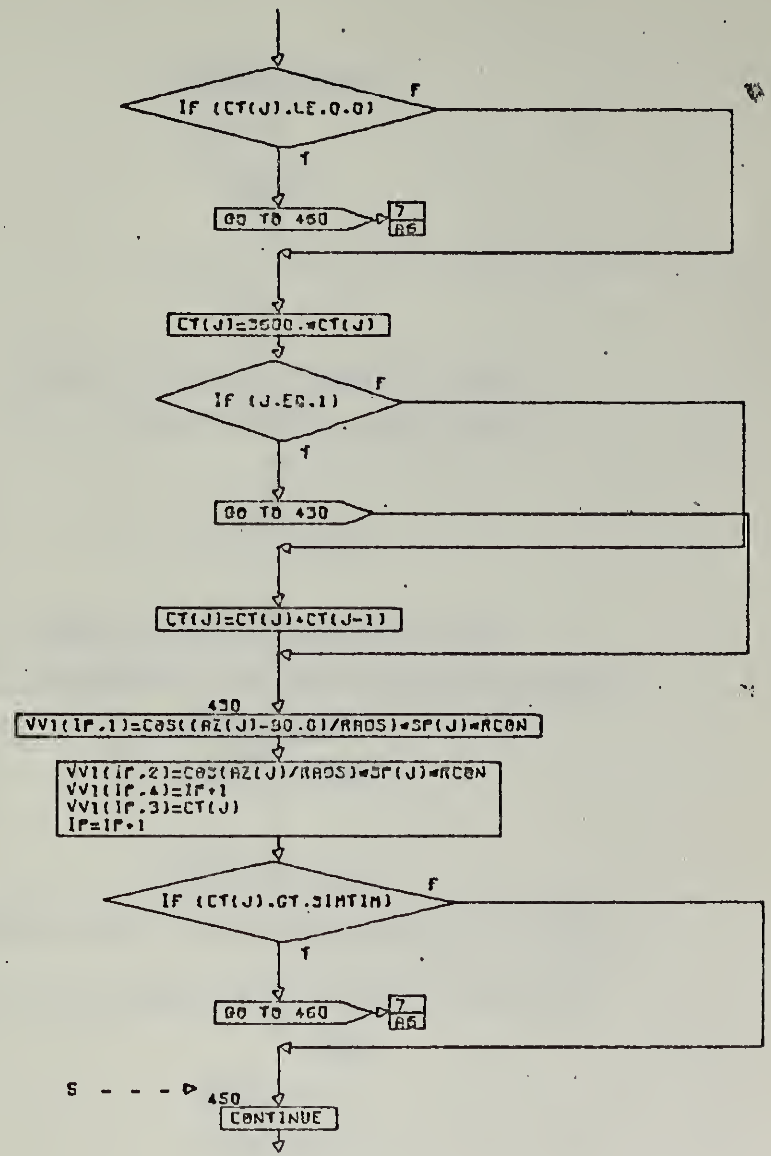
CONT. ON PG 4



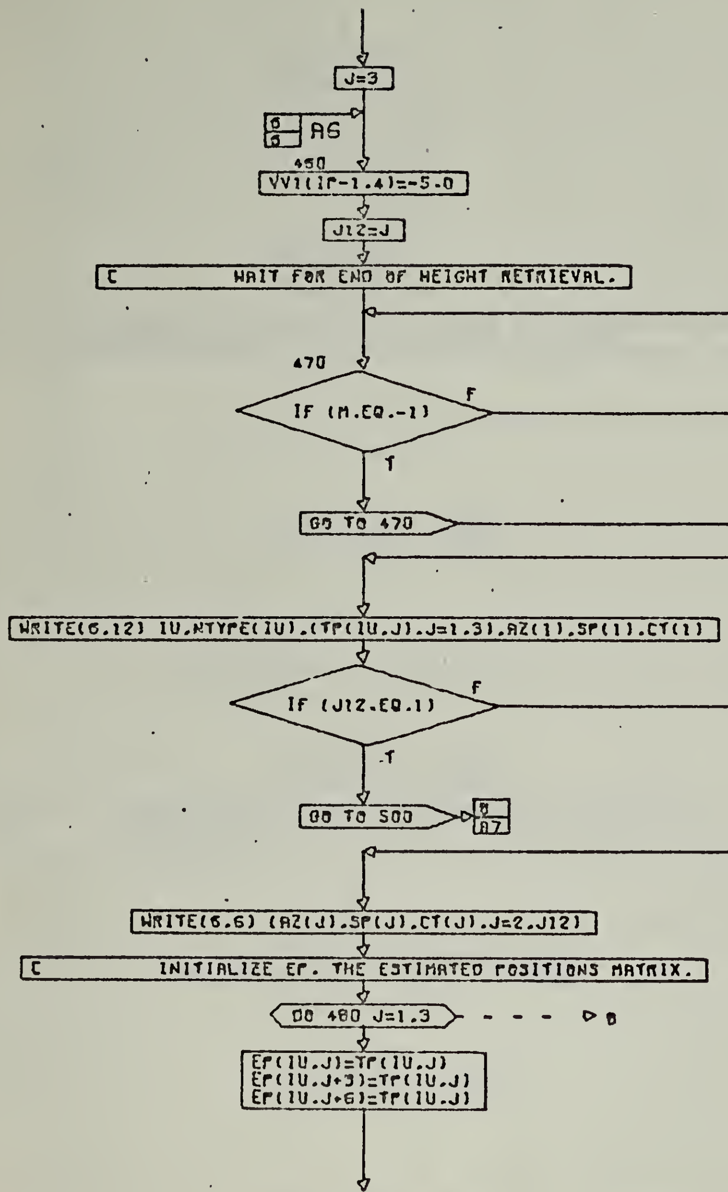
CONT. ON PG 5



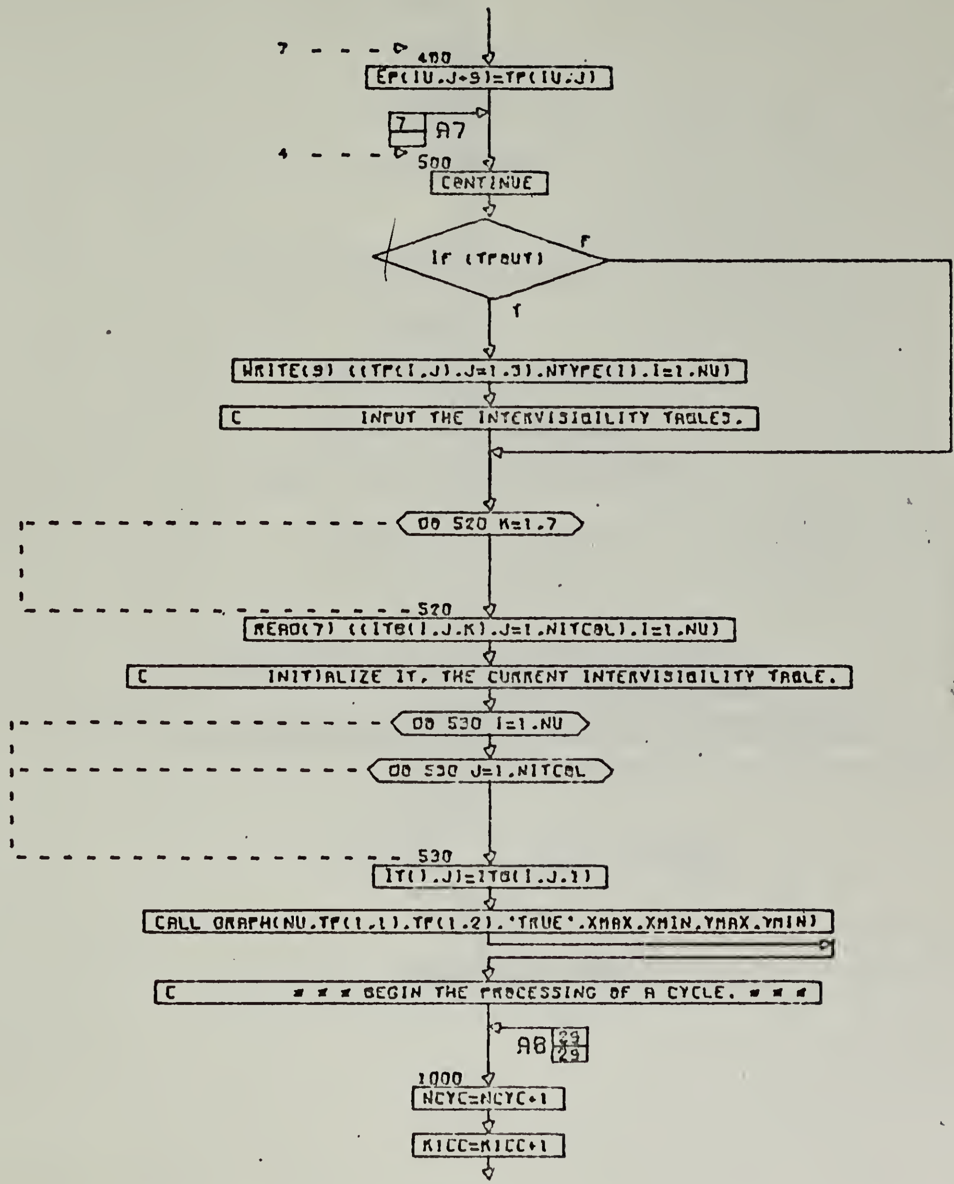
CONT. ON PG 6



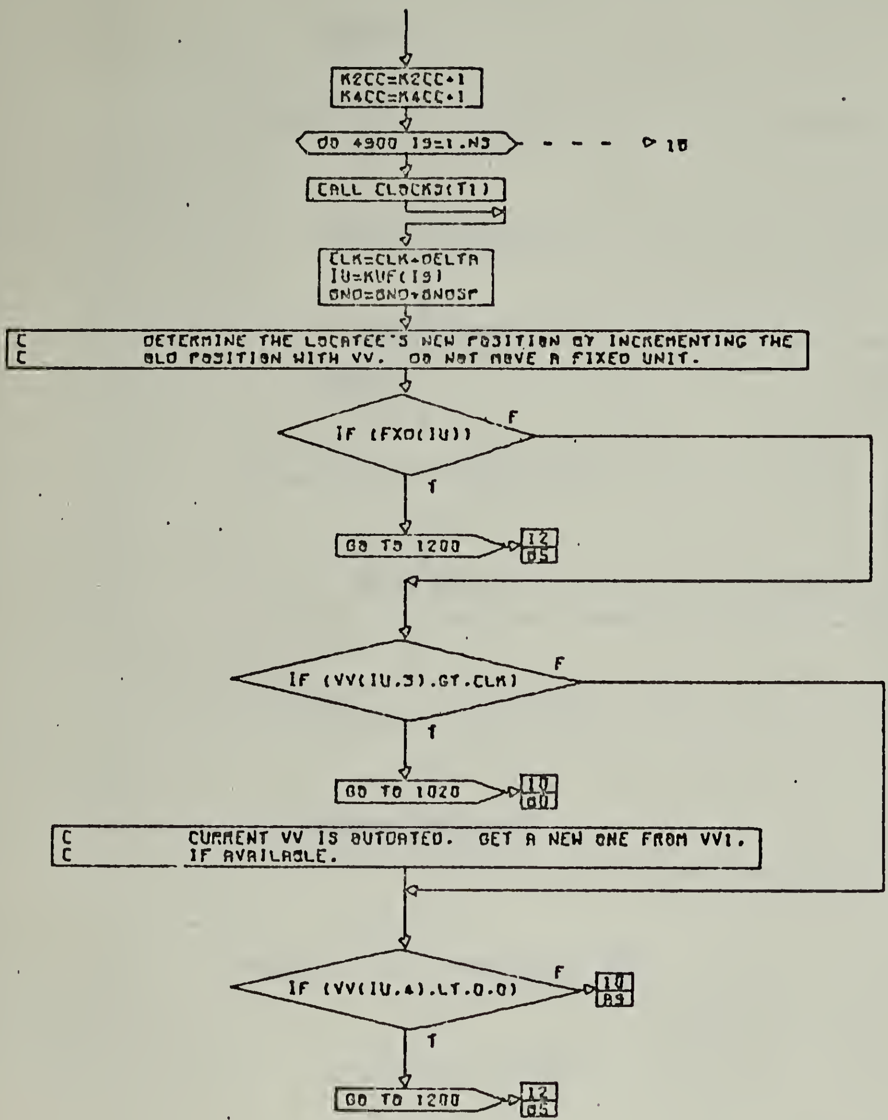
CONT. ON PG 7



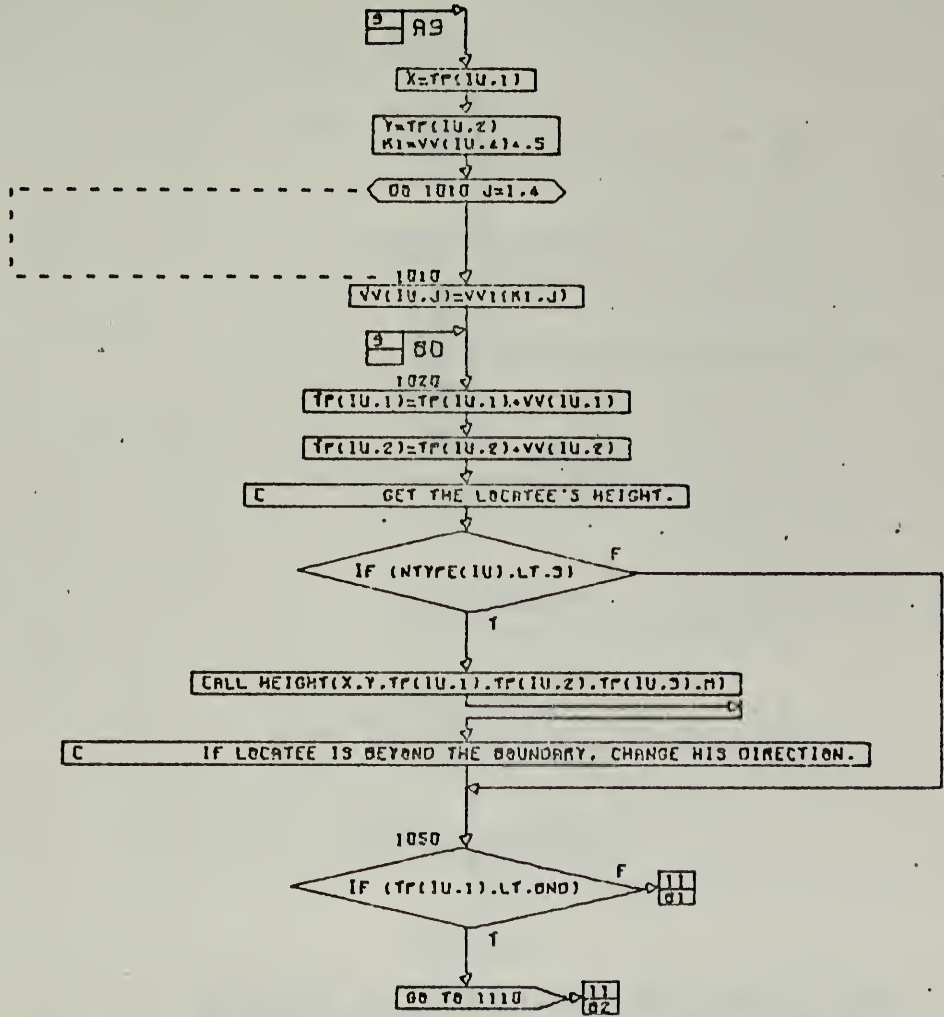
CONT. ON PG 8



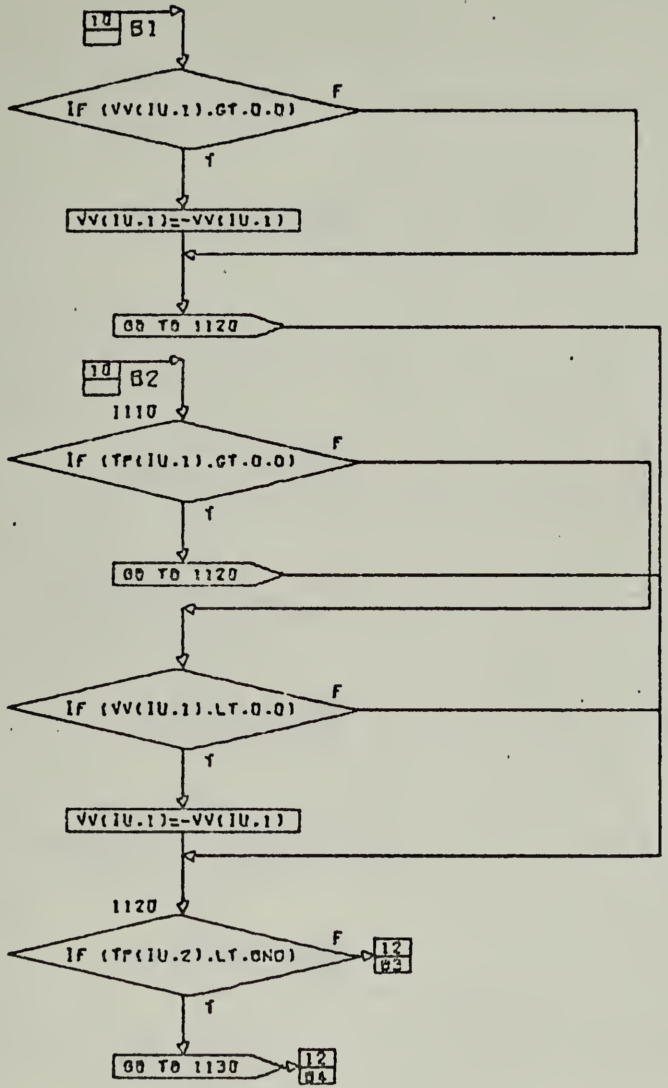
CONT. ON PG 9



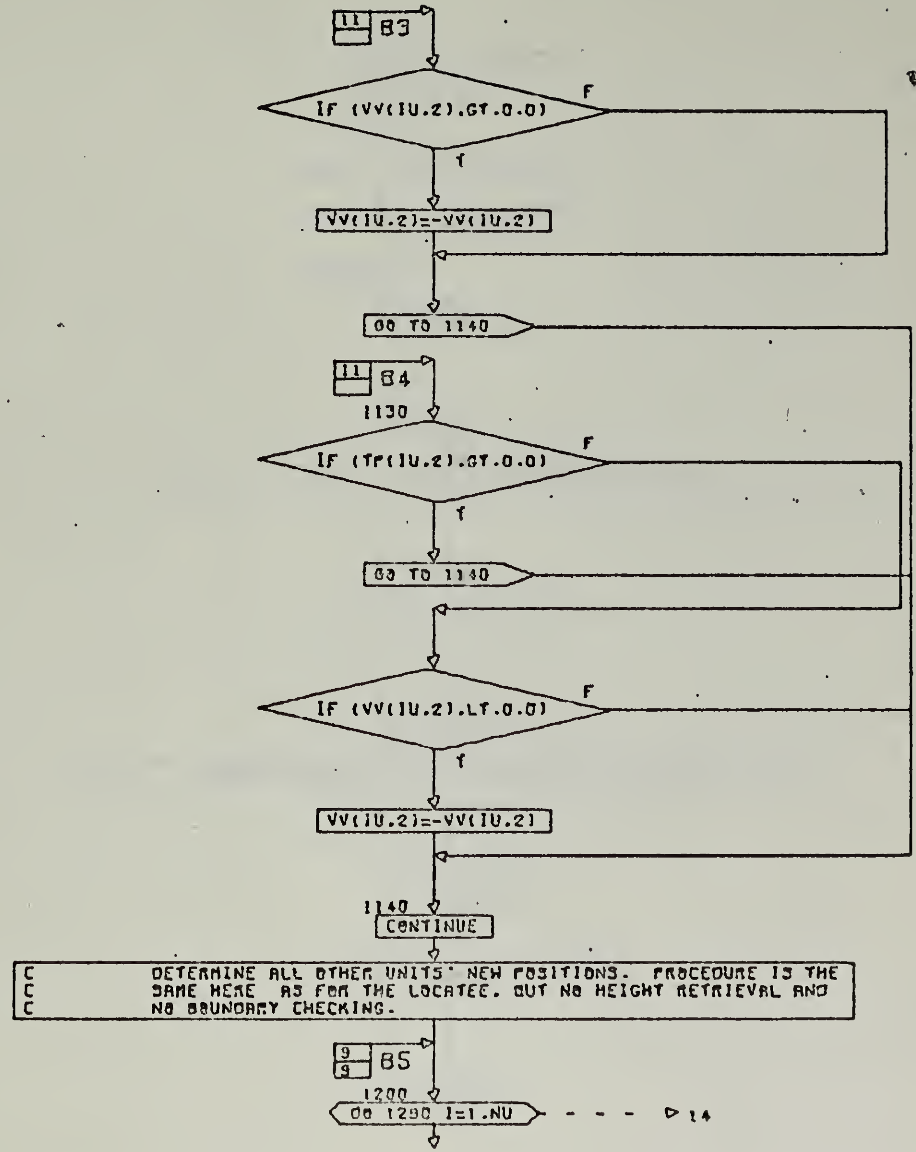
CONT. ON PG 10



