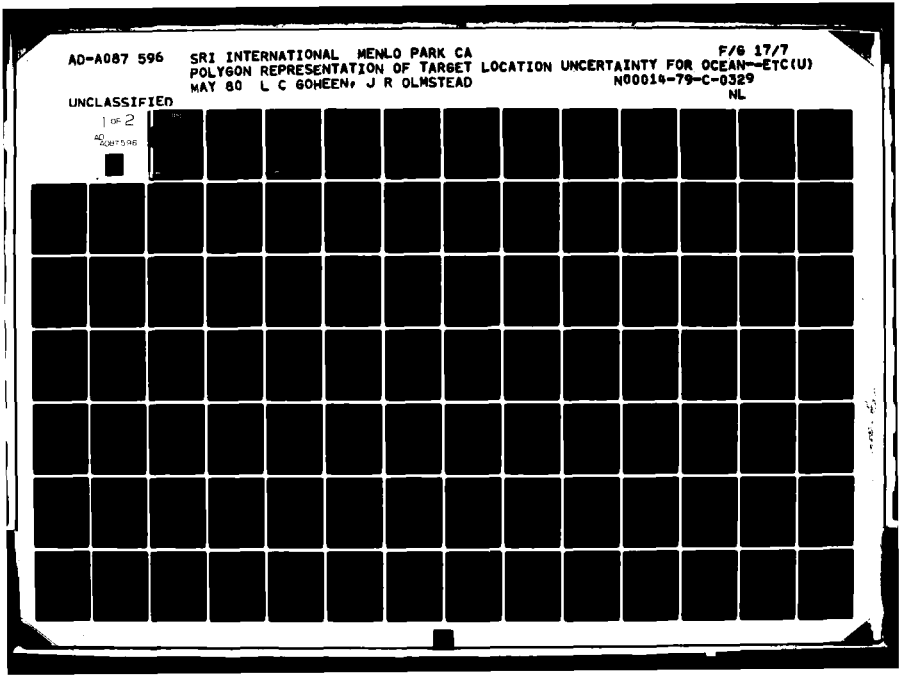
AD-A087 596 SRI INTERNATIONAL MENLO PARK CA F/G 17/7
POLYGON REPRESENTATION OF TARGET LOCATION UNCERTAINTY FOR OCEAN--ETC(U)
MAY 80 L C GOHEEN, J R OLMSTEAD N00014-79-C-0329
UNCLASSIFIED NL

1 OF 2
AD A087596



Technical Report *May 1980*

POLYGON REPRESENTATION OF TARGET LOCATION UNCERTAINTY FOR OCEAN SURVEILLANCE APPLICATION

By: LOLA C. GOHEEN JEFFREY R. OLMSTEAD

Prepared for:

NAVAL ANALYSIS PROGRAM (Code 431)
OFFICE OF NAVAL RESEARCH
DEPARTMENT OF THE NAVY
ARLINGTON, VIRGINIA 22217

CONTRACT N00014-79-C-0329
TASK NR 274-310

Approved for public release; distribution unlimited.

333 Ravenswood Avenue
Menlo Park, California 94025 U.S.A.
(415) 326-6200
Cable: SRI INTL MNP
TWX: 910-373-1246

Distribution List (Continued)

Name	Number of Copies
Naval Research Laboratory Washington, DC 20375 Code 2627 Code 5308 Code 7932 Code 7509	6
Naval Sea Systems Command Washington, DC 20360 Code 63R-1	1

SRI International

Technical Report *May 1980*

POLYGON REPRESENTATION OF TARGET LOCATION UNCERTAINTY FOR OCEAN SURVEILLANCE APPLICATION

By: LOLA C. GOHEEN JEFFREY R. OLMSTEAD

Prepared for:

NAVAL ANALYSIS PROGRAM (Code 431)
OFFICE OF NAVAL RESEARCH
DEPARTMENT OF THE NAVY
ARLINGTON, VIRGINIA 22217

CONTRACT N00014-79-C-0329
TASK NR 274-310

SRI Project 8385

Approved for public release; distribution unlimited.

Approved by:

JACQUES NAAR, *Director*
Center for Defense Analysis

DAVID D. ELLIOTT, *Executive Director*
Systems Research and Analysis Division

333 Ravenswood Avenue • Menlo Park, California 94025 • U.S.A.
(415) 326-6200 • Cable: SRI INTL MNP • TWX: 910-373-1246

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)

POLYGON REPRESENTATION OF TARGET LOCATION UNCERTAINTY FOR OCEAN SURVEILLANCE APPLICATION

5. TYPE OF REPORT & PERIOD COVERED

Final Report, Covering the [illegible] March 1979 - March 1980

6. PERFORMING ORG. REPORT NUMBER

SRI Project 8385

7. AUTHOR(s)

Lola C. Goheen, SRI International
Jeffrey R. Olmstead, now at ATAC

8. CONTRACT OR GRANT NUMBER(s)

N00014-79-C-0329

9. PERFORMING ORGANIZATION NAME AND ADDRESS

SRI International
333 Ravenswood Avenue
Menlo Park, California 94025

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

65152N
RØ145 NR 274-310

11. CONTROLLING OFFICE NAME AND ADDRESS

Naval Analysis Program (Code 431)
Office of Naval Research
Department of the Navy
Arlington, Virginia 22217

12. REPORT DATE

May 1980

13. NO. OF PAGES

170

14. MONITORING AGENCY NAME & ADDRESS (if diff. from Controlling Office)

15. SECURITY CLASS. (of this report)

UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this report)

Approved for public release; distribution unlimited.

Reproduction in whole or in part is permitted for any purpose of the United States Government.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Polygonal uncertainty area
Target uncertainty area
Ocean surveillance
Target localization and classification
Computer graphics
Algorithms to use polygonal uncertainty representation
Multisource information integration

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes polygonal location uncertainty representation and algorithms developed to use this uncertainty representation. The polygonal representation and algorithms provide a method for encoding and integrating target location information from many sources. The method has been implemented in an interactive FORTRAN IV computer program, which is described. Program output is graphically displayed. There are algorithms to fuse positive or negative location information with a target distribution, calculate the

(continued)

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

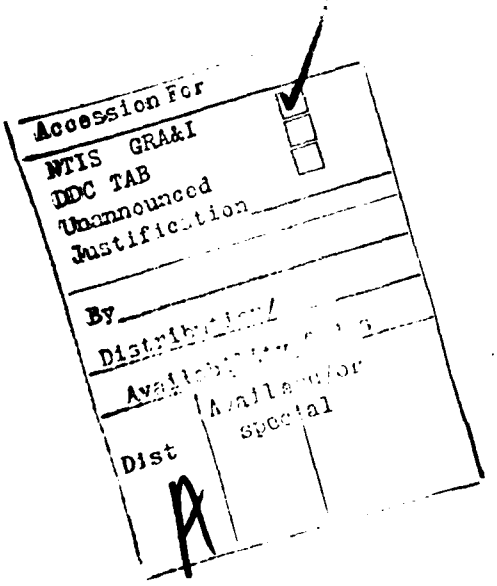
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD A090 581

19. KEY WORDS (Continued)

20 ABSTRACT (Continued)

conditional probability that a location report originated from a particular target, move target uncertainty areas in time, and allow target uncertainty areas to move along or around land masses.



CONTENTS

ILLUSTRATIONS

TABLES

I INTRODUCTION AND SUMMARY

This report discusses the use of multiple polygons to represent probability distributions in two dimensions. Research was performed on computer algorithms to manipulate multipolygonal distributions so that they may be useful in an ocean surveillance target localization application. The following sections discuss the project objectives, the relevance to the ocean surveillance problem, the algorithms required, the accomplishments of the project, and future research that is needed.

A. Project Objectives

The project objectives were to demonstrate (1) that a computational method based on polygons is feasible, and (2) that algorithms can be developed to graphically display target uncertainty areas for an ocean surveillance application. We have satisfied these objectives by developing a research computer program that can perform many functions needed in an ocean surveillance correlation center. The computational methods we developed appear feasible for application programs. Many basic algorithms were developed.

B. Ocean Surveillance Application

The research performed is relevant to the general ocean surveillance problem. The methodology may provide a new tool for target localization because it will permit an efficient representation that can integrate reports on target location.

Target location reports arise from a multiplicity of sensors and can be reports of target contact, no-contact, or other intelligence. Information on target location, including target contact reports arising from surveillance, appears as a geographic region within which the target lies at a fixed time with a given probability. This target uncertainty

region usually appears as a bearing sector, a bearing and range sector, an acoustic convergence zone, an ellipse, or a transit lane region. As the time of the report recedes into the past, the uncertainty region moves and deforms. The movement defines the target track, while the deformation represents increased positional uncertainty.

Negative information areas appear as geographic regions within which the target is known not to lie. Negative information can arise from regions within which it is physically impossible for the target to be located (such as a land mass when the target is a ship or submarine) or they can arise from target no-contact reports.

Target uncertainty areas can be represented as accurately as desired by polygons. Moreover, polygons can be represented efficiently on a digital computer. Thus, polygonal representation of uncertainty areas offers a versatile method of encoding target location information from many sources. The method permits the integration of target location information from multiple sources. Negative information reports can easily be incorporated, and land masses, choke points, and islands can be represented.

Two other methods of representing uncertainty are also applicable to ocean surveillance target localization: the Gaussian and Bayesian representations. Target localization may be performed by assuming a bivariate normal (Gaussian) probability distribution. More commonly, though, it is target tracking (rather than target localization) that utilizes the Gaussian representation. Target localization may also be performed with a Bayesian xy-cell representation. This method is not common, but it is a powerful approach if the target area is small enough that the number of xy-cells is manageable in a computer. The Bayesian method essentially develops a grid of xy-probabilities for each target by estimating probabilities of receiving a given report assuming that the target is in an xy-cell. Very complex probability distributions can be handled in this way.

The polygon method is compatible with both the Gaussian and the Bayesian methods. Polygons can be used before the uncertainty area

becomes small and unimodal; after that, the Gaussian method is more appropriate. Polygons can be used on a global scale where the Bayesian method is impractical or impossible. In a global situation where the Bayesian method would take very large amounts of computer time and memory, the polygon method would only use a modest amount of computer resources. It is in this sense that the polygon method is "computationally efficient." Even though it sounds as though there are areas of overlap and competition between the Bayesian and polygon methods, there is actually a natural way to use them together. This hybrid idea is discussed in Section I-E.

In summary, some localization problems are best described by a Gaussian representation, some by a Bayesian representation, and some by a polygon representation. Polygons appear more applicable to global ocean surveillance than to (1) tracking situations where Gaussian methods are preferred, and (2) small-area localization situations where the Bayesian method is feasible and may be preferred. In addition to using the polygon representation by itself, there is an opportunity to create hybrid representations that may prove useful and cover a wide range of applications, ocean surveillance or otherwise.

C. Algorithms Required

The kinds of algorithms that are needed for ocean surveillance application arise from the need to handle positive and negative information, and to do so over time and under geographic constraints.

Localization is dynamic and, therefore, prediction algorithms are needed. The target polygons must grow in time (negative information polygons shrink in time), and this implies the need for a velocity uncertainty representation.

To perform in a Bayesian-like way, there must be algorithms to calculate the geographic likelihoods of several targets with respect to a contact report. These conditional probabilities can then be used in a target classification scheme to associate contact reports with specific

targets. The idea of using polygons in a classification methodology is discussed in Section I-E, along with ideas of hybrid representations.

Algorithms are needed to combine a new report into current target uncertainty areas. New multipolygonal position and velocity distributions are thereby created.

Finally, algorithms are needed to allow uncertainty areas to interact with land. Uncertainty areas must deform and split apart to represent the uncertain behavior of a target as it maneuvers along coastal areas, through straits, or around islands.

D. Project Accomplishments

A research computer program was developed. It was used to investigate polygonal location-uncertainty representation and to test algorithms that use this uncertainty representation. The basic idea in representing location uncertainty is to build up a probability density function of xy-points by imagining polygons of various heights (polyhedra) being stacked on top of each other. In this way, a wide variety of density functions can be synthesized. This "multipolygon" idea was implemented and used to represent target position uncertainty.

One of the first algorithms developed was a subroutine to calculate the mean and covariance matrix of a general multipolygonal distribution. These distribution statistics were then used to display the associated error ellipse, and were also used as input to other algorithms. The "statistics algorithm" proved to be quite general in the sense that it could be used on distributions defined in a variety of ways.

A "set-operation algorithm" was implemented; it accepts two arbitrary polygons and returns the polygon (or polygons) of intersection, union, or negative intersection. We did not develop this algorithm but used a subroutine package developed by Oak Ridge National Laboratory[1]* and adapted it to our CDC 6400 computer. This basic set-operation

---

*References are listed at the end of this report.

algorithm was used by other algorithms to perform data association, prediction, and land interaction functions. Set-operations (intersection, union, and negative intersection) were the basic tools in developing polygon algorithms within an ocean surveillance context.

One application of set operations was an algorithm for calculating the multipolygonal distribution that resulted from an intersection or negative intersection of two other multipolygonal distributions. This "fusion algorithm" can be used to fuse (integrate, meld, filter) positive or negative location information with a target distribution. Fusion is carried out by intersections and negative intersections. The union of two probability distributions does not seem to have a natural application in the ocean surveillance context.

An algorithm was developed to calculate the conditional probability that a location report originated from a particular target. Both the reported location and the target location are represented by multipolygonal probability distributions. The conditional probability is the likelihood that a particular target is associated with the report, based only on the location information in the report. Likelihoods based on other information in the report (received radar characteristics, for example) are multiplied times the above likelihood to produce a total target likelihood with respect to all of the information in the report. Total likelihood for each candidate target is calculated, and the target with the maximum likelihood would be associated with the report. In summary, the "conditional probability algorithm" can be used to calculate target likelihoods with respect to location reports. In terms of ocean surveillance parlance,[2] the algorithm is used for "report-to-track" correlation. The algorithm can also be used for the other ocean surveillance functions: report-to-report, track-to-report, and track-to-track correlation. Finally, the algorithm can be used to calculate other useful conditional probabilities--for example: (1) the probability of finding a target, given a search distribution, or (2) the probability of killing a target, given a lethality distribution.

A "prediction algorithm" was developed to move target uncertainty areas in time. A multipolygonal target location distribution is assigned a multipolygonal velocity distribution. The algorithm calculates a new and larger location distribution based on the time step and the velocity distribution. We believe this is new ground, and that it is a major accomplishment of the project. From our literature search and contacts, there appears to be no previous work on moving multiple polygons to represent target location uncertainty.

Finally, a "land-interaction algorithm" was developed to allow target distributions to move along or around land masses such as coast lines, straits, and islands. The algorithm was designed to split the target distribution into two parts when approaching land. The probability of the direction the target will go is controlled by the terminal operator. The land-interaction algorithm makes for a more realistic display than simply having the target distribution accumulate against the shore line or, worse yet, having it disappear into an island only to reappear on the other side. The combination of the prediction algorithm and the land-interaction algorithm is the beginning of a software package that is a visually appealing and functionally versatile method for handling positive and negative information about targets of uncertain location.

The research program is written in FORTRAN IV for SRI's CDC 6400 computer. Program output is displayed on the Tektronics 4025 graphics terminal.

During this research effort, the decision was made to develop and implement as many algorithms as possible to use the polygonal location uncertainty representation, since the purpose of the project was to demonstrate (1) feasibility of the polygon data structure, and (2) feasibility of developing algorithms which use this data structure to graphically display target uncertainty areas. Thus while the deficiencies of some implemented algorithms were recognized and documented, effort was directed toward developing more algorithms rather than correcting the deficiencies of existing algorithms.

E. Future Research

The current research has met the objectives of the project but is far from being a complete software package that can be used on displays in an ocean surveillance correlation center. One area of future research is to continue building on the capability developed so far, improving current algorithms, and adding new ones. Included in this task would be target classification representation and report association research. Another research area is to investigate the usefulness of a hybrid Bayesian/polygon representation of localization and classification.

The eventual goal of the polygon research is to produce a software package than can (1) move target polygons around on a map, (2) accept new reports containing location and classification information, (3) associate the information with a target, (4) fuse the localization information with the target polygons to improve the location estimate, and (5) fuse the classification information with the target identity probabilities to improve the classification estimate.

A more immediate goal would be to expand the current research computer program to include a target identity representation, a classification information representation, algorithms to associate reports with targets, and algorithms to fuse classification information. In addition, continued work on current algorithms is necessary to ensure their general applicability.

Another line of research is to use polygons within a Bayesian approach. The Bayesian method uses detection and classification models to predict what should be observed, and then actual observations are compared to the prediction, and probabilities of xy-location and target identity are changed in response to the observation. The method can be effectively used in a multisensor/multitarget environment so long as the numbers of xy-locations and target identities do not combine to produce an impossible computational problem (even for large computers, computational limits are easily exceeded). The idea, then, is to use polygons to limit the xy-region in which the Bayesian calculations take place. The goal of the research would be to develop efficient algorithms that

marry the polygonal representation with the xy-point representation. Investigation is needed to find speedier ways to do the Bayesian/polygon hybrid equivalent of association and fusion. Perhaps a way could be found to let polygonal algorithms preprocess the information before passing it on to the Bayesian algorithms, thus saving computation time. If successful, the Bayesian/polygon hybrid methodology may free the powerful Bayesian method from its major drawback--a massive computational burden.

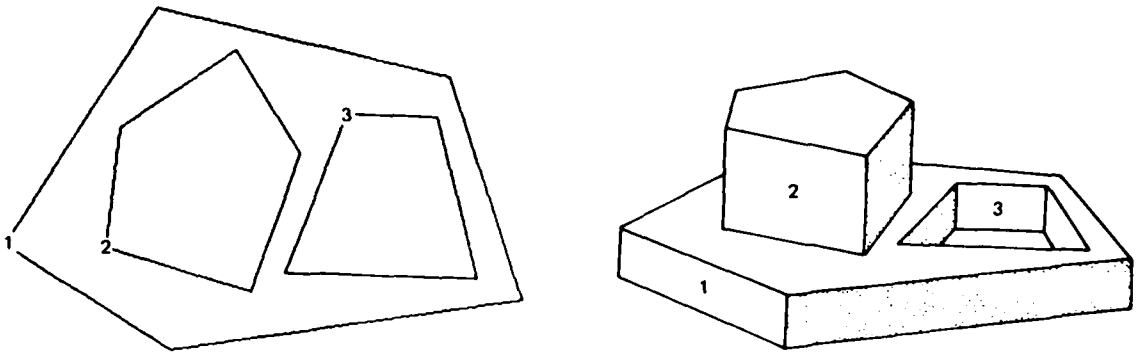
II LOCATION UNCERTAINTY REPRESENTATION

This section discusses how target location uncertainty and land masses are represented. Section II-A covers the concepts of multipolygonal probability distributions. Section II-B discusses the array and indexing scheme we used to represent target distributions in the computer. Section II-C discusses the array and indexing scheme used to represent land masses. It also describes three subroutines utilized in the definition of distributions and land masses.

A. Multipolygonal Probability Distributions

Figure 1 shows an example of a multipolygonal probability density function. The multiple polygons on the left are shown as they appeared on the display, and the figure on the right shows how the resulting probability surface looks in three dimensions. Notice that the three polygons were assigned heights of 1, 2, and -1 units. Thus Polygon 2 is shown twice as high (thick) as Polygon 1. Since Polygon 2 is stacked on top of Polygon 1, the value of the density function within the boundary of Polygon 2 is 1 + 2 = 3 units (actually this value is normalized so that the integral of the density function over all xy-space is unity). Polygons 2 and 3 are inside Polygon 1. Polygon 2 is stacked on top of Polygon 1 because it was given a positive weight. However, Polygon 3 cuts a hole out of Polygon 1 because it was given a negative weight. The hole goes all the way through Polygon 1 because Polygon 3 has height -1 unit, which is equal but opposite to the height of Polygon 1.

Summarizing, Figure 1 shows the general concept of multipolygonal distributions. Polygons are defined by drawing them on the display and assigning either a height value or a weight value to each one. If height is given, then weight is calculated by multiplying the given height times the area of the polygon (weight is equivalent to volume). The weights may be positive or negative--within the restrictions set forth



POLYGON	HEIGHT	WEIGHT
1	1.0	0.75
2	2.0	0.39
3	-1.0	-0.14

FIGURE 1 MULTIPOLYGONAL PROBABILITY DISTRIBUTION

below. Positive weight means that the polygon is stacked on top of other polygons; negative weight means that the polygon cuts out a hole in the other polygons.

1. Mathematical Representation

The above visual concept can be made more rigorous by a mathematical representation. Let Polygon j define the walls of a uniform probability density function, $u_j(x,y)$, such that there is a constant positive probability density for the xy-coordinates inside Polygon j and a zero probability density for the xy-coordinates outside Polygon j. The integral of $u_j(x,y)$ over all xy-space is unity.

A multipolygonal probability density function, $f(x,y)$ can then be defined by weighting and summing several such uniform density functions:

$$f(x,y) = \sum_j P_j \, u_j(x,y) \quad . \tag{1}$$

The polygon weights, P_j, can be positive or negative, but they are under some restriction because $f(x,y)$ cannot be negative at any xy-coordinate. This means that (1) polygons with negative weights must lie inside polygons with positive weights, and (2) the sum of positive heights must be equal to or greater than the sum of negative heights inside those areas enclosed by negatively weighted polygons. Another restriction on the weights is that they must sum to one:

$$\sum_j P_j = 1 \quad . \tag{2}$$

This restriction is because the integral of $f(x,y)$ over all xy-space is unity; therefore, integrating the uniform density functions to unity leaves the P_j to sum to one.

When all of the weights, P_j, are non-negative, they may be interpreted as probabilities (this is the reason for using the symbol "P"). The multipolygonal probability distribution may then be viewed in the following manner. Randomly choose, according to the probabilities, P_j, a polygon from the set of all polygons in the distribution; then the target location is uniformly distributed over the chosen polygon. This view stems from Eq. (1), where the $u_j(x,y)$ are really conditional density functions, conditional on j, providing that the P_j are non-negative. Note that P_j is <u>not</u>, in general, the probability that the target is inside Polygon j. The reason is that polygons may lie on top of each other and the density adds up. Only if the polygons are separate (disjoint) may the P_j be interpreted as containment probabilities.

2. Distributions with Holes

Figure 2 shows three ways to make a distribution with a hole in it. The methods shown on the left and in the middle use two polygons, while the method on the right uses a single polygon. The first two methods are essentially the same; in both cases the inside polygon is assigned a negative weight. In the first case, the negative weight is the result of a negative height multiplied by a positive area. We chose

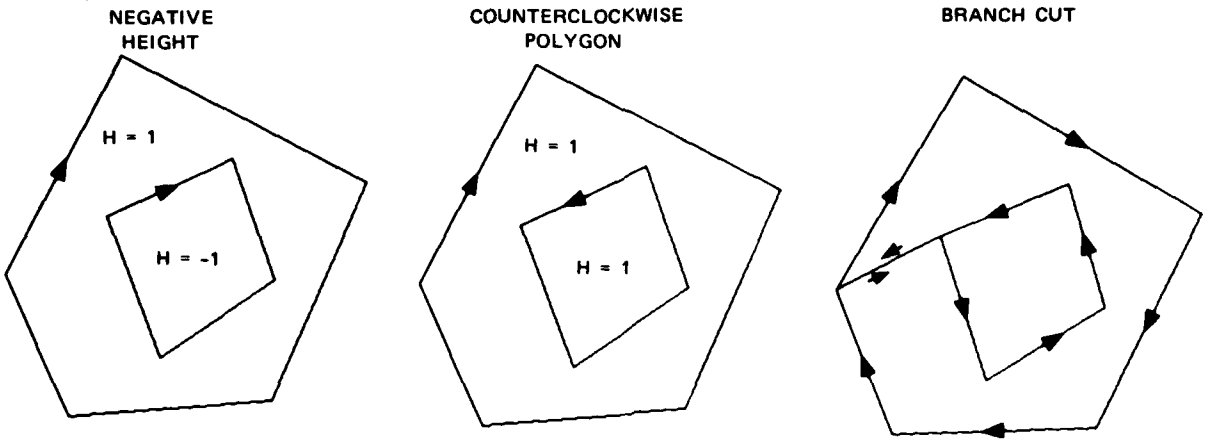


FIGURE 2 DISTRIBUTIONS WITH HOLES

to define polygons with vertices listed in a clockwise direction as containing positive areas. In the second case, the negative weight results from a positive height multiplied by a negative area (counterclockwise polygon).

The third method uses a single polygon and a "branch cut" to define the hole. The inside part of the polygon is a hole because of the counterclockwise sense of the vertices. The branch cut is invisible to algorithms using the distribution because the cut is traversed twice, once in one direction, and once in the opposite direction. When there is a choice, the branch-cut method is preferred because there are no negative weights associated with the distribution.

B. Computer Representation of Targets

The computer program is designed to accommodate several targets; each target is described by one position distribution and one velocity distribution for each of several time steps. The position distribution represents the probability that the target is at various positions within the area at a certain time step. The velocity distribution represents

all possible positions of the centroid of the position distribution at the next time step. Furthermore, each target distribution (position and velocity over time) may contain several polygons and each polygon may consist of many vertices. The memory requirement for this target description was too large to use a combinatorial indexing scheme. For example, 9 targets, 2 spaces (position and velocity space), 5 time steps, 10 polygons per distribution, and 50 xy-points per polygon require $9 \cdot 2 \cdot 5 \cdot 10 \cdot 50 \cdot 2 = 90{,}000$ words of memory. Although not impossible for a large computer, this massive memory allocation is unnecessary because it would be largely unused in any particular session at the terminal. Instead, we used index-pointers to define where polygons start and stop and where their vertices start and stop.

The xy-coordinates of polygon vertices are saved in the arrays, X(I) and Y(I), where I = 1, 1400. In other words, the program can save the coordinates of 1400 vertices. The set of I-indices that start with IA(J) and stop with IB(J) define the vertices of the J-th polygon (J = 1, 200). The first xy-point and the last xy-point of the J-th polygon are equal and describe the same vertex:

$$X(I1) = X(I2)$$

$$Y(I1) = Y(I2)$$

where I1 = IA(J) and I2 = IB(J). This redundance was convenient because the Tektronix 4025 terminal required the first and last points to be equal in order to display polygons.

The polygons are numbered from J = 1 to J = 200, and they are assigned to target distributions by index-pointers. The set of J-indices that start with JA(L,M,N) and stop with JB(L,M,N) define the polygons that belong to the L-th space (L = 1 position, L = 2 velocity), the M-th time step (M = 1, 5), and the N-th target (N = 1, 9).

Summarizing the above scheme, the distribution parameters position/velocity index L, time step index M, and target index N, determine the set of polygons, {J}, that belong to a particular LMN-distribution:

$$J = JA(L,M,N) \text{ to } JB(L,M,N)$$

and each polygon J is represented as a set of indices, {I}:

$$I = IA(J) \text{ to } IB(J)$$

and each index, I, is associated with the vertex:

$$X(I),\ Y(I) \quad .$$

This target-distribution representation takes (2 · 5 · 9 · 2) + (200 · 2) + (1400 · 2) = 3,380 words of memory--far less than the 90,000 words from before. We have not experienced any operating restriction due to the 200-polygon limit or the 1400-vertex limit. In fact, it is convenient to have available a large number of vertices per polygon (instead of a limit of 50, say) because the set-operation algorithm returns polygons with an unknown number of vertices.

The computer representation of targets consists of two parts: the X(I) and Y(I) vectors of polygon vertices, as already discussed, and polygon weights, P(J), which may be positive or negative. These polygon weights are associated with target distributions in exactly the same manner as above: the weights, P(J), starting with J = JA(L,M,N) and stopping with J = JB(L,M,N), are associated with the LMN-distribution. The sum of P(J) over the start-to-stop set of J-values is unity.

In addition to the arrays needed to describe and save multipolygonal probability distributions, the program also saves auxiliary arrays that are useful in polygon algorithms. The height of each polygon, H(J), is saved. The heights are normalized so that the total volume of the distribution is unity. The centroid of each polygon, (XCEN(J), YCEN(J)), is saved and used in the prediction algorithm. Statistics of each (L,M,N)-distribution, SE(I + KL) (for I from 1 to 9, and KL = KE(L,M,N)), are saved:

SE(1 + KL) is the mean of the x coordinates.

SE(2 + KL) is the mean of the y coordinates.

SE(3 + KL) is the minor semiaxis of the 2σ error ellipse.

SE(4 + KL) is the major semiaxis of the 2σ error ellipse.

SE(5 + KL) is sin θ.

SE(6 + KL) is cos θ, where θ is the clockwise angle from north to the major axis.

SE(7 + KL) is the variance of the x coordinates.

SE(8 + KL) is the variance of the y coordinates.

SE(9 + KL) is the covariance of the x and y coordinates.

KE is an array containing pointers that index SE as a function of (L,M,N). KE(L,M,N) yields the index in SE that immediately precedes the statistics of the distribution (L,M,N).

It is convenient to explain four variables that occur frequently in the program code, JJ, ILA, JLA, and KLA:

JJ is the polygon counter for a specific distribution.

ILA is the counter of the number of vertices stored in the X and Y arrays.

JLA is the counter of the number of polygons stored in the X and Y arrays.

KLA is the counter of the number of statistics stored in the SE array.

C. Computer Representation of Land Masses

The computer program is designed to accommodate several land masses. Each land mass is described by a polygon, which may consist of many vertices. The xy-coordinates of land mass vertices are stored, respectively, in arrays LAX and LAY. The scheme used to index into LAX and LAY is the

same as that used for the arrays X and Y. That is, the set of indices that start with LIA(L) and stop with LIB(L) define the vertices of the L-th land mass (L = 1, 200), with

$$LAX(LIA(L)) = LAX(LIB(L))$$

$$LAY(LIA(L)) = LAY(LIB(L)) \ .$$

Land masses are numbered from 1 up to 200, and LJB(1,1,1) is the number of land masses defined for any particular session. Table 1 gives the land mass definition algorithm.

When land is defined, the variable LAND is set to TRUE and subroutine LANDEF is called. Subroutine LANDEF controls the input of land. Land may be input from a properly prepared file named TAPE9, and/or from the terminal. Whenever a land mass is defined from the terminal the rectangle circumscribing it is computed. Thus, for land mass L:

LN(L) is the value of the largest y-coordinate.

LE(L) is the value of the largest x-coordinate.

LS(L) is the value of the smallest y-coordinate.

LW(L) is the value of the smallest x-coordinate.

The terminal operator has the option of saving defined land masses and circumscribing rectangles on a file named TAPE10.

Certain subroutines are utilized during the definition of target distributions and land masses. These are GET, PUT, and VEC. Subroutine GET allows the operator to define target distributions and land from the terminal. Subroutine PUT stores distributions in arrays X and Y, and land masses in arrays LAX and LAY. Subroutine VEC draws polygons.

Table 1

LAND MASS DEFINITION ALGORITHM

```
**
**     LAND
       LAND=.F.
       WRITE(6,60)
C      ENDFILE 6
       WRITE(8,60)
   60 FORMAT(*DEFINE LAND  (Y OR N)*)
       READ(5,14) Q
       WRITE(6,14) Q
       WRITE(7,14) Q
       WRITE(8,14) Q
       IF(Q.EQ.1HY) LAND=.T.
       IF(.NOT.LAND) GOTO 100
       CALL MESSAGC(7HPOL      ,7HLANDEF )
       CALL LANDEF(LAX,LAY,LIA,LIB,LJB,LN,LE,LS,LW,XW,YW)
       WRITE(6,28)
       WRITE(8,28)
       WRITE(6,70)
C      ENDFILE 6
       WRITE(8,70)
   70 FORMAT(*HOW CLOSE SHOULD BE THE DIFFERENCE IN AREAS (IN PERCENT*,
      $  * OF INPUT AREA)  F10.5*)
       READ(5,75) CLOSE
   75 FORMAT(F10.5)

       CLOSE=CLOSE/100.
       WRITE(6,85) INC,CLOSE
C      ENDFILE 6
       WRITE(7,85) INC,CLOSE
C      ENDFILE 7
       WRITE(8,85) INC,CLOSE
   85 FORMAT(*INC= *,G11.5,* CLOSE= *,G11.5)
       WRITE(6,96) COUNT
C      ENDFILE 6
       WRITE(7,96) COUNT
C      ENDFILE 7
       WRITE(8,96) COUNT
   96 FORMAT(*THE NUMBER OF ITERATIONS ALLOWED IS *, I3,*.*)



      SUBROUTINE LANDEF(LAX,LAY,LIA,LIB,LJB,LN,LE,LS,LW,
     $  XW,YW)
C
C      LANDEF AT USER OPTION DEFINES LAND MASSES BY ACCESSING A LAND FILE ON
C      UNIT 9 OR BY TERMINAL INPUT
C
       DIMENSION XW(1),YW(1),LIA(200),LIB(200),LJA(1,1,1),LJB(1,1,1)
       LOGICAL FLAG
       REAL LAX(1400),LAY(1400),LW(200),LN(200),LE(200),LS(200)
       DATA LIL,LJL/0,0/,JJ,J2/0,0/
       DATA I2/0/
       CALL MESSAGA(7HLANDEF )
       WRITE(6,7)
       WRITE(8,7)
    7 FORMAT(*$LIN 8*)
       LJB(1,1,1)=0
       FLAG=.F.
```

Table 1 (Concluded)

```
      WRITE(6,10)
      WRITE(8,10)
C     ENDFILE 6
   10 FORMAT(* USE LAND FILE? (Y OR N) *)
      READ(5,20) Q
      WRITE(6,20) Q
      WRITE(7,20) Q
      WRITE(8,20) Q
   20 FORMAT(A1)
      IF(Q.EQ.1HY) 100,200
  100 CONTINUE
      READ(9) LAX,LAY,LIA,LIB,LJA,LJB,LW,LE,LN,LS
      FLAG=.T.
      J2=LJB(1,1,1)
      DO 130 J=1,J2
      I1=LIA(J)
      I2=LIB(J)
      CALL VEC(I1,I2,LAX,LAY)
  130 CONTINUE
  200 CONTINUE
      WRITE(6,30)
      WRITE(8,30)
C     ENDFILE 6
   30 FORMAT(* DEFINE LAND FROM TERMINAL? (Y OR N)*)
      READ(5,20) Q
      WRITE(6,20) Q
      WRITE(7,20) Q
      WRITE(8,20) Q
      IF(Q.EQ.1HY) 300,400
  300 CONTINUE
      IF(FLAG) 320,325
  320 CONTINUE
      JJ=J2
  325 CONTINUE
      IMAX=99
      JJ=JJ+1
      CALL GET(IMAX,XW,YW,XO,YO,R1,R2,D1,D2)
      CALL VEC(1,IMAX,XW,YW)
      CALL PUT(1,1,1,JJ,IMAX,XW,YW,LIA,LIB,LJA,LJB,LJ,LAX,LAY,
     $  I2,J2)
      CALL MESSAGC(7HLANDEF ,7HRECTAN )
      CALL RECTAN(1,IMAX,XW,YW,LN(LJ),LE(LJ),LS(LJ),LW(LJ))
      CALL MESSAGA(7HLANDEF )
      WRITE(6,40)
C     ENDFILE 6
      WRITE(8,40)
   40 FORMAT(*MORE LAND? (Y OR N)*)
      READ(5,20) Q
      WRITE(6,20) Q
      WRITE(7,20) Q
      WRITE(8,20) Q
      IF(Q.EQ.1HY) GOTO 325
  400 CONTINUE
      WRITE(6,50)
      WRITE(8,50)
   50 FORMAT(* SAVE LAND ON UNIT 10?  (Y OR N)*)
      READ(5,20) Q
      WRITE(6,20) Q
      WRITE(7,20) Q
      WRITE(8,20) Q
      IF(Q.EQ.1HY) 500,600
  500 CONTINUE
      WRITE(10) LAX,LAY,LIA,LIB,LJA,LJB,LW,LE,LN,LS
      ENDFILE 10
  600 CONTINUE
      CALL MESSAGR(7HLANDEF )
      RETURN
      END
```

III POLYGON ALGORITHMS

This section describes six polygon algorithms that have been developed, how they are implemented, and any problems associated with their implementation.