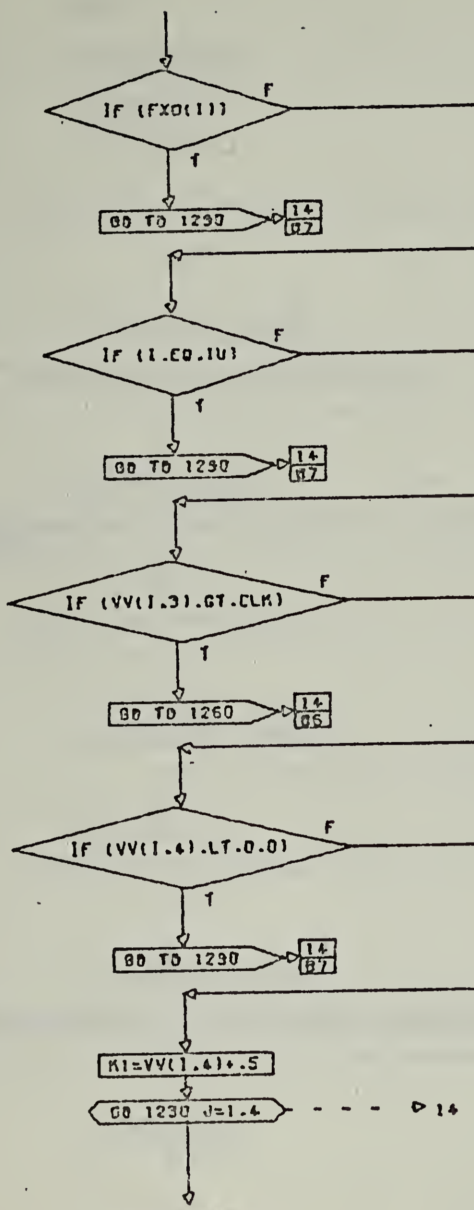
CONT. ON PG 11



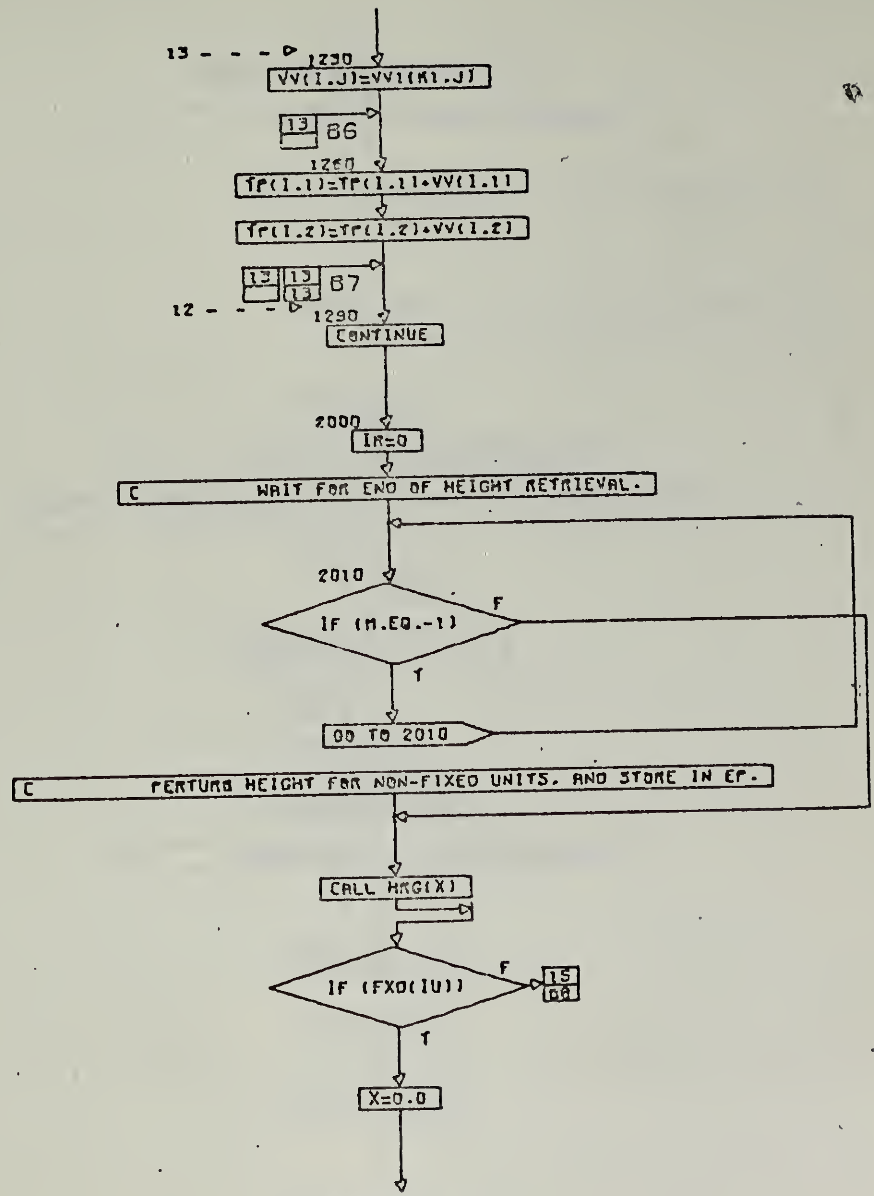
CONT. ON PG 12



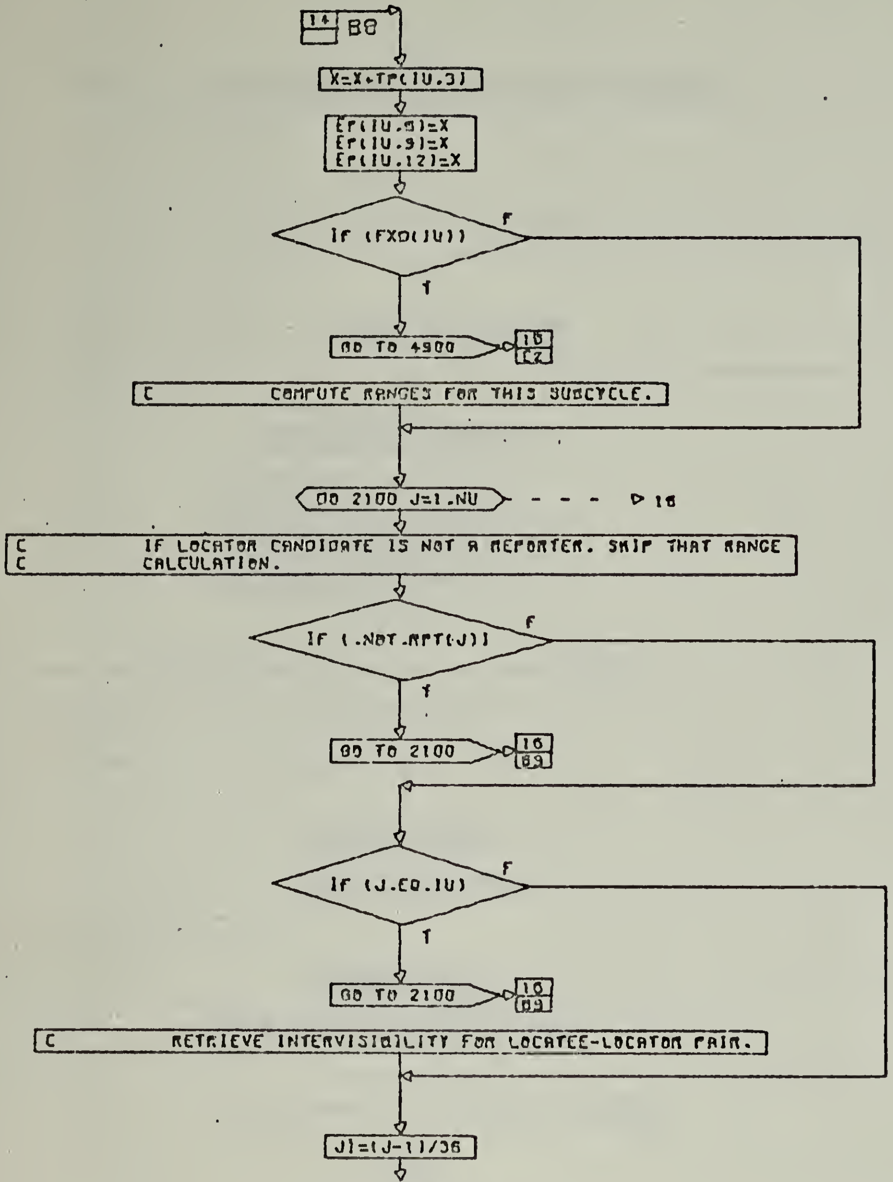
CONT. ON PG 13



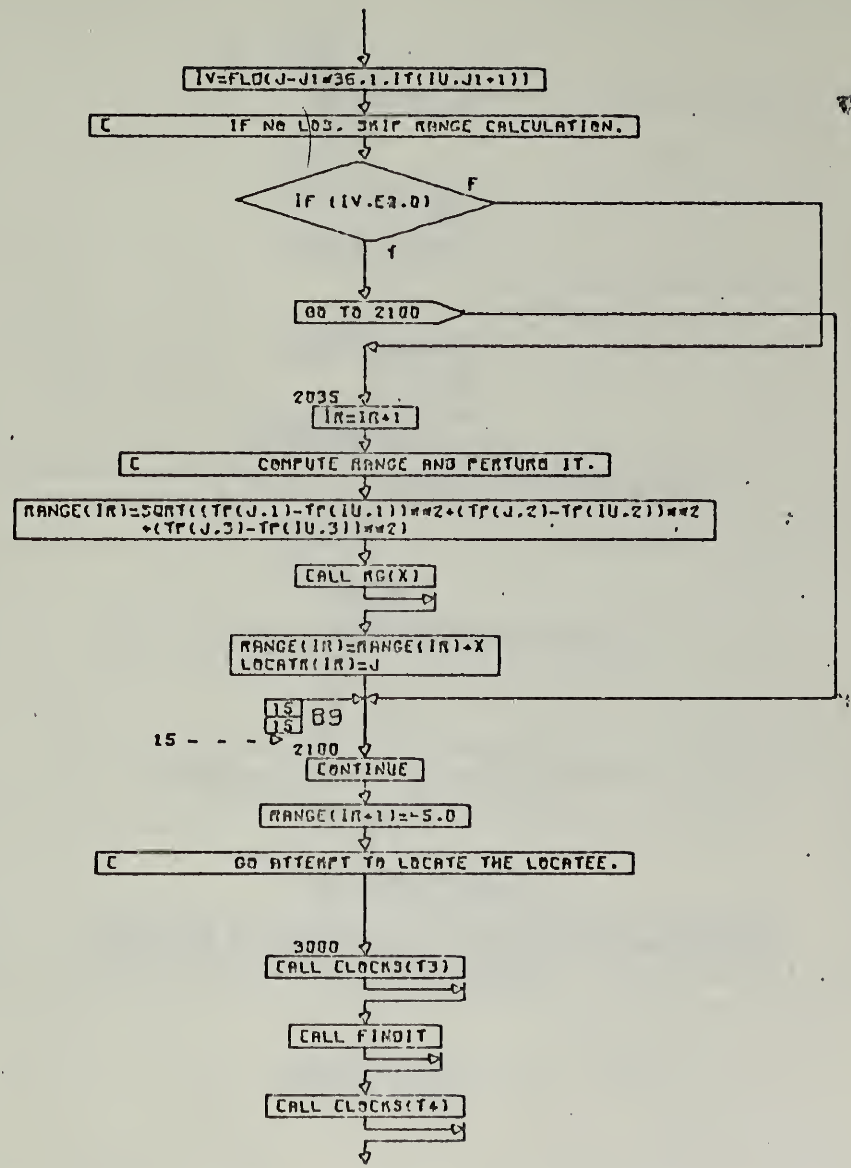
CONT. ON PG 14



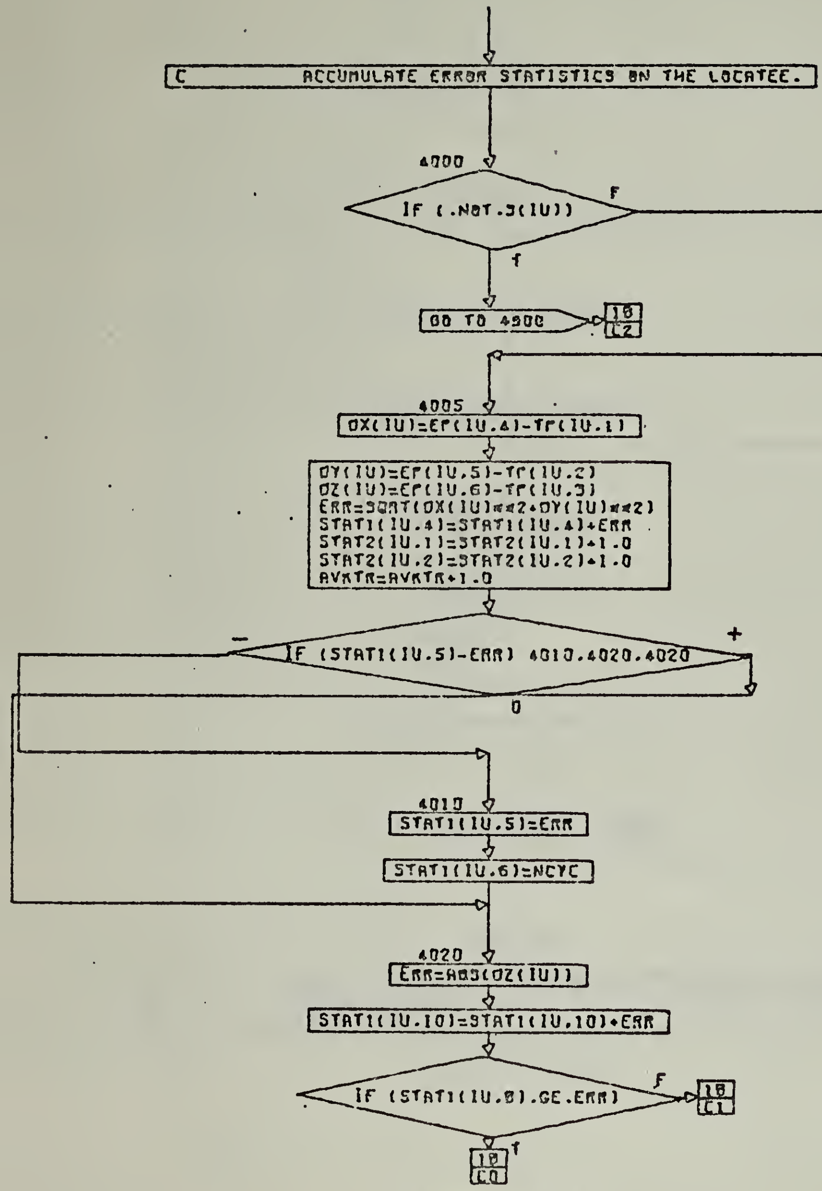
CONT. ON PG 15



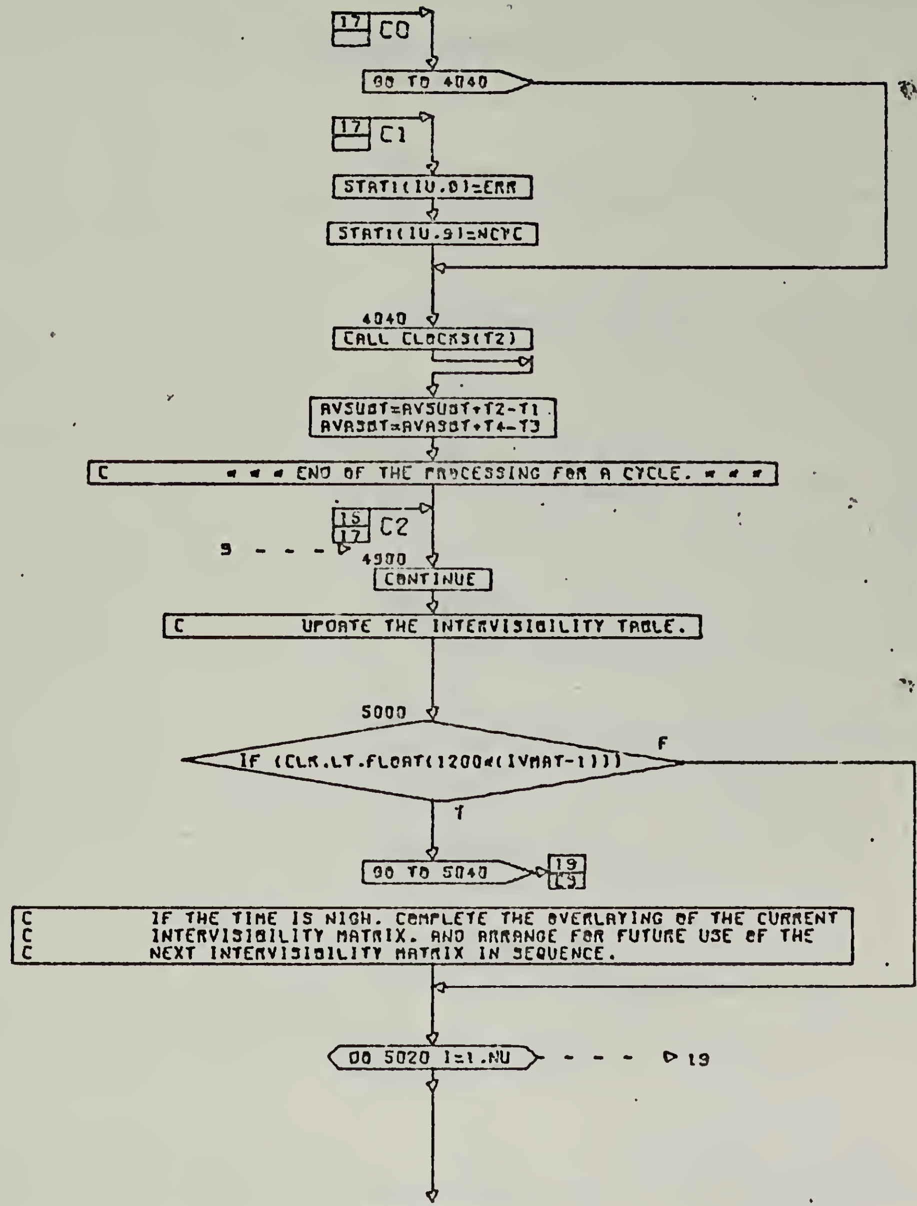
CONT. ON PG 16



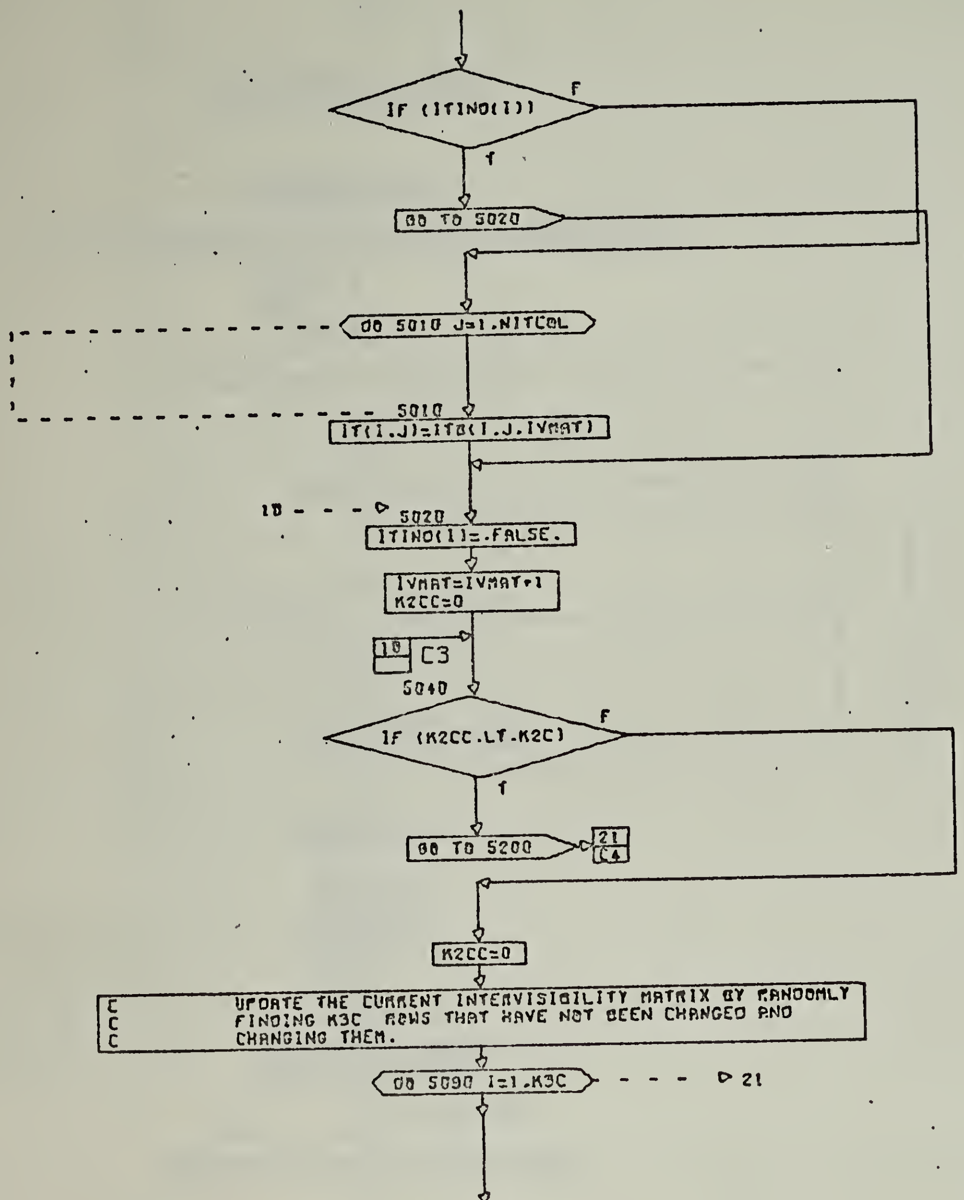
CONT. ON PG 17



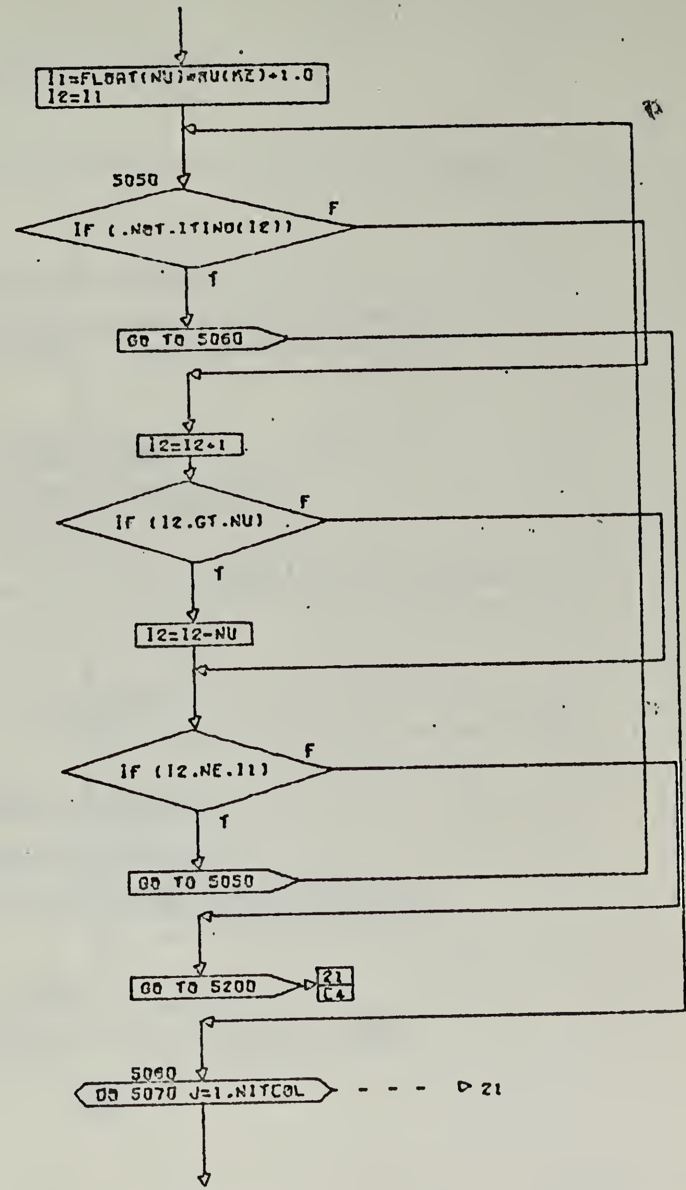
CONT. ON PG 18



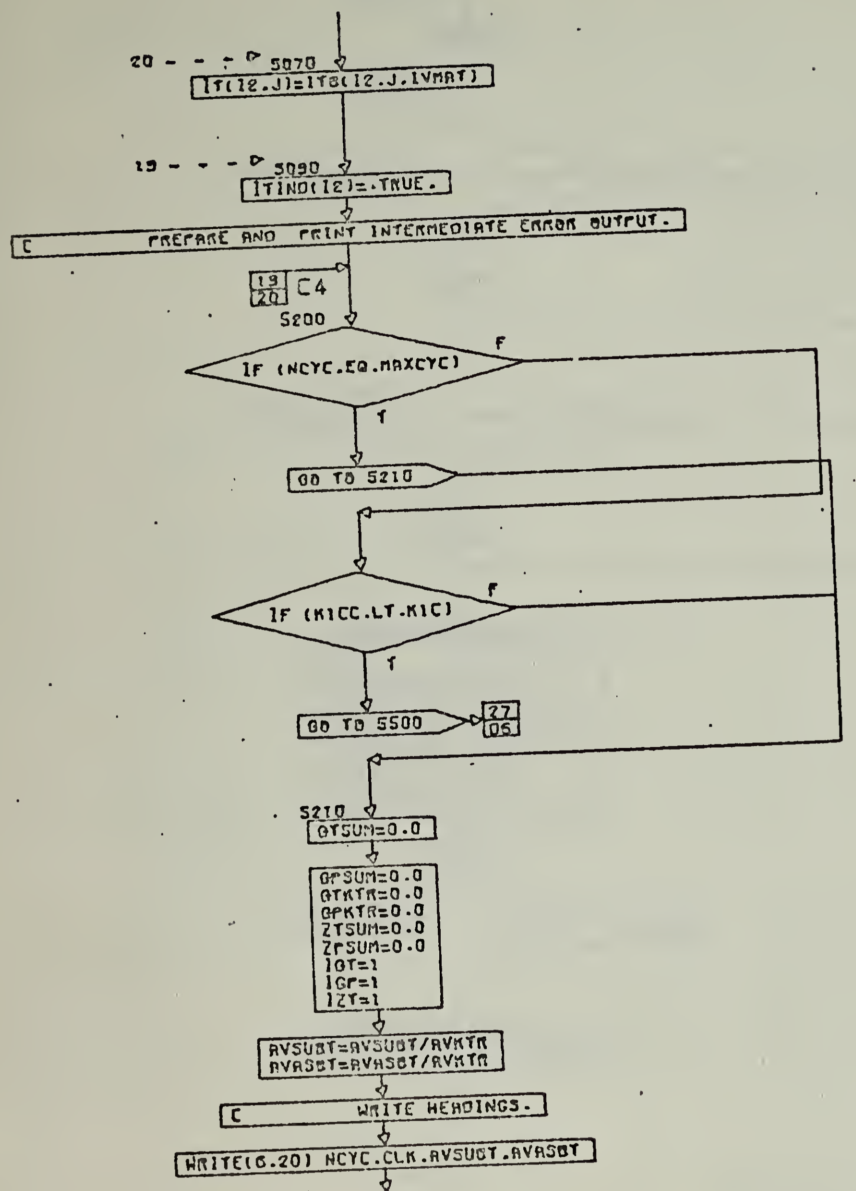
CONT. ON PG 19



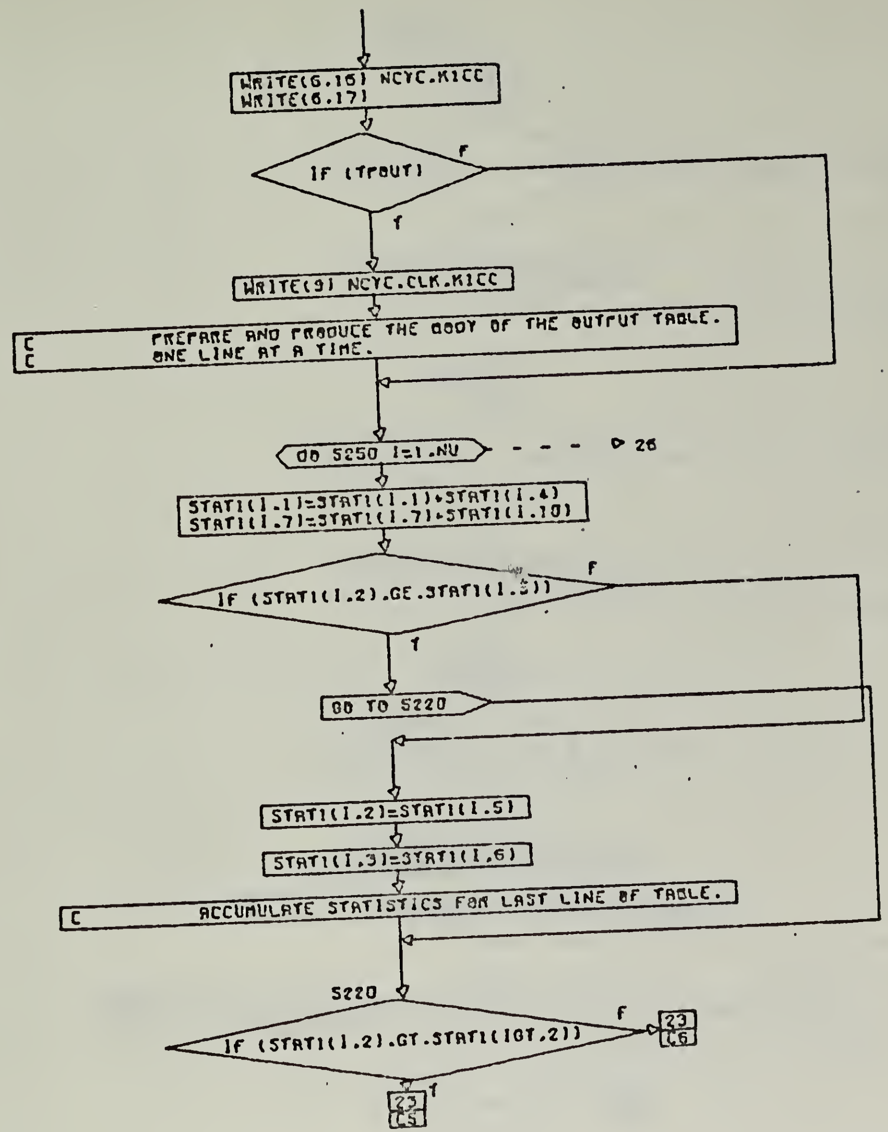
CONT. ON PG 20



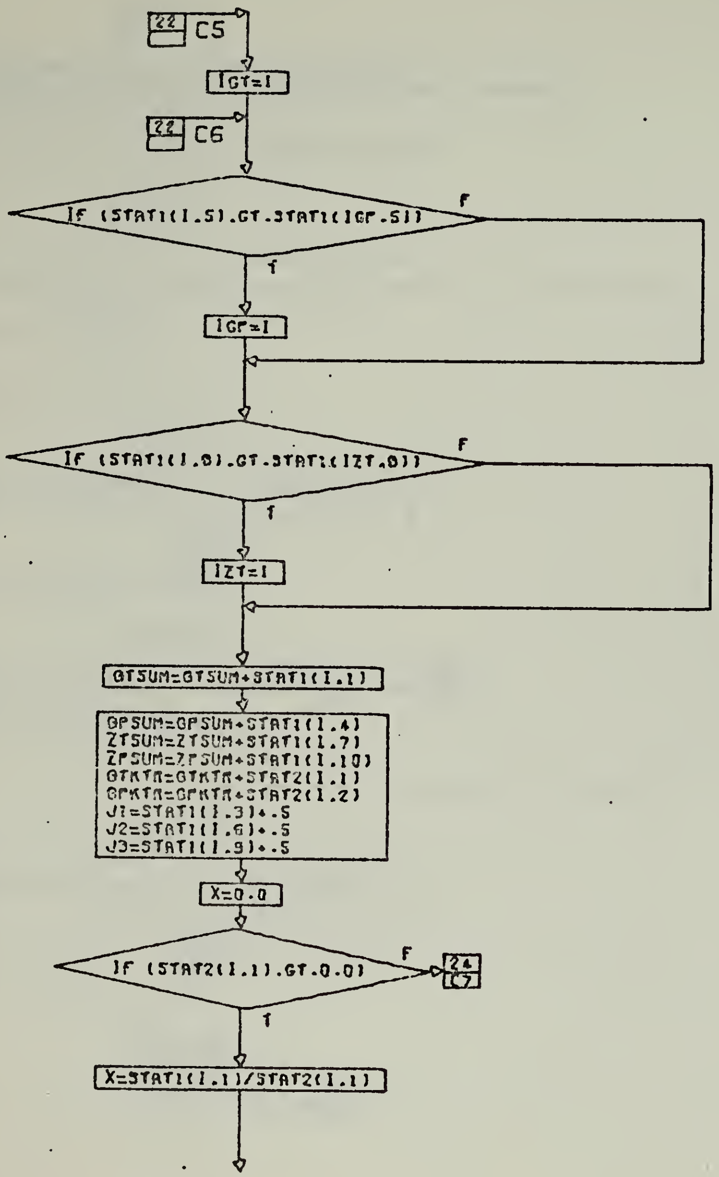
CONT. ON PG 21



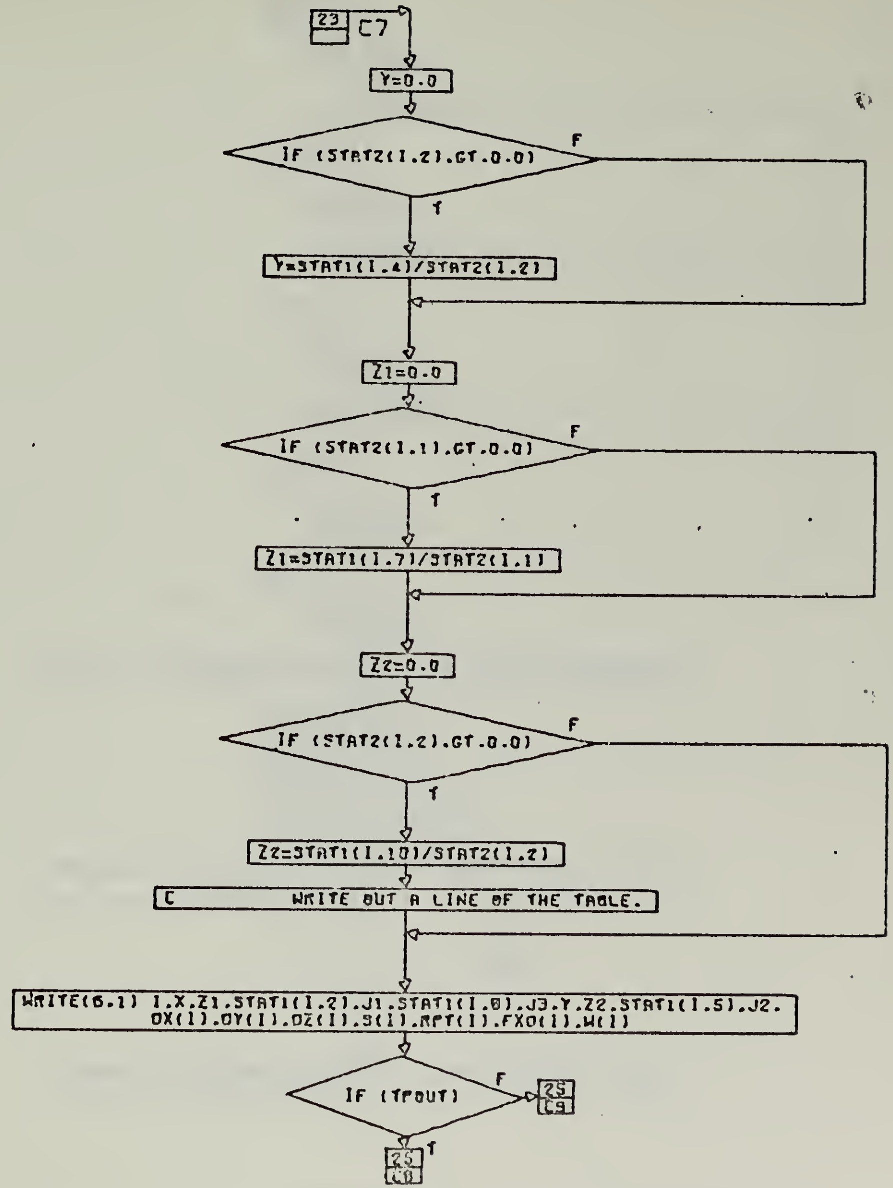
CONT. ON PG 22



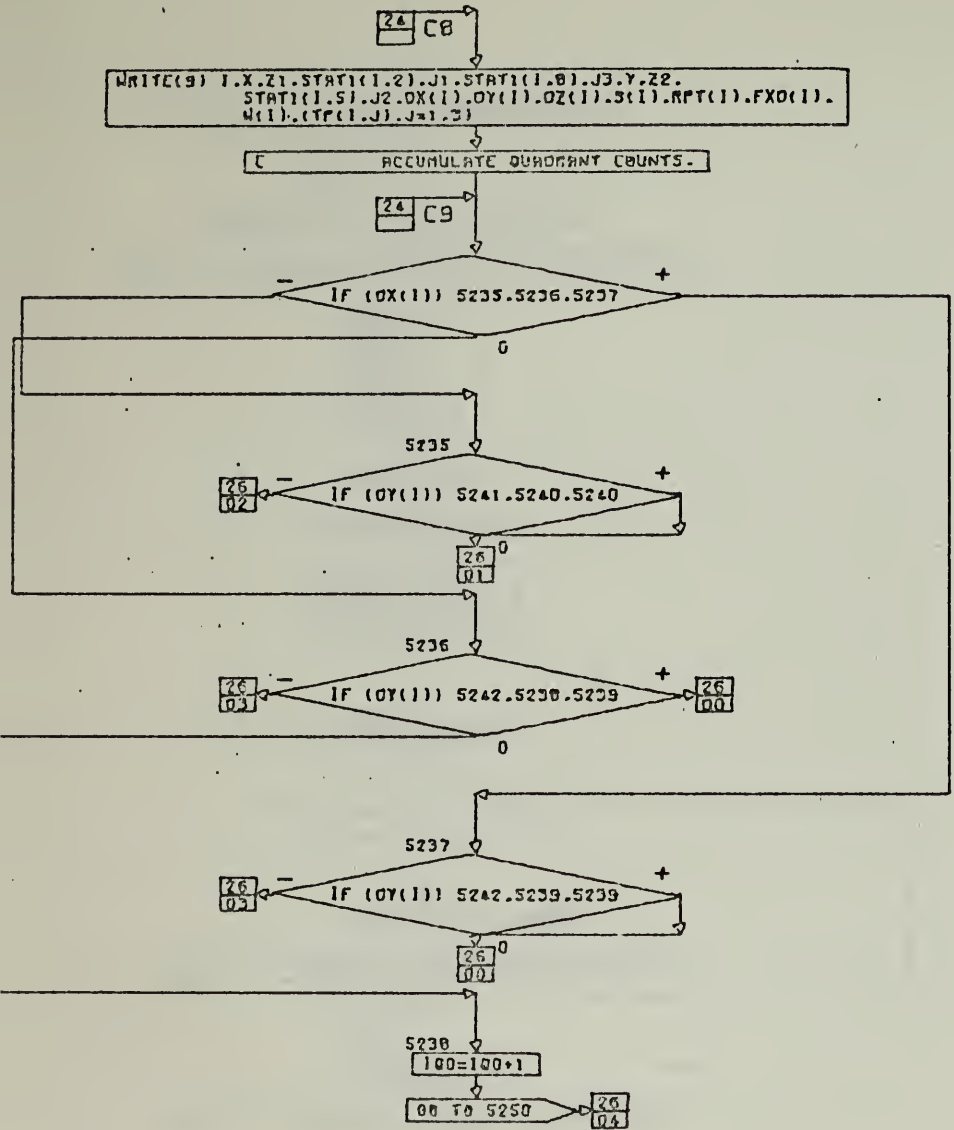
CONT. ON PG 23



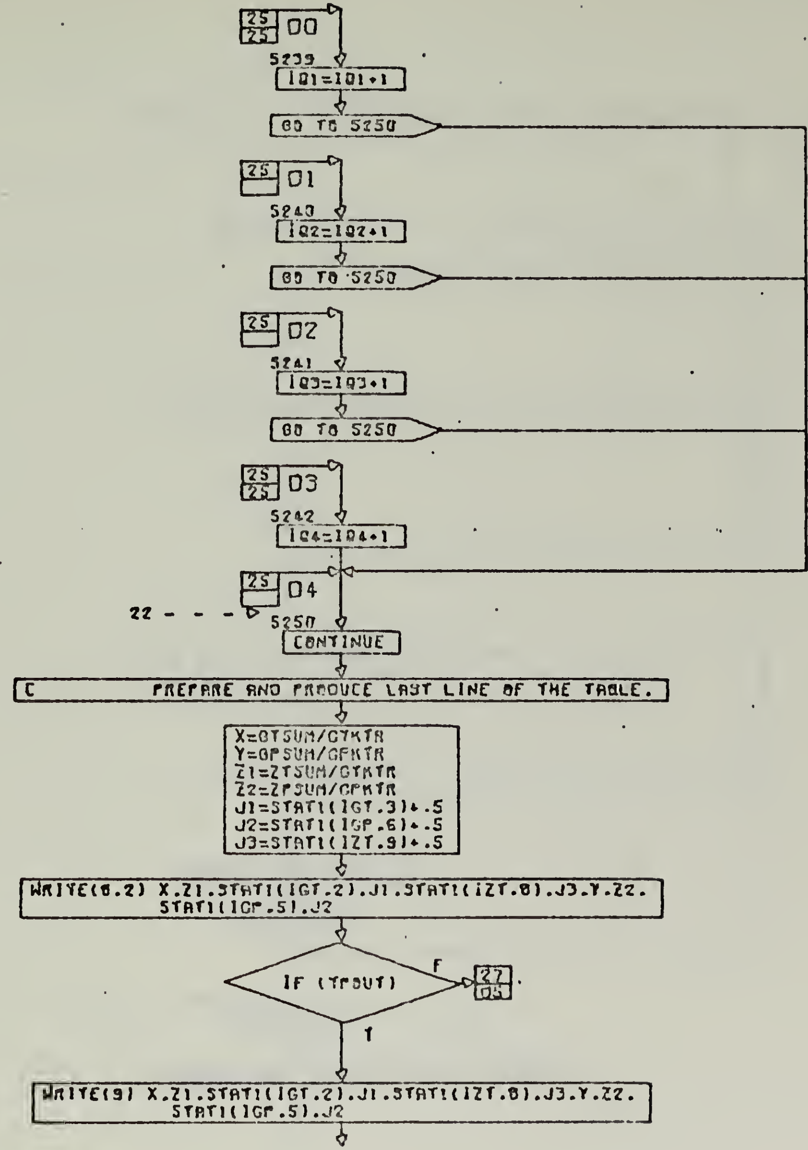