A. Statistics Algorithm

A "statistics algorithm" was developed to calculate the mean and covariance and the error ellipse of a multipolygonal probability distribution. The algorithm can be used on multipolygonal distributions, including distributions that have negative weights. The algorithm calculates the statistics of each individual polygon and then forms a weighted average over all the polygons in the distribution. The derivation of the algorithm is presented in Sections III-A-1, 2, and 3, below, and the implementation of the algorithm is presented in Sections III-A-4 and 5.

1. Polygon Moments

The algorithm calculates the area and the first and second moments of an arbitrary polygon. Moments are defined as:

$$\bar{f} = \frac{1}{A}\int_P \int f \, dx \, dy \tag{3}$$

where f assumes the following five functional values:

$$f = x, \; y, \; x^2, \; y^2, \; xy \quad . \tag{4}$$

The region of integration is inside a polygon, P, and the area of P is given by

$$A = \int_P \int dx \, dy \quad . \tag{5}$$

A polygon may be thought of as a two-dimensional probability distribution: points outside the polygon have zero probability density, and the points inside the polygon have constant probability density equal to 1/A. Therefore, integrating over the polygon and dividing by the area is equivalent to integrating over the probability distribution.

The method for calculating the moments is: divide the polygon into triangles, find the area-weighted moments of each triangle, add them up, and then divide by the total area. The legitimacy of this approach is found in Stokes's Theorem, which relates line integrals to surface integrals. For the two-dimensional case, the theorem may be written:

$$\oint_C \vec{F} \cdot \vec{dr} = \int_P \int (\vec{\nabla} \times \vec{F}) \cdot \vec{k} \, dx \, dy \qquad (6)$$

where the contour, C, encircles the area P; and $\vec{k}$ is the unit vector in the z-direction. For example, if F is defined as:

$$\vec{F} = (0, x^3/3, 0) \qquad (7)$$

then

$$\oint_C \vec{F} \cdot \vec{dr} = \int_P \int x^2 \, dx \, dy \quad . \qquad (8)$$

For the purposes of the following discussion, the integral over an area, P, is positive when the contour, C, is counterclockwise, and the integral is negative if C is clockwise.

Figure 3 shows the polygon, P, broken into triangles, P_1 P_2 P_3. The line integral over the contour, C, is equivalent to the sum of line integrals over the triangular contours, C_1 C_2 C_3. This equivalence can be symbolically expressed as:

$$\oint_C = \oint_{C_1} + \oint_{C_2} + \oint_{C_3} \quad . \qquad (9)$$

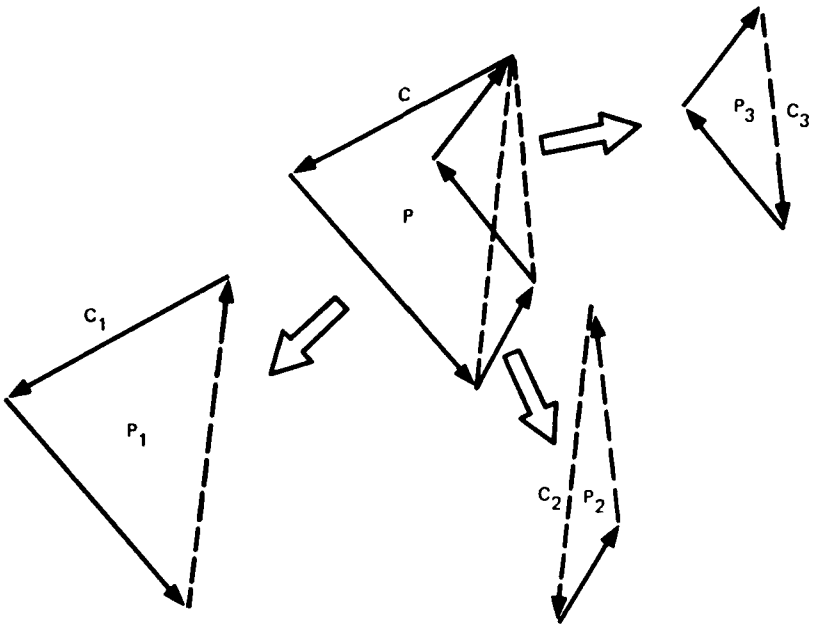


FIGURE 3 A POLYGON BROKEN INTO TRIANGULAR PARTS

The reason the three integrals add up to the single integral over C is that the dashed sides of the triangles are traversed twice, once in a given direction for a triangle and then again in the opposite direction for the adjacent triangle. Thus, the dashed sides of the triangles do not contribute to the sum; only the solid sides contribute, and when added together, they produce the integral over C. By Stokes's Theorem the sum of line integrals is also a sum of area integrals and can be symbolically expressed as:

$$\int_P = \int_{P_1} + \int_{P_2} - \int_{P_3} \quad . \tag{10}$$

Notice that the integral over P_3 is subtracted because the contour, C_3, is in a clockwise direction. Thus, the conclusion is that the moments of a polygon can be calculated by a sum of positive and negative integrals over triangular parts of a polygon.

An alternative method for calculating polygon moments is to use Stokes's Theorem directly and integrate around the perimeter of the polygon. This method was not implemented for two reasons. First, the triangle-area method results in equations with fewer terms than the line-integral method; thus, derivation and programming are simplified. Second, the number of triangles in a polygon is two less than the number of sides of a polygon; thus the triangle-area method is slightly faster. Although the line-integral method is theoretically more straightforward, we chose to derive and implement the triangle-area method for the above reasons.

2. Triangle Moments

The next task was to derive the first and second moments of an arbitrary triangle. The general equations for the first moments and special equations for the x and y moments of inertia of a triangle are given in Pearson.[3] We did not readily find a reference on the general equations for the second moments of a triangle.

The derivation uses an oblique coordinate system as shown in Figure 4.* Integration is over the rs-space inside the triangle: first r is integrated from 0 to R and then s is integrated from 0 to C. The equation for R is found by similarity of triangles:

$$R = B(1 - s/C) \quad . \tag{11}$$

The moments are then given by:

---

*In this derivation, a positive-area triangle has its points ordered in a clockwise direction.

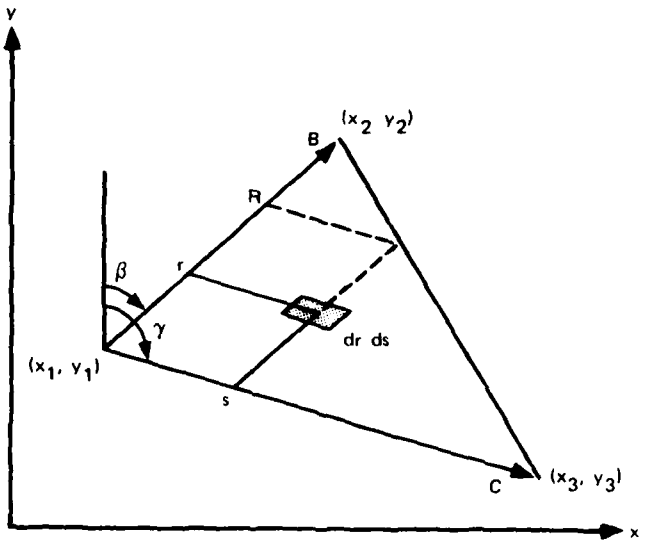


FIGURE 4 OBLIQUE COORDINATES FOR AREA INTEGRATION

$$\bar{f} = \frac{1}{A} \int_0^C \int_0^R f \sin(\gamma-\beta)\, dr\, ds \quad . \tag{12}$$

As before,

$$f = x,\ y,\ x^2,\ y^2,\ xy \tag{13}$$

where x and y are now functions of r and s.

The area of a triangle, A, is easily calculated from the cross product of vector $\vec{C}$ with vector $\vec{B}$:

$$A = (C_x B_y - B_x C_y)/2 \quad . \tag{14}$$

Also, by geometry:

$$A = BC \sin (\gamma-\beta)/2 \quad . \tag{15}$$

Therefore the moments may be written:

$$\bar{f} = \frac{2}{BC}\int_0^C \int_0^R f \, dr \, ds \quad . \tag{16}$$

The second moments are most easily calculated by using the identities:

$$\begin{aligned} \overline{x^2} &= \bar{x}^2 + \overline{\Delta x^2} \\ \overline{y^2} &= \bar{y}^2 + \overline{\Delta y^2} \\ \overline{xy} &= \bar{x}\,\bar{y} + \overline{\Delta x \Delta y} \end{aligned} \tag{17}$$

where $\Delta x = x - \bar{x}$, and $\Delta y = y - \bar{y}$.

The transformation equations from rs-space to xy-space are:

$$\begin{aligned} x &= x_1 + r \sin \beta + s \sin \gamma \\ y &= y_1 + r \cos \beta + s \cos \gamma \quad . \end{aligned} \tag{18}$$

Using these equations, the second central moments of a triangle in xy-space can be written in terms of the second central moments in rs-space; for example:

$$\begin{aligned} \overline{\Delta x \Delta y} = {} & B_x B_y \, \overline{\Delta r^2}/B^2 \\ & + (B_x C_y + C_x B_y) \, \overline{\Delta r \Delta s}/BC \\ & + C_x C_y \, \overline{\Delta s^2}/C^2 \end{aligned} \tag{19}$$

where $\sin \beta = B_x/B$, etc.

The second central moments in rs-space are calculated using identities such as:

$$\overline{\Delta r \Delta s} = \overline{rs} - \bar{r}\,\bar{s}$$

where the r,s moments are derived by integration; for example:

$$\overline{rs} = \frac{2}{BC}\int_0^C \int_0^R rs\, dr\, ds \qquad (20)$$

$$\overline{rs} = \frac{B}{C}\int_0^C (1 - s/C)^2\, s\, ds \quad .$$

The results are:

$$\begin{aligned} \overline{r} &= B/3 \\ \overline{s} &= C/3 \\ \overline{r^2} &= B^2/6 \\ \overline{s^2} &= C^2/6 \\ \overline{rs} &= BC/12 \quad . \end{aligned} \qquad (21)$$

The first moments and the second central moments for a triangle in xy-space are then computed:

$$\begin{aligned} \overline{x} &= x_1 + (B_x + C_x)/3 \\ \overline{y} &= y_1 + (B_y + C_y)/3 \\ \overline{\Delta x^2} &= (B_x^2 - B_x C_x + C_x^2)/18 \\ \overline{\Delta y^2} &= (B_y^2 - B_y C_y + C_y^2)/18 \\ \overline{\Delta x \Delta y} &= [B_x B_y - (B_x C_y + C_x B_y)/2 + C_x C_y]/18 \quad . \end{aligned} \qquad (22)$$

3. Eigenvalues

The "error ellipse" that corresponds to a polygonal probability distribution can be determined from the distribution's statistics. The center of the ellipse is at $(\overline{x},\overline{y})$. If a 2-by-2 covariance matrix, V, is defined as:

$$V_{11} = \overline{x^2} - \bar{x}^2$$

$$V_{22} = \overline{y^2} - \bar{y}^2 \qquad (23)$$

$$V_{12} = \overline{xy} - \bar{x}\,\bar{y} = V_{21}$$

then the orientation and principal-axis standard deviations of the ellipse are determined by an eigenvalue calculation, as derived below.

The eigenvalue problem can be written as:

$$V\vec{u} = \sigma^2 \vec{u} \qquad (24)$$

where $\vec{u}$ is a 2-by-1 vector and σ^2 is a scalar. The eigenvalues are derived by noting that the determinant of the equation must be zero:

$$\left|V - \sigma^2 I\right| = 0 \quad . \qquad (25)$$

This quadradic equation yields the two eigenvalues:

$$\sigma_1^2 = (P - Q)/2 \qquad (26)$$

$$\sigma_2^2 = (P + Q)/2$$

where:

$$P = V_{11} + V_{22}$$

$$Q = \left[(V_{11} - V_{22})^2 + 4V_{12}^2\right]^{1/2} \quad . \qquad (27)$$

The eigenvector, $\vec{u}_2$, that corresponds to the larger of the two sigmas, σ_2, is found by using one of the linear equations implied by:

$$(V - \sigma_2^2 I)\,\vec{u}_2 = 0 \quad . \qquad (28)$$

The equation determines a ratio between the components of $\vec{u}_2$, and therefore an angle can be calculated:

$$\alpha = \arctan\left[V_{12}/(\sigma_2^2 - V_{11})\right] \tag{29}$$

where α is the angle from the y axis to the long axis of the ellipse.

4. Mean and Covariance Subroutine

Table 2 shows the subroutines that calculate polygon distribution statistics. The input parameters to subroutine MOM are (1) the polygon-index limits, J1 and J2; (2) the vectors of vertex-index limits, IA and IB; (3) the vector of polygon vertices, X and Y; and (4) the vector of polygon weights, P. The output parameters are (1) the vector of polygon heights, H; (2) the means of the distribution, EX and EY; (3) the variances, EXX and EYY, and the covariance, EXY; and (4) the vector of polygon centroids, XCEN and YCEN. The input parameters to subroutine CENTRD are (1) the vector-index limits, I1, I2, of the distinct vertices of a polygon; and (2) the vectors of polygon vertices, X and Y. The output parameters are (1) the area of the polygon, A; (2) values proportional to the first moments of the polygon, AX and AY; and (3) values proportional to the second moments of the polygon, AXX, AYY, AXY.

In subroutine CENTRD, the loop is over the triangles constituting the polygon. The number of triangles, I22, is two less than the number of polygon vertices. The vectors (BX,BY) and (CX,CY) determine two sides of the triangle. The formulas for the area and moments of a triangle are easily recognized by referring to the previous section. The triangle moments are weighted by the area of each triangle (which may be positive or negative) and then summed over all triangles in the polygon.

In subroutine MOM, the loop is over the polygons that constitute the distribution. MOM calls CENTRD once for each polygon. When control returns to MOM from CENTRD, the centroid of the polygon is calculated; the x coordinate of the centroid is stored in XCEN(J), and the

Table 2

MEAN AND COVARIANCE SUBROUTINES:
MOM AND CENTRD

```
      SUBROUTINE MOM(J1,J2,IA,IB,X,Y,P,H,EX,EY,EXX,EYY,EXY,XCEN,YCEN)
      DIMENSION XCEN(1),YCEN(1),IA(1),IB(1),X(1),Y(1),P(1),H(1)
      EX=EY=EXX=EYY=EXY=0.
      DO 55 J=J1,J2
      I1=IA(J)
      I2=IB(J)-1
      CALL MESSAGC(7HMOM      ,7HCENTRD )
      CALL CENTRD(I1,I2,X,Y,A,AX,AY,AXX,AYY,AXY)
      XCEN(J)=AX/A
      YCEN(J)=AY/A
      Z=P(J)/A
      EX=EX+Z*AX
      EY=EY+Z*AY
      EXX=EXX+Z*AXX
      EYY=EYY+Z*AYY
      EXY=EXY+Z*AXY
      H(J)=P(J)/ABS(A)
   55 CONTINUE
      EXX=EXX-EX*EX
      EYY=EYY-EY*EY
      EXY=EXY-EX*EY
      RETURN
      END
```

```
            SUBROUTINE CENTRD(I1,I2,X,Y,A,AX,AY,AXX,AYY,AXY)
      C
      C     CENTRD CALCULATES THE AREA,AND FIRST MOMENTS AND VALUES
      C     PROPORTIONAL TO THE SECOND MOMENTS OF A POLYGON
      C     WHOSE DISTINCT VERTICES ARE STORED IN IUCALC OUTPUT FORMAT
      C     IN X(I),Y(I),I=I1,I2.
      C     CENTRD IS CALLED BY GOLDSEC AND MOM.
      C
            DIMENSION X(1),Y(1)
            CALL MESSAGA(7HCENTRD )
            I22=I2-2
            A=AX=AY=AXX=AYY=AXY=0.
            DO 50 I=I1,I22
              BX=X(I+1)-X(I1)
              BY=Y(I+1)-Y(I1)
              CX=X(I+2)-X(I1)
              CY=Y(I+2)-Y(I1)
              AI=0.5*(CX*BY-BX*CY)
              A=A+AI
              XI=X(I1)+(BX+CX)/3.
              YI=Y(I1)+(BY+CY)/3.
              AX=AX+AI*XI
              AY=AY+AI*YI
              AXX=AXX+AI*(XI*XI+(BX*BX-BX*CX+CX*CX)/18.)
              AYY=AYY+AI*(YI*YI+(BY*BY-BY*CY+CY*CY)/18.)
              AXY=AXY+AI*(XI*YI+(BX*BY-0.5*(BX*CY+CX*BY)+CX*CY)/18.)
         50   CONTINUE
            WRITE(7,20) A,AX,AY,AXX,AYY,AXY
         20 FORMAT(*THE OUTPUT OF CENTRD IS *,/(5G15.5))
            CALL MESSAGR(7HCENTRD )
            RETURN
            END
```

y coordinate in YCEN(J). Then each area-weighted sum of triangle moments is normalized by the area of the polygon, weighted by the P(J) values of the polygon (which may be positive or negative), and then summed over all polygons in the distribution. The moments that result from the DO-55 loop do not have to be normalized by the sum of P-weights because the P(J) are already defined so that they sum to one. The heights of the polygons are calculated and saved in this subroutine because it proved a convenient place to do so. After leaving the DO-55 loop, the second moments are converted to covariance parameters for output.

5. Error Ellipse Subroutine

Table 3 shows the computer algorithm that performs the eigenvalue calculation. The input parameters are the covariance terms, EXX, EYY, and EXY. The output parameters are the 2-sigma values, R1 and R2, which lie along the minor and major axes; and the sine and cosine, D1 and D2, of the angle from the y-axis to the major axis of the ellipse. The first part of the subroutine takes care of two special cases when the ellipse is aligned with the coordinate axis; the remainder of the subroutine follows the previous derivation.

Figure 5 shows an example of a 2-sigma error ellipse superimposed on its multipolygonal probability distribution.

Figure 6 shows how a bivariate normal distribution can be approximated with three 16-sided elliptically shaped polygons that are stacked on top of each other. The resulting 2-sigma error ellipse nearly coincides with the polygon that represents the 2-sigma ellipse of the bivariate normal distribution. This is a result of the particular weighting scheme used for the three polygons.

When a single elliptically shaped polygon is computed, the 2-sigma error ellipse almost coincides with the polygon and thus the area of the polygon and ellipse are nearly equal. Experience has shown that the area of the 2-sigma error ellipse is always larger than the area of an arbitrary polygon; therefore, the ellipse-to-polygon area ratio is a convenient measure of the ellipticity of a polygon. As the polygon approaches an elliptical shape, the area ratio approaches a value of one.

Table 3

ERROR ELLIPSE SUBROUTINE:
EIGEN

```
      SUBROUTINE EIGEN(EXX,EYY,EXY,R1,R2,D1,D2)
      IF(EXY.NE.0.) GO TO 70
      IF(EXX.GT.EYY) GO TO 60
      D1=0. $D2=1.
      U=EXX $V=EYY $GO TO 80
   60 D1=1. $D2=0.
      U=EYY $V=EXX $GO TO 80
   70 EE=EXY*EXY
      P=EXX+EYY
      Q=SQRT((EXX-EYY)**2+4.*EE)
      U=(P-Q)/2.
      V=(P+Q)/2.
      W=V-EXX
      D=SQRT(EE+W*W)
      D1=EXY/D
      D2=W/D
   80 R1=0.
      IF(U/P.GT.1.E-100) R1=2.*SQRT(U)
      R2=2.*SQRT(V)
      RETURN
      END
```

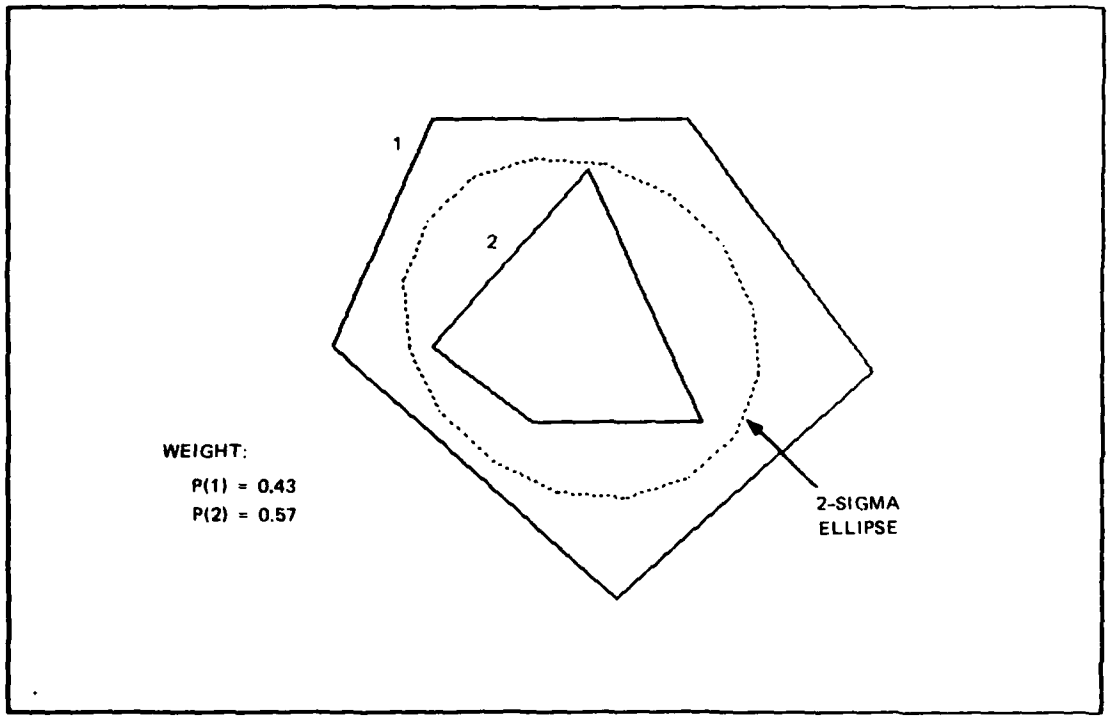


FIGURE 5 TWO-SIGMA ERROR ELLIPSE

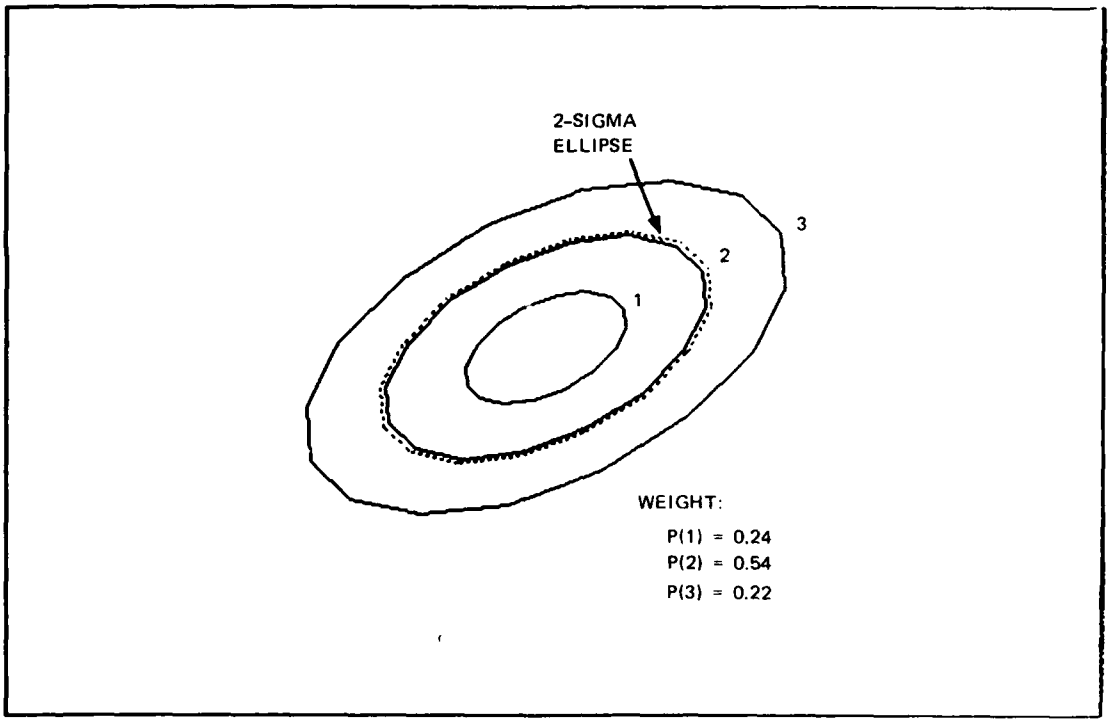


FIGURE 6 BIVARIATE NORMAL APPROXIMATION

B. Set-Operation Algorithm

A card deck of the IUCALC package (a set of FORTRAN subroutines that calculate polygon intersections, unions, and negative intersections) was obtained from the Computer Sciences Division, Oak Ridge National Laboratory, Oak Ridge, Tennessee.[1] A copy of IUCALC was modified in order to make the package compatible with the SRI CDC 6400 computer.

1. Subroutines in the IUCALC Package

The IUCALC package consists of eight subroutines and functions as shown in Figure 7. These are briefly described below. Subroutine IUCALC is the main routine of the package. Its primary function is to call Subroutine IUSUB1 and, when IUSUB1 terminates normally, to call Subroutine IUSUB2. The parameters of IUCALC are discussed in a separate

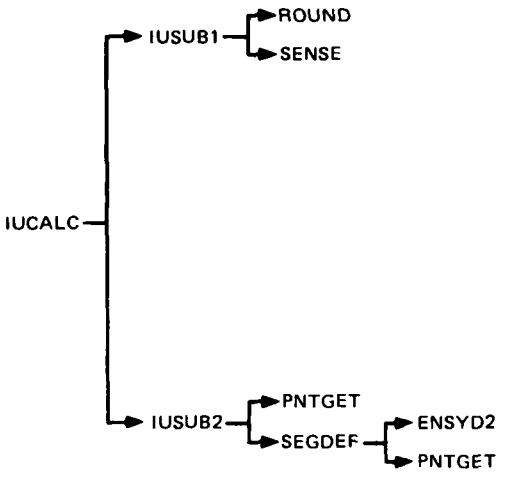


FIGURE 7 STRUCTURE OF THE IUCALC PACKAGE

section below. Subroutine IUSUB1 generates coordinate points, each of which is the intersection of a side of a polygon with a line segment of a simply connected chain. (A simply connected chain is a sequence of distinct directed line segments, e_1, e_2, ..., e_n. Each e_i begins at coordinate point (x_i,y_i) and ends at coordinate point (x_{i+1},y_{i+1}), $1 \leq i \leq n$. (x_i,y_i) and (x_{i+1},y_{i+1}) are called the endpoints of e_i. Other than at an endpoint, a line segment shares no point with another line segment. The coordinate points (x_1,y_1), (x_2,y_2), ..., (x_{n+1},y_{n+1}) are distinct with the exception that $(x_1,y_1) = (x_{n+1},y_{n+1})$ is allowed, in which case the chain is called a closed chain (or a polygon); when $(x_1,y_1) \neq (x_{n+1},y_{n+1})$ the chain is called an open chain.) IUSUB1 calls the functions SENSE and ROUND. Subroutine IUSUB2 generates the coordinate points of each resulting polygon or chain. IUSUB2 calls the subroutines SEGDEF and PNTGET.

Function SENSE determines whether the vertices of a polygon are in clockwise or counterclockwise order. Function ROUND rounds double precision numbers to single precision and is machine-dependent.

A CDC 6400 compatible ROUND function was written at SRI and is a part of the IUCALC package operating on the CDC 6400.

Subroutine SEGDEF generates a matrix that defines the segments of the resulting polygons or chains. (A segment is a series of successive sides of a polygon or a series of successive line segments of a chain, all of which must be entirely inside the other polygon, or all of which must be entirely outside the other polygon, or all of which must be entirely on the boundary of the other polygon.) SEGDEF calls Subroutine PNTGET and Function ENSYD2.

Subroutine PNTGET returns a certain coordinate point selected from among the coordinates of the input polygon and chain and the calculated points of intersection. Function ENSYD2 determines if a given point is inside or outside a given polygon, and assumes the point is not on a boundary of the polygon.

2. Parameters of Subroutine IUCALC

Subroutine IUCALC has 15 parameters. The first six parameters define the polygon and chain to be operated on. In particular, the parameters are:

APX The real array containing the x coordinates of polygon A.

APY The real array containing the y coordinates of polygon A.

NOAP The number of coordinate pairs in polygon A.

BCX The real array containing the x coordinates of chain B.

BCY The real array containing the y coordinates of chain B.

NOBC The number of coordinate pairs in chain B.

KALC The integer specifying the desired set operation--specifically:

- KALC = 1 Union of polygons A and B
- KALC = 2 Intersection of polygons A and B
- KALC = 3 Relative difference of polygon A to polygon B (A intersect not-B)
- KALC = 4 Relative difference of polygon B to polygon A (B intersect not-A)
- KALC = 5 Subchains of open chain B on the boundary of polygon A

KALC = 6 Subchains of open chain B strictly exterior to polygon A

KALC = 7 Subchains of open chain B strictly interior to polygon A

KALC = 8 Subchains of open chain B exterior to or on the boundary of polygon A

KALC = 9 Subchains of open chain B interior to or on the boundary of polygon A.

The next two parameters define the work area. They are:

WORK The real array used as a work area. The dimension of the array can be estimated by the formula

$$NOAP + NOBC + 6K + 2$$

where K is the number of points of intersection.

WRKMAX The dimension of the work space WORK.

The final six parameters contain the results of the calculations. In particular, the parameters are:

RCX The real array containing the x coordinates of the result polygons or chains.

RCY The real array containing the y coordinates of the result polygons or chains.

NRCMAX The dimension of arrays RCX and RCY.

INORC The integer array that contains the index of the start of each result polygon or chain in RCX and RCY, and the number of coordinate pairs in each result polygon or chain.

INOMAX The maximum allowed number of result polygons or chains. The dimension of INORC is 2 by INOMAX.

NORC The number of result polygons or chains calculated; or an error flag when the arrays INORC, or RCX and RCY are exceeded.

3. Interface with IUCALC

It was convenient to develop two subroutines to communicate with IUCALC. Subroutine OAK was developed because IUCALC cannot operate directly on the multipolygonal vectors, X(I) and Y(I). Rather, IUCALC requires vectors that describe single polygons. Furthermore, IUCALC

requires only the vertices of a polygon, not the first-equals-last redundancy built into the XY-representation. Subroutine PREIU was developed because the iterative nature of two algorithms made it desirable to call IUCALC and transfer the IUCALC result to one pair of the input arrays with a single subroutine call.

The interface subroutine OAK is shown in Table 4. Subroutine OAK has as input parameters:

KL	The integer specifying the desired set operation.
JP	The index defining the first input polygon
JQ	The index defining the second input polygon
IA	The array of vertex-start indices
IB	The array of vertex-stop indices
X	The x-coordinates of all polygons
Y	The y-coordinates of all polygons.

The output parameters of subroutine OAK are:

RX	The array of x-coordinates of the resultant polygons
RY	The array of y-coordinates of the resultant polygons
IR	The integer array that contains the index of the start of each resultant polygon and the number of vertices in each resultant polygon
JMAX	The number of resultant polygons (including zero) or an error flag indicating abnormal termination of processing by IUCALC.

Subroutine OAK first calculates the number of vertices, NP and NQ, in the two input polygons. Then pointing indices, IP and IQ, are computed and used in the DO-loops that copy X and Y values into (PX,PY) and (QX,QY) arrays for input to IUCALC.