CONT. ON PG 24



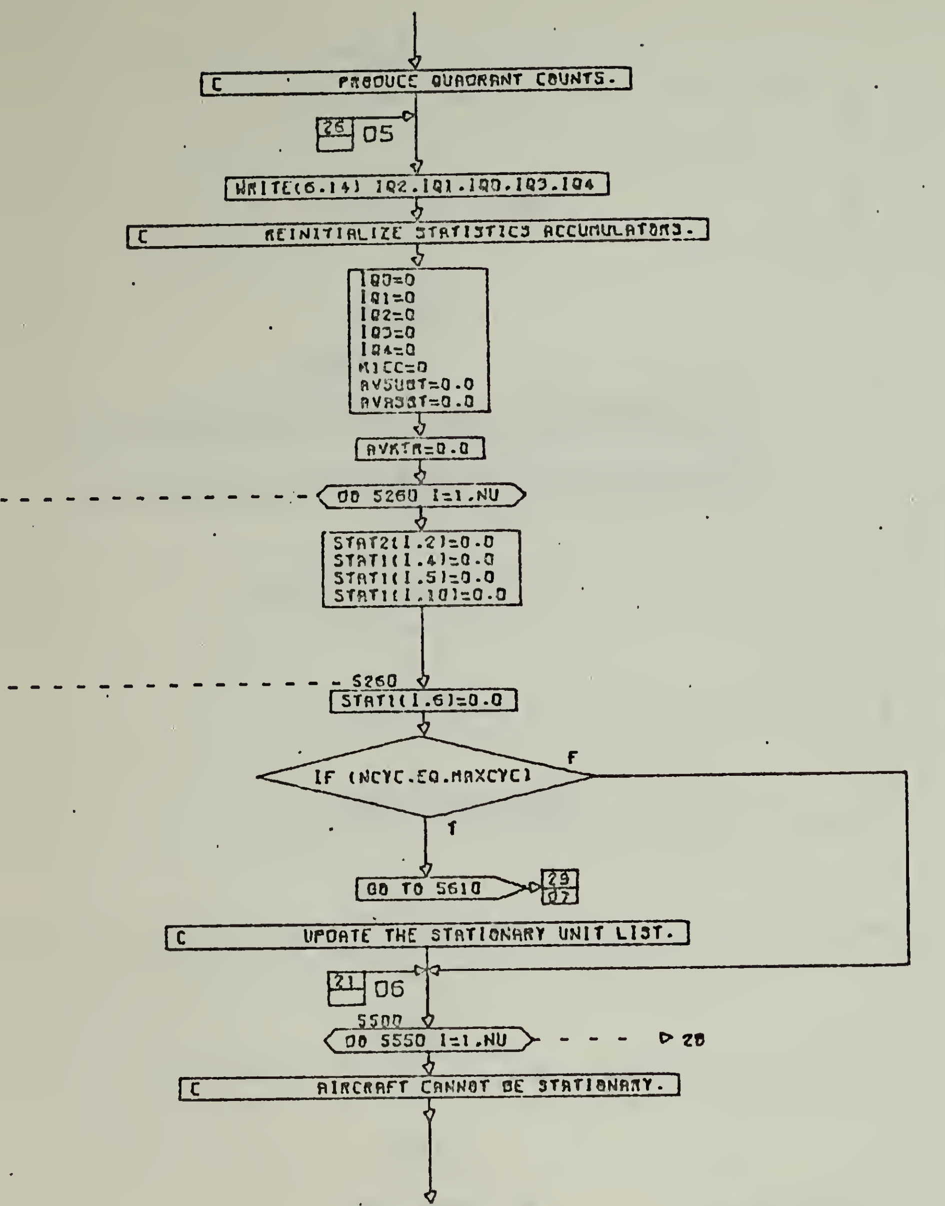
CONT. ON PG 25



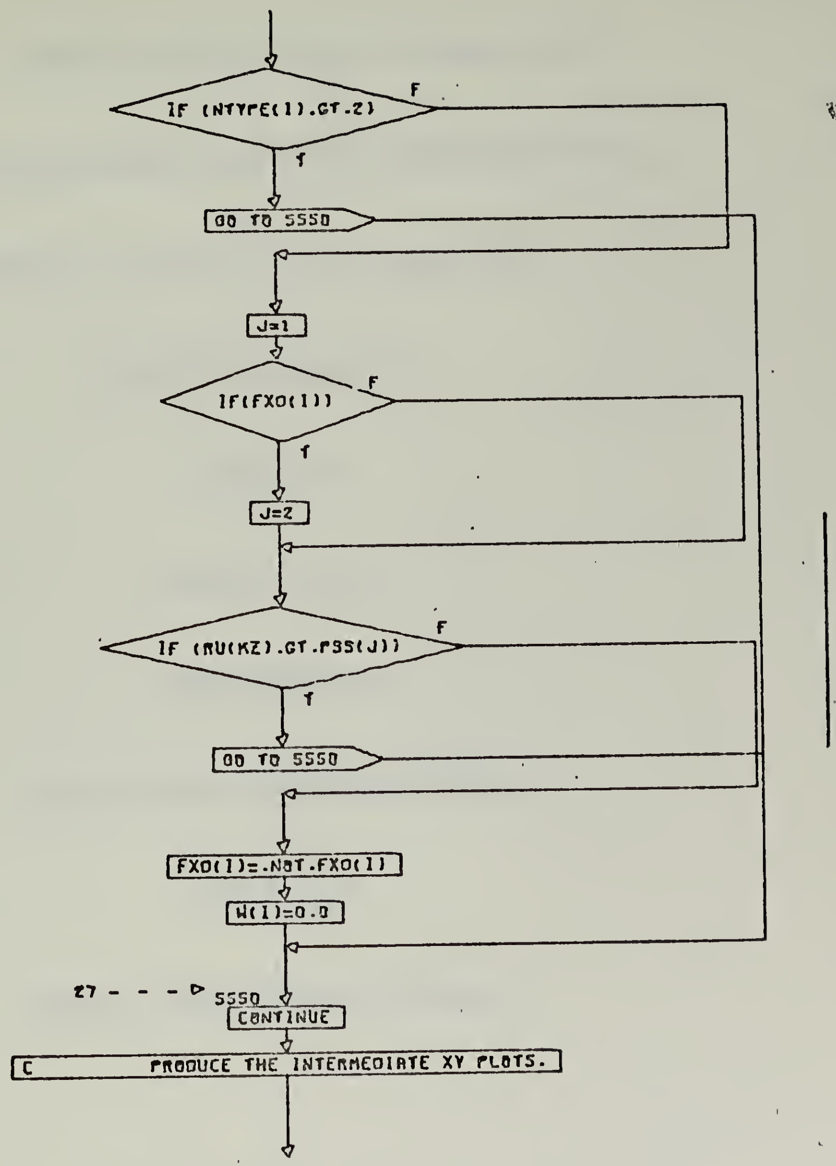
CONT. ON PG 26



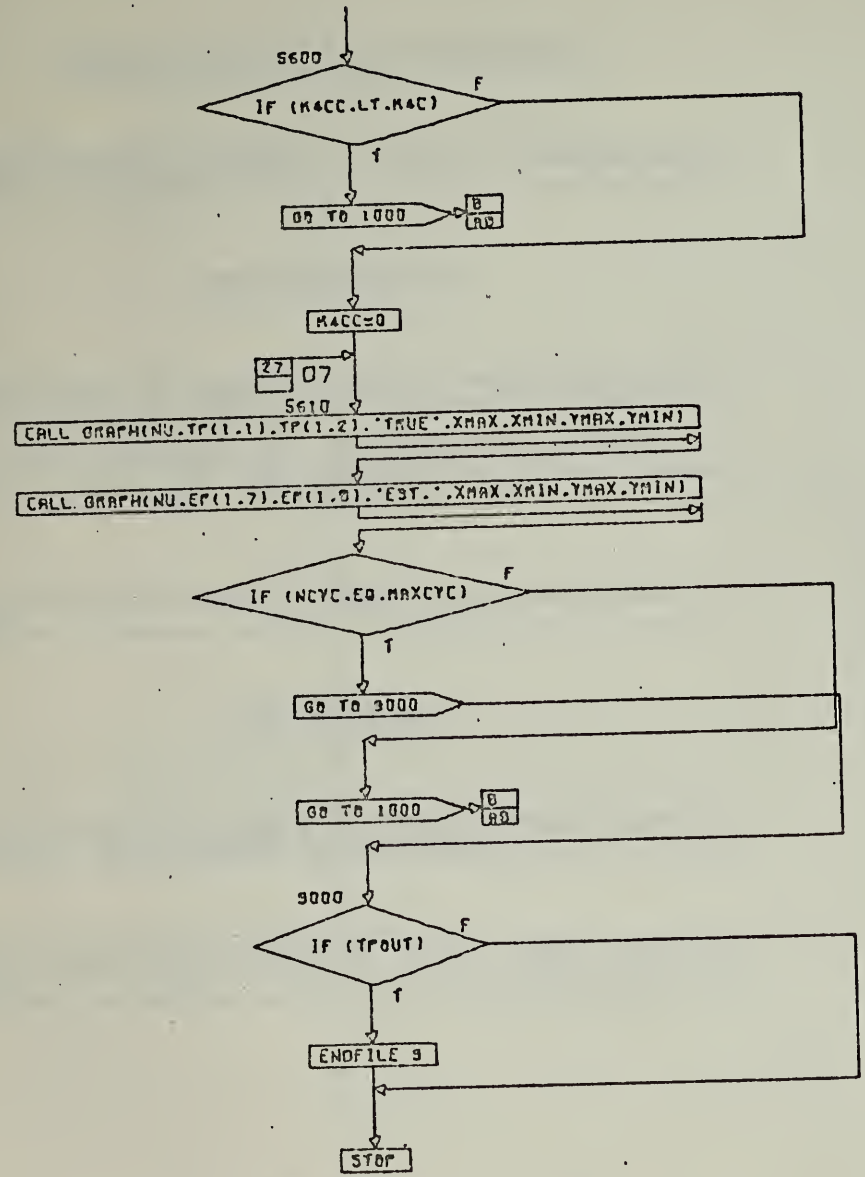
CONT. ON PG 27



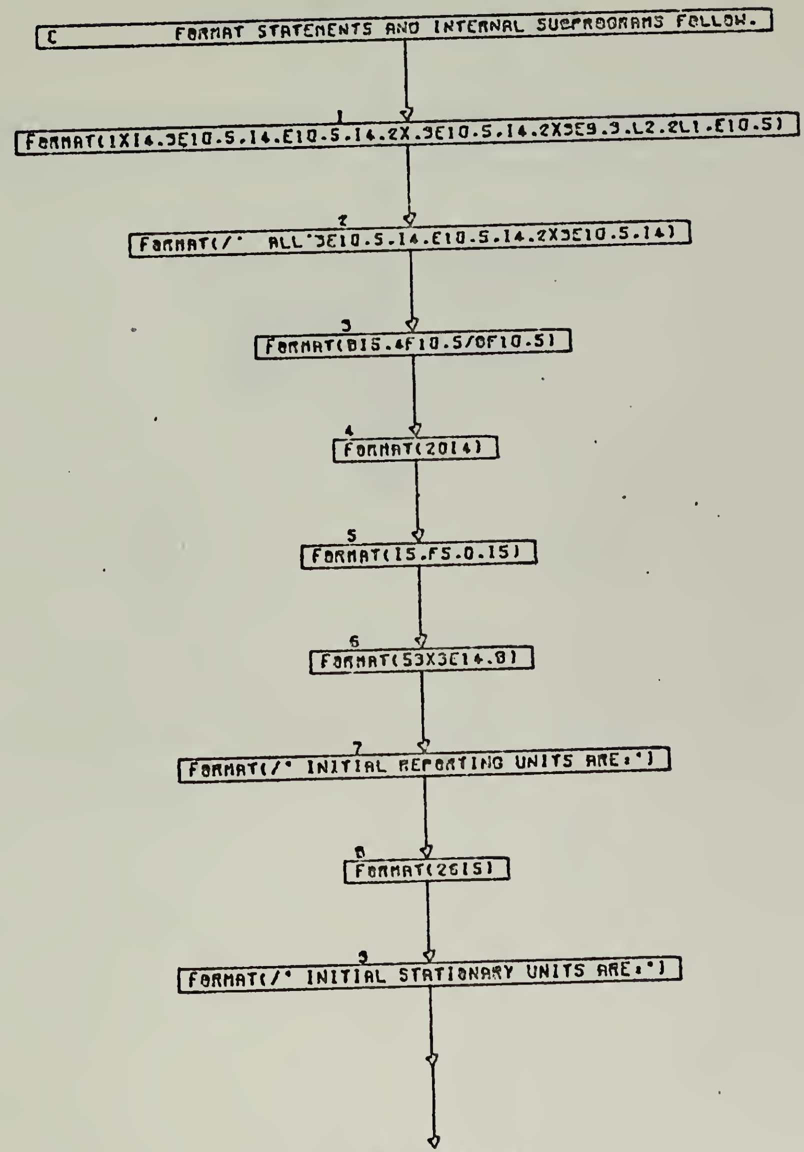
CONT. ON PG 28



CONT. ON PG 29



CONT. ON PG 30

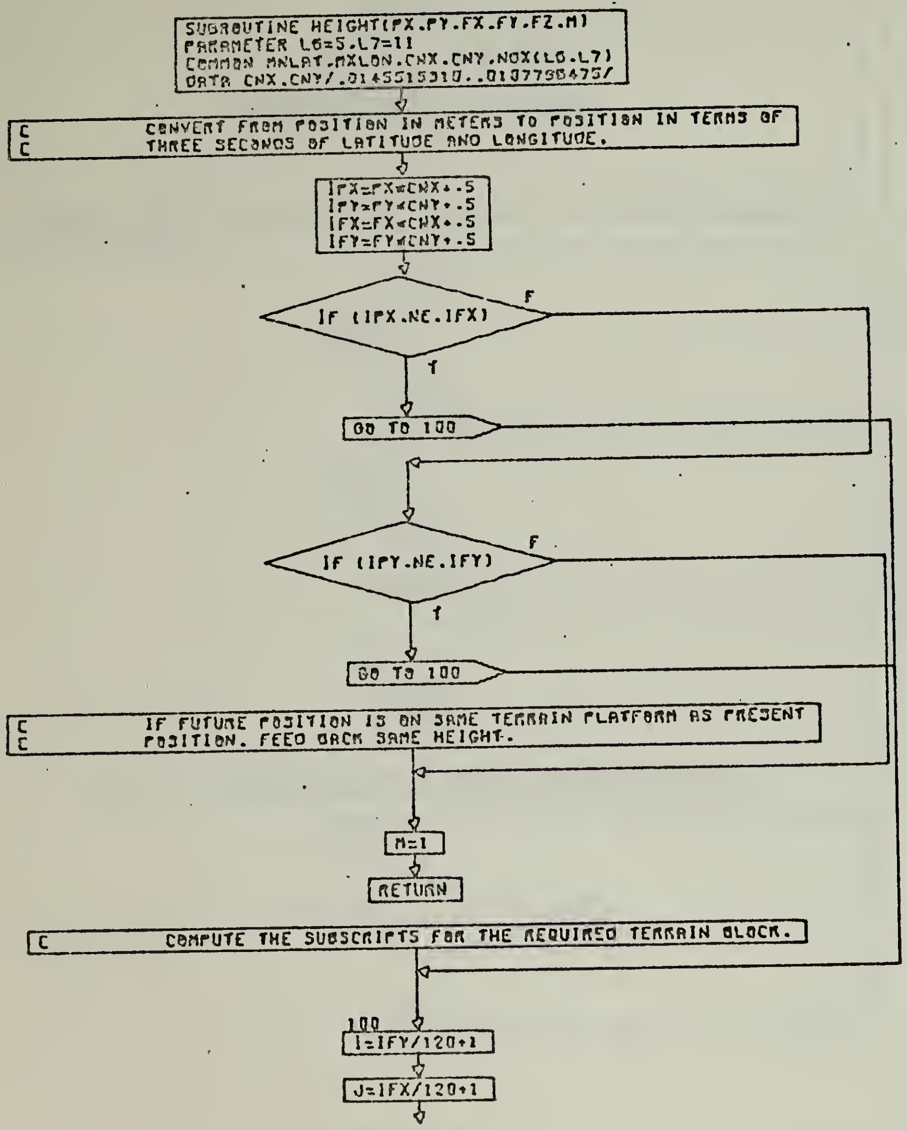


CONT. ON PG 31

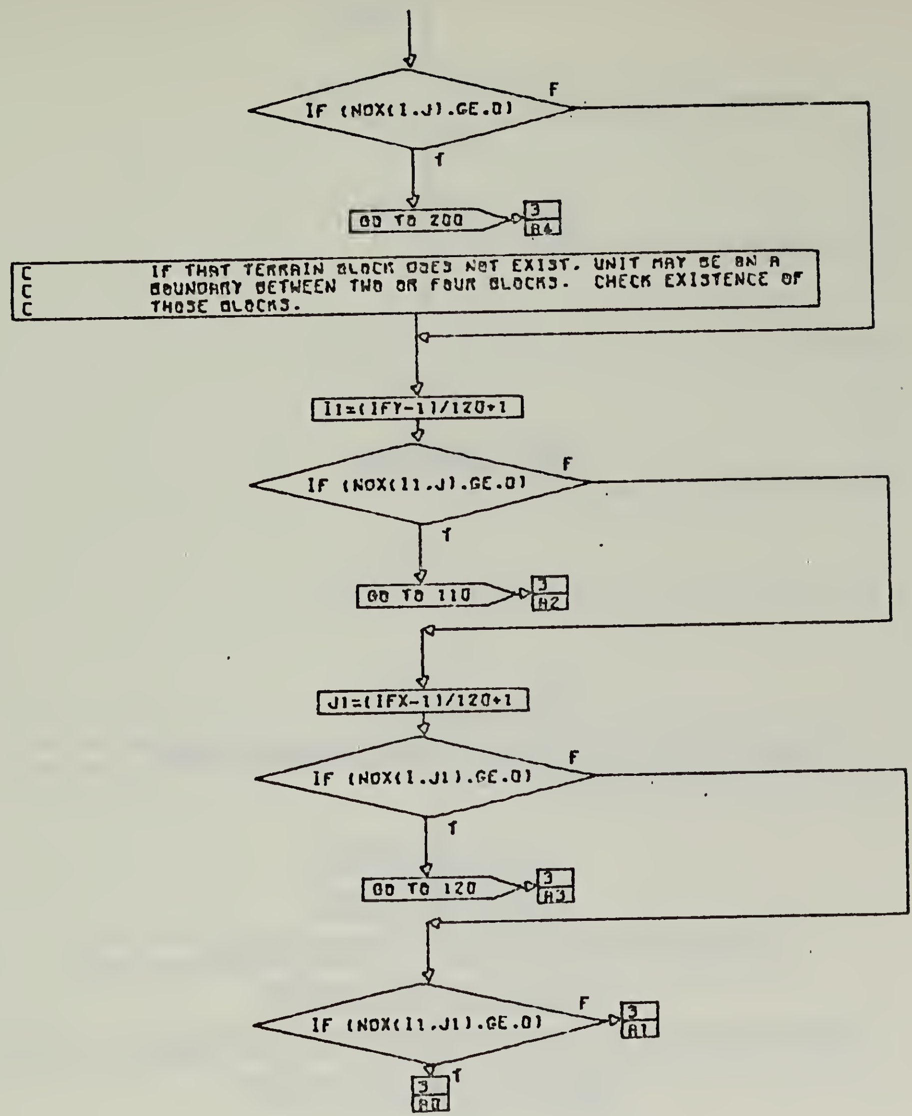