The interface subroutine PREIU is shown in Table 4. Subroutine PREIU has as input parameters:

XW	The array of x-coordinates of the first input polygon
YW	The array of y-coordinates of the first input polygon
IW1	The number of distinct vertices in the first polygon
PX	The array of x-coordinates of the second input polygon

Table 4

IUCALC INTERFACE SUBROUTINES:
OAK AND PREIU

```
      SUBROUTINE OAK(KL,JP,JQ,IA,IB,X,Y,RX,RY,IR,JMAX)
      DIMENSION PX(50),PY(50),QX(50),QY(50),WK(300),
     $          RX(1),RY(1),IR(2,1),X(1),Y(1),IA(1),IB(1)
      NP=IB(JP)-IA(JP)
      NQ=IB(JQ)-IA(JQ)
      IP=IA(JP)-1
      IQ=IA(JQ)-1
      DO 127 I=1,NP
      PX(I)=X(I+IP)
  127 PY(I)=Y(I+IP)
      DO 128 I=1,NQ
      QX(I)=X(I+IQ)
  128 QY(I)=Y(I+IQ)
      CALL IUCALC(PX,PY,NP,QX,QY,NQ,KL,WK,300,JMAX,IR,20,RX,RY,200)
      RETURN
      END
```

```
      SUBROUTINE PREIU(XW,YW,IW1,PX,PY,M1,KL,IS,XS,YS,
     $  SUB,FLAG,FLAG1,JMAX)
      LOGICAL FLAG,FLAG1
      DIMENSION XW(1),YW(1),PX(1),PY(1),XS(1),YS(1),WK(300),
     $  IS(2,1)
C
C     PREIU CALLS IUCALC. TESTS JMAX FOR IUCALC AND CALLING ROUTINE ERRORS.
C     THE IUCALC RESULT IS TRANSFERED TO XW,YW.
C     IF ERROR THEN AN APPROPRIATE MESSAGE IS WRITTEN.
C     SUB CONTAINS THE NAME OF CALLING ROUTINE.
      CALL MESSAGA(7HPREIU  )
      FLAG=.T.
      FLAG1=.T.
      WRITE(7,30) (XW(I),YW(I),I=1,IW1)
      WRITE(7,30) (PX(I),PY(I),I=1,M1)
      CALL MESSAGC(7HPREIU  ,7HIUCALC )
      CALL IUCALC(XW,YW,IW1,PX,PY,M1,KL,WK,300,JMAX,IS,20,XS,YS,200)
      CALL MESSAGA(7HPREIU  )
      IF(JMAX) 100,200,300
  100 CONTINUE
      WRITE(6,15) JMAX,SUB
      WRITE(7,15) JMAX,SUB
C     ENDFILE 7
   15 FORMAT(*IUCALC ERROR *,I3,* OCCURS IN *,A7)
      WRITE(6,25)
      WRITE(8,25)
   25 FORMAT(*$LIN 7*)
      IW11=IW1+1
      XW(IW11)=XW(1)
      YW(IW11)=YW(1)
      M11=M1+1
      PX(M11)=PX(1)
      PY(M11)=PY(1)
   30 FORMAT(5(G13.5,G13.5))
      CALL VEC(1,IW11,XW,YW)
      CALL VEC(1,M11,PX,PY)
      FLAG=.F.
      CALL MESSAGR(7HPREIU  )
      RETURN
  200 CONTINUE
```

Table 4 (Concluded)

```
C     WRITE(6,40) SUB,JMAX
      WRITE(7,40) SUB,JMAX
   40 FORMAT(A7,*RESULT IS THE SAME AS ORIGINAL DATA OR NO RESULT *,
     $  *EXISTS. *,I3)
C     ENDFILE 7
C
C     RESULT IS THE SAME AS ORIGINAL DATA OR NO RESULT EXISTS
C
      FLAG1=.F.
      CALL MESSAGR(7HPREIU  )
      RETURN
  300 CONTINUE
C     WRITE(6,50) JMAX
      WRITE(7,50) JMAX
C     ENDFILE 7
   50 FORMAT(*PREIU CALLS RECOVER *,I3)
      CALL RECOVER(JMAX,XS,YS,IS,XW,YW,IW1,SUB)
      CALL MESSAGA(7HPREIU  )
      IF(JMAX.EQ.0) FLAG1=.F.
      CALL MESSAGR(7HPREIU  )
      RETURN
      END
```

PY The array of y-coordinates of the second input polygon

M1 The number of distinct vertices in the second input polygon

KL The same as KL in OAK

SUB The name of the routine calling PREIU; this is used in error messages.

Subroutine PREIU has as output parameters:

XW The array of x-coordinates of a selected resultant polygon

YW The array of y-coordinates of a selected resultant polygon

IW1 The number of distinct vertices in the selected polygon

XS The same as RX in OAK

YS The same as RY in OAK

IS The same as IR in OAK

FLAG1 = FALSE The signal that the result is the same as the original data or no result exists.

FLAG = FALSE The error flag that IUCALC has abnormally terminated processing.

JMAX The same as JMAX in OAK.

FLAG and FLAG1 are set to true upon entry into subroutine PREIU. Then subroutine IUCALC is called. Upon return from IUCALC, the

value of JMAX is evaluated. When JMAX indicates that IUCALC has abnormally terminated processing, FLAG is set to false, error messages are written, and PREIU returns. When JMAX indicates that the result is the same as the original data or no result exists, FLAG1 is set to false, messages are written, and PREIU returns. When JMAX indicates that one or more resultant polygons have been computed, PREIU calls subroutine RECOVER. Subroutine RECOVER searches XS, YS, and IS in order to eliminate spurious vertices generated by IUCALC and select a resultant polygon.

The calling sequence of IUCALC is given below in terms of the symbols in the previous section as compared to the symbols in subroutines OAK and PREIU:

Input			Output		
IUCALC	OAK	PREIU	IUCALC	OAK	PREIU
APX	PX	XW	NORC	JMAX	JMAX
APY	PY	YW	INORC	IR	IS
NOAP	NP	IW1	INOMAX	20	20
BCX	QX	PX	RCX	RX	XS
BCY	QY	PY	RCY	RY	YS
NOBC	NQ	M1	NRCMAX	200	200
KALC	KL	KL			
WORK	WK	WK			
WRKMAX	300	300			

4. Examples of Set Operations

Figure 8 shows examples of set operations. Two polygons, A and B, were input to IUCALC and three results were computed:

- Intersection of A with B, A * B
- Union of A and B, A + B
- Negative intersection of A with B, $\overline{A}$ * B.

The other negative intersection, $\overline{B}$ * A, is not shown. Notice that all of the set operations can produce multiple polygons. IUCALC returns polygon vertices in a clockwise sense; holes in polygons are returned

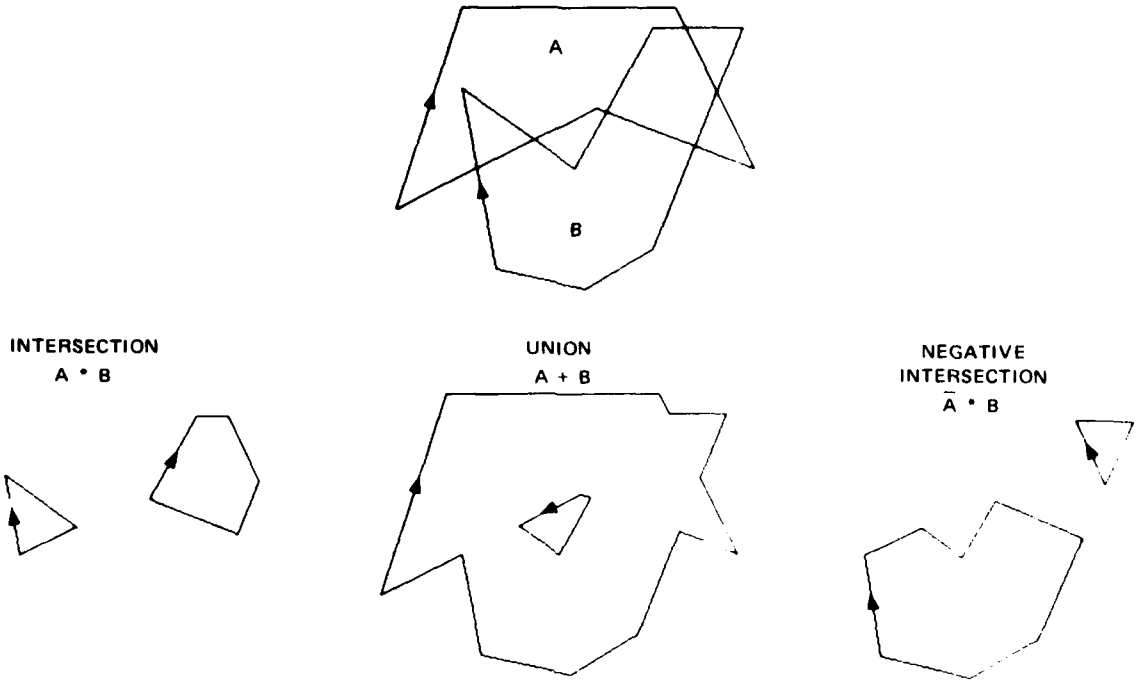


FIGURE 8 POLYGON SET OPERATIONS

as polygons with vertices in a counterclockwise sense. Counterclockwise polygons are always inside clockwise polygons.

C. Fusion Algorithm

A "fusion algorithm" was developed to fuse (integrate, meld, filter) two probability distributions into one distribution. Two different kinds of fusion are possible, and they are multipolygonal counterparts to the intersection and negative intersection set operations. Thus, a location-report distribution can be fused with a target distribution by using intersection fusion; and a negative-information distribution can be fused with a target distribution by using negative-intersection fusion. The algorithm cannot perform union fusion. Since it is not clear what the union distribution represents in an ocean surveillance context, we did not develop the algorithm (although it is possible to do so).

1. Program Logic

Assume that a location report is described by a single polygon, R, and that the target uncertainty area, to which the report is associated, is described by a single polygon, T. Since the target is assumed to be inside both polygons R and T at the same time, the intersection of the two polygons must be used to define the target uncertainty region after the fusion of the report. This single-polygon example is derived mathematically and then the more complicated multipolygonal case is deduced.

An appropriate mathematical method for describing fusion is the Bayesian method. For this method, (1) a likelihood function, $g(R/x,y)$, is defined; (2) a prior probability function, $f(x,y/T)$, is defined; and (3) the posterior probability density (the result of fusing the report and target distributions) is calculated by Bayes's formula:

$$f(x,y/T') = K\ g(R/x,y)\ f(x,y/T) \tag{30}$$

where K is a normalization constant that makes the integral of the posterior density over all xy-space equal to unity.

In the single-polygon example, the "likelihood" of point (x,y) with respect to the report is defined to be a function that is proportional to the uniform density function that is associated with Polygon R:

$$g(R/x,y) = G\ u(x,y/R) \tag{31}$$

where G can be any positive constant--in particular, unity.

The "prior" probability density is the probability density of locating target, T, at point (x,y) before the report distribution is fused with the prior distribution. For the single polygon example, the prior density is just the uniform density associated with Polygon T:

$$f(x,y/T) = u(x,y/T) \quad . \tag{32}$$

The integral of the prior density over all xy-space is unity.

When Bayes's formula, Eq. (30), is used with the likelihood function, Eq. (31), and prior density, Eq. (32), the result is:

$$f(x,y/T') = K\ G\ u(x,y/R)\ u(x,y/T) \quad . \tag{33}$$

The product of the two uniform densities is proportional to a third uniform density, $u(x,y/T')$, because the product density is positive only for those xy-points with positive density in both $u(x,y/R)$ and $u(x,y/T)$. In other words, the resulting uniform distribution is defined by the intersection, $T' = R * T$, and thus the posterior distribution is also defined by this intersection. For the single-polygon example, the Bayesian method produces the same result as the intuitive method of intersecting two polygons.

The Bayesian methodology can be extended to handle the fusion of two multipolygonal distributions. Assume that the report, R, is given in terms of a multipolygonal likelihood function:

$$g(R/x,y) = G \sum_j P(R_j)\ u(x,y/R_j) \tag{34}$$

where G is a positive constant, R_j represents the j-th polygon in the distribution, $P(R_j)$ is the weight of the j-th polygon, and $u(x,y/R_j)$ is a uniform density function defined by the j-th polygon. Assume further that the prior density for the target, T, is a multipolygonal density function:

$$f(x,y/T) = \sum_k P(T_k)\ u(x,y/T_k) \tag{35}$$

where T_k is the k-th polygon in the distribution, $P(T_k)$ is the weight of the k-th polygon, and $u(x,y/T_k)$ is a uniform density defined by the k-th polygon.

By using Bayes's formula, Eq. (30), with the multipolygonal representations of the likelihood function, Eq. (34), and prior density, Eq. (35), the posterior density is written:

$$f(x,y/T') = KG \sum_j \sum_k P(R_j)\ P(T_k)\ u(x,y/R_j)\ u(x,y/T_j) \quad . \quad (36)$$

The product of two uniform densities is proportional to a single uniform density:

$$u(x,y/R_j)\ u(x,y/T_k) = C_{jk}\ u(x,y/R_j*T_k) \quad (37)$$

where C_{jk} is a positive constant of proportionality. C_{jk} is easily derived because at any xy-point with positive density, the value of the density product on the left must equal C_{jk} times the value of the density on the right; therefore,

$$C_{jk} = A(R_j*T_k) \ / \ [|A(R_j)|\ |A(T_k)|] \quad (38)$$

where A(X) is defined as the area of the set of polygons, X.

The area of a set of polygons is the sum of the positive areas (clockwise polygons) and negative areas (counterclockwise polygons). Because of an IUCALC convention, the intersection, $R_j * T_k$, produces only clockwise polygons. A negative intersection also produces clockwise polygons, but a union produces both clockwise and counterclockwise polygons. The area of the set of polygons resulting from an intersection, negative intersection, or union is always positive.

The set of pologons that are the result of the intersection, $R_j * T_k$, can be denoted T'_{jkn}, where n = 1, N_{jk}. With this convention, the above uniform density can be synthesized in terms of uniform density functions that are defined by single polygons:

$$u(x,y/R_j*T_k) = \sum_n Q_{jkn}\ u(x,y/T'_{jkn}) \quad (39)$$

where the weights, Q_{jkn}, are defined so that the value of the left side is constant over all xy-points that have positive density. Under this

condition, the weights must be proportional to the areas of the elemental polygons:

$$Q_{jkn} = A(T'_{jkn}) \; / \; A(R_j * T_k) \quad . \tag{40}$$

The denominator is just the sum of the areas over n = 1, N_{jk}.

Putting the various pieces together, the posterior probability density function can now be written as:

$$f(x,y/T') = \sum_j \sum_k \sum_n P(T'_{jkn}) \; u(x,y/T'_{jkn}) \tag{41}$$

where the weights are given by:

$$P(T'_{jkn}) = KG \; P(R_j) \; P(T_k) \; C_{jk} \; Q_{jkn} \quad . \tag{42}$$

Using Eqs. (38) and (40), the equation for the weights can be reduced to the following:

$$P(T'_{jkn}) = K \; G \; H(R_j) \; H(T_k) \; A(T'_{jkn}) \tag{43}$$

where the height of polygon X, H(X), can be positive or negative, and is defined as:

$$H(X) = P(X) \; / \; |A(X)| \quad . \tag{44}$$

Note that the weights have to be normalized by the constant K such that they sum to 1 over all jkn-polygons.

For those intersections, $R_j * T_k$, that have no resultant polygons (R_j and T_k do not overlap), the jk-elements in Eq. (41) are skipped over and not recorded. In the computer program, the three-index jkn-list is transformed to a list based on a single index so that the representation of f(x,y/T') is similar to f(x,y/T), Eq. (35). The list of polygons

includes only those intersection polygons that are formed by overlapping R_j and T_k polygons.

In summary, Eq. (41) describes the fusion of two multipolygonal distributions. The polygons, T'_{jkn}, are formed by using the intersection operation in IUCALC, and the polygon weights are computed by using Eq. (43).

The preceding discussion was concerned with the fusion of positive information, but it could equally well be applied to negative information. The only difference is that negative intersections are used in place of (positive) intersections. For example, if a report, R, consists of negative information (e.g., the target is not in Polygon R_j), then the fused polygons are given by: $\overline{R}_j * T_k$, where $\overline{R}_j$ means the area outside Polygon R_j. The intersection produces Polygons T'_{jkn} and the calculation of the weights proceeds as before.

2. Example of Fusion

Figure 9 shows an example in which a multipolygonal report distribution is fused with a multipolygonal target distribution. Assume that the report distribution is given by Polygons R1 and R2, and that the height of R2 equals the negative height of R1 so that the probability density is concentrated in a rectangular annulus, as shown by the shaded area. Assume also that the target distribution is given by Polygons T1 and T2 and that their heights are also equal but opposite. The heights are given by:

$$\begin{aligned} H(R1) &= 1 \;/\; [A(R1) - A(R2)] \\ H(R2) &= -1 \;/\; [A(R1) - A(R2)] \\ H(T1) &= 1 \;/\; [A(T1) - A(T2)] \\ H(T2) &= -1 \;/\; [A(T1) - A(T2)] \end{aligned} \tag{45}$$

where all of the areas, A(X), are positive values. These values for height are derived by requiring the associated weights to sum to one. For example, $H(R1)\,A(R1) + H(R2)\,A(R2) = 1$.

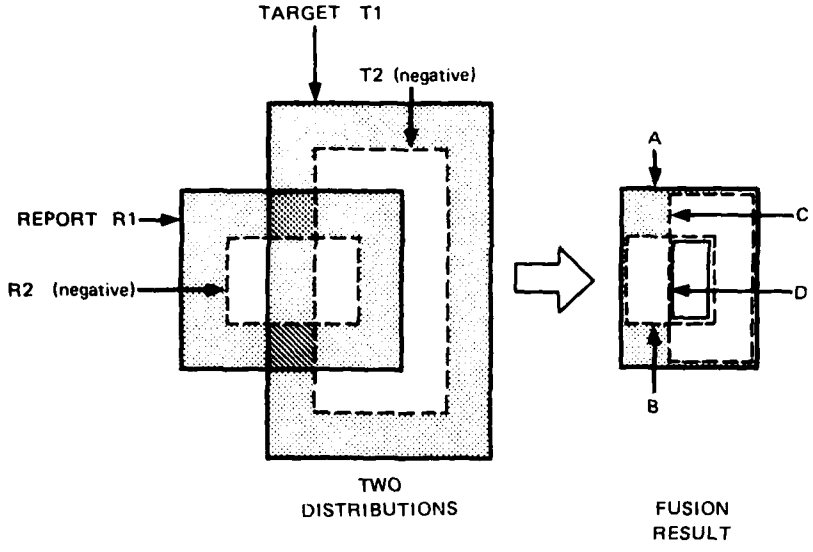


FIGURE 9 INTERSECTION FUSION OF TWO DISTRIBUTIONS WITH NEGATIVE WEIGHTS

Since there are 2 polygons in each input distribution, there are 4 polygons in the output distribution. These polygons are denoted A through D in Figure 9 and are the result of the following intersections:

$$\begin{aligned} A &= R1 * T1 \\ B &= R2 * T1 \\ C &= R1 * T2 \\ D &= R2 * T2 \end{aligned} \tag{46}$$

The weights for Polygons A through D are found by using Eq. (43):

$$\begin{aligned} P(A) &= K\ G\ H(R1)\ H(T1)\ A(A) \\ P(B) &= K\ G\ H(R2)\ H(T1)\ A(B) \\ P(C) &= K\ G\ H(R1)\ H(T2)\ A(C) \\ P(D) &= K\ G\ H(R2)\ H(T2)\ A(D) \end{aligned} \tag{47}$$

Substituting Eq. (45) into Eq. (47), the weights are seen to be proportional to the areas:

$$
\begin{aligned}
P(A) &= Z\ A(A) \\
P(B) &= -Z\ A(B) \\
P(C) &= -Z\ A(C) \\
P(D) &= Z\ A(D)
\end{aligned}
\tag{48}
$$

where the positive constant, Z, is the magnitude of the height of each polygon:

$$
Z = K\ G\ /\ ([A(R1) - A(R2)]\ [A(T1) - A(T2)]) \quad . \tag{49}
$$

Since the weights must sum to one, Z is also given by:

$$
Z = 1\ /\ [A(A) - A(B) - A(C) + A(D)] \quad . \tag{50}
$$

These two equations for Z may be used to derive the normalization constant, K.

Summarizing, Polygons A and D have positive height, Z, and Polygons C and B have negative height, -Z (negative polygons are shown by dashed lines on Figure 9). Therefore, by stacking the polygons on top of each other, the probability density adds and subtracts, and as a result, it is positive in only the two small squares shown in Figure 9. This is, of course, the intuitive answer obtained by visually analyzing Figure 9. Other cases can be imagined that are not at all intuitive. Then the power of the algorithm comes into play.

3. Program Code

Table 5 shows the section of code in program POL that controls the calculation of multipolygonal fusion distributions. The operator chooses one of two fusion options: intersection (KL = 2) or negative intersection (KL = 4). The operator also inputs the indices (L,M,N) of the "P" report distribution, and the indices (L,M,N2) of the "Q" target distribution. The J-limits (JP1,JP2) and (JQ1,JQ2) define the polygons in the P and Q distributions.

The DO-249 loops set up an outer product of P-polygons and Q-polygons. For a particular pair of polygons, JP and JQ, the IUCALC

Table 5

FUSION ALGORITHM

```
**
**FUSION (INTERSECTION, UNION, AND NEGATIVE INTERSECTION)
  230 KL=2  $GO TO 241
  235 KL=1  $GO TO 241
  240 KL=4
  241 JP1=JA(L,M,N)
      JP2=JB(L,M,N)
      JQ1=JA(L,M,N2)
      JQ2=JB(L,M,N2)
      JJ=0
      DO 249 JP=JP1,JP2
      DO 249 JQ=JQ1,JQ2
      CALL OAK(KL,JP,JQ,IA,IB,X,Y,XW,YW,IW,JMAX)
      IF(JMAX)255,249,243
  243 HPQ=H(JP)*H(JQ)
      DO 248 JO=1,JMAX
      IO=IW(1,JO)
      IM=IW(2,JO)
      DO 246 I=1,IM
      XS(I)=XW(I+IO)
  246 YS(I)=YW(I+IO)
      IM=IM+1
      XS(IM)=XS(1)
      YS(IM)=YS(1)
      CALL VEC(1,IM,XS,YS)
      JJ=JJ+1
      CALL PUT(L,M,N3,JJ,IM,XS,YS,IA,IB,JA,JB,J,X,Y,ILA,JLA)
  248 P(J)=HPQ*AREA(XW,YW,IW,JO)
  249 CONTINUE
      N=N3
      GO TO 185
  185 J1=JA(L,M,N)
      J2=JB(L,M,N)
      CALL NORM(J1,J2,P)
  186 CALL MOM(J1,J2,IA,IB,X,Y,P,H,EX,EY,EXX,EYY,EXY,XCEN,YCEN)
      IF(EXX.GT.0..AND.EYY.GT.0.) GO TO 187
      WRITE(6,25)
C     ENDFILE 6
      WRITE(8,25)
   25 FORMAT(*NEGATIVE VARIANCE*/)
      GO TO 100
  187 CALL EIGEN(EXX,EYY,EXY,R1,R2,D1,D2)
      CALL SAVE(L,M,N,EX,EY,R1,R2,D1,D2,EXX,EYY,EXY,SE,KE,KLA)
      INPUT(L,M,N)=.T.
      WRITE(6,28)
      WRITE(8,28)
C     ENDFILE 6
      WRITE(8,28)
   28 FORMAT(*$LIN 1*)
      GO TO 100
```

package is called through the interface subroutine, OAK. JMAX is checked to see how many polygons are returned; if there are no polygons, then a new pair of input polygons is processed. If there are resultant polygons, then the heights of the input polygons, JP and JQ, are found and multiplied together.

The DO-248 loop is over each one of the new polygons resulting from the intersection or negative intersection of polygons P and Q. First, the vertex points of the new polygon are read into arrays XS and YS and the redundant vertex points are added. Then, subroutine VEC is called to draw the new polygon on the display. Subroutine PUT is called to save the polygon in a new distribution with indices (L,M,N3). The new polygon is the JJ-th polygon in the distribution. The arrays IA, IB, JA, JB, X, Y are updated to reflect the addition. The last action of the DO-248 loop is to calculate the (unnormalized) weight of the new polygon. All polygon pairs are processed and the new distribution is completed when the DO-249 loops are finished.

After the new distribution is created, subroutine NORM is called to normalize the weights so that they sum to one. Subroutines MOM and EIGEN are called to calculate statistical parameters (see Section III-A) so that they may be saved by calling subroutine SAVE.

D. Conditional Probability Algorithm

Before using the fusion algorithm, a decision must be made as to which target distribution should be fused with the report distribution. The "conditional probability algorithm" can help make that decision. The idea is to calculate the likelihood of each target distribution with respect to the report distribution and then choose the target with the maximum likelihood. Figure 10 illustrates this idea. The report is associated with Target A because its likelihood is higher than that of Target B.

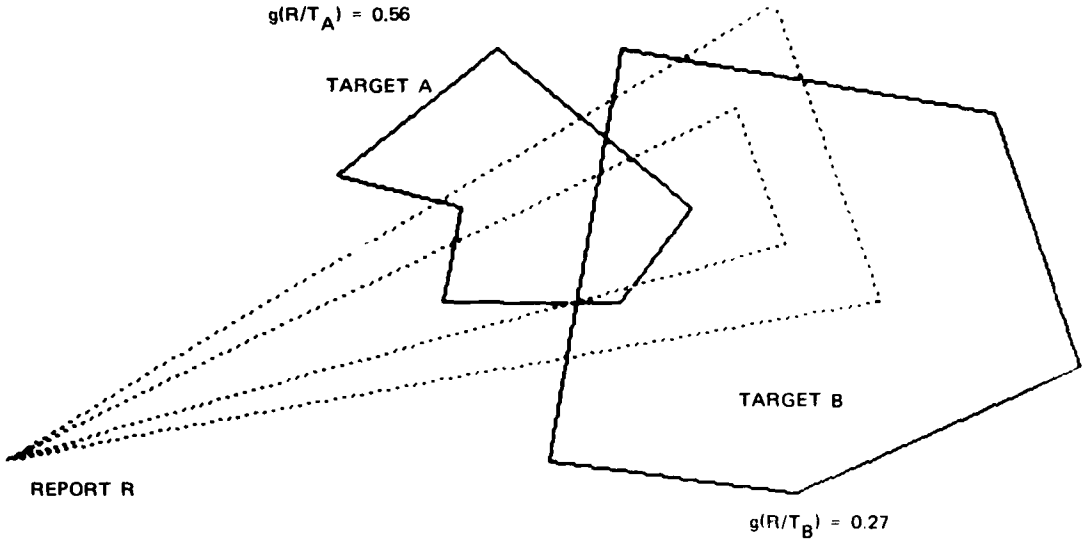


FIGURE 10 TARGET LIKELIHOOD

1. Program Logic

The conditional probability, g(R/T), is the probability of event R, given target T. It is computed by finding the average value of the likelihood function, g(R/x,y), assuming that the target probability density is given by f(x,y/T). Thus, the conditional probability is an integral over all xy-space:

$$g(R/T) = \iint g(R/x,y)\ f(x,y/T)\ dx\ dy \quad . \tag{51}$$

The multipolygonal representations of g(R/x,y) and f(x,y/T) were previously given in Eqs. (34) and (35).

The derivation of an expression for g(R/T) can be simplified by noting from Eq. (30) that the above integrand is just f(x,y/T′)/K. Then, by using Eq. (41), the xy-integration is simple because it involves a sum of uniform density functions; the result is:

$$g(R/T) = \sum_j \sum_k \sum_n P(T'_{jkn}) \; / \; K \quad . \tag{52}$$

This reduces to the desired expression:

$$g(R/T) = G \sum_j \sum_k H(R_j) \; H(T_k) \; A(R_j * T_k) \tag{53}$$

where G is a positive constant, $H(R_j)$ is the height of polygon R_j (as defined by Eq. (44)), $H(T_k)$ is the height of polygon T_k, and $A(R_j * T_k)$ is the total positive area of the polygons resulting from the intersection of polygons R_j and T_k.

For g(R/T) to be a conditional probability, the constant, G, must be defined such that g(R/x,y) is between 0 and 1 inclusive. The function g(R/x,y) is then the probability of the event R, given that the target is at point (x,y). For example, if R is the report of a detection event, then g(R/x,y) is the probability of detection, given that the target is at (x,y). If R is the report of a location measurement (latitude-longitude, range, bearing, etc.), then g(R/x,y) is the probability of the measurement occurring, given that the target is at (x,y).

2. Example of Conditional Probability

The example in the previous section (the rectangular annuli, Figure 9) is continued to demonstrate the calculation of conditional probability. Equations (45) and (46) are substituted into Eq. (53) and the result is:

$$g(R/T) = G \; \frac{A(A) - A(B) - A(C) + A(D)}{[A(R1) - A(R2)]\,[A(T1) - A(T1)]} \quad . \tag{54}$$

This is the likelihood of target T, with respect to report R. The result is more easily interpreted if the constant, G, is set equal to the shaded report area (see Figure 9):

$$G = A(R1) - A(R2) \quad . \tag{55}$$

The assumption is that if the target is in the shaded report area, then there is a 100-percent chance that the location measurement occurs. With G so defined, the likelihood--Eq. (54)--becomes a conditional probability. It is just the ratio of the area of the two small squares, A(A) - A(B) - A(C) + A(D), divided by the area of the target annulus, A(T1) - A(T2). In other words, the probability of the target being in both the report and target areas, given that it is in the target area, is the ratio of the two areas--again, an intuitive result.

Figure 11 shows a second example of a problem in which the conditional probability algorithm would be useful. Instead of a localization report likelihood function, a search pattern likelihood function is defined. If the target is inside the small dotted polygon, then there is a 100-percent probability of detecting the target. If the target is inside the large dotted polygon but outside the small dotted polygon, then there is a 50-percent probability of detecting the target. The conditional probability of detection, g(R/T), given a target distribution (the three ellipses), can then be computed by using Eq. (53).

3. Program Code

Table 6 shows the section of code used to calculate conditional probabilities. The operator inputs the indices of the P-distribution (L,M,N) and the Q-distribution (L,M,N2). The conditional probability is the probability of P, given Q.

The DO-254 loops set up an outer product of P and Q polygons, JP and JQ. Subroutine OAK returns a set of polygons that are the result of the intersection of polygons JP and JQ. If there are polygons in the intersection (JMAX positive), then the heights of the input polygons, JP and JQ, are found. Function AREA is used to calculate the area of each polygon in the intersection; and the product of the heights and the area is summed in the DO-252 loop.

Once all of the polygon pairs are intersected, the resulting sum, PROB, is divided by the magnitude of the height of the first polygon in the P-distribution; thus, the G-factor [see Eq. (53)] is a

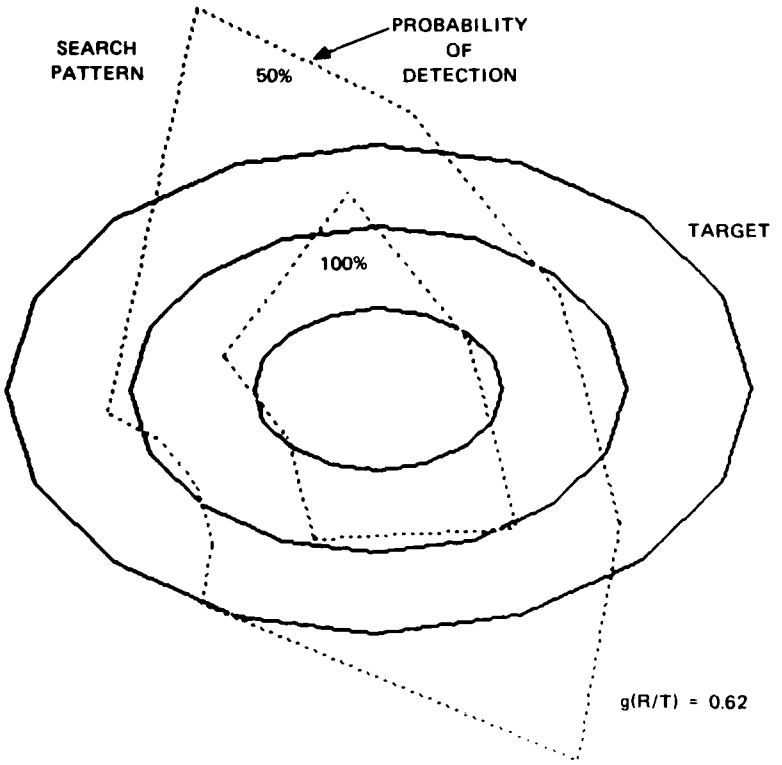


FIGURE 11 PROBABILITY OF DETECTION

reciprocal height. For single-polygon likelihood functions, this scheme is equivalent to assuming a 100-percent probability of receiving the localization report, conditioned on the target being at an xy-point inside the polygon. For multipolygonal likelihood functions, this scheme is not a very good one. What is needed is a simple algorithm to find the maximum height of a likelihood function when G = 1. Then a new G-factor can be defined so that the maximum height is equal to an input probability, such as 100 percent. The maximum-height algorithm has not been developed yet.

Table 6

CONDITIONAL PROBABILITY ALGORITHM

```
**
** CONDITIONAL PROBABILITY
  250 PROB=0.
      JP1=JA(L,M,N)
      JP2=JB(L,M,N)
      JQ1=JA(L,M,N2)
      JQ2=JB(L,M,N2)
      DO 254 JP=JP1,JP2
      DO 254 JQ=JQ1,JQ2
      CALL OAK(2,JP,JQ,IA,IB,X,Y,XW,YW,IW,JMAX)
      IF(JMAX)255,254,251
  251 HPQ=H(JP)*H(JQ)
      DO 252 JO=1,JMAX
  252 PROB=PROB+HPQ*AREA(XW,YW,IW,JO)
  254 CONTINUE
      PROB=PROB/ABS(H(JP1))
      WRITE(6,34) N,N2,PROB
      WRITE(8,34) N,N2,PROB
   34 FORMAT(*PROB OF *,I2,* GIVEN *,I2,* =*,F5.2)
      GO TO 100
  255 WRITE(6,35) JMAX
      WRITE(8,35) JMAX
   35 FORMAT(*1UCALC ERROR *,I3)
      GO TO 100
```

E. Prediction Algorithm

A "prediction algorithm" was developed to move target uncertainty areas in time. A multipolygonal target position distribution, (1,M,N), is assigned a multipolygonal velocity distribution, (2,M,N). The algorithm calculates a predicted position polygon for each polygon in (1,M,N) based on each polygon in the velocity distribution (2,M,N). Furthermore, a predicted velocity distribution is calculated.

1. Program Logic

a. Predicted Position Polygons

Recall that the velocity distribution, (2,M,N), when defined with respect to a position distribution, (1,M,N), represents all possible positions of the centroid of the position distribution at the next time step, M + 1. Consequently, all possible positions at time step M + 1 of the centroid of any polygon, P, in the position distribution are represented by a translation of said velocity distribution.

This translation is determined by adding to all the vertices of each polygon in (2,M,N) the vector determined by subtracting the centroid of the position distribution (1,M,N) from the centroid of the position polygon P. That is, if V_i is a polygon in velocity distribution (2,M,N) with vertices given by v_{ij}, and S is the centroid of position distribution (1,M,N), and $\bar{p}$ is the centroid of polygon P, then the vertices of the translated polygon $V_i(\bar{p})$ are given by:

$$v_{ij}(\bar{p}) = v_{ij} + \bar{p} - S, \text{ for all } j. \tag{56}$$

Similarly, if p is any point of positive information in position polygon P, all possible positions of p at time step M + 1 are represented by translating the velocity distribution (2,M,N) by the vector determined by subtracting S from p. The vertices of this translated distribution, $\{V_i(p)\}$, are given by $v_{ij}(p) = v_{ij} + p - S$, for all i and j.

This argument is sufficient to deduce the predicted position polygon defined by a position polygon, P, and a velocity polygon, V, and is valid for negative as well as positive information polygons. The predicted position polygon thus derived is always a positive information area. In the context of this argument, however, negative information polygons must be interpreted in terms of positive information. That is, a negative information polygon has no interior; it consists solely of a boundary of positive information. Moreover, a negative information area can exist only when enclosed by a positive information area. Thus, the predicted position polygon determined by P and V is the union of the V(p) over all p in P:

$$\left(\bigcup_{p \epsilon P} V(p) \right) \quad .$$

One can visualize the predicted position polygon as constructed by the following process. The centroid, $\bar{v}(\bar{p})$, of the translated velocity polygon $V(\bar{p})$ is determined. The position polygon, P, is rigidly translated to have its centroid at $\bar{v}(\bar{p})$. The predicted position polygon then is the figure resulting when the velocity polygon is rigidly translated to

have its centroid successively at each point of the translated position polygon. It is not hard to prove the equivalence of

$$\bigcup_{p \in P} V(p) \text{ to } \bigcup_{v \in V(\bar{p})} P(v),$$

where P(v) is the rigid translation of the position polygon that has its centroid at v. If the vertices of P are given by p_j, the vertices of translated polygon P(v) are given by:

$$p_j(v) = p_j + v - \bar{p}, \text{ for all } j \quad . \tag{57}$$

It was conjectured and then proved that the algorithm described below computes the predicted position polygon, given a position polygon, P, and a velocity polygon, V.

The Exact Algorithm. The translated velocity polygon $V(\bar{p})$ is first computed. The vertices of $V(\bar{p})$ are given by Eq. (56), and the centroid of $V(\bar{p})$ is given by $\bar{v}(\bar{p}) = \bar{v} + \bar{p} - S$, where $\bar{v}$ is the centroid of V. Subsequently, the translated position polygons $P(v_j(\bar{p}))$ are determined for all vertices, $v_j(\bar{p})$, of the velocity polygon. The vertices of polygon $P(v_j(\bar{p}))$ are given by Eq. (57)--that is, $p_i(v_j(\bar{p})) = p_i + v_j(\bar{p}) - \bar{p}$. The centroid of $P(v_j(\bar{p}))$ is $v_j(\bar{p})$. A parallelogram is generated corresponding to each pair of consecutive vertices, p_i and p_{i+1}, of position polygon P, and each vertex, v_j, of velocity polygon $V(\bar{p})$. Thus, for each pair of i and j indices, the parallelogram consists of vertices $p_i(v_j)$ and $p_{i+1}(v_j)$ of polygon $P(v_j)$, and vertices $p_{i+1}(v_{j+1})$ and $p_i(v_{j+1})$ of polygon $P(v_{j+1})$. Then the union of all the $P(v_j)$ and all the parallelograms is computed iteratively (see Figure 12).

When the position polygon and velocity polygon represent positive information areas, the predicted position polygon is all the area surrounded by the outer boundary of the union, including the outer boundary. When either the position polygon or the velocity polygon (but not both) represent negative information, the predicted position polygon is the union. In this case, a predicted negative information area need

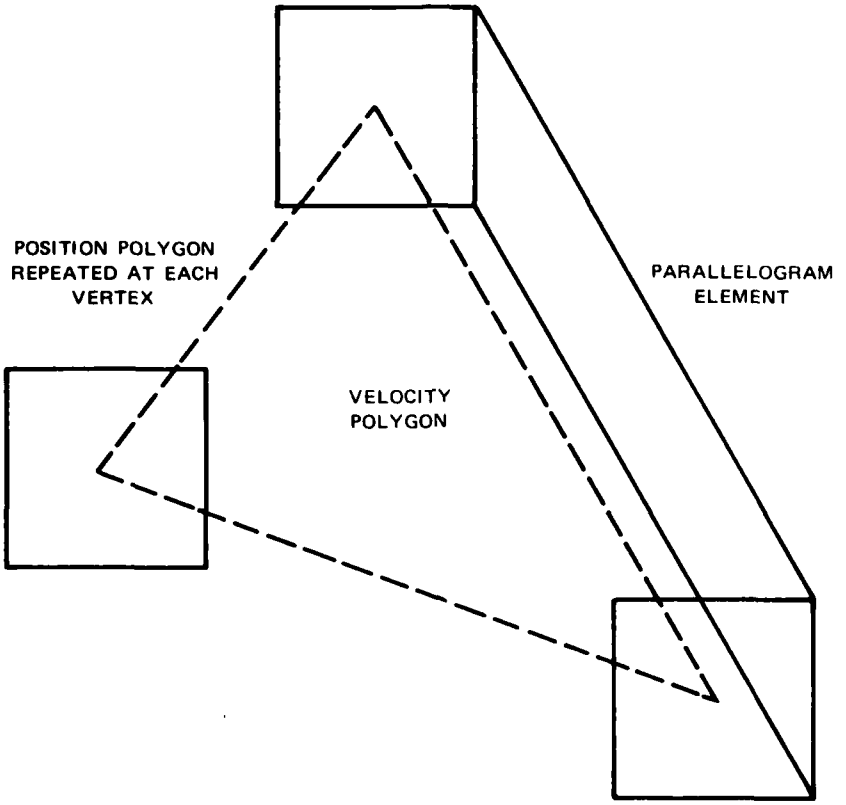
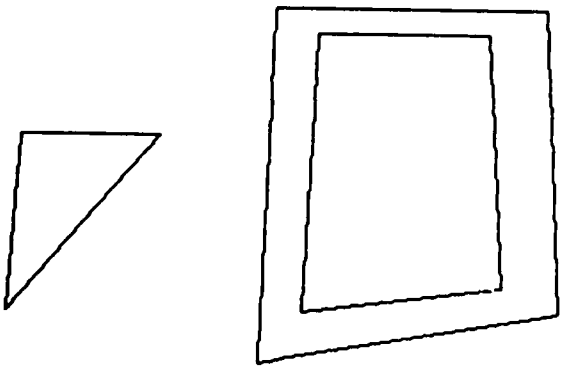


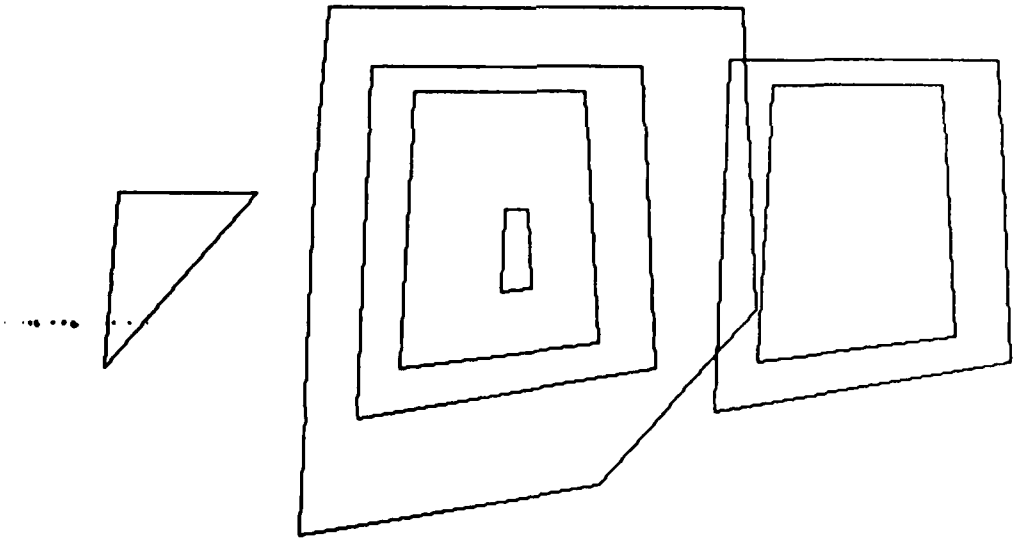
FIGURE 12 GRAPHIC OUTLINE OF EXACT ALGORITHM

not exist. When the union has an inner boundary, the area enclosed by the inner boundary is the negative information area (see Figure 13). When there is no inner boundary, the negative information area does not exist. When both position polygon and velocity polygon represent negative information, it can be proven that the predicted position polygon generated by them is a subset of the predicted position polygon generated by the positive information position and velocity polygons that enclose them.

The exact algorithm is based on complete enumeration. It generally requires that (M1+1) · M2 unions be performed, where M1 and M2, respectively, are the number of distinct vertices in the position and velocity polygons. Algorithms to compute the predicted position polygon that are more efficient, though less obvious, than the exact algorithm



(a) INPUT POSITION DISTRIBUTION (left polygon) AND INPUT VELOCITY DISTRIBUTION (quadrilaterals). THE INNER QUADRILATERAL OF THE VELOCITY DISTRIBUTION IS A NEGATIVE INFORMATION AREA



(b) PREDICTED POSITION DISTRIBUTION (largest polygon--a positive information area; smallest polygon--a negative information area) AND PREDICTED VELOCITY DISTRIBUTION (right pair of quadrilaterals)

FIGURE 13 EXAMPLE OF TRIANGULAR POSITION DISTRIBUTION AND QUADRILATERAL VELOCITY DISTRIBUTION PREDICTION

were expected to exist. A much more efficient, but heuristic, algorithm has been developed (based on study of the exact algorithm) to compute the predicted position polygon determined by convex positive information position polygon P and convex or nonconvex positive information velocity polygon V. It does not appear applicable to negative information polygons. This algorithm is described in the following paragraphs.

The Quick Algorithm. The algorithm assumes that the velocity and position polygons have vertices ordered in the clockwise direction. The translated polygon $V(\bar{p})$ is first computed. A vertex known to be on the boundary of the predicted position polygon is selected. (Such a vertex is the northernmost vertex of the position polygon when translated so that its centroid is at the northernmost vertex of the polygon $V(\bar{p})$.) Suppose the last vertex selected for the predicted position polygon corresponds to a vertex--say, with index k--of the position polygon translated to have its centroid at a vertex, v_j, of polygon $V(\bar{p})$. The equation of this vertex, $p_k(v_j)$, is given by Eq. (57), specifically $p_k(v_j) = p_k + v_j - \bar{p}$. Usually the next vertex of the predicted position polygon is given by either the vertex with index k + 1 of polygon $P(v_j)$, $p_{k+1}(v_j)$, or the vertex with index k of polygon $P(v_{j+1})$, $p_k(v_{j+1})$. The oriented area of the triangle defined by the three points $p_k(v_j)$, $p_{k+1}(v_j)$, and $p_k(v_{j+1})$ is computed. [It is assumed that a triangle with vertices ordered in a clockwise direction has positive area (refer to III-A).] When the area of the triangle is strictly positive, the next vertex of the predicted position polygon is usually $p_{k+1}(v_j)$. When the area is strictly negative, the next vertex is usually $p_k(v_{j+1})$. When the area is zero, either vertex will usually do. Vertices continue to be chosen in the manner described above until the first vertex selected is chosen again. The algorithm in this way determines a polygon, U, consisting of points that usually are vertices of the predicted position polygon.

It has been shown that sometimes some point selected in this manner is actually in the interior of the predicted position polygon. Although this is an infrequent occurrence, a further procedure of the algorithm was developed to replace such interior points with the

proper vertices. This procedure is to replace the polygon U, with the union of U and all the $P(v_j)$. Thus, the algorithm yields the final result:

$$U \cup (\bigcup_j P(v_j))$$

for the predicted position polygon. The heuristic result compares very well to the result computed by the exact algorithm.

The predicted position polygon computed by the above methods is stored as a polygon in distribution (1,M+1,N).

b. Predicted Velocity Polygons

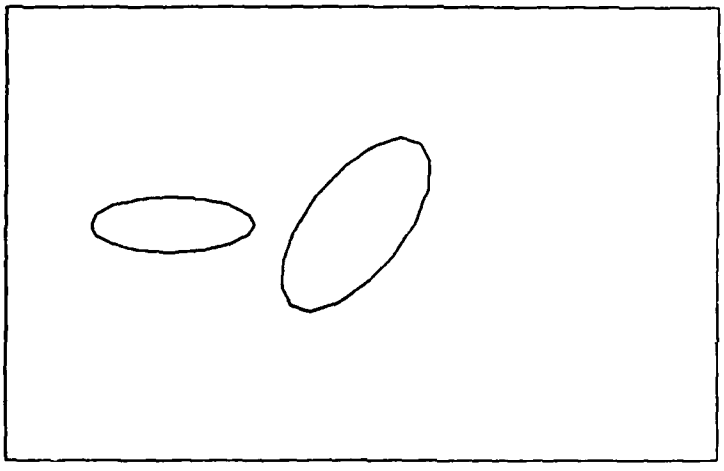
The following algorithm provides the logic for the computation of the predicted velocity distribution (2,M+1,N). The algorithm determines the time step, M_o, of the most recently input velocity distribution associated with target N. The predicted velocity distribution is then computed by translating velocity distribution $(2,M_o,N)$ by the vector derived by subtracting the centroid of distribution $(1,M_o,N)$ from the centroid of (1,M+1,N). If V_j is a polygon in velocity distribution $(2,M_o,N)$, with vertices given by v_{ij}, if S_o and S_1 are the centroids of distributions $(1,M_o,N)$ and (1,M+1,N), respectively, then the vertices of the predicted velocity polygon determined by V_j are given by

$$v_{ij}(S_1) = v_{ij} + S_1 - S_o \quad . \tag{58}$$

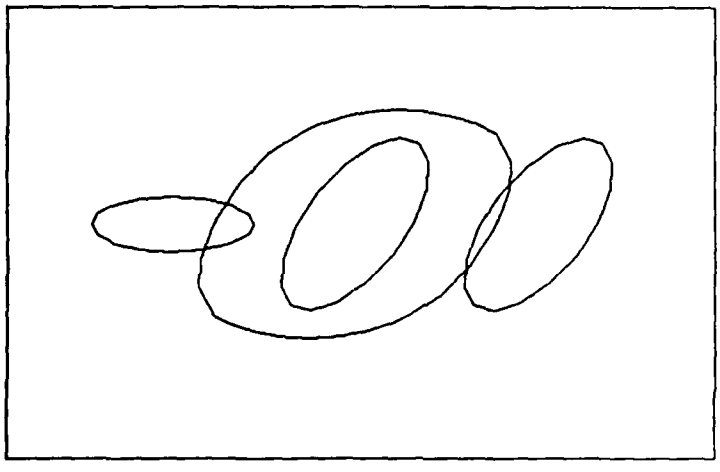
2. Examples of Prediction

Figure 14 illustrates the evolution over one time step of an elliptical position and velocity distribution, Figure 14(a). The predicted position distribution is the large elliptical polygon in Figure 14(b). The predicted velocity distribution is also elliptical [the rightmost polygon in Figure 14(b)].

Figure 13 shows the evolution over one time step of a triangular position distribution and a velocity distribution composed of nested



(a) INPUT POSITION DISTRIBUTION (left polygon) AND
INPUT VELOCITY DISTRIBUTION (right polygon)



(b) PREDICTED POSITION DISTRIBUTION (largest polygon) AND
PREDICTED VELOCITY DISTRIBUTION (right polygon)

FIGURE 14 EXAMPLE OF ELLIPTICAL POSITION AND VELOCITY DISTRIBUTION PREDICTION

quadrilaterals, Figure 13(a). The inner quadrilateral is a negative information area. At time step 2, Figure 13(b), the predicted position distribution has a positive and a negative information area. The positive information area is the pentagon and the negative information area is the smallest quadrilateral within it.

3. Program Code

Table 7 shows the section of code in program POL that controls the calculation of predicted position and velocity distributions. The terminal operator inputs M, the time step from which the prediction is calculated, and N, the distribution number.

The screen is erased, then subroutine DRAW is called so that the position distribution and velocity distribution that are input to the prediction algorithm can be redrawn. If land masses have been defined (that is, when LAND = TRUE), subroutine VEC is called so that the land masses can be redrawn.

Subroutine MOVMENT is called to calculate the predicted position distribution. Subroutine MOVECEN is called to calculate the predicted velocity distribution. When an error is detected by subroutines MOVMENT or MOVECEN or any subroutines called by them, control automatically returns to label 100 of the main program POL.

a. Position Prediction

Subroutine MOVMENT calls subroutines CONCLDE, CUMUN, NUCUMUN, SELECT, UPDATE, and VEC. Its input parameters are: M, N, JA, JB, X, Y, IA, IB, P, XCEN, YCEN, XW, J, ILA, JLA, SE, KE, KLA, H, LAX, LAY, LIA, LIB, LAN, LAND, LN, LE, LS, LW, INC, CLOSE. The arrays XW and YW are work areas. The remaining parameters are as defined in Sections II-B and II-C. The output parameters are JA, JB, X, Y, IA, IB, P, XCEN, YCEN, J, ILA, JLA, SE, KE, KLA, and H.

Subroutine MOVMENT first determines the time step, M1, for which the predicted position is computed. In fact, M1 = M + 1. The

Table 7

PREDICTION ALGORITHM

```
**
**PREDICTION ALGORITHM
  500 CONTINUE
      M1=M+1
      WRITE(8,48)
      WRITE(6,48)
   48 FORMAT(*$ERA G*)
      WRITE(8,50)
      WRITE(6,50)
   50 FORMAT(*$LIN 4*)
      CALL DRAW(1,M,N,IA,IB,JA,JB,X,Y,P)
      CALL DRAW(2,M,N,IA,IB,JA,JB,X,Y,P)
      IF(.NOT.LAND) GOTO 700
      WRITE(6,55)
      WRITE(8,55)
   55 FORMAT(*$LIN 8*)
      LAN=LJB(1,1,1)
      DO 600 J=1,LAN
        I1=LIA(J)
        I2=LIB(J)
        CALL VEC(I1,I2,LAX,LAY)
  600 CONTINUE
  700 CONTINUE
      WRITE(6,28)
C     ENDFILE 6
      WRITE(8,28)
      CALL MESSAGC(7HPOL     ,7HMOVMENT)
      CALL MOVMENT(M,N,JA,JB,X,Y,IA,IB,P,XCEN,YCEN,
     $  XW,YW,J,ILA,JLA,FLAG,SE,KE,KLA,H,LAX,LAY,LIA,LIB,
     $  LAN,LAND,LN,LE,LS,LW,INC,CLOSE)
      CALL MESSAGA(7HPOL     )
      IF(.NOT.FLAG) GOTO 100
      CALL MESSAGC(7HPOL     ,7HMOVCEN )
      CALL MOVECEN(M,N,JA,JB,X,Y,IA,IB,P,SE,KE,KLA,
     $  XW,YW,J,ILA,JLA,XCEN,YCEN,H,LAX,LAY,LIA,LIB,
     $  LAN,LAND,LN,LE,LS,LW,INPUT,COUNT,FLAG)
      CALL MESSAGA(7HPOL     )
      GOTO 100
```

DO-1000 loop is over each polygon in the input velocity distribution, (2,M,N), while the DO-900 loop is over each polygon in the input position distribution, (1,M,N).

For each execution of the DO-900 loop, a translation of the velocity polygon is computed with respect to the position polygon and stored in arrays XC and YC, redundant vertex points included. The weights of the velocity and position polygons are evaluated in order to determine whether positive or negative information areas are represented by them. When two positive information areas are indicated, subroutine

NUCUMUN or subroutine CUMUN is called at operator option. When one negative information area is indicated, subroutine CUMUN is called. When two negative information areas are indicated, no further operation in the DO-900 loop is executed.

Subroutine NUCUMUN performs the heuristic quick algorithm (refer to Section III-E-1-a) to calculate the predicted position polygon. NUCUMUN assumes that the vertices of the input polygons are in clockwise order. NUCUMUN calls the function NORTH, which calculates the index of the most north vertex of an input polygon; the function AREA, which calculates the area of an input polygon using the triangle-area method; and subroutine PREIU. The input parameters of NUCUMUN are:

NGON1	The polygon number of the position polygon
NGON2	The polygon number of the velocity polygon
IA	The array of vertex-start indices
IB	The array of vertex-stop indices
X	The x-coordinates of all polygons
Y	The y-coordinates of all polygons
XCEN	The array of x coordinates of polygon centroids
YCEN	The array of y coordinates of polygon centroids
PGX	The x coordinates of the translation of the velocity polygon with respect to NGON1
PGY	The y coordinates of the translation of the velocity polygon with respect to NGON1.

Certain scratch arrays are passed to NUCUMUN for work areas. These are XW, YW, IW, XS, YS, IS, and WK. The output parameters of NUCUMUN are:

FLAG = FALSE	The signal that NUCUMUN or IUCALC has abnormally terminated processing
RX	The array of x coordinates of the predicted position polygon
RY	The array of y coordinates of the predicted position polygon
IR1	The number of distinct vertices in the predicted position polygon
JMAX	The number of resultant polygons (including zero) computed by IUCALC or an error flag indicating that IUCALC has abnormally terminated processing.

The arrays PARX and PARY contain the x coordinates and y coordinates, respectively, of triangles, the signs of whose areas determine the next vertex.

Subroutine NUCUMUN first calls function NORTH to compute the most north vertex in PGX, which is stored in NONGON2, and the most north vertex in NGON1, which is stored in NONGON1. Using this information, the starting vertex on the boundary of the predicted position polygon is then determined and stored in both (RX(1), RY(1)) and (PARX(1), PARY(1)). The formula for this vertex is given by Eq. (57). The DO-2999 loop determines the two candidates for the next vertex on the boundary. The first candidate (the next vertex of NGON1 translated to the vertex of NGON2 associated with the last vertex selected for the boundary) is stored in (PARX(2), PARY(2)); the second candidate (the same vertex of NGON1 translated to the next vertex of NGON2) is stored in (PARX(3), PARY(3)). The formula for these vertices is given by Eq. (57). The area of the triangle defined by the arrays PARX and PARY is then evaluated and stored in A. When A is close to zero, the two candidates are collinear. In order that the next vertex be as far from the last one as possible, the two distances D2 and D3 are computed. D2 is the distance between the last vertex selected and the first candidate; D3 is the distance between the last vertex selected and the second candidate. The first or second candidate is selected, depending, respectively, on whether D2 or D3 is greater. When A is positive and not close to zero, the first candidate is selected. When A is negative and not close to zero, the second candidate is selected. The vertex selected is stored in the next position of arrays RX and RY. The DO-2999 loop terminates normally when the starting vertex is selected as the next vertex. In this case, the DO-5000 loop is executed. This loop successively computes the union of NGON1 translated to each vertex of NGON2, with the polygon in arrays RX and RY. This is done by repeated calls to PREIU. If no error is detected by PREIU, the final result is the boundary of the predicted position polygon. It is stored in the arrays RX and RY, and then control returns to MOVMENT. An abnormal termination of the DO-2999 loop

results in an error message with control returning from NUCUMUN to label 100 of the main program POL.

Subroutine NUCUMUN provides a quick method of generating the predicted position polygon. It is a heuristic method, however. When this algorithm is applied to nonconvex position polygons, the predicted position polygon it calculates is not correct in all particulars. Errors in NUCUMUN calculations result in the generation of polygon-like figures where sides intersect at interior points. An example of this is shown in Figure 15. The operator can avoid this situation by using NUCUMUN only when the input position polygon is convex. When questionable NUCUMUN results are generated, the code allows the operator to recalculate the predicted target distribution using subroutine CUMUN.

To generate the predicted position polygon, subroutine CUMUN uses a method that has been proved correct. This method, however, requires many more calculations than NUCUMUN to generate the predicted position polygon.

Subroutine CUMUN performs the exact algorithm (refer to Section III-E-1-a) to calculate the predicted position polygon. CUMUN calls the function COLL, which determines whether four vertices computed by CUMUN (whose x and y coordinates are stored, respectively, in arrays PARX and PARY) form a parallelogram, and it also calls subroutine PREIU. The input parameters of CUMUN are exactly the same as the input parameters of NUCUMUN. The scratch arrays passed to CUMUN for work areas are XW, YW, IW, WK, RX, RY, and IR. The output parameters of CUMUN are XS, YS, IS, FLAG, and JMAX. FLAG and JMAX are the same as in NUCUMUN. For the remaining output parameters we have:

XS	The array of x coordinates of the resulting predicted positive information area
YS	The array of y coordinates of the resulting predicted positive information area
IS	The integer array that contains the index of the start of each polygon and the number of vertices of each polygon in the predicted positive information area.

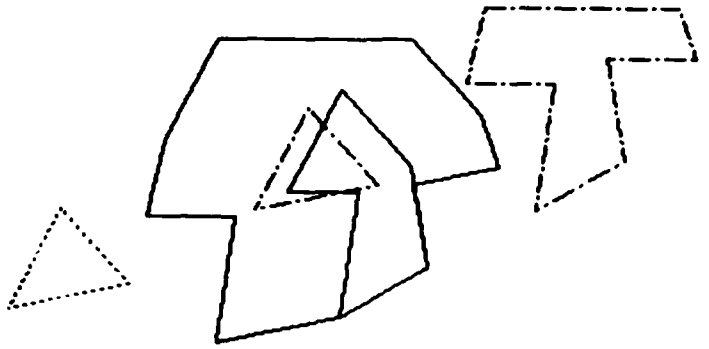


FIGURE 15 ERRONEOUS NUCUMUN RESULT

The DO-1500 loop of subroutine CUMUN is over the number of distinct vertices in NGON2. When the DO-1500 loop is executed the J-th time, NGON1 is translated so that the translation's centroid is the J-th vertex of NGON2. The translation is stored in arrays XW and YW, redundant vertex not included. The formulas for the vertices of this polygon are given by Eq. (57). The DO-1000 loop is over the number of distinct vertices in NGON1. When the DO-1000 loop is executed the I-th time, four vertices are computed. Two vertices are vertices I and I + 1 of NGON1 translated to have its centroid at vertex J of NGON2. Two vertices are vertices I + 1 and I of NGON1 translated to have its centroid at vertex J + 1 of NGON2. The x and y coordinates of these vertices are stored, respectively, in PARX and PARY. Function COLL determines when the points in PARX and PARY define a parallelogram, in this case COLL = FALSE. When PARX and PARY define a parallelogram, the union of the parallelogram and the polygon currently stored in arrays XW and YW is computed by a call to subroutine PREIU. After the DO-1000 loop completes processing, and if the DO-1500 loop is in its first iteration, the contents of arrays XW and YW are transferred, respectively, to arrays RX and RY. After the DO-1000 loop completes processing and for all iterations of the DO-1500 loop except the first, the union of the polygon stored in arrays XW and YW and the polygon stored in arrays RX and RY is computed by a call to subroutine PREIU. When the DO-1500 loop completes

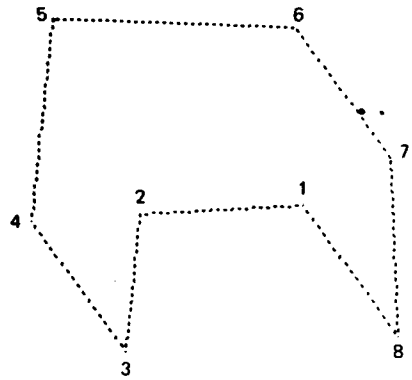
processing, the final result is stored in XS, YS, IS, and control returns to MOVMENT.

During the execution of subroutine CUMUN, problems are sometimes encountered. The exact algorithm that CUMUN implements requires that frequently the union be computed between two polygons having two or more vertices in common. Erroneous results have been generated by the IUCALC subroutines, which are used to compute the unions in this algorithm. IUCALC may generate polygon-like figures where sides intersect in interior points as exhibited by Figure 16. Or it may generate figures with coinciding sides as exhibited by Figure 17. Investigation seems to implicate the routine, ENSYD2, which determines when a point is inside a polygon. It was discovered that this routine has limited application because it must assume that the point is on no ray coincident with a side of the polygon.[4] A condition that is necessarily unsatisfied by the algorithm using it. At this time the problem remains uncorrected.

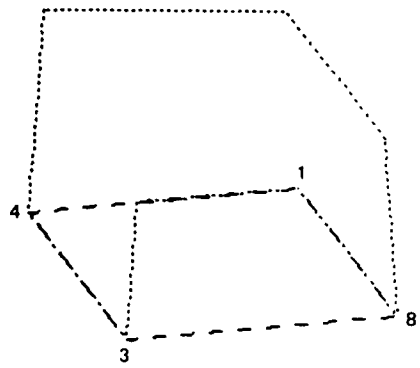
When control returns from CUMUN to MOVMENT, MOVMENT calls subroutine SELECT. Subroutine SELECT, depending on the value of K, extracts a polygon from arrays XS, YS, and IS and stores it in arrays XPC and YPC. The input parameters of subroutine SELECT are K, JMAX, XS, YS, IS.

When K = -1, the polygon representing the outer boundary of the positive information area is selected; when K = -2, the polygon representing the inner boundary of the positive information area is selected, if it exists. The parameters JMAX, XS, YS, IS are defined as in CUMUN. The output parameters are K, XPC, YPC, and IC1. The signal that the inner boundary of the positive information area was to be selected, but that it did not exist, is K = 0. When a polygon is selected from XS, YS, and IS, its x and y coordinates are stored, respectively, in arrays XPC and YPC. The variable IC1 contains the number of distinct vertices in the polygon.

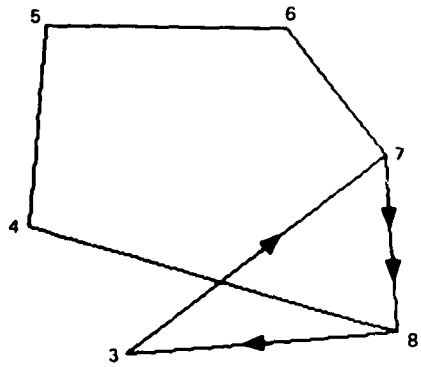
When control returns from subroutine SELECT to MOVMENT, but no polygon was selected (that is KAY = 0), no further operation in the DO-900 loop is executed. When control returns from SELECT to MOVMENT



(a) POLYGON



(b) POLYGON AND PARRALELOGRAM



(c) ERRONEOUS RESULT

FIGURE 16 DEVELOPMENT OF ERRONEOUS IUCALC RESULT

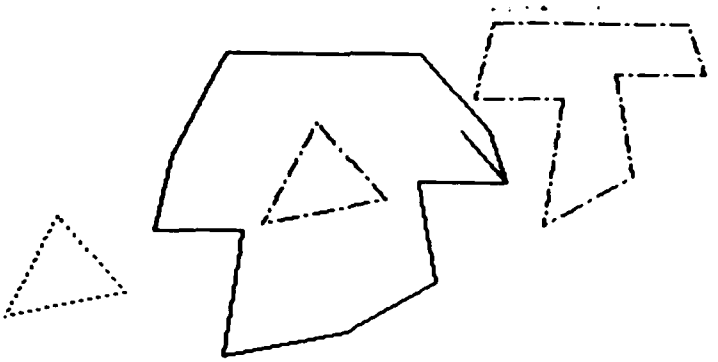


FIGURE 17 ERRONEOUS IUCALC RESULT

and a polygon was selected or when control returns from NUCUMUN to MOVMENT, MOVMENT calls subroutine CONCLDE. Subroutine CONCLDE calls the land interaction algorithm and stores the resulting predicted position polygon. This is discussed in a later section. When control returns to MOVMENT and no error has been detected, an iteration of the DO-900 and DO-1000 loops has been completed. When the DO-1000 and DO-900 loops have completed execution, MOVMENT calls subroutine UPDATE to compute and save the statistics for the predicted position distribution. Subroutine UPDATE calls subroutines NORM, MOM, and SAVE.

b. Velocity Prediction

Subroutine MOVECEN calls subroutines NEWCEN, UPDATE, and VEC; and functions ENSYD2 and MINIMAX. Its input parameters are: M, N, JA, JB, X, Y, IA, IB, P, SE, KE, KLA, XW, YW, J, ILA, JLA, XCEN, YCEN, H, LAX, LAY, LIA, LIB, LAN, LAND, LN, LE, LS, LW, INPUT, and COUNT. The arrays XW and YW are work areas. The remaining parameters are defined in Sections II-B and II-C. The output parameters are JA, JB, X, Y, IA, IB, P, SE, KE, KLA, J, ILA, JLA, XCEN, YCEN, and H.