```
10   FORMAT(/' ASSIGNMENT OF UNITS TO SLOTS I3:')

11   FORMAT(/' UNIT  TYPE   STARTING X     STARTING Y     STARTING Z'6X'A
     ZIMUTH'8X'SPEED'6X'CHANGE TIME')

12   FORMAT(I5.I6.6E14.0)

13   FORMAT(' NUMBER  OF  UNITS ='I9/' NUMBER  OF  SLOTS ='I9/' OUTPUT
        COUNTER ='I9/' IV UPDATE COUNTER ='I9/' ROWS IN IV UPDATE ='I9/
     ' XY  PLOT  COUNTER ='I9/
     ' MAX NUMBER RANGES ='I9/' NUM.  STAT. UNITS ='I9/' SIMULATION   T
     IME ='F9.2/' MAX NUMBER CYCLES ='I9/' SUBCYCLE  DELTA T ='F9.4/' A
     VG  TIME  MOVING ='F9.2/' AVG  TIME STOPPED ='F9.2/' UPPER  BOUNDA
     RIES ='F9.4/' BOUNDARY    SPEED ='F9.2)

14   FORMAT(/53X'QUADRANT COUNTS OF ERRORS'//60XI3.5XI3/64XI3/60XI3.
     5XI3)

15   FORMAT(21(1XSI1))

16   FORMAT(/14X'ERROR STATS. OVER ALL'I4.' CYCLES'11X'ERROR STATS. OVE