Subroutine MOVECEN first determines the time step, M1, for which the predicted velocity distribution is computed. The DO-200

loop determines the time step, MK, of the most recently input velocity distribution associated with target N. If this loop is completed with no error detected, the vector defined by the centroid of the predicted position distribution (1,M1,N) minus the centroid of the position distribution (1,MK,N) is evaluated and the resulting x and y coordinates are stored in variables SE1 and SE2, respectively. The DO-4000 loop is over the polygons in the velocity distribution (2,MK,N). At iteration J1 of the DO-4000 loop when no land is defined, control passes to label 3100 where velocity polygon J1 is translated by vector (SE1,SE2). Then subroutine MOVECEN calls VEC to draw the translated polygon on the display. Subroutine PUT is called to store the translated polygon in a new distribution with indices (2,M1,N) by updating the arrays IA, IB, JA, JB, X and Y, and the variables J, ILA, and JLA. The translated polygon is the J-th polygon in X and Y. The weight of polygon J is set equal to polygon J1 in the last calculation of the DO-4000 loop. Before returning to the main program, subroutine MOVECEN calls subroutine UPDATE to compute and save the statistics of the predicted velocity distribution. The behavior of MOVECEN when land is defined is discussed in a later section.

F. Land Interaction Algorithm

A "land interaction algorithm" was wanted that would allow target distributions to move along or around land masses such as coastlines, straits, and islands. The land interaction algorithm was to be used in conjunction with the already implemented prediction algorithm. The implementation of such an algorithm would provide more realistic display of target distributions, allow the investigation of assumptions underlying the definition of target distributions, and show by example the capabilities of the polygonal data structure.

Initially, the only requirement of the land interaction algorithm was that the consequent graphic display, of target distributions interacting with land, should provide reasonable suggestions for the uncertain behavior of targets as they maneuver around land. There were thus conceivably many satisfactory approaches to development of the algorithm.

As an introduction to the algorithm finally implemented, some approaches to the land interaction algorithm are described. Through an investigation of these approaches, additional requirements were adopted. The approaches considered are by no means exhaustive. Since the purpose of the research was to demonstrate the feasibility of the polygonal data structure concept, rather than to select the most appropriate models of target motion, the algorithm implemented should not be regarded as optimal. It should be regarded as an example of the types of algorithm made possible by the use of this data structure.

As seen in the development of the prediction algorithm, in the absence of land the uncertainty characterized by the position and velocity distribution at a specified time step completely determines position uncertainty at the next time step. A predicted negative information polygon indicates that there exists some positive area outside. That is, it is possible that the target is on the boundary of the negative information polygon. Thus when velocity uncertainty increases, position uncertainty cannot decrease. Consequently, the ocean area included in the predicted negative information polygon cannot increase; to do so would indicate a decrease in position uncertainty. The approaches considered, therefore, treat negative information polygons the same.

1. Approaches Considered for Land Interaction

a. Approaches Considered for Position and Land Interaction

Approach 1. One approach to the interaction of a predicted position polygon with land masses is to eliminate all portions of the polygon overlapping land. (It is clear that so long as a portion of the corresponding velocity polygon is outside of all land masses, some portion of the predicted position polygon will be outside of all land masses.) Thus a predicted position polygon that overlapped a land mass would be recomputed to be that portion of the position polygon outside of the land. This approach makes no effort to try to account for velocity uncertainties introduced by the proximity of land. It seems reasonable to treat positive and negative information position polygons

alike if this approach is used. Thus, this approach is simple to implement and comparatively economical of computer time.

Approach 2. Another approach is to treat positive and negative information position polygons differently. A negative information predicted position polygon overlapping land is recomputed to be that portion of the predicted position outside land. A positive information predicted position polygon overlapping land would be recomputed to be a polygon overlapping no land, but close in area to the initial predicted position polygon, and similar in appearance to those portions of the initial predicted position polygon outside of land. One way to maintain similarity of appearance between the initial predicted position polygon and the recomputed polygon is to compute, for each vertex of the initial polygon, the equation of the line through the vertex and centroid of the polygon. Then choose as vertices of the new polygon one point from each of the lines determined above that is a distance δ from the corresponding vertex. Thus the approach determines an all-ocean polygon close in area to the initial predicted positive information polygon. This approach attempts to account for velocity uncertainties introduced by the proximity of land by increasing the ocean area included in the predicted positive information polygon, and not increasing the ocean area included in the predicted negative information polygon. This approach is conceptually simple. Compared to the first approach described, it is harder to implement and has more demanding computational requirements.

Approach 3. The third approach considered also treats positive and negative information areas differently. Negative information predicted position polygons are treated just as in the previous approaches. Positive information predicted position polygons overlapping land are recomputed to be a polygon overlapping no land and close in area to the initial predicted position polygon. The requirement of similarity in appearance is relaxed in this approach, however. The idea would be to use heuristic devices, controlled by the terminal operator, to determine the form of the recomputed polygon. The operator, in this

approach, would control the final shape of the recomputed polygon. That is in contrast to the previously described approaches, where the shape of the recomputed polygon is completely determined by the initial predicted position polygon. The heuristics required by this third approach could be difficult to implement and extreme in computational requirements.

Many other approaches can be conceptualized, but those described above are perhaps the simplest to implement. It is interesting to note that the ability to define and redefine target distributions, subject only to space available, means that the terminal operator by undertaking the computational burden (for instance, estimation of the predicted polygon area) can approximate any of these approaches, and furthermore that no land interaction software is then required. The purpose of the land interaction algorithm is to free the terminal operator from this computational burden.

Approach 2 was selected for implementation because it attempts to account for velocity uncertainties introduced by proximity to land, yet is not difficult to conceptualize or implement.

b. Approaches Considered for Velocity and Land Interaction

Approaches similar to those considered for position and land interaction can be used for velocity and land interaction. Two other approaches were also considered. Those two are described below.

Approach 4. One approach to the interaction of a predicted velocity polygon with land masses is to create two copies of any predicted velocity polygon that overlaps a land mass. These copies are chosen so that they overlap as little as possible with land masses, but they are the same distance as the predicted velocity polygon from the centroid, S_1, of the predicted position distribution to which the predicted velocity polygon is associated. One copy is placed on each side of the line joining the centroid of the predicted velocity polygon to S_1.

Approach 5. Approach 5 is very similar to the preceding one. But now the two copies are chosen so that the centroid of each copy is not within a land mass. This approach is easy to implement and is not computationally burdensome. Approach 5 was selected for implementation.

2. Program Logic

a. Program Logic for Position and Land Interaction

Positive Information. An algorithm was needed to recompute the predicted position polygon when the predicted positive information position polygon generated by the prediction algorithm intersects a land mass. The recomputed polygon needed to possess the properties of intersecting no land mass and being similar in appearance and close in area to the predicted position polygon provided by the prediction algorithm. Two iterative methods of recomputing predicted position polygons were developed. These methods differ only in the final step. They are discussed below. The methods determine a scale factor, α, based on the method of Golden Section.[5] Then the methods compute a polygon, Q, with vertices q_i, from α and the vertices p_i and centroid $\bar{p}$ of the predicted position polygon provided by the prediction algorithm. The equations of the vertices of Q are:

$$q_i = \alpha p_i + (1 - \alpha)\bar{p} \quad . \tag{59}$$

At the next step, both methods compute a set of remainder polygons from Q and the land masses. The remainder polygons are the parts of Q that are outside all the land masses.

Method 1 of recomputing the predicted position polygons computes the total area contained in the set of remainder polygons. When the operator considers this area close enough to the area of the predicted position polygon provided by the prediction algorithm, Method 1 selects that set of remainder polygons as the recomputed predicted position polygons. Otherwise, another scale factor is selected according to the method of Golden Section, and the iterations continue. Method 2 of

generating the recomputed predicted position polygon computes the area of the largest polygon in the set of remainder polygons. When the display operator considers this area close enough to the area of the predicted position polygon provided by the prediction algorithm, the method selects the largest remainder polygon as the recomputed predicted position polygon. Otherwise, another scale factor is selected according to the method of Golden Section and the iterations continue.

Method 1 has the feature that multiple recomputed predicted position polygons are frequently generated from one predicted position polygon. Furthermore, these multiple polygons cwn be widely separated from each other--for example, on opposite sides of a land mass. Thus, the multiple polygons that result in certain cases are difficult to interpret, and this proliferation of polygons creates an unnecessary burden on the software.

Method 2 avoids this feature because only one recomputed predicted position polygon is generated. Method 2 was the method adopted.

Negative Information. An algorithm was needed to recompute the predicted position polygon when the predicted negative information position polygon generated by the prediction algorithm intersects a land mass. The recomputed polygon generated by the method was to be that portion of the predicted position polygon outside land. The method developed in the previous section is used to recompute the polygon by setting $\alpha = 1$.

b. Program Logic for Velocity and Land Interaction

A method was developed for recomputing predicted velocity polygons. The method checks each predicted velocity polygon to determine if its centroid is contained in any land mass. When a centroid is in a land mass, one copy of the predicted velocity polygon is placed on each side of the line joining the centroid of the predicted velocity polygon to the centroid of the predicted position distribution with which it is associated, such that the centroid of each copy of not contained in any land mass.

Each copy has associated with it a proportion of the weight assigned the predicted velocity polygon generated by the prediction algorithm. The proportion is specified by the operator.

An iterative algorithm is employed to first compute a centroid of each copy outside all land masses. This algorithm is described below. Consider the circle with center at the centroid, S_1, of the predicted position distribution and radius given by the distance between S_1 and the centroid of the predicted velocity polygon. When the centroid of the predicted velocity polygon, (x_o,y_o), is in a land mass, the algorithm computes the point on the circle 10 degrees clockwise of (x_o,y_o). This point is checked to determine if it is contained in the same land mass as (x_o,y_o). If it is, the algorithm computes the point on the circle 20 degrees clockwise of (x_o,y_o) and this point is checked. The algorithm continues to compute points clockwise from (x_o,y_o) and 10 degrees apart until a point, (x_1,y_1), is calculated that is not in the same land mass as (x_o,y_o). Then, (x_1,y_1) is checked to determine if it is contained in any other land mass. When it is not, (x_1,y_1) is the centroid of the clockwise copy of the predicted velocity polygon.

When (x_1,y_1) is contained in another land mass, the point, (x,y), computed by the algorithm immediately preceding (x_1,y_1) is in the same land mass as (x_o,y_o); therefore the algorithm computes δ, the angle separating (x,y) and (x_1,y_1) divided by 10 and computes the point, (x_2,y_2), on the circle δ degrees clockwise of (x,y). The algorithm checks (x_2,y_2) for containment in the same land mass as (x_o,y_o), and so long as that is true the algorithm continues to compute points clockwise from (x,y) and δ degrees apart until a point is calculated that is not in the same land mass as (x_o,y_o). This point is then treated as (x_1,y_1) (see above) and the iterations continue.

The centroid of the counterclockwise copy of the predicted velocity polygon is computed in a similar way.

3. Examples of Land Interaction

Figure 18 illustrates the evolution of a target distribution over three time steps around a land mass. The position polygon at Step 1 is the solid triangle, while velocity is given by the dotted triangle. At Step 2 the position uncertainty area has evolved to the hexagon. The velocity distribution has evolved to two triangles in response to its proximity to the land mass. At Step 3, the two polygons in the velocity distribution at Step 2 cause two position uncertainty areas to evolve around the land mass.

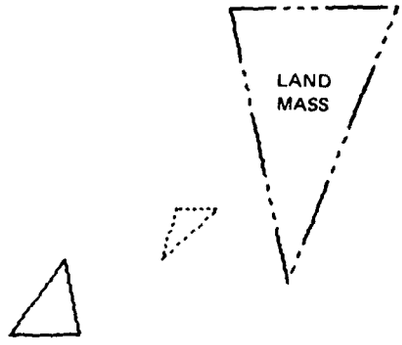
Figure 19 illustrates the evolution of another target distribution near a choke point. At Step 1, the nested quadrilaterals are the position distribution, consisting of positive and negative information areas. The nested triangles are the velocity distribution, consisting of positive and negative information areas. At Step 2, the position distribution has evolved to the solid polygons. It still has a negative information area. The velocity distribution has evolved to four triangles.

4. Program Code

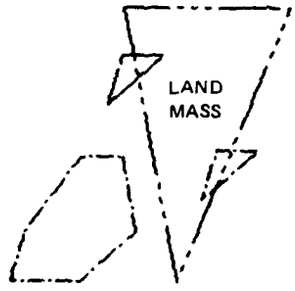
a. Position and Land Interaction

Subroutine CONCLDE calls the land interaction algorithm and stores the recomputed predicted position polygon in distribution (1,M1,N). The predicted position polygon computed by the prediction algorithm has IC1 distinct vertices; its x and y coordinates are stored respectively in arrays XPC and YPC. Subroutine CONCLDE first checks the value of LAND. Only when land masses are defined is LAND = TRUE.

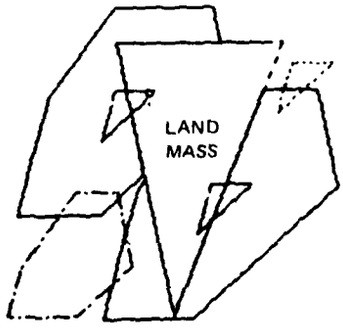
When no land is defined, control passes to label 900, where the redundant vertex is added to the predicted position polygon. Subroutine VEC is called to draw the polygon on the display. Subroutine PUT is called to store the polygon as polygon JJ in distribution (1,M1,N) by updating the arrays IA, IB, JA, JB, X, and Y, and the variables J, ILA, and JLA. This polygon then is the J-th polygon stored in X and Y. The weight of this polygon is set equal to the product of the weight of



(a) TIME STEP 1

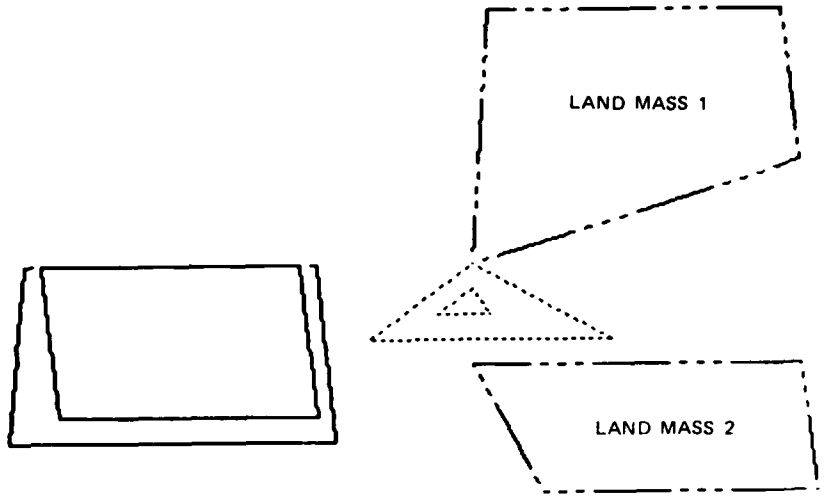


(b) TIME STEP 2

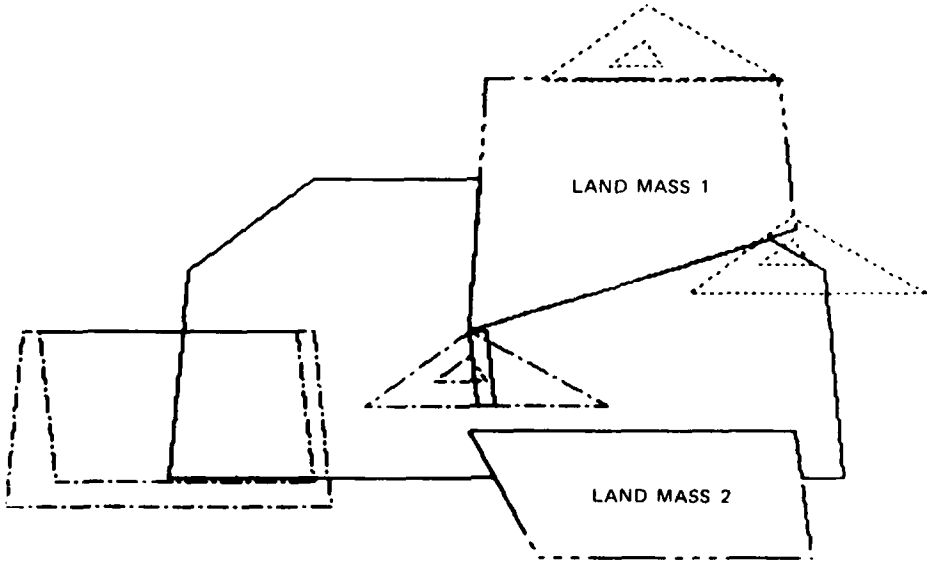


(c) TIME STEP 3

FIGURE 18 EXAMPLE OF TARGET DISTRIBUTION EVOLUTION AROUND A LAND MASS



(a) TIME STEP 1



(b) TIME STEP 2

FIGURE 19 EXAMPLE OF TARGET DISTRIBUTION EVOLUTION NEAR A CHOKE POINT

the generating position polygon and the weight of the generating velocity polygon, P(J) = PP x PC. Then control returns to subroutine MOVMENT.

When land is defined, subroutine CONCLDE calls subroutine POSINT to determine whether the predicted position polygon intersects any land mass. POSINT calls subroutine RECTAN to compute the rectangle circumscribing the predicted position polygon. The input parameters of RECTAN are:

I1	The index of the first vertex of the polygon
I2	The index of the last distinct vertex of the polygon
X	The array in which the x coordinates of the polygon are stored
Y	The array in which the y coordinates of the polygon are stored.

The output parameters of RECTAN are:

N	The value of the largest y coordinate of the polygon
E	The value of the largest x coordinate of the polygon
S	The value of the smallest y coordinate of the polygon
W	The value of the smallest x coordinate of the polygon.

Thus N,E,S, and W determine a circumscribing rectangle with vertices (N,E), (S,E), (S,W), and (N,W).

Next, subroutine POSINT initializes the variables, FLAG1 and LANINT so that they are TRUE.

The DO-1000 loop is over the number, LAN, of defined land masses. At the J-th iteration of the DO-1000 loop, the rectangle circumscribing the predicted position is tested for overlap with the rectangle circumscribing land mass J. This is accomplished by evaluating function MINIMAX. MINIMAX = TRUE only when there is no overlap between the rectangle circumscribing polygon A and the rectangle circumscribing polygon B. The input parameters to MINIMAX are:

N	The value of the largest y coordinate in polygon A
E	The value of the largest x coordinate in polygon A
S	The value of the smallest y coordinate in polygon A

W	The value of the smallest x coordinate in polygon A
LN	The value of the largest y coordinate in polygon B
LE	The value of the largest x coordinate in polygon B
LS	The value of the smallest y coordinate in polygon B
LW	The value of the smallest x coordinate in polygon B.

When MINIMAX = TRUE, the predicted position polygon does not intersect land mass J, and no further operation in the DO-1000 loop is executed. When MINIMAX = FALSE, an intersection with land mass J is possible; thus, the next operation of the DO-1000 loop is to test the predicted position polygon for a nonempty intersection with land mass J. In order to make the test, the DO-100 loop copies the distinct vertices of land mass J from arrays LAX and LAY into XW and YW, respectively, so that IUCALC may be called. The intersection is computed by a call to IUCALC. When control returns to POSINT, JMAX is evaluated. JMAX = 0 indicates that the intersection is empty, whence no further operation in the DO-1000 loop is executed.

JMAX > 0 indicates that the intersection is nonempty. In this event the weights of the generating position and velocity polygon are checked to determine if the predicted position polygon is a positive or negative information area. In the case of a negative information polygon, subroutine REMAIN is called. In the case of a positive information polygon subroutine GOLDSEC is called.

Given a polygon, Q, stored in arrays RX, RY, and IR as if output by IUCALC, subroutine REMAIN computes those portions of Q outside all land masses. Polygon Q is computed in REMAIN from an input polygon contained in arrays XPC and YPC; from the centroid, (XCEN, YCEN), of the input polygon; and from a scale factor, ALPHA.

The DO-1000 loop computes Q using Eq. (59). The DO-4000 loop is over the number of land masses. At iteration IL of the DO-4000 loop, the number, NO, of polygons currently stored in RX, RY, IR is determined. The DO-3000 loop is over NO.

At iteration IP of the DO-3000 loop, the IP-th polygon in RX, RY, IR is tested for intersection with land mass IL. RECTAN is

called to compute the rectangle circumscribing polygon IP. Function MINIMAX is applied to the rectangles circumscribing polygon IP and land mass IL. At this point, if the polygon does not intersect the land mass, the polygon is copied into arrays XWO and YWO by subroutine INTO and no further operation in the DO-3000 loop is executed. If a nonempty intersection between the polygon and land mass is still possible, the negative intersection of the land mass with the polygon is determined by a call to IUCALC. If no error is detected (in fact, if JMAX > 0), the last operation of the DO-3000 loop copies the result into arrays XWO and YWO by calling subroutine INTO.

Next, the DO-4000 loop copies the polygons computed by the DO-3000 loop from XWO and YWO to RX and RY, respectively. The last operation of the DO-4000 loop transfers the index of the start of each polygon and the number of vertices in each polygon from NUM to IR.

Thus the portions of Q outside all land masses are determined. These are called the remainder polygons, and the x and y coordinates of the IP-th remainder polygon are stored, respectively, in

XWO(NUM(1,REM(1,LL)+IP)+I) and YWO(NUM(1,REM(1,LL)+IP)+I),
$1 \le I \le$ NUM(2,REM(1,LL)+IP).

The last operation in REMAIN computes the area of the largest polygon in the remainder polygons. As currently implemented, the subroutine does not incorporate the negative contribution of holes in calculating the areas of the remainder polygons.

Subroutine INTO has as input parameters arrays XS, YS, and IS (which are assumed to contain a set of polygons as if output by IUCALC); and JMIN, JMAX, LL, MAXPOL, IP, and IL. Given these input parameters, the DO-3000 loop of INTO copies polygon JO, JMIN $\le$ JO $\le$ JMAX, from XS and YS into XWO and YWO, respectively. The x and y coordinates of polygon JO are stored, respectively, in XWO(NUM(1,KO)+I) and YWO(NUM(1,KO)+I), where

$1 \le I \le$ NUM(2,KO), REM(1,LL)+MAXPOL + 1 $\le$ KO $\le$ REM(1,LL) + MAXPOL + JMAX-JMIN+1, and NUM(2,KO) = IS(2,JO).

INTO then updates NUM, REM, and MAXPOL.

Given an input polygon, P, contained in arrays XPC and YPC, subroutine GOLDSEC uses the method of Golden Section to generate a polygon Q according to Eq. (59) such that the largest remainder polygon in Q, q*, has area close to that of the input polygon. Thus the method of Golden Section determines an optimal scale factor. The subroutine first computes the area, A, and the centroid, (XCEN,YCEN), of the input polygon. The stopping criterion, ACLOS, is also computed. Subroutine GOLDSEC returns control to POSINT whenever the area of q* is within ACLOS of A. In order to apply the method of Golden Section, upper and lower bounds on the optimal scale factor must be determined; thus GOLDSEC next determines upper and lower bounds for the optimal scale factor. The lower bound for the optimal scale factor is always one. GOLDSEC determines the area of q* when the scale factor generating Q is 1. This is accomplished by the first call to subroutine REMAIN. If the stopping criterion remains unsatisfied, the DO-1000 loop is executed. The DO-1000 loop determines an upper bound for the optimal scale factor. At each iteration, the DO-1000 loop computes a larger scale factor, NEXT; determines the area of the corresponding q* by calling REMAIN; and compares the area to A. The iterations continue until the stopping criterion is satisfied; or the area exceeds A, in which case NEXT is the upper bound for the optimal scale factor and control is transferred to label 1100; or the upper limit, KAY, of the DO-1000 counter is exceeded. In the event KAY is exceeded, the subroutine gives the operator the option of returning to label 100 of the main program or continuing the calculation of the upper bound.

Label 1100 marks the beginning of the Golden Section iterations. The upper and lower bounds of the optimal scale factor are, respectively, NEXT and SCALE. The DO-3000 loop determines a new interval containing the optimal scale factor, computes a new scale factor, determines the area of the q* corresponding to this new scale factor, and compares this area to A. The iterations continue until the stopping criterion is satisfied or the upper limit, KAY, of the DO-3000 counter is exceeded. In the event KAY is exceeded, the subroutine gives the operator the option of accepting one of the last two scale factors

generated, or returning to label 100 of the main program, or continuing the Golden Section iterations.

b. Velocity and Land Interaction

Subroutine MOVECEN calls the land interaction algorithm and stores the recomputed velocity distribution in distribution (2,M1,N). The DO-4000 loop is over the number of polygons in the velocity distribution. At iteration J1 of the DO-4000 loop when land is defined, the centroid, (XO,YO), of predicted polygon J1 is computed. Then the DO-3000 loop is executed. The DO-3000 loop is over the number of land masses. At iteration IL of the DO-3000 loop, (XO,YO) is checked for containment within land IL. This check is accomplished by the following operations. First a call to function MINIMAX determines whether (XO,YO) is contained in the rectangle circumscribing land IL. When (XO,YO) is not contained in the rectangle, no further operation of the DO-3000 loop is executed. When (XO,YO) is in the rectangle, containment in land IL is possible. Consequently, function ENSYD2 is called to determine if (XO,YO) is in land IL.

If (XO,YO) is not in land IL, no further operation of the DO-3000 loop is executed. Otherwise, two copies of polygon J1 are generated such that their centroids are not in any land mass. First, the operator associates to each copy a proportion of the weight assigned to polygon J1. [CPROB(1) and CPROB(2) are the proportions associated with, respectively, the counterclockwise and clockwise copies.] Next, subroutine NEWCEN is called to compute a centroid of each copy outside all land masses. (VX(1),VY(1)) and (VX(2),VY(2)) are the centroids, respectively, of the counterclockwise and clockwise copies.

Subroutine NEWCEN implements the algorithm for computing centroids outside of land masses described in the previous section. The DO-3000 loop is executed twice, once for each copy. The variable TIME counts the absolute number of candidates generated for a particular copy. When TIME exceeds COUNT, the display operator has the option of returning control to label 100 of the main program or continuing the iterations.

The DO-2000 loop iteratively computes a candidate for the centroid (X1,Y1), and then checks it for containment in any land mass. The check is primarily accomplished by the DO-1000 loop. This loop also recomputes the variables, D1 and D2, controlling the angle between successive centroid candidates.

When control returns to MOVECEN, the DO-1000 loop is executed. At iteration K, the DO-1000 loop computes a copy of polygon J1 with its centroid at (VX(K),VY(K)). After the copy is computed, VEC is called to draw it on the display; PUT is called to store it as polygon JJ of distribution (2,M1,N); and its weight is computed, P(J) = P(J1) · CPROB(K). After the DO-4000 loop finishes execution, subroutine UPDATE is called to compute distribution statistics.

IV POLYGON RESEARCH COMPUTER PROGRAM

A. General

This section discusses the interactive capability of the computer program used to test the algorithms that we developed. Appendix A lists the FORTRAN source code for Program POL. This program controls the graphics terminal and calls algorithm subroutines. All of the subroutines that we developed are in Appendix A. Figure 20 shows the structure of the polygon research program. Appendix B lists the FORTRAN source code for IUCALC, which was obtained from Oak Ridge National Laboratory and modified for the CDC 6400.

B. Interactive Capability

The computer program is designed to be used in an interactive mode on the Tektronix 4025 graphics terminal. When the binary file is executed, the computer initializes variables, teaches the terminal certain function keys, and draws a border around the graphics area.

First, the computer allows the operator to define land masses. The following message is printed below the graphics area:

DEFINE LAND (Y OR N)

The computer waits for the operator to type Y (for yes) or N (for no). When the operator types Y, the computer responds with the message:

USE LAND FILE? (Y OR N)

If the operator types Y, the computer assumes that a properly formatted file named TAPE9 is available to be read with an unformatted READ statement. If the user types N, or, after reading TAPE9 the computer prints the message:

DEFINE LAND FROM TERMINAL? (Y OR N)

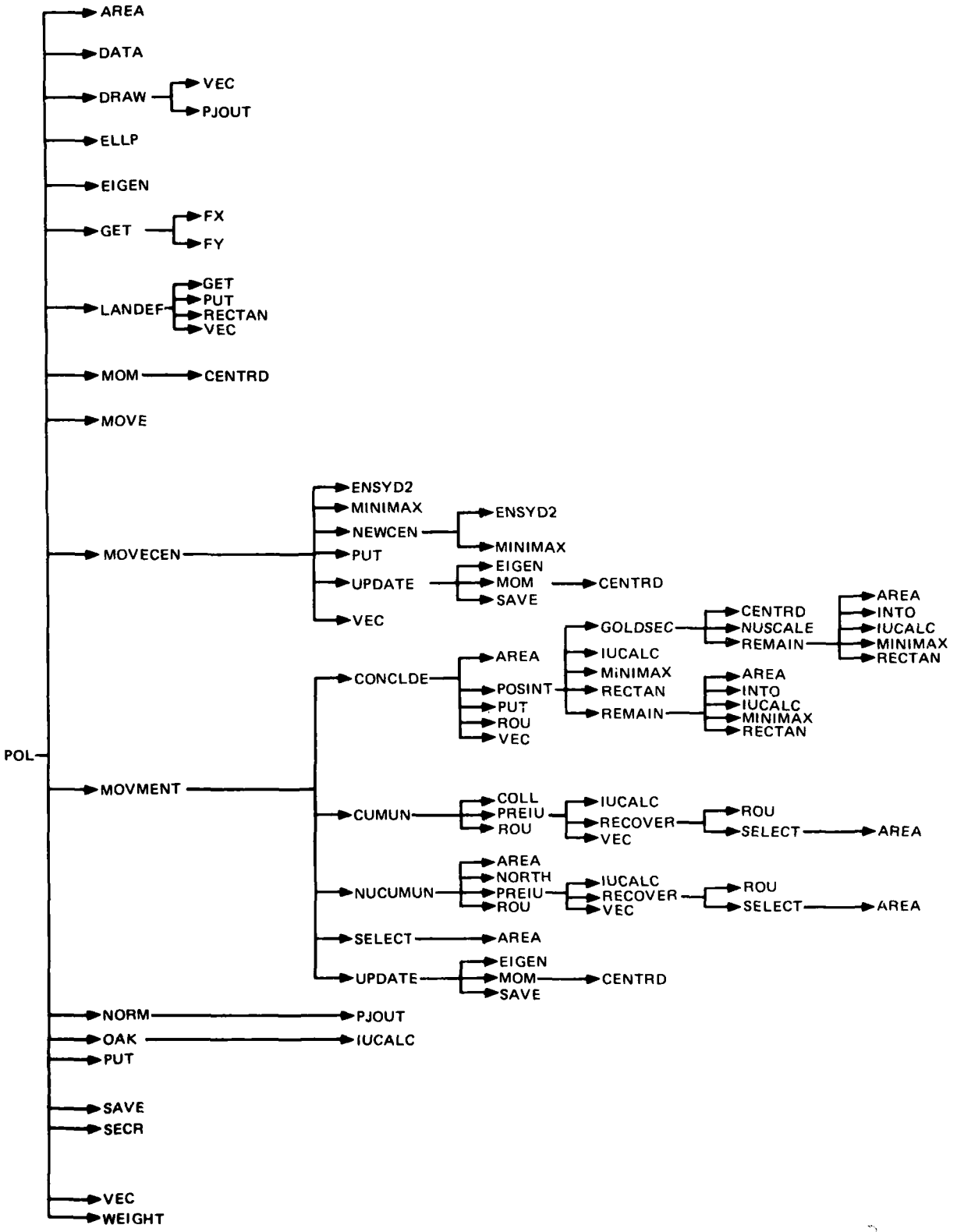
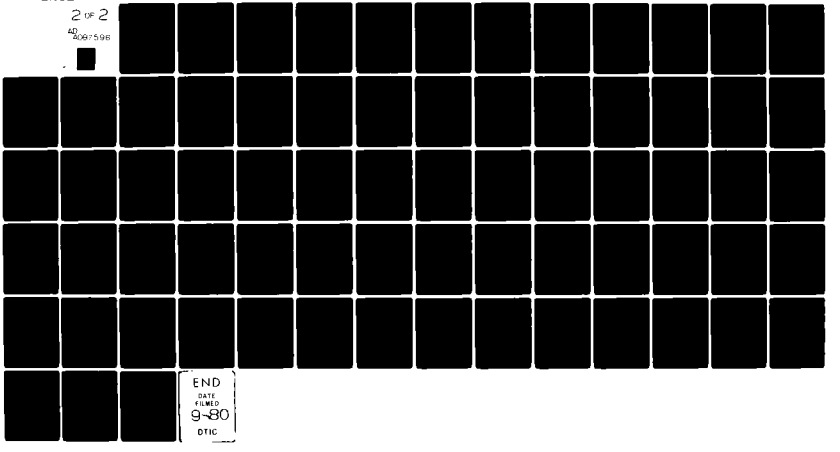


FIGURE 20 STRUCTURE OF POLYGON RESEARCH PROGRAM

AD-A087 596 SRI INTERNATIONAL MENLO PARK CA F/G 17/7
POLYGON REPRESENTATION OF TARGET LOCATION UNCERTAINTY FOR OCEAN--ETC(U)
MAY 80 L C GOHEEN, J R OLMSTEAD N00014-79-C-0329
UNCLASSIFIED NL

2 OF 2
AD A087596



END
DATE FILMED
9-80
DTIC

If the operator types Y, the computer permits the vertex-by-vertex input of as many land masses as space permits. After inputting a land mass from the terminal, the computer queries:

MORE LAND? (Y OR N)

The operator may input another land mass if Y is typed. If the operator types N, the computer will respond with:

SAVE LAND ON UNIT 10? (Y OR N)

Typing Y will save the land masses currently displayed on the terminal on a local file named TAPE10. (Unless saved on a permanent file, TAPE10 ceases to exist at the end of the terminal session.) When the computer prints:

HOW CLOSE SHOULD BE THE DIFFERENCE IN AREAS (IN PERCENT OF INPUT AREA) F10.5

The operator should respond with the percent of allowed discrepancy between the area of the predicted position polygon, as computed by the prediction algorithm, and the area of the predicted position polygon after land interaction. The quantity is stored in CLOSE. The percent should be expressed in F10.5 format.

Then the computer prints the following message below the graphics area:

FUNCTION: D E R L M C I U N P S V T

and waits for the operator to respond with one of the above code letters (followed by three (or fewer) integers (for example, D6, I127, or V). The code letters stand for functions that are available to the operator:

- D Define a multipolygonal probability distribution.
- E Draw the error ellipse associated with a distribution.
- R Redraw a distribution.
- L Label a distribution.
- M Translate and rotate a distribution.
- C Change the polygon weights of a distribution.
- I Perform an intersection fusion of two distributions.

U	Perform a union of two distributions.
N	Perform a negative intersection fusion of two distributions.
P	Calculate the conditional probability of one distribution, given another distribution.
S	Display the amount of space left in the distribution arrays.
V	Change to velocity space.
T	Set the time step or calculate the predicted target distribution.

In addition to the capability implied by the above letter functions, the terminal is taught certain function keys:

PT	Send the position of the cursor to the computer and print an asterisk at the point.
]	Designate the last point of a polygon (type before using PT).
RUB OUT	Erase the entire graphics area, including the border.
/	Redraw the border.
Shift ERASE	Draw erase vectors.
!	Draw solid line vectors.
@	Draw dashed line vectors.

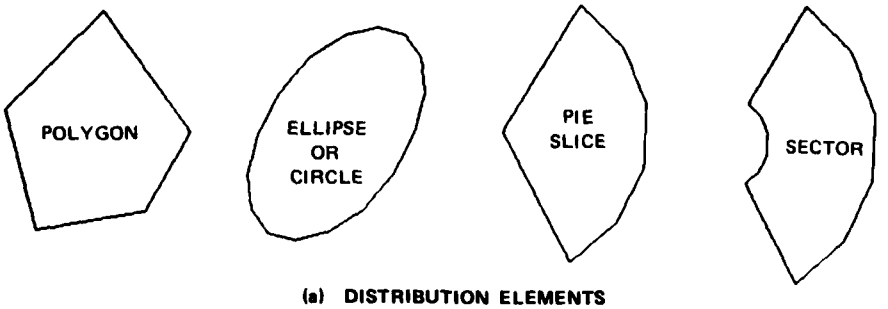
The remainder of this section discusses the use of the letter-functions.

1. Defining a Distribution

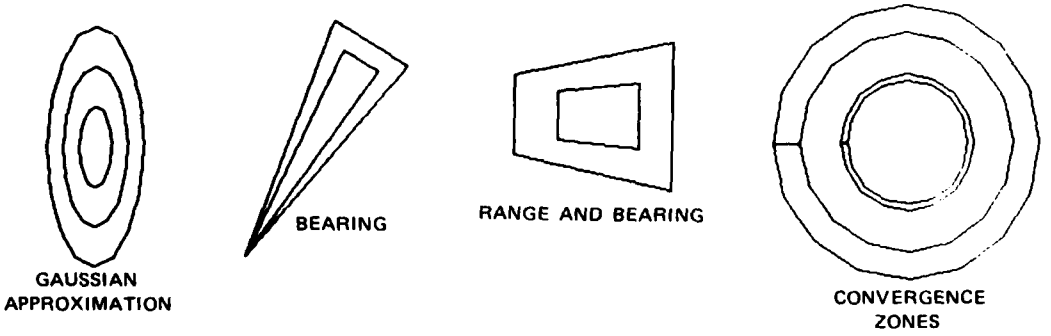
A multipolygonal probability distribution is defined by typing D and the number, N, of the target distribution (N = 1, 9); for example, D6 will define target distribution number 6. The computer then returns the following message:

PO EL PS SE GA BO RB CZ

and the operator picks one of the above letter combinations and types it on the terminal. The letter combinations stand for various kinds of polygons as shown in Figure 21 and described below:



(a) DISTRIBUTION ELEMENTS



(b) PROGRAMMED DISTRIBUTIONS

FIGURE 21 DEFINING DISTRIBUTIONS

PO Arbitrary polygon that has each vertex designated by the operator. The vertices may be entered clockwise or counterclockwise.

EL Elliptical polygon of 16 sides. The center is designated first, then the end of the major axis, and finally the end of the minor axis (the direction of the minor axis is not important, only the magnitude is used).

PS Pie slice polygon. The center point and the end of the radius at the midpoint of the angle are designated. Then the value of the angle is typed in.

SE Sector polygon. The center point, the end of the outer radius, and the end of the inner radius are designated, and the angle of the sector typed in.

GA Gaussian approximation distribution. The 2-sigma ellipse is designated as in EL above. The distribution is a weighted combination of the 1,2,3-sigma elliptical polygons.

BO Bearing-only distribution. The origin and the maximum likely range (in the direction of the bearing line) is designated, then the bearing sigma is typed in. The

computer draws the 1- and 2-sigma polygons. The 2-sigma polygon is drawn with the designated range plus 10 percent. The polygons are weighted to simulate normally distributed bearing.

RB Range and bearing distribution. The origin and the range/bearing point are designated, then the range and bearing sigmas are typed in. The two polygons are weighted to represent the 1- and 2-sigma contours of normal distributions in range and bearing.

CZ Convergence zone distribution. The center and the outer radius of the first zone is designated. The operator supplies the weights for the two zones.

After designating a polygon (PO EL PS SE), the computer asks the operator to type in a weight for the polygon. If a height is desired instead, then the operator types in 0 and the computer will ask for the height of the polygon. Both height and weight are relative measures and are normalized by the computer.

Upon receiving the polygon weight the computer returns the message:

MORE POLYGONS? (Y OR N)

If the operator wishes more polygons in the multipolygonal distribution, then he types in Y for yes. Both the polygonal elements (PO EL PS SE) and the pre-programmed distributions (GA BO RB CZ) can be used to build complex distributions. Upon typing N for no, the computer normalizes the weights and calculates and saves the statistics of the distribution. When finished with these computations, the computer prints out the normalized weight of each polygon in the distribution, prints the previously discussed FUNCTION statement, and waits for a new letter code. The distribution is always a position distribution (L = 1) unless the operator puts the program in the velocity mode (L = 2) just prior to using the D-function.

2. Drawing an Error Ellipse

The 2-sigma error ellipse (a 16-sided polygon) of a multipolygonal distribution is drawn by typing E and the number of the distribution (for example, E6). The distribution must have been previously

defined. The ellipse is centered on the mean of the distribution. If the operator wishes the ellipse to be drawn with a dashed line, he pushes the @ key before typing E6.

3. Redrawing a Distribution

A previously defined distribution may be drawn again by typing R and the number of the distribution (for example, R6). This function is useful when the display is cleared by pushing RUB OUT so that distributions can be redrawn as desired.

4. Labeling a Distribution

The order of the polygons in a distribution can be labeled by typing L and the number of the distribution (for example, L6). The integer labels (1 for Polygon 1, etc.) are placed on the first vertex of each polygon.

5. Moving a Distribution

A distribution may be translated and/or rotated by typing M and the number of the distribution (for example, M6). The computer then puts the cursor in the graphics area and two points are designated by the operator. The first point is the point around which the rotation takes place; it is also the start of the translation vector. The second point is the end of the translation vector. The computer then asks for the angle of rotation. The distribution is rotated around the first point designated, then it is rigidly translated according to the vector. Finally, the distribution statistics are recalculated. Although the old points of the distribution are still in the XY-arrays, there is no mechanism to retrieve them; only the newly moved distribution is available to the operator.

6. Changing Polygon Weights

The weights of polygons can be changed by typing C and the number of the distribution (for example, C6). The computer asks for

the weight (or height) of each polygon, and when finished, normalizes the weights and calculates the statistics.

7. Fusing Two Distributions

The fusion of two distributions into a third distribution is performed by typing I or N and the numbers of the distributions being fused and the number of the resultant distribution. For example, the intersection of Distribution 4 with Distribution 6 (the result saved as Distribution 7) is denoted: I467. The negative intersection, $\overline{4} * 6$, is denoted: N467. If the output distribution is not designated, then the result is put in Distribution 9.

Usually the display is cleared before typing in the fusion function because if the input distributions are still on the CRT, then the output polygons cannot be seen because they lie on top of the input polygons. The program computes the vertices of the output polygons, their weights, and the distribution statistics.

8. Conditional Probability

The conditional probability of one distribution, given another distribution, is calculated by typing C and the numbers of the two distributions. For example, the conditional probability of Distribution 6, given Distribution 4, is denoted: P64. The output is printed below the graphics area and looks like the following:

PROB OF 6 GIVEN 4 = .27 .

The algorithm really computes a number that is proportional to the conditional probability; only when the first distribution is a single polygon does the algorithm return a number that is always between zero and one.

9. Array Status

The memory status of the arrays can be checked at any time by typing in S. The computer returns a message that gives the amount of space left in the various arrays; for example:

MEMORY LEFT:

POINTS ... 1370

POLYGONS ... 195

STATISTICS ... 864

"Points" refers to the X and Y arrays; "Polygons" refers to the P, H, XCEN, YCEN, IA, IB arrays; and "Statistics" refers to the SE array.

10. Velocity Space

A velocity distribution can be defined by typing V, and, after the computer responds with the FUNCTION statement, typing D and the number of the distribution for which velocity is defined. The velocity distribution is defined the same as a position distribution. The polygons that are drawn represent the uncertainty of the centroid of the position distribution one time step into the future. The V-function can also precede other functions. For example, to get an error ellipse of the velocity distribution associated with Distribution 6, type V then type E6. The program is automatically returned to position space after a function is performed.

11. Changing Time

The time index is changed for all distributions by typing T0 and the time step desired. For example, T03 changes the time to Step 3. The time step remains the same until changed; time is initialized to Step 1.

12. Prediction

The predicted position and velocity of a target distribution, N, at time step, M, are calculated by typing TNM, (N = 1, 9; M = 1, 4). Distribution N must be defined for time step M. For example if distribution 6 has been defined at time step 2, T62 calculates the predicted position and velocity of target 6 at the next time step, 3.

The screen is erased and the specified input distribution is redrawn. When the predicted position polygon will be a positive information area, the computer asks:

CALL NUCUMUN (Y OR N).

A response of Y results in the use of the quick algorithm for generating positive information predicted position polygons and is usually given when the input position polygon is convex. A response of N results in the use of the exact algorithm for generating predicted position polygons. Upon calculation, the predicted target distribution is drawn and its statistics are saved. The computer prints the normalized weights for each polygon in the distribution.

REFERENCES

1. R. G. Edwards and P. R. Coleman, "IUCALC-A FORTRAN Subroutine for Calculating Polygon-Line Intersections, and Polygon-Polygon Intersections, Unions, and Relative Differences," ORNL/CSD/TM-12, Geographic Data Systems Group, Computing Applications Department, Computer Sciences Division, Oak Ridge National Laboratory, Oak Ridge, Tennessee (August 1976).

2. H. L. Wiener, W. W. Willman, and I. R. Goodman, "Naval Ocean-Surveillance Correlation Handbook, 1978," NRL Report 8340, Naval Research Laboratory (October 1979), UNCLASSIFIED.