     R LAST'I3.' CYCLES'4X'CURRENT COORDINATE ERRORS')

17   FORMAT(' UNIT  XY AVG'5X'Z AVG'4X'XY MAX   CYC   Z MAX   CYC'4X'XY
      AVG'5X'Z AVG'4X'XY MAX   CYC'5X'EX-TX'4X'EY-TY'4X'EZ-TZ  SAF  WEI
     GHT'/)
```

CONT. ON PG 32

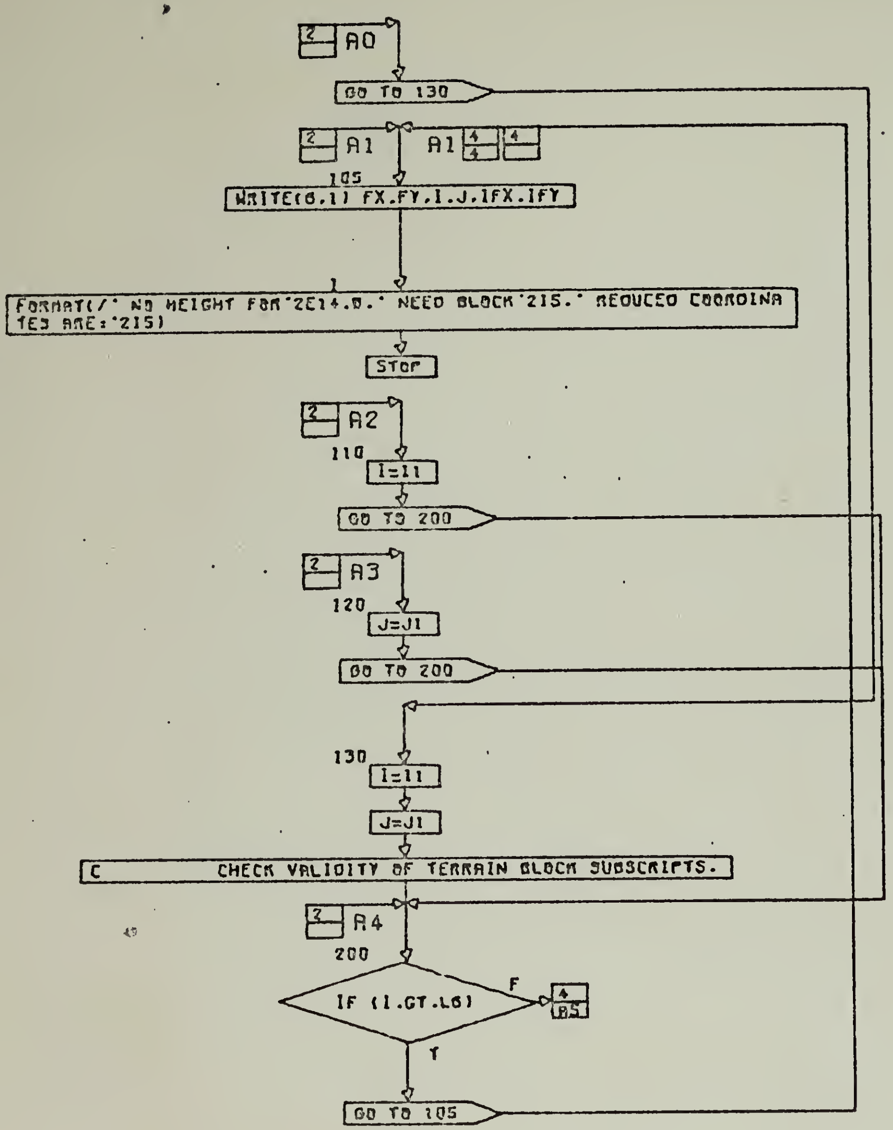

```
18   FORMAT(/' THE UNIT TYPES ARE:')

20   FORMAT(///' CYCLE NUMBER'I5.'.  TIME ='F10.3.' S.  AVERAGE SIMULAT