3. C. E. Pearson, Ed., Handbook of Applied Mathematics, p. 72 (Van Nostrand Reinhold Company, New York, New York, 1974).

4. S. Nordbeck and B. Rystedt, "Computer Cartography Point-In-Polygon Programs," BIT, Vol. 7, 45-46 (1967).

5. W. I. Zangwill, Nonlinear Programming, p. 121 (Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1969).

Appendix A

PROGRAM POL SOURCE CODE

```
      PROGRAM POL(INPUT,OUTPUT,TAPE5=INPUT,TAPE6,TAPE8=OUTPUT,TAPE7,
     $     TAPE9,TAPE10)
      LOGICAL FLAG,LAND,INPUT(2,5,9)
      DIMENSION XCEN(200),YCEN(200),IA(200),IB(200),JA(2,5,9),JB(2,5,9),
     $          X(1400),Y(1400),P(200),H(200),SE(900),KE(2,5,9),
     $          LJA(1,1,1),LJB(1,1,1),
     $          LIA(200),LIB(200),
     $          XW(200),YW(200),IW(2,20),SB(17),CB(17),XS(50),YS(50)
      INTEGER COUNT
      REAL LAX(1400),LAY(1400),INC,LN(200),LE(200),LS(200),LW(200)
      DATA INPUT/90*.FALSE./
      DATA INC,COUNT/0.5,1/
      DATA PI/3.141592653/
      DATA M,N3/1,9/
      DATA ILA,JLA,KLA/0,0,0/
      WRITE(6,10)
C     ENDFILE 6
      WRITE(8,10)
   10 FORMAT(*$WOR 30 $GRA 2,29,2,79 $LIN 1*/
     $       *$VEC 0,0 0,391 623,391 623,0 0,0*/
     $       *$LEA PT "$REP 01" 13 42*/
     $       *$LEA J 93 08*/
     $       *$LEA 127 "$ERA G" 13*/
     $       *$LEA ◆ "$VEC 0 0 0 391 623 391 623 0 0 0" 13*/
     $       *$LEA 173 "$LIN E" 13*/
     $       *$LEA I "$LIN 1" 13*/
     $       *$LEA @ "$LIN 2" 13*)
      BB=2.*PI/16.
      DO 5 I=1,17
      B=BB*(I-1)
      SB(I)=SIN(B)
    5 CB(I)=COS(B)
      AN=9HANGLE
      WE=9HWEIGHT
      HE=9HHEIGHT
      SGB=9HSIG BRG
      SGR=9HSIG RNG
**
**    LAND
      LAND=.F.
      WRITE(6,60)
C     ENDFILE 6
      WRITE(8,60)
   60 FORMAT(*DEFINE LAND  (Y OR N)*)
      READ(5,14) Q
      WRITE(6,14) Q
      WRITE(7,14) Q
      WRITE(8,14) Q
      IF(Q.EQ.1HY) LAND=.T.
      IF(.NOT.LAND) GOTO 100
      CALL MESSAGC(7HPOL     ,7HLANDEF )
      CALL LANDEF(LAX,LAY,LIA,LIB,LJB,LN,LE,LS,LW,XW,YW)
      WRITE(6,28)
      WRITE(8,28)
      WRITE(6,70)
C     ENDFILE 6
      WRITE(8,70)
   70 FORMAT(*HOW CLOSE SHOULD BE THE DIFFERENCE IN AREAS (IN PERCENT*,
     $  * OF INPUT AREA)  F10.5*)
      READ(5,75) CLOSE
   75 FORMAT(F10.5)
```

```
      CLOSE=CLOSE/100.
      WRITE(6,85) INC,CLOSE
C     ENDFILE 6
      WRITE(7,85) INC,CLOSE
C     ENDFILE 7
      WRITE(8,85) INC,CLOSE
   85 FORMAT(*INC= *,G11.5,* CLOSE= *,G11.5)
      WRITE(6,96) COUNT
C     ENDFILE 6
      WRITE(7,96) COUNT
C     ENDFILE 7
      WRITE(8,96) COUNT
   96 FORMAT(*THE NUMBER OF ITERATIONS ALLOWED IS *, I3,*,*)

**FUNCTION
  100 L=1
  106 CONTINUE
      WRITE(6,12)
      WRITE(8,12)
C     ENDFILE 6
   12 FORMAT(/*FUNCTION: D E R L M C I U N P S V T*)
  101 READ(5,14) F,N,N2,N3
      WRITE(6,14) F,N,N2,N3
      WRITE(7,14) F,N,N2,N3
      WRITE(8,14) F,N,N2,N3
   14 FORMAT(A1,3I1)
      IF(F.EQ.1HD) GO TO 110
      IF(F.EQ.1HE) GO TO 190
      IF(F.EQ.1HR) GO TO 200
      IF(F.EQ.1HL) GO TO 205
      IF(F.EQ.1HM) GO TO 210
      IF(F.EQ.1HC) GO TO 220
      IF(F.EQ.1HI) GO TO 230
      IF(F.EQ.1HU) GO TO 235
      IF(F.EQ.1HN) GO TO 240
      IF(F.EQ.1HP) GO TO 250
      IF(F.EQ.1HS) GO TO 260
      IF(F.NE.1HV) GO TO 105
      L=2  $GO TO 106
  105 IF(F.NE.1HT) GO TO 100
      M=N2
      IF(N.EQ.0) GOTO 100
      GOTO 500
**
**DEFINE PROBABILITY DISTRIBUTION
  110 JJ=0
      IF(L.EQ.2) WRITE(6,18)
      IF(L.EQ.2) WRITE(8,18)
   18 FORMAT(*$LIN 2*)
  111 JJ=JJ+1
      WRITE(6,20)
C     ENDFILE 6
      WRITE(8,20)
   20 FORMAT(*PO EL PS SE GA BO RB CZ*)
  112 READ(5,22) FF
      WRITE(6,22) FF
      WRITE(7,22) FF
      WRITE(8,22) FF
   22 FORMAT(A2)
      IF(FF.EQ.2HPO) GO TO 120
      IF(FF.EQ.2HEL) GO TO 125
```

```
      IF(FF.EQ.2HPS) GO TO 130
      IF(FF.EQ.2HSE) GO TO 135
      IF(FF.EQ.2HGA) GO TO 140
      IF(FF.EQ.2HBO) GO TO 145
      IF(FF.EQ.2HRB) GO TO 150
      IF(FF.EQ.2HCZ) GO TO 155
      GO TO 112
**
**POLYGON
  120 IMAX=99
      CALL GET(IMAX,XW,YW,XO,YO,R1,R2,D1,D2)
  121 CALL VEC(1,IMAX,XW,YW)
      CALL PUT(L,M,N,JJ,IMAX,XW,YW,IA,IB,JA,JB,J,X,Y,ILA,JLA)
      P(J)=DATA(WE)
      IF(P(J)) 180,122,180
  122 HI=DATA(HE)
      CALL WEIGHT(HI,J,P,IA,IB,X,Y)
      GO TO 180
**
**ELLIPSE
  125 IMAX=3
      CALL GET(IMAX,XW,YW,XO,YO,R1,R2,D1,D2)
      CALL ELLP(XO,YO,R1,R2,D1,D2,SB,CB,XW,YW)
      IMAX=17
      GO TO 121
**
**PIE SLICE
  130 IMAX=2
      CALL GET(IMAX,XW,YW,XO,YO,R1,R2,D1,D2)
      R1=0.
  131 BSD=DATA(AN)
      CALL SECR(PI,BB,BSD,XO,YO,R1,R2,D1,D2,XW,YW,IMAX)
      IF(L.EQ.1) GO TO 121
      KO=KE(1,M,N)
      XC=SE(1+KO)
      YC=SE(2+KO)
      CALL MOVE(XO,YO,XC,YC,0.,1,IMAX,XW,YW)
      GO TO 121
**
**SECTOR
  135 IMAX=3
      CALL GET(IMAX,XW,YW,XO,YO,R1,R2,D1,D2)
      GO TO 131
**
**GAUSSIAN APPROXIMATION
  140 IMAX=3
      CALL GET(IMAX,XW,YW,XO,YO,R1,R2,D1,D2)
      S1=.5*R1
      S2=.5*R2
      DO 142 NC=1,3
      R1=S1*NC
      R2=S2*NC
      CALL ELLP(XO,YO,R1,R2,D1,D2,SB,CB,XW,YW)
      IMAX=17
      CALL VEC(1,IMAX,XW,YW)
      CALL PUT(L,M,N,JJ,IMAX,XW,YW,IA,IB,JA,JB,J,X,Y,ILA,JLA)
  142 JJ=JJ+1
      JJ=JJ-1
      PW=DATA(WE)
      P(J)=PW*.225
      P(J-1)=PW*.536
```

```
      P(J-2)=PW*.239
      GO TO 180
**
**BEARING ONLY
  145 IMAX=2
      CALL GET(IMAX,XW,YW,XO,YO,R1,R2,D1,D2)
      R1=0.
      R=R2
      BC=DATA(SGB)*2.
      DO 147 NC=1,2
      BSD=BC*NC
      R2=R+R*.1*(NC-1)
      CALL SECR(PI,BB,BSD,XO,YO,R1,R2,D1,D2,XW,YW,IMAX)
      IF(L.EQ.1) GO TO 146
      KO=KE(1,M,N)
      XC=SE(1+KO)
      YC=SE(2+KO)
      CALL MOVE(XO,YO,XC,YC,0.,1,IMAX,XW,YW)
  146 CALL VEC(1,IMAX,XW,YW)
      CALL PUT(L,M,N,JJ,IMAX,XW,YW,IA,IB,JA,JB,J,X,Y,ILA,JLA)
  147 JJ=JJ+1
      JJ=JJ-1
      PW=DATA(WE)
      P(J)=PW*.568
      P(J-1)=PW*.432
      GO TO 180
**
**RANGE AND BEARING
  150 IMAX=2
      CALL GET(IMAX,XW,YW,XO,YO,R1,R2,D1,D2)
      RC=DATA(SGR)*R2*.01
      BC=DATA(SGB)*2.
      R=R2
      DO 152 NC=1,2
      D=RC*NC
      R1=R-D    $R2=R+D
      BSD=BC*NC
      CALL SECR(PI,BB,BSD,XO,YO,R1,R2,D1,D2,XW,YW,IMAX)
      IF(L.EQ.1) GO TO 151
      KO=KE(1,M,N)
      XC=SE(1+KO)
      YC=SE(2+KO)
      CALL MOVE(XO,YO,XC,YC,0.,1,IMAX,XW,YW)
  151 CALL VEC(1,IMAX,XW,YW)
      CALL PUT(L,M,N,JJ,IMAX,XW,YW,IA,IB,JA,JB,J,X,Y,ILA,JLA)
  152 JJ=JJ+1
      JJ=JJ-1
      PW=DATA(WE)
      P(J)=PW*.651
      P(J-1)=PW*.349
      GO TO 180
**
**CONVERGENCE ZONES
  155 IMAX=2
      CALL GET(IMAX,XW,YW,XO,YO,R1,R2,D1,D2)
      DO 156 ND=1,2
      RO=R2*ND
      RI=RO*(1.-.1*ND)
      CALL SECR(PI,BB,360.,XO,YO,RI,RO,D1,D2,XW,YW,IMAX)
      CALL VEC(1,IMAX,XW,YW)
      CALL PUT(L,M,N,JJ,IMAX,XW,YW,IA,IB,JA,JB,J,X,Y,ILA,JLA)
```

```
      P(J)=DATA(WE)
      IF(P(J).GT.0.) GO TO 156
      HI=DATA(HE)
      CALL WEIGHT(HI,J,P,IA,IB,X,Y)
  156 JJ=JJ+1
      JJ=JJ-1
      GO TO 180
**
**TEST FOR FINISH AND CALCULATE STATISTICS
  180 WRITE(6,24)
C     ENDFILE 6
      WRITE(8,24)
   24 FORMAT(*MORE POLYGONS? (Y OR N)*)
      READ(5,14) Q
      WRITE(6,14) Q
      WRITE(7,14) Q
      WRITE(8,14) Q
      IF(Q.EQ.1HY) GO TO 111
  185 J1=JA(L,M,N)
      J2=JB(L,M,N)
      CALL NORM(J1,J2,P)
  186 CALL MOM(J1,J2,IA,IB,X,Y,P,H,EX,EY,EXX,EYY,EXY,XCEN,YCEN)
      IF(EXX.GT.0..AND.EYY.GT.0.) GO TO 187
      WRITE(6,25)
C     ENDFILE 6
      WRITE(8,25)
   25 FORMAT(*NEGATIVE VARIANCE*/)
      GO TO 100
  187 CALL EIGEN(EXX,EYY,EXY,R1,R2,D1,D2)
      CALL SAVE(L,M,N,EX,EY,R1,R2,D1,D2,EXX,EYY,EXY,SE,KE,KLA)
      INPUT(L,M,N)=.T.
      WRITE(6,28)
      WRITE(8,28)
C     ENDFILE 6
      WRITE(8,28)
   28 FORMAT(*$LIN 1*)
      GO TO 100
**
**DRAW ERROR ELLIPSE
  190 KO=KE(L,M,N)
      EX=SE(1+KO)
      EY=SE(2+KO)
      R1=SE(3+KO)
      R2=SE(4+KO)
      D1=SE(5+KO)
      D2=SE(6+KO)
      CALL ELLP(EX,EY,R1,R2,D1,D2,SB,CB,XW,YW)
      CALL VEC(1,17,XW,YW)
      WRITE(6,28)
      WRITE(8,28)
      GO TO 100
**
**REPEAT DISTRIBUTION
  200 CALL DRAW(L,M,N,IA,IB,JA,JB,X,Y,P)
      WRITE(6,28)
      GO TO 100
**
**LABEL POLYGONS
  205 J1=JA(L,M,N)
      J2=JB(L,M,N)
      JC=0
```

```
      DO 206 J=J1,J2
      JC=JC+1
      I1=IA(J)
      X1=X(I1)
      Y1=Y(I1)
  206 WRITE(6,30) X1,Y1,JC
      WRITE(8,30) X1,Y1,JC
   30 FORMAT(*$VEC *,2(F10.0,1X),*$STR **,I1,***)
      GO TO 100
**
**MOVE DISTRIBUTION
  210 IMAX=2
      CALL GET(IMAX,XW,YW,XO,YO,R1,R2,D1,D2)
      XC=XO+R2*D1
      YC=YO+R2*D2
      ANG=DATA(AN)
      J1=JA(L,M,N)
      J2=JB(L,M,N)
      DO 211 J=J1,J2
      I1=IA(J)
      I2=IB(J)
  211 CALL MOVE(XO,YO,XC,YC,ANG,I1,I2,X,Y)
      CALL DRAW(L,M,N,IA,IB,JA,JB,X,Y,P)
      WRITE(6,28)
      WRITE(8,28)
      GO TO 186
**
**CHANGE WEIGHTS
  220 J1=JA(L,M,N)
      J2=JB(L,M,N)
      DO 223 J=J1,J2
      JP=J-J1+1
      XS(J)=P(J)/H(J)
      WRITE(6,32) JP
      WRITE(8,32) JP
   32 FORMAT(*WEIGHT OF *,I2)
      READ(5,33) P(J)
      WRITE(6,33) P(J)
      WRITE(7,33) P(J)
      WRITE(8,33) P(J)
   33 FORMAT(F10.0)
      IF(P(J)) 223,222,223
  222 WRITE(6,38) JP
      WRITE(8,38) JP
   38 FORMAT(*HEIGHT OF *,I2)
      READ(5,33) HI
      WRITE(6,33) HI
      WRITE(7,33) HI
      WRITE(8,33) HI
      CALL WEIGHT(HI,J,P,IA,IB,X,Y)
  223 CONTINUE
      CALL NORM(J1,J2,P)
      DO 225 J=J1,J2
  225 H(J)=P(J)/XS(J)
      GO TO 186
**
**FUSION (INTERSECTION, UNION, AND NEGATIVE INTERSECTION)
  230 KL=2  $GO TO 241
  235 KL=1  $GO TO 241
  240 KL=4
  241 JP1=JA(L,M,N)
```

```
      JP2=JB(L,M,N)
      JQ1=JA(L,M,N2)
      JQ2=JB(L,M,N2)
      JJ=0
      DO 249 JP=JP1,JP2
      DO 249 JQ=JQ1,JQ2
      CALL OAK(KL,JP,JQ,IA,IB,X,Y,XW,YW,IW,JMAX)
      IF(JMAX)255,249,243
  243 HPQ=H(JP)*H(JQ)
      DO 248 JO=1,JMAX
      IO=IW(1,JO)
      IM=IW(2,JO)
      DO 246 I=1,IM
      XS(I)=XW(I+IO)
  246 YS(I)=YW(I+IO)
      IM=IM+1
      XS(IM)=XS(1)
      YS(IM)=YS(1)
      CALL VEC(1,IM,XS,YS)
      JJ=JJ+1
      CALL PUT(L,M,N3,JJ,IM,XS,YS,IA,IB,JA,JB,J,X,Y,ILA,JLA)
  248 P(J)=HPQ*AREA(XW,YW,IW,JO)
  249 CONTINUE
      N=N3
      GO TO 185
**
** CONDITIONAL PROBABILITY
  250 PROB=0.
      JP1=JA(L,M,N)
      JP2=JB(L,M,N)
      JQ1=JA(L,M,N2)
      JQ2=JB(L,M,N2)
      DO 254 JP=JP1,JP2
      DO 254 JQ=JQ1,JQ2
      CALL OAK(2,JP,JQ,IA,IB,X,Y,XW,YW,IW,JMAX)
      IF(JMAX)255,254,251
  251 HPQ=H(JP)*H(JQ)
      DO 252 JO=1,JMAX
  252 PROB=PROB+HPQ*AREA(XW,YW,IW,JO)
  254 CONTINUE
      PROB=PROB/ABS(H(JP1))
      WRITE(6,34) N,N2,PROB
      WRITE(8,34) N,N2,PROB
   34 FORMAT(*PROB OF *,I2,* GIVEN *,I2,* =*,F5.2)
      GO TO 100
  255 WRITE(6,35) JMAX
      WRITE(8,35) JMAX
   35 FORMAT(*!UCALC ERROR *,I3)
      GO TO 100
**
**STATUS
  260 ICO=1400-ILA
      JCO=200-JLA
      KCO=900-KLA
      WRITE(8,36) ICO,JCO,KCO
      WRITE(6,36) ICO,JCO,KCO
   36 FORMAT(*MEMORY LEFT:*,3X,
     $       * POINTS...*,I4,3X,
     $       * POLYGONS...*,I3,3X,
     $       * STATISTICS...*,I3)
      GO TO 100
```

```
**
**PREDICTION ALGORITHM
  500 CONTINUE
      M1=M+1
      WRITE(8,48)
      WRITE(6,48)
   48 FORMAT(*$ERA G*)
      WRITE(8,50)
      WRITE(6,50)
   50 FORMAT(*$LIN 4*)
      CALL DRAW(1,M,N,IA,IB,JA,JB,X,Y,P)
      CALL DRAW(2,M,N,IA,IB,JA,JB,X,Y,P)
      IF(.NOT.LAND) GOTO 700
      WRITE(6,55)
      WRITE(8,55)
   55 FORMAT(*$LIN 8*)
      LAN=LJB(1,1,1)
      DO 600 J=1,LAN
        I1=LIA(J)
        I2=LIB(J)
        CALL VEC(I1,I2,LAX,LAY)
  600 CONTINUE
  700 CONTINUE
      WRITE(6,28)
C     ENDFILE 6
      WRITE(8,28)
      CALL MESSAGC(7HPOL     ,7HMOVMENT)
      CALL MOVMENT(M,N,JA,JB,X,Y,IA,IB,P,XCEN,YCEN,
     $  XW,YW,J,ILA,JLA,FLAG,SE,KE,KLA,H,LAX,LAY,LIA,LIB,
     $  LAN,LAND,LN,LE,LS,LW,INC,CLOSE)
      CALL MESSAGA(7HPOL     )
      IF(.NOT.FLAG) GOTO 100
      CALL MESSAGC(7HPOL     ,7HMOVCEN )
      CALL MOVECEN(M,N,JA,JB,X,Y,IA,IB,P,SE,KE,KLA,
     $  XW,YW,J,ILA,JLA,XCEN,YCEN,H,LAX,LAY,LIA,LIB,
     $  LAN,LAND,LN,LE,LS,LW,INPUT,COUNT,FLAG)
      CALL MESSAGA(7HPOL     )
      GOTO 100
      END


      FUNCTION FX(C)
      FX=3+8*(C-2)
      RETURN
      END

      FUNCTION FY(R)
      FY=7+14*(29-R)
      RETURN
      END

      FUNCTION COL(X)
      COL=2.5+(X-3.)/8.
      RETURN
      END

      FUNCTION ROW(Y)
      ROW=29.5-(Y-7.)/14.
      RETURN
      END
```

```
      FUNCTION DATA(A)
      WRITE(6,200) A
C     ENDFILE 6
      WRITE(8,200) A
      READ(5,300) DATA
      WRITE(6,300) DATA
      WRITE(7,300) DATA
      WRITE(8,300) DATA
  200 FORMAT(A9)
  300 FORMAT(F10.0)
      RETURN
      END

      SUBROUTINE GET(IM,X,Y,X1,Y1,R1,R2,D1,D2)
      DIMENSION X(1),Y(1),R(3)
  100 FORMAT(*$JUM 15,40*/*$WOR K*)
  200 FORMAT(*POINT*)
  300 FORMAT(8X,F3.0,1X,F3.0,1X,A1)
  400 FORMAT(*$MON K*)
  500 FORMAT(13(4(*$VEC *,2F7.0,*$STR 136*)/))
      WRITE(6,100)
C     ENDFILE 6
      WRITE(8,100)
      DO 10 I=1,IM
      WRITE(6,200)
C     ENDFILE 6
      WRITE(8,200)
      READ(5,300) ROW,COL,SYM
      X(I)=FX(COL)
      Y(I)=FY(ROW)
      IF(SYM.EQ.1H]) GO TO 30
   10 CONTINUE
      DO 20 I=2,IM
      P=X(I)-X(1)
      Q=Y(I)-Y(1)
   20 R(I)=SQRT(P*P+Q*Q)
      X1=X(1)   $Y1=Y(1)
      R1=R(3)
      R2=R(2)
      D1=(X(2)-X(1))/R2
      D2=(Y(2)-Y(1))/R2
      GO TO 40
   30 IM=I+1
      X(IM)=X(1)
      Y(IM)=Y(1)
   40 WRITE(6,400)
C     ENDFILE 6
      WRITE(8,400)
      WRITE(6,500)(X(I),Y(I),I=1,IM)
C     ENDFILE 6
      WRITE(8,500) (X(I),Y(I),I=1,IM)
      RETURN
      END

      SUBROUTINE PUT(L,M,N,JJ,IM,XW,YW,IA,IB,JA,JB,J,X,Y,IL,JL)
      DIMENSION XW(1),YW(1),IA(1),IB(1),JA(2,5,9),JB(2,5,9),
     $          X(1),Y(1)
      JL=JL+1
      IF(JJ.EQ.1) JA(L,M,N)=JL
      J=JB(L,M,N)=JL
      DO 10 I=1,IM
```

```
      X(IL+I)=XW(I)
   10 Y(IL+I)=YW(I)
      IA(J)=IL+1
      IL=IL+IM
      IB(J)=IL
      RETURN
      END

      SUBROUTINE MOVE(X1,Y1,X2,Y2,ANG,I1,I2,X,Y)
      DIMENSION X(1),Y(1)
      A=ANG*3.1416/180.
      SA=SIN(A)
      CA=COS(A)
      DO 10 I=I1,I2
      XI=X(I)-X1
      YI=Y(I)-Y1
      X(I)=X2+XI*CA+YI*SA
   10 Y(I)=Y2+YI*CA-XI*SA
      RETURN
      END

      SUBROUTINE NORM(J1,J2,P)
      DIMENSION P(1)
      SUM=0.
      DO 5 J=J1,J2
    5 SUM=SUM+P(J)
      IF(SUM.LE.0.) STOP 1
      DO 20 J=J1,J2
   20 P(J)=P(J)/SUM
      CALL PJOUT(J1,J2,P)
      RETURN
      END

      SUBROUTINE PJOUT(J1,J2,P)
      DIMENSION P(1)
      JM=J2-J1+1
      WRITE(6,100) (JC,JC=1,JM)
      WRITE(7,100) (JC,JC=1,JM)
      WRITE(8,100) (JC,JC=1,JM)
      WRITE(6,200) (P(J),J=J1,J2)
      WRITE(7,200) (P(J),J=J1,J2)
      WRITE(8,200) (P(J),J=J1,J2)
  100 FORMAT(* POLYGON: *,9(I4,3X))
  200 FORMAT(*  WEIGHT: *,9(F5.2,2X))
      RETURN
      END

      SUBROUTINE ELLP(XC,YC,R1,R2,D1,D2,SB,CB,XW,YW)
      DIMENSION SB(1),CB(1),XW(1),YW(1)
      DO 10 I=1,17
      XB=R1*SB(I)
      YB=R2*CB(I)
      XW(I)=XC+XB*D2+YB*D1
   10 YW(I)=YC+YB*D2-XB*D1
      RETURN
      END

      SUBROUTINE DRAW(L,M,N,IA,IB,JA,JB,X,Y,P)
      DIMENSION IA(1),IB(1),JA(2,5,9),JB(2,5,9),X(1),Y(1),P(1)
      J1=JA(L,M,N)
      J2=JB(L,M,N)
```

```
      DO 20 J=J1,J2
      I1=IA(J)
      I2=IB(J)
   20 CALL VEC(I1,I2,X,Y)
      CALL PJOUT(J1,J2,P)
      RETURN
      END

      SUBROUTINE VEC(I1,I2,X,Y)
      DIMENSION X(1),Y(1)
      WRITE(6,560) ((X(I),Y(I)),I=I1,I2)
C     ENDFILE 6
      WRITE(8,560) ((X(I),Y(I)),I=I1,I2)
  560 FORMAT(*$VEC *,9(18F7.0,* & */))
      RETURN
      END

      SUBROUTINE MOM(J1,J2,IA,IB,X,Y,P,H,EX,EY,EXX,EYY,EXY,XCEN,YCEN)
      DIMENSION XCEN(1),YCEN(1),IA(1),IB(1),X(1),Y(1),P(1),H(1)
      EX=EY=EXX=EYY=EXY=0.
      DO 55 J=J1,J2
      I1=IA(J)
      I2=IB(J)-1
      CALL MESSAGC(7HMOM      ,7HCENTRD )
      CALL CENTRD(I1,I2,X,Y,A,AX,AY,AXX,AYY,AXY)
      XCEN(J)=AX/A
      YCEN(J)=AY/A
      Z=P(J)/A
      EX=EX+Z*AX
      EY=EY+Z*AY
      EXX=EXX+Z*AXX
      EYY=EYY+Z*AYY
      EXY=EXY+Z*AXY
      H(J)=P(J)/ABS(A)
   55 CONTINUE
      EXX=EXX-EX*EX
      EYY=EYY-EY*EY
      EXY=EXY-EX*EY
      RETURN
      END

      SUBROUTINE EIGEN(EXX,EYY,EXY,R1,R2,D1,D2)
      IF(EXY.NE.0.) GO TO 70
      IF(EXX.GT.EYY) GO TO 60
      D1=0. $D2=1.
      U=EXX $V=EYY $GO TO 80
   60 D1=1. $D2=0.
      U=EYY $V=EXX $GO TO 80
   70 EE=EXY*EXY
      P=EXX+EYY
      Q=SQRT((EXX-EYY)**2+4.*EE)
      U=(P-Q)/2.
      V=(P+Q)/2.
      W=V-EXX
      D=SQRT(EE+W*W)
      D1=EXY/D
      D2=W/D
   80 R1=0.
      IF(U/P.GT.1.E-100) R1=2.*SQRT(U)
      R2=2.*SQRT(V)
      RETURN
```

```
      END

      SUBROUTINE SAVE(L,M,N,EX,EY,R1,R2,D1,D2,EXX,EYY,EXY,SE,KE,KL)
      DIMENSION SE(1),KE(2,5,9)
      SE(1+KL)=EX
      SE(2+KL)=EY
      SE(3+KL)=R1
      SE(4+KL)=R2
      SE(5+KL)=D1
      SE(6+KL)=D2
      SE(7+KL)=EXX
      SE(8+KL)=EYY
      SE(9+KL)=EXY
      KE(L,M,N)=KL
      KL=KL+9
      RETURN
      END

      SUBROUTINE SECR(PI,BB,BSD,XC,YC,RI,RO,RXO,RYO,XW,YW,IMAX)
      DIMENSION XW(1),YW(1)
      BS=BSD*PI/180.
      AL=ATAN2(RXO,RYO)-BS/2.
      IB=BS/BB+.5
      IF(IB.LE.0) IB=1
      BI=BS/IB
      IBO=IB+1
      IMAX=2*IBO+1
      DO 96 I=1,IBO
      B=BI*(I-1)+AL
      SBB=SIN(B)
      CBB=COS(B)
      XW(I)=XC+RO*SBB
      YW(I)=YC+RO*CBB
      IF(RI.EQ.0.) GO TO 96
      J=IMAX-I
      XW(J)=XC+RI*SBB
      YW(J)=YC+RI*CBB
   96 CONTINUE
      IF(RI.GT.0.) GO TO 98
      IMAX=IBO+2
      XW(IBO+1)=XC
      YW(IBO+1)=YC
   98 XW(IMAX)=XW(1)
      YW(IMAX)=YW(1)
      RETURN
      END

      SUBROUTINE OAK(KL,JP,JQ,IA,IB,X,Y,RX,RY,IR,JMAX)
      DIMENSION PX(50),PY(50),QX(50),QY(50),WK(300),
     $          RX(1),RY(1),IR(2,1),X(1),Y(1),IA(1),IB(1)
      NP=IB(JP)-IA(JP)
      NQ=IB(JQ)-IA(JQ)
      IP=IA(JP)-1
      IQ=IA(JQ)-1
      DO 127 I=1,NP
      PX(I)=X(I+IP)
  127 PY(I)=Y(I+IP)
      DO 128 I=1,NQ
      QX(I)=X(I+IQ)
  128 QY(I)=Y(I+IQ)
      CALL IUCALC(PX,PY,NP,QX,QY,NQ,KL,WK,300,JMAX,IR,20,RX,RY,200)
```

```
      RETURN
      END

      FUNCTION AREA(XW,YW,IW,JO)
      DIMENSION XW(1),YW(1),IW(2,1),XS(50),YS(50)
      CALL MESSAGA(7HAREA     )
      AREA=0.
      IO=IW(1,JO)+1
      IMM=IW(2,JO)+IW(1,JO)-2
      XW1=XW(IO)
      YW1=YW(IO)
      DO 50 I=IO,IMM
      BX=XW(I+1)-XW1
      BY=YW(I+1)-YW1
      CX=XW(I+2)-XW1
      CY=YW(I+2)-YW1
      AI=.5*(CX*BY-BX*CY)
   50 AREA=AREA+AI
      WRITE(7,60) AREA
   60  FORMAT(*AREA IS *,G15.5)
      CALL MESSAGR(7HAREA    )
      RETURN
      END


      SUBROUTINE WEIGHT(H,J,P,IA,IB,X,Y)
      DIMENSION P(1),IA(1),IB(1),X(1),Y(1)
      I1=IA(J)
      I2=IB(J)-3
      A=0.
      X1=X(I1)
      Y1=Y(I1)
      DO 50 I=I1,I2
      BX=X(I+1)-X1
      BY=Y(I+1)-Y1
      CX=X(I+2)-X1
      CY=Y(I+2)-Y1
      AI=.5*(CX*BY-BX*CY)
      A=A+AI
   50 CONTINUE
      P(J)=A*H
      RETURN
      END

      SUBROUTINE LANDEF(LAX,LAY,LIA,LIB,LJB,LN,LE,LS,LW,
     $  XW,YW)
C
C     LANDEF AT USER OPTION DEFINES LAND MASSES BY ACCESSING A LAND FILE ON
C     UNIT 9 OR BY TERMINAL INPUT
C
      DIMENSION XW(1),YW(1),LIA(200),LIB(200),LJA(1,1,1),LJB(1,1,1)
      LOGICAL FLAG
      REAL LAX(1400),LAY(1400),LW(200),LN(200),LE(200),LS(200)
      DATA LIL,LJL/0,0/,JJ,J2/0,0/
      DATA I2/0/
      CALL MESSAGA(7HLANDEF )
      WRITE(6,7)
      WRITE(8,7)
    7 FORMAT(*$LIN 8*)
      LJB(1,1,1)=0
      FLAG=.F.
```

```
      WRITE(6,10)
      WRITE(8,10)
C     ENDFILE 6
   10 FORMAT(* USE LAND FILE? (Y OR N) *)
      READ(5,20) Q
      WRITE(6,20) Q
      WRITE(7,20) Q
      WRITE(8,20) Q
   20 FORMAT(A1)
      IF(Q.EQ.1HY) 100,200
  100 CONTINUE
      READ(9) LAX,LAY,LIA,LIB,LJA,LJB,LW,LE,LN,LS
      FLAG=.T.
      J2=LJB(1,1,1)
      DO 130 J=1,J2
      I1=LIA(J)
      I2=LIB(J)
      CALL VEC(I1,I2,LAX,LAY)
  130 CONTINUE
  200 CONTINUE
      WRITE(6,30)
      WRITE(8,30)
C     ENDFILE 6
   30 FORMAT(* DEFINE LAND FROM TERMINAL? (Y OR N)*)
      READ(5,20) Q
      WRITE(6,20) Q
      WRITE(7,20) Q
      WRITE(8,20) Q
      IF(Q.EQ.1HY) 300,400
  300 CONTINUE
      IF(FLAG) 320,325
  320 CONTINUE
      JJ=J2
  325 CONTINUE
      IMAX=99
      JJ=JJ+1
      CALL GET(IMAX,XW,YW,XO,YO,R1,R2,D1,D2)
      CALL VEC(1,IMAX,XW,YW)
      CALL PUT(1,1,1,JJ,IMAX,XW,YW,LIA,LIB,LJA,LJB,LJ,LAX,LAY,
     $  I2,J2)
      CALL MESSAGC(7HLANDEF ,7HRECTAN )
      CALL RECTAN(1,IMAX,XW,YW,LN(LJ),LE(LJ),LS(LJ),LW(LJ))
      CALL MESSAGA(7HLANDEF )
      WRITE(6,40)
C     ENDFILE 6
      WRITE(8,40)
   40 FORMAT(*MORE LAND? (Y OR N)*)
      READ(5,20) Q
      WRITE(6,20) Q
      WRITE(7,20) Q
      WRITE(8,20) Q
      IF(Q.EQ.1HY) GOTO 325
  400 CONTINUE
      WRITE(6,50)
      WRITE(8,50)
   50 FORMAT(* SAVE LAND ON UNIT 10?  (Y OR N)*)
      READ(5,20) Q
      WRITE(6,20) Q
      WRITE(7,20) Q
      WRITE(8,20) Q
      IF(Q.EQ.1HY) 500,600
  500 CONTINUE
      WRITE(10) LAX,LAY,LIA,LIB,LJA,LJB,LW,LE,LN,LS
      ENDFILE 10
  600 CONTINUE
      CALL MESSAGR(7HLANDEF )
      RETURN
      END
```

```
      SUBROUTINE MOVMENT(M,N,JA,JB,X,Y,IA,IB,P,XCEN,YCEN,
     $   XW,YW,J,ILA,JLA,FLAG,SE,KE,KLA,H,LAX,LAY,LIA,LIB,
     $   LAN,LAND,LN,LE,LS,LW,INC,CLOSE)
C
C     POSITION DISTRIBUTION IS MOVED TO ACCOUNT FOR THE PASSAGE OF TIME
C     INCORPORATING SPEED AND COURSE UNCERTAINTY AND LAND INTERACTION.
C
      LOGICAL FLAG,LAND,LANINT
      DIMENSION JA(2,5,9),JB(2,5,9),X(1),Y(1),IA(1),IB(1),
     $   P(1),XCEN(1),YCEN(1),XW(1),YW(1),SE(1),H(1),IW(2,1),
     $   KE(2,5,9),WK(300),RX(200),RY(200),
     $   IR(2,20),XPC(200),YPC(200),XC(200),YC(200),XS(200),YS(200),
     $   IS(2,20),LIA(1),LIB(1)
      REAL LAX(1),LAY(1),LN(1),LE(1),LS(1),LW(1),INC
      DATA IR(1,1)/0/
      CALL MESSAGA(7HMOVMENT)
      FLAG=.T.
      SUB=7HMOVMENT
      M1=M+1
      JJ=0
      JA2=JA(2,M,N)
      JB2=JB(2,M,N)
      JA1=JA(1,M,N)
      JB1=JB(1,M,N)
      DO 1000 NC=JA2,JB2
        IA2=IA(NC)
        IB2=IB(NC)
        IC2=IB2-IA2+1
        PC=P(NC)
        DO 900 NP=JA1,JB1
C
C       GENERATE THE VELOCITY POLYGON WITH RESPECT TO POSITION POLYGON. THE
C       RESULT IS STORED IN XC,YC.
C
          SENC1=XCEN(NC)-XCEN(NP)
          SENC2=YCEN(NC)-YCEN(NP)
          SE1=-SE(1+KE(1,M,N))+XCEN(NP)
          SE2=-SE(2+KE(1,M,N))+YCEN(NP)
          DO 100 I=IA2,IB2
            IU=I-IA2+1
            XC(IU)=X(I)+SE1
            YC(IU)=Y(I)+SE2
  100     CONTINUE
C         WRITE(6,6)
C         WRITE(8,6)
C    6    FORMAT(*$LIN 5*)
C         CALL VEC(1,IC2,XC,YC)
          IA1=IA(NP)
          IB1=IB(NP)
          IC1=IB1-IA1+1
          PP=P(NP)
          IF((PP.LT.0.).AND.(PC.LT.0.)) GOTO 900
          IF(PP.LT.0.) GOTO 500
          IF(PC.LT.0.) GOTO 700
  200     CONTINUE
C
C         TWO POSITIVE INFORMATION AREAS.
C         GENERATE CUM(NP,NC),SEE DOCUMENTATION. RESULT IS STORED IN XS,YS.
C
          WRITE(7,7)
    7     FORMAT(*THE CENTROID POLYGON WITH RESPECT TO THE*
```

```
     $          * POSITION POLYGON IS *)
                WRITE(7,8) (XC(I),YC(I),I=1,IC2)
      8         FORMAT(5(G13.6,G13.6))
C               ENDFILE 7
                WRITE(6,19)
                WRITE(7,19)
                WRITE(8,19)
     19         FORMAT(*CALL NUCUMUN (Y OR N)*)
                READ(5,30) Q
                WRITE(6,30) Q
                WRITE(7,30) Q
                WRITE(8,30) Q
                IF(Q.EQ.1HN) GOTO 250
                CALL MESSAGC(7HPIAAM   ,7HNUCUMUN   )
                CALL NUCUMUN(NP,NC,IA,IB,X,Y,XW,YW,XS,YS,IS,IW,XCEN,YCEN,
     $            WK,XC,YC,FLAG,XPC,YPC,IC1,JMAX)
                CALL MESSAGA(7HMOVMENT)
                GOTO 300
    250         CONTINUE
     30         FORMAT(A1)
                CALL MESSAGC(7HPIAAM   ,7HCUMUN   )
                CALL CUMUN(NP,NC,IA,IB,X,Y,XW,YW,XS,YS,IS,IW,XCEN,YCEN,
     $            WK,XC,YC,FLAG,RX,RY,IR,JMAX)
                CALL MESSAGA(7HMOVMENT)
C
C               GENERATE PIAAM(NP,NC)= CUM(NP,NC) U NC(C) U NP(C)=OUTER BOUNDARY
C               OF CUM(NP,NC)=CLOCKWISE TURNING POLYGON. STORE RESULT IN XPC,YPC
C
                IF(.NOT.FLAG) RETURN
                CALL MESSAGC(7HPIAAM    ,7HSELECT )
                KAY=-1
                CALL SELECT(KAY,JMAX,XS,YS,IS,XPC,YPC,IC1)
                CALL MESSAGA(7HMOVMENT)
                IF(KAY.NE.0) GOTO 300
                WRITE(6,15)
                WRITE(7,15)
C               ENDFILE 7
                WRITE(8,15)
     15         FORMAT(*ERROR IN PIAAM CALCULATION*)
                FLAG=.F.
                CALL MESSAGR(7HMOVMENT)
                RETURN
    300         CONTINUE
                CALL CONCLDE(M1,N,JA,JB,X,Y,IA,IB,P,
     $            XCEN,YCEN,J,ILA,JLA,FLAG,SE,KE,KLA,H,LAX,LAY,LIA,
     $            LIB,LAN,LAND,LN,LE,LS,LW,INC,CLOSE,JJ,IC1,XPC,YPC,
     $            PP,PC,XW,YW,IW,RX,RY,IR,XS,YS,IS,LANINT)
                GOTO 900
    500         CONTINUE
C
C               POSITION POLYGON IS A NEGATIVE INFORMATION AREA
C               TRANSLATE THE NEGATIVE INFORMATION POSITION AREA
C               TO HAVE CENTROID AT ITS VELOCITY CENTROID.
C               STORE IN XPC,YPC.
C
                DO 525 I=IA1,IB1
                  IU=I-IA1+1
                  XPC(IU)=X(I)+SENC1
                  YPC(IU)=Y(I)+SENC2
    525         CONTINUE
                CALL MESSAGC(7HNIAAMPO,7HCUMUN   )
```

```
            WRITE(7,8) (XPC(I),YPC(I),I=1,IC1)
C           ENDFILE 7
            CALL CUMUN(NC,NP,IA,IB,X,Y,XW,YW,XS,YS,IS,
     $        IW,XCEN,YCEN,WK,XPC,YPC,
     $        FLAG,RX,RY,IR,JMAX)
            CALL MESSAGA(7HMOVMENT)
C
C           GENERATE NIAAM(NP,NC)=NP(C) & ¬CUM(NP,NC)=INNER BOUNDARY OF
C           CUM(NP,NC)=COUNTERCLOCKWISE POLYGON. STORE RESULT IN XPC,YPC.
C
            IF(.NOT.FLAG) RETURN
            CALL MESSAGC(7HNIAAMPO,7HSELECT )
            KAY=-2
            CALL SELECT(KAY,JMAX,XS,YS,IS,XPC,YPC,IC1)
            CALL MESSAGA(7HMOVMENT)
C
C           IF KAY=0 NIAAM IS THE EMPTY SET
C
            IF(KAY.EQ.0) GOTO 900
            CALL CONCLDE(M1,N,JA,JB,X,Y,IA,IB,P,
     $        XCEN,YCEN,J,ILA,JLA,FLAG,SE,KE,KLA,H,LAX,LAY,LIA,LIB,
     $        LAN,LAND,LN,LE,LS,LW,INC,CLOSE,JJ,IC1,XPC,YPC,PP,PC,XW,
     $        YW,IW,RX,RY,IR,XS,YS,IS,LANINT)
            IF(.NOT.FLAG) RETURN
            GOTO 900
  700       CONTINUE
C
C           CENTROID POLYGON IS A NEGATIVE INFORMATION AREA
C
            CALL MESSAGC(7HNIAAMCE,7HCUMUN  )
            CALL CUMUN(NP,NC,IA,IB,X,Y,XW,YW,XS,YS,IS,
     $        IW,XCEN,YCEN,WK,XC,YC,FLAG,
     $        RX,RY,IR,JMAX)
            CALL MESSAGA(7HMOVMENT)
C
C           GENERATE NIAAM(NP,NC)=NC(C) & ¬CUM(NP,NC)=INNER BOUNDARY OF
C           CUM(NP,NC)=COUNTERCLOCKWISE POLYGON. STORE RESULT IN XPC,YPC.
C
            IF(.NOT.FLAG) RETURN
            CALL MESSAGC(7HNIAAMCE,7HSELECT )
            KAY=-2
            CALL SELECT(KAY,JMAX,XS,YS,IS,XPC,YPC,IC1)
            CALL MESSAGA(7HMOVMENT)
C
C           IF KAY=0 NIAAM IS THE EMPTY SET
C
            IF(KAY.EQ.0) GOTO 900
            CALL CONCLDE(M1,N,JA,JB,X,Y,IA,IB,P,
     $        XCEN,YCEN,J,ILA,JLA,FLAG,SE,KE,KLA,H,LAX,LAY,LIA,LIB,
     $        LAN,LAND,LN,LE,LS,LW,INC,CLOSE,JJ,IC1,XPC,YPC,PP,PC,XW,
     $        YW,IW,RX,RY,IR,XS,YS,IS,LANINT)
            IF(.NOT.FLAG) RETURN
  900     CONTINUE
 1000   CONTINUE
        CALL MESSAGC(7HMOVMENT,7HUPDATE )
        CALL UPDATE(M1,N,JA,JB,X,Y,IA,IB,P,XCEN,YCEN,
     $    J,ILA,JLA,SE,KE,H,KLA,FLAG,1)
        CALL MESSAGA(7HMOVMENT)
        IF(.NOT.FLAG) RETURN
        CALL MESSAGR(7HMOVMENT)
        RETURN
```

```
      END
            SUBROUTINE CONCLDE(M1,N,JA,JB,X,Y,IA,IB,P,XCEN,YCEN,
     $      J,ILA,JLA,FLAG,SE,KE,KLA,H,LAX,LAY,LIA,LIB,LAN,LAND,LN,
     $      LE,LS,LW,INC,CLOSE,JJ,IC1,XPC,YPC,PP,PC,XW,YW,IW,RX,RY,
     $      IR,XS,YS,IS,LANINT)
C
C           CONCLDE STORES THE RESULTING POSITION POLYGON TAKING ACCOUNT
C           OF LAND INTERACTION.
C
            LOGICAL FLAG,LAND,LANINT
            DIMENSION JA(2,5,9),JB(2,5,9),X(1),Y(1),IA(1),IB(1),P(1),
     $        XCEN(1),YCEN(1),XW(1),YW(1),H(1),IW(2,1),SE(1),
     $        KE(2,5,9),WK(1),RX(1),RY(1),IR(2,1),XPC(1),YPC(1),
     $        XS(1),YS(1),IS(2,1),XWO(1400),YWO(1400),