     ION SUBCYCLE TIME ='F7.4.' S.  AVERAGE ALGORITHM SUBCYCLE TIME ='F
     7.4.' S.')

30   FORMAT(I3.1PX2(I3.2I2).3(1X3F4.0))

     FUNCTION RU(KZ)
     KZ=KZ*2045
     KZ=KZ-(KZ/4194304)*4194304
     RU=KZ
     RU=RU*2.38418580E-7

     RETURN

     END
```



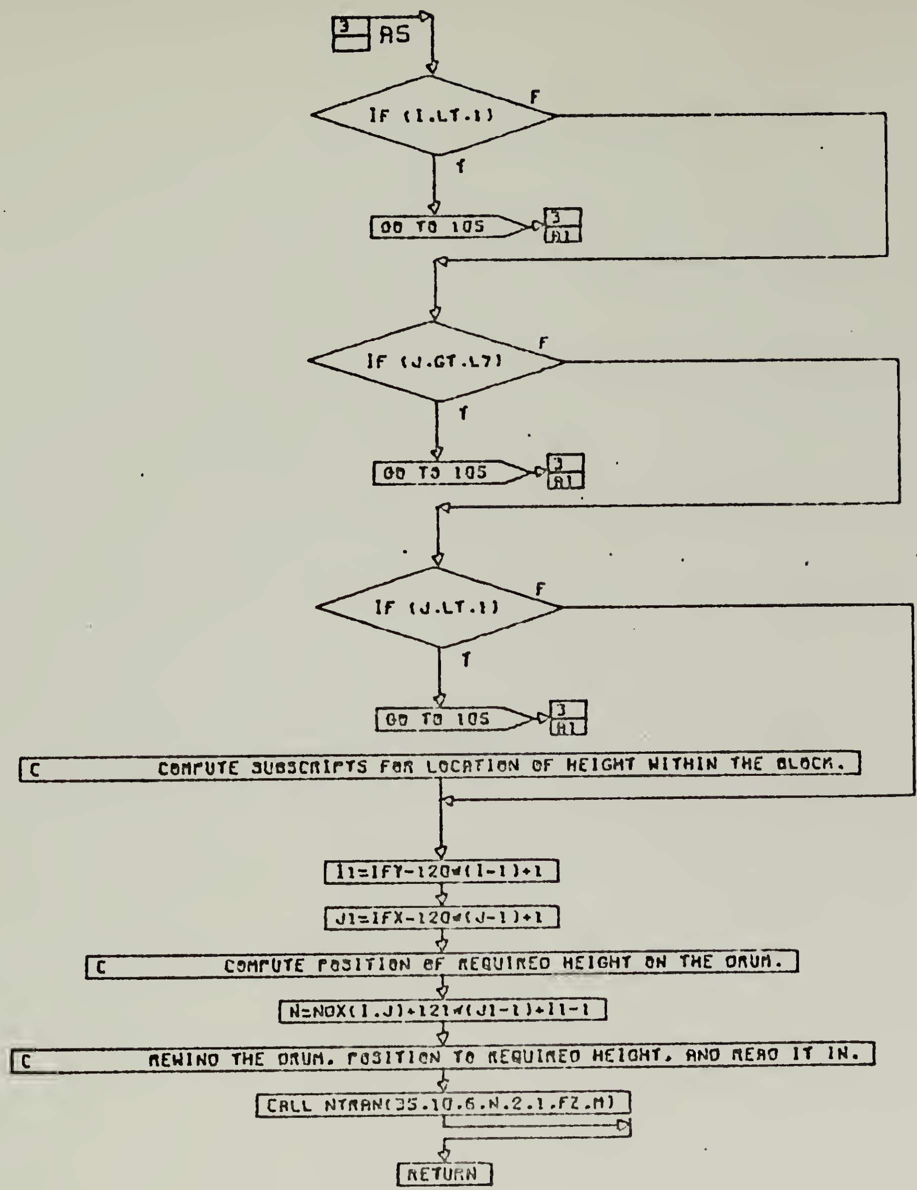
CONT. ON PG 2



CONT. ON PG 3



CONT. ON PG 4



CONT. ON PG 5

END

EO 5 FINAL