     $        NUM(2,200),LIA(1),LIB(1)
            REAL LAX(1),LAY(1),LN(1),LE(1),LS(1),LW(1),INC
            INTEGER REM(2,20),INUM(50)
            DATA REM(1,1),NUM(1,1)/2*0/
            CALL MESSAGA(7HCONCLDE)
            IF(.NOT.LAND) GOTO 900
            CALL MESSAGC(7HCONCLDE,7HPOSINT )
C
C           LAN=LJB(1,1,1)
C
            CALL POSINT(XPC,YPC,IC1,LAX,LAY,LIA,LIB,LAN,LN,LE,LS,LW,XW
     $        ,YW,IW,RX,RY,IR,XS,YS,IS,LANINT,INC,CLOSE,FLAG,XWO,YWO,
     $        NUM,REM,IREM,INUM,PP,PC)
            CALL MESSAGA(7HCONCLDE)
            IF(.NOT.FLAG) RETURN
C
C           WHEN (XPC,YPC) INTERSECTS SOME LAND MASS,(XWO,YWO)
C           CONTAINS POSITION POLYGONS DETERMINED BY (XPC,YPC) AND
C           THE LAND MASSES (LAX,LAY). THE CONTENTS OF (XPC,YPC)
C           ARE UNCHANGED. IREM IS THE INDEX OF THE OPTIMAL SET OF
C           REMAINDER POLYGONS. INUM(IREM) IS THE INDEX OF THE LARGEST
C           REMAINDER POLYGON IN SET IREM.  THE VARIOUS SETS OF
C           REMAINDER POLYGONS ARE SPECIFIED BY (XWO,YWO,NUM,REM).
C           IF LANINT=.FALSE.,THEN (XPC,YPC) INTERSECTS NO LAND MASS
C
            IF(.NOT.LANINT) GOTO 900
            IF((PP.GT.0.).AND.(PC.GT.0.)) GOTO 800
            AR=0.
C
C           THIS CODE HANDLES NEGATIVE INFORMATION AREAS.
C
            J1=REM(1,1)
            J2=REM(2,1)
            DO 600 JO=1,J2
            IS1=NUM(1,J1+JO)
            IC1=NUM(2,J1+JO)
            DO 500 I=1,IC1
            XPC(I)=ROU(XWO(IS1+I))
            YPC(I)=ROU(YWO(IS1+I))
  500       CONTINUE
            IC1=IC1+1
            XPC(IC1)=XPC(1)
            YPC(IC1)=YPC(1)
            WRITE(7,60) JO,IREM
            WRITE(7,8) (XPC(I),YPC(I),I=1,IC1)
C           ENDFILE 7
            WRITE(6,40)
```

```
                WRITE(8,40)
                CALL VEC(1,IC1,XPC,YPC)
                JJ=JJ+1
                CALL PUT(1,M1,N,JJ,IC1,XPC,YPC,IA,IB,JA,JB,J,X,Y,ILA,JLA)
                A=AREA(XWO,YWO,NUM,J1+JO)
                AR=AR+A
                P(J)=PP*PC*A
                IF(JO.EQ.1) JJ1=J
                IF(JO.EQ.J2) JJ2=J
  600           CONTINUE
                DO 700 JO=JJ1,JJ2
                P(JO)=P(JO)/AR
  700           CONTINUE
                CALL MESSAGR(7HCONCLDE)
                RETURN
  800           CONTINUE
                WRITE(7,60) INUM(IREM),IREM
   60           FORMAT(*POLYGON *,I3,* IN THE OPTIMAL REMAINDER SET *,I3)
                JO=REM(1,IREM)+INUM(IREM)
                IS1=NUM(1,JO)
                IC1=NUM(2,JO)
                DO 860 I=1,IC1
                  XPC(I)=ROU(XWO(IS1+I))
                  YPC(I)=ROU(YWO(IS1+I))
  860           CONTINUE
  900           CONTINUE
                IC1=IC1+1
                XPC(IC1)=XPC(1)
                YPC(IC1)=YPC(1)
                WRITE(7,8) (XPC(I),YPC(I),I=1,IC1)
    8           FORMAT(5(G13.6,G13.6))
C               ENDFILE 7
                WRITE(6,40)
                WRITE(8,40)
   40           FORMAT(*$LIN 1*)
                CALL VEC(1,IC1,XPC,YPC)
                JJ=JJ+1
                CALL PUT(1,M1,N,JJ,IC1,XPC,YPC,IA,IB,JA,JB,J,X,Y,ILA,JLA)
                P(J)=PP*PC
                CALL MESSAGR(7HCONCLDE)
                RETURN
                END
      LOGICAL FUNCTION COLL(PARX,PARY)
C
C     COLL IS TRUE WHEN POINTS IN PARX,PARY DO NOT FORM A PARALLELOGRAM.
C
      DIMENSION PARX(4),PARY(4)
      CALL MESSAGA(7HCOLL    )
      COLL=.F.
      E=.0001
      EP=1.
      DO 50 I=1,3
      I1=I+1
      DO 40 J=I1,4
      IF((ABS(PARX(I)-PARX(J)).GT.EP).OR.(ABS(PARY(I)-PARY(J)).GT.EP))
     $  GOTO 39
      COLL=.T.
      CALL MESSAGR(7HCOLL    )
      RETURN
   39 CONTINUE
   40 CONTINUE
```

```
   50 CONTINUE
      A1=PARY(2)-PARY(1)
      B1=PARX(2)-PARX(1)
      C1=(A1*PARX(1))-(PARY(1)*B1)
      A2=PARY(3)-PARY(4)
      B2=PARX(3)-PARX(4)
      C2=(A2*PARX(4))-(PARY(4)*B2)
      IF(ABS(A2).LT.E) 100,200
  100 CONTINUE
      IF(ABS((C1/B1)-(C2/B2)).LT.EP) COLL=.T.
      CALL MESSAGR(7HCOLL    )
      RETURN
  200 CONTINUE
      IF(ABS((C1/A1)-(C2/A2)).LT.EP) COLL=.T.
      CALL MESSAGR(7HCOLL    )
      RETURN
      END
      SUBROUTINE CUMUN(NGON1,NGON2,IA,IB,X,Y,XW,YW,XS,
     $  YS,IS,IW,XCEN,YCEN,WK,PGX,PGY,FLAG,
     $  RX,RY,IR,JMAX)
C
C     GIVEN POLYGONS NGON1 AND NGON2,GENERATE CUM(NGON1,NGON2).
C     THIS IS THE SET RESULTING FROM MOVING NGON1 AROUND THE VERTICES
C     OF NGON2.  FOR EACH J EVALUATE NGON1 AT VERTEX J OF NGON2,CALL
C     IT NGON1(J).  UNION TO NGON1(J) ALL PARALLELOGRAMS GENERATED
C     BY NGON1(J) AND NGON1(J+1).  TAKE THE UNION OF THIS RESULT OVER
C     ALL J.  CUM(NGON1,NGON2) ACTUALLY EQUALS THE UNION OVER ALL J
C     OF ALL PARALLELOGRAMS GENERATED BY NGON1(J) AND NGON1(J+1).
C     BUT SOMETIMES WHEN JUST PARALLELOGRAM UNIONS WERE USED,
C     COMPUTATIONAL PROBLEMS CAUSED MORE THAN ONE FIGURE TO RESULT
C     AT INCONVENIENT POINTS.  THIS PROBLEM WAS ALLEVIATED BY
C     INCORPORATING NGON1(J) EXPLICITLY IN THE CALCULATIONS.
C     XS,YS CONTAINS THE RESULT CUM(NGON1,NGON2).
C
      LOGICAL COLL,FLAG,FLAG1
      DIMENSION X(1),Y(1),IA(1),IB(1),XW(1),YW(1),IW(2,1),XS(1),
     $  YS(1),PX(50),PY(50),QX(50),QY(50),WK(1),PGX(1),PGY(1),
     $  PARX(4),PARY(4),RX(1),RY(1),IS(2,1),IR(2,1),
     $  XCEN(1),YCEN(1)
C
C     PGX,PGY CONTAINS X AND Y COORDINATES OF NGON2.M2 IS THE NUMBER OF
C     DIFFERENT VERTICES OF NGON2
C     PX,PY CONTAINS NGON1 AT THE J VERTEX OF NGON2
C     QX,QY CONTAINS NGON1 AT THE J+1 VERTEX OF NGON2
C     PARX,PARY CONTAIN THE VERTICES OF PARALLOLOGRAM I,J
C
      CALL MESSAGA(7HCUMUN   )
      IA1=IA(NGON1)
      M1=IB(NGON1)-IA1
      M11=M1+1
      M2=IB(NGON2)-IA(NGON2)
      M22=M2+1
      M21=M2-1
      SUB=7HCUMUN
      XC1=-XCEN(NGON1)+PGX(1)
      YC1=-YCEN(NGON1)+PGY(1)
      DO 1500 J=1,M2
        XC2=-XCEN(NGON1)+PGX(J+1)
        YC2=-YCEN(NGON1)+PGY(J+1)
C
C       EVALUATE AND SAVE NGON1(J) AND NGON1(J+1)
```

```
C
      IF(J.EQ.1) 100,200
  100 DO 120 I=1,M1
        IA11=IA1+I-1
        PX(I)=ROU(X(IA11)+XC1)
        PY(I)=ROU(Y(IA11)+YC1)
  120 CONTINUE
      PX(M11)=PX(1)
      PY(M11)=PY(1)
      GOTO 400
  200 CONTINUE
      DO 220 I=1,M11
        PX(I)=QX(I)
        PY(I)=QY(I)
  220 CONTINUE
  400 CONTINUE
      DO 500 I=1,M1
        IA11=IA1+I-1
        QX(I)=ROU(X(IA11)+XC2)
        QY(I)=ROU(Y(IA11)+YC2)
  500 CONTINUE
      QX(M11)=QX(1)
      QY(M11)=QY(1)
  800 CONTINUE
      DO 1000 I=1,M1
      IF(I.EQ.1) 820,840
  820 CONTINUE
      IW1=M1
      DO 830 K=1,IW1
        XW(K)=PX(K)
        YW(K)=PY(K)
  830 CONTINUE
  840 CONTINUE
C
C     GENERATE PARALLELOGRAM I
C
      PARX(1)=PX(I)
      PARY(1)=PY(I)
      PARX(2)=PX(I+1)
      PARY(2)=PY(I+1)
      PARX(3)=QX(I+1)
      PARY(3)=QY(I+1)
      PARX(4)=QX(I)
      PARY(4)=QY(I)
      CALL MESSAGC(7HCUMUN  ,7HCOLL    )
      IF(COLL(PARX,PARY)) GOTO 1000
C
C     GENERATE UNION OF PARALLELOGRAMS(I,J),I=1,M1,= XW,YW
C
      CALL MESSAGC(7HCUMUN  ,7HPREIU 1)
      CALL PREIU(XW,YW,IW1,PARX,PARY,4,1,IS,XS,YS,SUB,FLAG,
     $   FLAG1,JMAX)
      CALL MESSAGA(7HCUMUN  )
      IF(.NOT.FLAG) RETURN
 1000 CONTINUE
 1200 CONTINUE
C
C     UNION XW,YW WITH PREVIOUS RESULTS  STORE RESULT IN XS,YS,AND (WHEN
C     RESULT IS ONE POLYGON) IN RX,RY.
C
      IF(J.EQ.1) 1300,1400
```

```
 1300    CONTINUE
         IR1=IR(2,1)=IW1
         DO 1325 K=1,IR1
           RX(K)=XW(K)
           RY(K)=YW(K)
 1325    CONTINUE
         GOTO 1500
 1400    CONTINUE
         CALL MESSAGC(7HCUMUN  ,7HPREIU 3)
         CALL PREIU(RX,RY,IR1,XW,YW,IW1,1,IS,XS,YS,SUB,FLAG,FLAG1,JMAX)
         CALL MESSAGA(7HCUMUN  )
C
C        WHEN J=M2,RESULT COULD BE A POLYGON AND A HOLE. THIS IS THE CASE
C        WHEN CUM SURROUNDS A NONEMPTY AREA. THEN FIND RESULT IN XS,YS.
         IF(.NOT.FLAG) RETURN
 1500 CONTINUE
      CALL MESSAGR(7HCUMUN  )
      RETURN
      END
      SUBROUTINE PREIU(XW,YW,IW1,PX,PY,M1,KL,IS,XS,YS,
     $  SUB,FLAG,FLAG1,JMAX)
      LOGICAL FLAG,FLAG1
      DIMENSION XW(1),YW(1),PX(1),PY(1),XS(1),YS(1),WK(300),
     $  IS(2,1)
C
C     PREIU CALLS IUCALC. TESTS JMAX FOR IUCALC AND CALLING ROUTINE ERRORS.
C     THE IUCALC RESULT IS TRANSFERED TO XW,YW.
C     IF ERROR THEN AN APPROPRIATE MESSAGE IS WRITTEN.
C     SUB CONTAINS THE NAME OF CALLING ROUTINE.
      CALL MESSAGA(7HPREIU  )
      FLAG=.T.
      FLAG1=.T.
      WRITE(7,30) (XW(I),YW(I),I=1,IW1)
      WRITE(7,30) (PX(I),PY(I),I=1,M1)
      CALL MESSAGC(7HPREIU  ,7HIUCALC )
      CALL IUCALC(XW,YW,IW1,PX,PY,M1,KL,WK,300,JMAX,IS,20,XS,YS,200)
      CALL MESSAGA(7HPREIU  )
      IF(JMAX) 100,200,300
  100 CONTINUE
      WRITE(6,15) JMAX,SUB
      WRITE(7,15) JMAX,SUB
C     ENDFILE 7
   15 FORMAT(*IUCALC ERROR *,I3,* OCCURS IN *,A7)
      WRITE(6,25)
      WRITE(8,25)
   25 FORMAT(*$LIN 7*)
      IW11=IW1+1
      XW(IW11)=XW(1)
      YW(IW11)=YW(1)
      M11=M1+1
      PX(M11)=PX(1)
      PY(M11)=PY(1)
   30 FORMAT(5(G13.5,G13.5))
      CALL VEC(1,IW11,XW,YW)
      CALL VEC(1,M11,PX,PY)
      FLAG=.F.
      CALL MESSAGR(7HPREIU  )
      RETURN
  200 CONTINUE
C     WRITE(6,40) SUB,JMAX
      WRITE(7,40) SUB,JMAX
```

```
   40 FORMAT(A7,*RESULT IS THE SAME AS ORIGINAL DATA OR NO RESULT *,
     $  *EXISTS. *,I3)
C      ENDFILE 7
C
C      RESULT IS THE SAME AS ORIGINAL DATA OR NO RESULT EXISTS
C
       FLAG1=.F.
       CALL MESSAGR(7HPREIU  )
       RETURN
  300  CONTINUE
C      WRITE(6,50) JMAX
       WRITE(7,50) JMAX
C      ENDFILE 7
   50  FORMAT(*PREIU CALLS RECOVER *,I3)
       CALL RECOVER(JMAX,XS,YS,IS,XW,YW,IW1,SUB)
       CALL MESSAGA(7HPREIU  )
       IF(JMAX.EQ.0) FLAG1=.F.
       CALL MESSAGR(7HPREIU  )
       RETURN
       END
       SUBROUTINE SELECT(K,JMAX,XS,YS,IS,XPC,YPC,IC1)
C
C      SELECT EXTRACTS THE APPROPRIATE POLYGON FROM CUM(NP,NC),
C      STORES IT IN (XPC,YPC).
C      IF K=-1 THE OUTER BOUNDARY OF CUM IS SELECTED,THIS IS A CLOCKWISE
C      POLYGON AND PIAAM AND AREA.GT.0.
C      IF K=-2 THE INNER BOUNDARY,IF IT EXISTS,IS SELECTED,THIS IS A COUNTER-
C      CLOCKWISE POLYGON AND NIAAM AND AREA.LT.0.
C      K=0 IF NIAAM DOES NOT EXIST.
C
       DIMENSION XS(1),YS(1),IS(2,1),XPC(1),YPC(1)
       CALL MESSAGA(7HSELECT )
       IF(JMAX.EQ.0) STOP 4
       IF(JMAX.EQ.1) 100,500
  100  IF(K.EQ.-1) 200,300
  200  CONTINUE
       IC1=IS(2,1)
       DO 225 I=1,IC1
         XPC(I)=XS(I)
         YPC(I)=YS(I)
  225  CONTINUE
       WRITE(7,10)
   10  FORMAT(*SELECT CHOOSES THE POLYGON DEFINED BY *)
       WRITE(7,15) (XPC(I),YPC(I),I=1,IC1)
   15  FORMAT(5(G13.5,G13.5))
C      ENDFILE 7
       CALL MESSAGR(7HSELECT )
       RETURN
  300  CONTINUE
       K=0
       CALL MESSAGR(7HSELECT )
       RETURN
  500  CONTINUE
       MAX=JMAX
       DO 600 J=1,MAX
C
C      CALCULATE AREA OF POLYGON IN IUCALC OUTPUT FORMAT.
C
       CALL MESSAGC(7HSELECT ,7HAREA    )
       A=AREA(XS,YS,IS,J)
       CALL MESSAGA(7HSELECT )
```

```
      IF((K.EQ.-1).AND.(A.LT.0.)) GOTO 600
      IF((K.EQ.-2).AND.(A.GT.0.)) GOTO 600
      I00=IS(1,J)+1
      IC1=IS(2,J)
      I0M=IS(1,J)+IC1
      DO 560 I=I00,I0M
        IU=I-I00+1
        XPC(IU)=XS(I)
        YPC(IU)=YS(I)
  560 CONTINUE
      WRITE(7,10)
      WRITE(7,15) (XPC(I),YPC(I),I=1,IC1)
C     ENDFILE 7
      CALL MESSAGR(7HSELECT )
      RETURN
  600 CONTINUE
      K=0
      CALL MESSAGR(7HSELECT )
      RETURN
      END
      SUBROUTINE UPDATE(M1,N,JA,JB,X,Y,IA,IB,P,XCEN,YCEN,
     $  J,ILA,JLA,SE,KE,H,KLA,FLAG,L)
C
C     UPDATE CALCULATES STATISTICS FOR THE NEW DISTRIBUTION
C
      LOGICAL FLAG
      DIMENSION JA(2,5,9),JB(2,5,9),X(1),Y(1),IA(1),IB(1),P(1),
     $  XCEN(1),YCEN(1),SE(1),KE(2,5,9),H(1)
      CALL MESSAGA(7HUPDATE )
      FLAG=.T.
      J1=JA(L,M1,N)
      J2=JB(L,M1,N)
      CALL NORM(J1,J2,P)
      CALL MOM(J1,J2,IA,IB,X,Y,P,H,EX,EY,EXX,EYY,EXY,XCEN,YCEN)
      IF((EXX.GT.0.).AND.(EYY.GT.0.)) GOTO 100
      WRITE(6,10)
      WRITE(7,10)
C     ENDFILE 7
   10 FORMAT(*NEGATIVE VARIANCE WAS COMPUTED IN UPDATE*/)
      FLAG=.F.
      CALL MESSAGR(7HUPDATE )
      RETURN
  100 CONTINUE
      CALL EIGEN(EXX,EYY,EXY,R1,R2,D1,D2)
      CALL SAVE(L,M1,N,EX,EY,R1,R2,D1,D2,EXX,EYY,EXY,SE,KE,KLA)
      CALL MESSAGR(7HUPDATE )
      RETURN
      END
      REAL FUNCTION ROU(X)
      ROU=AINT(X+0.5)
      END
      SUBROUTINE RECOVER(JMAX,XS,YS,IS,XW,YW,IW1,SUB)
      REAL IE
      DIMENSION XS(1),YS(1),XW(1),YW(1),XR(200),YR(200),IS(2,1),
     $  IR(2,20)
C     COINCIDENT VERTICES AND DEGENERATE POLYGONS ARE REMOVED.
C
      CALL MESSAGA(7HRECOVER)
      IR1=0
      IR(1,1)=IR1
      MAX=JMAX
```

```
      IE=1.
      DO 600 J=1,JMAX
      IS1=IS(1,J)
      IS2=IS(2,J)
      K1=1
      R=XR(IR1+1)=ROU(XS(IS1+1))
      S=YR(IR1+1)=ROU(YS(IS1+1))
      DO 450 K=2,IS2
      R1=ROU(XS(IS1+K))
      S1=ROU(YS(IS1+K))
      IF((ABS(R-R1).LE.IE).AND.(ABS(S-S1).LE.IE))
     $  GOTO 450
      K1=K1+1
      R=XR(IR1+K1)=R1
      S=YR(IR1+K1)=S1
  450 CONTINUE
      R1=XR(IR1+1)
      S1=YR(IR1+1)
      IF((ABS(R-R1).LE.IE).AND.(ABS(S-S1).LE.IE))
     $  K1=K1-1
      IF(K1.LE.2) 500,550
  500 CONTINUE
      MAX=MAX-1
      GOTO 600
  550 CONTINUE
      IR(2,J)=K1
      IR1=IR(1,J+1)=IR(1,J)+K1
  600 CONTINUE
      JMAX=MAX
      IF(JMAX.EQ.1) 650,700
  650 CONTINUE
      IW1=IR(2,1)
      DO 675 I=1,IW1
      XW(I)=XR(I)
      YW(I)=YR(I)
  675 CONTINUE
      WRITE(7,10) (XW(I),YW(I),I=1,IW1)
C     ENDFILE 7
   10 FORMAT(5(G13.5,G13.5))
      CALL MESSAGR(7HRECOVER)
      RETURN
  700 CONTINUE
      IF(JMAX.EQ.0) 725,735
  725 CONTINUE
      WRITE(7,15) SUB,JMAX
   15 FORMAT(A7,* RESULT IS THE SAME AS ORIGINAL DATA OR NO RESULT *,
     $  * EXISTS. *,I3)
C     ENDFILE 7
      RETURN
  735 CONTINUE
C     WRITE(6,20) SUB,JMAX
      WRITE(7,20) SUB,JMAX
C     ENDFILE 7
   20 FORMAT(A7,* ERRCR *,I3)
      DO 750 J=1,JMAX
      IR1=IR(1,J)
      IR2=IR(2,J)
      WRITE(7,10) (XR(IR1+I),YR(IR1+I),I=1,IR2)
C     ENDFILE 7
  750 CONTINUE
      CALL MESSAGC(7HRECOVER,7HSELECT )
```

```
      CALL SELECT(-1,JMAX,XR,YR,IR,XW,YW,IW1)
      CALL MESSAGA(7HRECOVER)
      CALL MESSAGR(7HRECOVER)
      RETURN
      END
      SUBROUTINE MESSAGR(NAME)
C
C     MESSAGR IS CALLED MANY PLACES.  IT LEAVES RETURN MESSAGES
C
C     WRITE(6,10) NAME
C     ENDFILE 6
      WRITE(7,10) NAME
C     ENDFILE 7
C     WRITE(8,10) NAME
   10 FORMAT(A7,* RETURNS *)
      RETURN
      END
```

```
      SUBROUTINE CENTRD(I1,I2,X,Y,A,AX,AY,AXX,AYY,AXY)
C
C     CENTRD CALCULATES THE AREA,AND FIRST MOMENTS AND VALUES
C     PROPORTIONAL TO THE SECOND MOMENTS OF A POLYGON
C     WHOSE DISTINCT VERTICES ARE STORED IN IUCALC OUTPUT FORMAT
C     IN X(I),Y(I),I=I1,I2.
C     CENTRD IS CALLED BY GOLDSEC AND MOM.
C
      DIMENSION X(1),Y(1)
      CALL MESSAGA(7HCENTRD )
      I22=I2-2
      A=AX=AY=AXX=AYY=AXY=0.
      DO 50 I=I1,I22
        BX=X(I+1)-X(I1)
        BY=Y(I+1)-Y(I1)
        CX=X(I+2)-X(I1)
        CY=Y(I+2)-Y(I1)
        AI=0.5*(CX*BY-BX*CY)
        A=A+AI
        XI=X(I1)+(BX+CX)/3.
        YI=Y(I1)+(BY+CY)/3.
        AX=AX+AI*XI
        AY=AY+AI*YI
        AXX=AXX+AI*(XI*XI+(BX*BX-BX*CX+CX*CX)/18.)
        AYY=AYY+AI*(YI*YI+(BY*BY-BY*CY+CY*CY)/18.)
        AXY=AXY+AI*(XI*YI+(BX*BY-0.5*(BX*CY+CX*BY)+CX*CY)/18.)
   50 CONTINUE
      WRITE(7,20) A,AX,AY,AXX,AYY,AXY
   20 FORMAT(*THE OUTPUT OF CENTRD IS *,/(5G15.5))
      CALL MESSAGR(7HCENTRD )
      RETURN
      END
      SUBROUTINE MESSAGA(NAME)
C
C     MESSAGA IS CALLED MANY PLACES. IT LEAVES ANSWER MESSAGES.
C
C     WRITE(6,10) NAME
C     ENDFILE 6
      WRITE(7,10) NAME
C     ENDFILE 7
C     WRITE(8,10) NAME
   10 FORMAT(A7,* ANSWERS *)
      RETURN
      END
      SUBROUTINE MESSAGC(NAME1,NAME2)
C
C     MESSAGC IS CALLED MANY PLACES. IT LEAVES CALL MESSAGES.
C
C     WRITE(6,10) NAME1,NAME2
C     ENDFILE 6
      WRITE(7,10) NAME1,NAME2
C     ENDFILE 7
C     WRITE(8,10) NAME1,NAME2
   10 FORMAT(A7,* CALLS *,A7)
      RETURN
      END
      REAL FUNCTION NUSCALE(SCALE,INC)
C
C     NUSCALE IS CALLED BY GOLDSEC.
C     NUSCALE USES SCALE AND INC TO GENERATE THE NEXT TERM IN THE SEQUENCE
C
```

```
      REAL INC
      NUSCALE=SCALE+INC
      RETURN
      END
        SUBROUTINE RECTAN(I1,I2,X,Y,N,E,S,W)
C
C       RECTAN DETERMINES A RECTANGLE CIRCUMSCRIBING P=(I1,I2,X,Y) AND STORES
C       THE EXTREME X COORDINATES IN W (=WEST) AND E (=EAST) AND THE EXTREME
C       Y COORDINATES IN N (=NORTH) AND S (=SOUTH).
C       (X(I),Y(I),I=I1,I2) ARE DISTINCT VERTICES OF P
C       RECTAN IS CALLED BY LANDEF, POSINT, AND REMAIN.
        REAL W,E,N,S
        DIMENSION X(1),Y(1)
        CALL MESSAGA(7HRECTAN )
        WRITE(7,12)(X(I),Y(I),I=I1,I2)
C       ENDFILE 7
   12   FORMAT(* INPUT TO RECTAN*,/5(G13.6,G13.6))
        EP=0.0001
        W=E=X(I1)
        N=S=Y(I1)
        III=I1+1
        DO 1000 I=III,I2
          PX=X(I)
          PY=Y(I)
          IF(PX.LE.W-EP) W=PX
          IF(PX.GE.E+EP) E=PX
          IF(PY.LE.S-EP) S=PY
          IF(PY.GE.N+EP) N=PY
 1000   CONTINUE
C       WRITE(8,15) N,E,S,W
C       WRITE(6,15) N,E,S,W
C       ENDFILE 6
        WRITE(7,15) N,E,S,W
C       ENDFILE 7
   15   FORMAT(*NORTH=*,G15.5,*EAST=*,G15.5,*SOUTH=*,G15.5,*WEST=*,
     $    G15.5)
        CALL MESSAGR(7HRECTAN )
        RETURN
        END
        LOGICAL FUNCTION MINIMAX(N,E,S,W,LN,LE,LS,LW)
C
C       MINIMAX DETERMINES IF THE RECTANGLES GIVEN BY (W,N,E,S) AND
C       (LW,LN,LE,LS) OVERLAP. IT IS USED TO PRESCREEN POSITION POLYGONS AND
C       LAND MASSES. MINIMAX = .T. IF THERE IS NO OVERLAP, = .F. OTHERWISE.
C       THIS TEST IS FROM GILOI,W.K.,P.158; EXCEPT IF THE ONLY OVERLAP OCCURS
C       BETWEEN BOUNDARIES OF THE RECTANGLES, MINIMAX= .T.
C       MINIMAX IS CALLED BY POSINT, REMAIN, MOVECEN, NEWCEN.
C
        REAL W,E,N,S,LW,LE,LN,LS
        CALL MESSAGA(7HMINIMAX)
        MINIMAX=.F.
        EP=0.01
        IF((E.LE.LW+EP).OR.(LE.LE.W+EP).OR.(N.LE.LS+EP).OR.(LN.LE.S+EP))
     $    MINIMAX=.TRUE.
C       WRITE(6,20) MINIMAX
C       ENDFILE 6
        WRITE(7,20) MINIMAX
C       ENDFILE 7
C       WRITE(8,20) MINIMAX
   20   FORMAT(*MINIMAX=*,L3)
        CALL MESSAGR(7HMINIMAX)
```

```
      RETURN
      END
      SUBROUTINE POSINT(XPC,YPC,IC1,LAX,LAY,LIA,LIB,LAN,LN,LE,LS,LW,
     $   XW,YW,IW,RX,RY,IR,XS,YS,IS,LANINT,INC,CLOSE,FLAG,XWO,YWO,NUM,
     $   REM,IREM,INUM,PC,PP)
C
C     POSINT, GIVEN A POSITION POLYGON AND LAND MASSES, DETERMINES THE
C     POLYGON(S) WHICH CONSERVE AREA YET INTERSECT NO LAND.
C     POSINT IS CALLED BY CONCLDE.
C
      DIMENSION LIA(1),LIB(1),XPC(1),YPC(1),XW(1),YW(1),
     $   IW(2,1),RX(1),RY(1),IR(2,1),XS(1),YS(1),IS(2,1),WK(300),
     $   XWO(1),YWO(1),NUM(2,1),INUM(1),AREAS(50)
      INTEGER REM(2,1)
      REAL LAX(1),LAY(1),LW(1),LE(1),LN(1),LS(1),INC,NORTH,EAST,SOUTH,
     $   WEST
      LOGICAL MINIMAX,FLAG,FLAG1,LANINT
C
C     XPC,YPC,IC1 CONTAINS THE CURRENT POSITION POLYGON
C
      CALL MESSAGA(7HPOSINT )
      CALL MESSAGC(7HPOSINT ,7HRECTAN )
C
C     DETERMINE CIRCUMSCRIBING RECTANGLE OF XPC,YPC
C
      CALL RECTAN(1,IC1,XPC,YPC,NORTH,EAST,SOUTH,WEST)
      CALL MESSAGA(7HPOSINT )
C
C     LAN=LJB(1,1,1)
C
      FLAG1=.T.
      LANINT=.T.
      DO 1000 J=1,LAN
C
C       TEST FOR OVERLAP OF CIRCUMSCRIBING RECTANGLE OF (XPC,YPC) AND
C       CIRCUMSCRIBING RECTANGLE OF (LAX(I),LAY(I)) I=LIA(J),LIB(J).
C
        CALL MESSAGC(7HPOSINT ,7HMINIMAX)
        IF(MINIMAX(NORTH,EAST,SOUTH,WEST,LN(J),LE(J),LS(J),LW(J)))
     $     GOTO 900
C
C       TEST FOR OVERLAP OF (XPC,YPC) AND POSSIBLE (LAX(I),LAY(I)) I=LIA(J),
C       LIB(J).
C
        I1=LIA(J)
        I2=LIB(J)-1
        DO 100 IN=I1,I2
          I=IN-I1+1
          XW(I)=LAX(IN)
          YW(I)=LAY(IN)
  100   CONTINUE
        IW1=I2-I1+1
        CALL MESSAGC(7HPOSINT ,7HIUCALC )
        CALL IUCALC(XW,YW,IW1,XPC,YPC,IC1,2,WK,300,JMAX,IS,20,XS,YS,
     $     200)
        CALL MESSAGA(7HPOSINT )
        IF(JMAX.LT.0) 300,400
  300   CONTINUE
        WRITE(6,40) J,JMAX
        WRITE(6,40) J,JMAX
C       ENDFILE 6
```

```
            WRITE(7,40) J,JMAX
C           ENDFILE 7
   40       FORMAT(*IUCALC ERROR IN POSINT. J= *,I3,* JMAX= *,I3)
            STOP 15
  400       CONTINUE
C           WRITE(8,45) J,JMAX
C           WRITE(6,45) J,JMAX
C           ENDFILE 6
            WRITE(7,45) J,JMAX
C           ENDFILE 7
   45       FORMAT(*J= *,I3,*JMAX= *,I3)
C
C           JMAX.EQ.0 WHEN THE INTERSECTION IS EMPTY, OR THE INPUT POLYGONS TO
C           IUCALC ARE IDENTICAL. THE ASSUMPTION HERE IS THAT THE INPUT
C           POLYGONS ARE NEVER IDENTICAL.
C
            IF(JMAX.EQ.0) GOTO 900
C
C           A NONEMPTY INTERSECTION RESULTS.
C           (XWO,YWO) CONTAINS THE POSITON POLYGONS DETERMINED BY THE INTERACTION
C           OF (XPC,YPC) WITH THE LAND MASSES (LAX,LAY).  IREM IS THE INDEX OF THE
C           OPTIMAL SET OF REMAINDER POLYGONS.  MAXPOL IS THE NUMBER IN THE SET.
C
            IF(PP*PC.GT.0.) 600,700
  600       CONTINUE
            CALL MESSAGC(7HPOSINT ,7HGOLDSEC)
            CALL GOLDSEC(XPC,YPC,IC1,LAX,LAY,LIA,LIB,LAN,XW,YW,IW,XS,YS,
     $        IS,RX,RY,IR,INC,LN,LE,LS,LW,CLOSE,FLAG,XWO,YWO,NUM,REM,
     $        IREM,INUM,AREAS)
            CALL MESSAGA(7HPOSINT )
            CALL MESSAGR(7HPOSINT )
            RETURN
  700       CONTINUE
            CALL MESSAGC(7HPOSINT ,7HREMAIN )
            CALL REMAIN(XS,YS,IS,LAX,LAY,LIA,LIB,LAN,XW,YW,IW,
     $        RX,RY,IP,LN,LE,LS,LW,1.,XPC,YPC,IC1,0.,0.,XWO,YWO,
     $        NUM,REM,1,AREAS,INUM)
            CALL MESSAGA(7HPOSINT )
            IREM=1
            CALL MESSAGR(7HPOSINT )
            RETURN
  900       CONTINUE
 1000     CONTINUE
C
C         XPC,YPC INTERSECTS NO LAND MASS.
C
            LANINT=.F.
C           WRITE(8,60)
C           WRITE(6,60)
C           ENDFILE 6
            WRITE(7,60)
C           ENDFILE 7
   60       FORMAT(*XPC,YPC INTERSECTS NO LAND MASS.*)
            CALL MESSAGR(7HPOSINT )
            RETURN
            END
          SUBROUTINE NUCUMUN(NGON1,NGON2,IA,IB,X,Y,
     $    XW,YW,XS,YS,IS,IW,XCEN,YCEN,WK,
     $    PGX,PGY,FLAG,RX,RY,IR1,JMAX)
C
C       GIVEN NGON1 AND NGON2, TWO POLYGONS REPRESENTING POSITIVE INFORMATION,
```

```
C     GENERATE PIAAM. START WITH A VERTEX KNOWN TO BE ON BOUNDARY OF PIAAM,
C     IN THIS CASE THE MOST NORTH VERTX OF NGON1 EVALUATED AT THE MOST
C     NORTH VERTEX OF NGON2.  THE NEXT VERTEX ON THE BOUNDARY OF PIAAM IS
C     USUALLY EITHER THE NEXT VERTEX OF NGON1 EVALUATED AT THE SAME VERTEX
C     OF NGON2, OR THE SAME VERTEX OF NGON1 EVALUATD AT THE NEXT VERTEX
C     OF NGON2.  CHOOSE THE FORMER WHEN IT IS TO THE LEFT OF THE LATTER,
C     OTHERWISE CHOOSE THE LATTER.  TO ALLOW FOR THE CASES WHEN NEITHER
C     ABOVE VERTEX IS NEXT, UNION TO THE POLYGON FORMED BY THE ABOVE
C     NGON1 EVALUATD AT ALL VERTICES OF NGONN2.
C
      LOGICAL FLAG,FLAG1
      DIMENSION X(1),Y(1),IA(1),IB(1),XW(1),YW(1),IW(2,1),XS(1),
     $  YS(1),PX(50),PY(50),WK(1),PGX(1),PGY(1),IPAR(2,1),
     $  PARX(3),PARY(3),RX(1),RY(1),IS(2,1),XCEN(1),YCEN(1)
C
C     PGX,PGY CONTAINS X,Y COORDINATES OF NGON2.  M2 IS THE NUMBER OF
C     DIFFERENT VERTICES OF NGON2. RX,RY CONTAINS THE POLYGON DETERMINED
C     THE "USUALLY THE NEXT VERTEX" ALGORITHM, IR1 IS THE NUMBER
C     OF ENTRIES IN RX,RY.  THE FINAL NUCUMUN RESULT IS STORED IN RX,RY.
C
      CALL MESSAGA(7HNUCUMUN)
      DATA IPAR(1,1),IPAR(2,1)/0,3/
      IA1=IA(NGON1)
      IB1=IB(NGON1)
      M1=IB1-IA1
      M2=IB(NGON2)-IA(NGON2)
      EP=1.
      E=.001
      FLAG=.T.
C
C     DETERMINE MOST NORTH VERTEX OF PIAAM.
C
      CALL MESSAGC(7HNUCUMUN,7HNORTH  )
      NONGON2=NON=NORTH(1,M2,PGY)
      NONGON1=NORTH(IA1,IB1-1,Y)
      I=1
      XC=XCEN(NGON1)
      YC=YCEN(NGON1)
      WRITE(7,5) XC,YC
    5 FORMAT(*THE CENTROID OF NGON1 IS *,2G15.5)
      XC1=PGX(NONGON2)-XC
      YC1=PGY(NONGON2)-YC
      XC2=PGX(NONGON2+1)-XC
      YC2=PGY(NONGON2+1)-YC
      RX(I)=ROU(X(NONGON1)+XC1)
      RY(I)=ROU(Y(NONGON1)+YC1)
      WRITE(7,10) I,RX(I),RY(I)
C     ENDFILE 7
   10 FORMAT(*THE *,I3,* ENTRY IN RX,RY IS *,2G15.5)
C
C     DETERMINE CANDIDATES FOR NEXT VERTEX.
C
      MX=(M1+1)*(M2+1)
      DO 2999 I=2,MX
      PARX(1)=RX(I-1)
      PARY(1)=RY(I-1)
C
C     DETERMINE THE NEXT VERTEX OF NGON1 EVALUATED AT THE VERTEX OF
C     NGON2 ASSOCIATED WITH RX(I-1),RY(I-1).
C
      NEXT1=NONGON1+1
```

```
      IF(NEXT1.GT.IB1) NEXT1=IA1+MOD(NEXT1-IA1,IB1-IA1)
      WRITE(7,15) NEXT1
   15 FORMAT(*NEXT1= *,I3)
C     ENDFILE 7
      PARX(2)=ROU(X(NEXT1)+XC1)
      PARY(2)=ROU(Y(NEXT1)+YC1)
      WRITE(7,20) PARX(2),PARY(2)
   20 FORMAT(*PARX(2),PARY(2) IS *,2G15.5)
C     ENDFILE 7
C
C     DETERMINE THE SAME VERTEX OF NGON1 EVALUATED AT THE NEXT VERTEX OF
C     NGON2.
C
      PARX(3)=ROU(X(NONGON1)+XC2)
      PARY(3)=ROU(Y(NONGON1)+YC2)
      WRITE(7,30) PARX(3),PARY(3)
   30 FORMAT(*PARX(3),PARY(3) IS *,2G15.5)
C     ENDFILE 7
      A=AREA(PARX,PARY,IPAR,1)
      IF(ABS(A).GT.E) GOTO 1000
      F2X=PARX(2)-RX(I-1)
      F2Y=PARY(2)-RY(I-1)
      F3X=PARX(3)-RX(I-1)
      F3Y=PARY(3)-RY(I-1)
      D2=SQRT(F2X*F2X+F2Y*F2Y)
      D3=SQRT(F3X*F3X+F3Y*F3Y)
      IF(D2.GE.D3+E) 1500,2000
 1000 CONTINUE
      IF(A.GE.E) 1500,2000
 1500 CONTINUE
C
C     NEXT VERTEX IS (PARX(2),PARY(2)).  THE NEXT VERTEX OF NGON1 EVALUATED AT
C     THE VERTEX OF NGON2 ASSOCIATED WITH RX(I-1),RY(I-1).
C
      RX(I)=PARX(2)
      RY(I)=PARY(2)
      NONGON1=NEXT1
      WRITE(7,10) I,RX(I),RY(I)
      IF((ABS(RX(I)-RX(1)).LE.EP).AND.(ABS(RY(I)-RY(1)).LE.EP))
     $  GOTO 3000
      GOTO 2999
 2000 CONTINUE
C
C     NEXT VERTEX IS (PARX(3),PARY(3)).  THE SAME VERTEX OF NGON1 EVALUATED AT
C     THE NEXT VERTEX OF NGON2.
C
      RX(I)=PARX(3)
      RY(I)=PARY(3)
      WRITE(7,10) I,RX(I),RY(I)
      IF((ABS(RX(I)-RX(1)).LE.EP).AND.(ABS(RY(I)-RY(1)).LE.EP))
     $  GOTO 3000
      NONGON2=NONGON2+1
      IF(NONGON2.GT.M2) NONGON2=MOD(NONGON2,M2)
      WRITE(7,35) NONGON2
   35 FORMAT(*NONGON2= *,I3)
C     ENDFILE 7
      XC1=PGX(NONGON2)-XC
      YC1=PGY(NONGON2)-YC
      XC2=PGX(NONGON2+1)-XC
      YC2=PGY(NONGON2+1)-YC
 2999 CONTINUE
```

```
      FLAG=.F.
      WRITE(6,40)
      WRITE(7,40)
      WRITE(8,40)
   40 FORMAT(*NUCUMUN ERROR *)
      CALL MESSAGR(7HNUCUMUN)
      RETURN
 3000 CONTINUE
C
C     UNION TO THE POLYGON FORMED BY THE ABOVE OPERATIONS NGON1 EVALUATED
C     AT ALL VERTICES OF NGON2.
C
      IR1=I-1
      WRITE(7,50) IR1,(RX(I),RY(I),I=1,IR1)
   50 FORMAT(*RX,RY HAS *,I3,* ENTRIES.  (RX,RY) IS *,/5(G13.5,G13.5))
C     ENDFILE 7
      DO 5000 J=1,M2
      XC1=PGX(J)-XC
      YC1=PGY(J)-YC
      DO 3500 I=1,M1
      IA11=IA1+I-1
      PX(I)=ROU(X(IA11)+XC1)
      PY(I)=ROU(Y(IA11)+YC1)
 3500 CONTINUE
      CALL MESSAGC(7HNUCUMUN,7HPREIU  )
      SUB=7HNUCUMUN
      CALL PREIU(RX,RY,IR1,PX,PY,M1,1,IS,XS,YS,SUB,FLAG,FLAG1,JMAX)
      CALL MESSAGA(7HNUCUMUN)
      IF(.NOT.FLAG) RETURN
 5000 CONTINUE
      WRITE(7,60)
   60 FORMAT(*THE FINAL OUTPUT OF NUCUMUN IS *)
      WRITE(7,50) IR1,(RX(I),RY(I),I=1,IR1)
C     ENDFILE 7
      CALL MESSAGR(7HNUCUMUN)
      RETURN
      END
      FUNCTION NORTH(I1,I2,Y)
C
C     NORTH,GIVEN POLYGON (I1,I2,X,Y), RETURNS THE INDEX OF THE
C     MOST NORTH VERTEX OF THE POLYGON.  THE NUMBER OF DISTINCT
C     VERTICES IN THE POLYGON IS I2-I1+1.
C
      DIMENSION Y(1)
      REAL N
      CALL MESSAGA(7HNORTH  )
      EP=0.0001
      N=Y(I1)
      NORTH=I1
      WRITE(7,10) (Y(I),I=I1,I2)
   10 FORMAT(*INPUT TO NORTH IS *,/5(G13.6,G13.6))
      I11=I1+1
      DO 1000 I=I11,I2
      PY=Y(I)
      IF(PY.LT.N+EP) GOTO 1000
      N=PY
      NORTH=I
 1000 CONTINUE
      WRITE(7,20) NORTH,N
   20 FORMAT(*NORTH OUTPUT.  NORTH=*,I3,* N=*,G15.5)
      CALL MESSAGR(7HNORTH  )
```

```
      SUBROUTINE GOLDSEC(XPC,YPC,IC1,LAX,LAY,LIA,LIB,LAN,XW,YW,
     $   IW,XS,YS,IS,RX,RY,IR,INC,LN,LE,LS,LW,CLOSE,FLAG,
     $   XWO,YWO,NUM,REM,IREM,INUM,AREAS)
C
C     GOLDSEC USES THE METHOD OF GOLDEN SECTIONS TO GENERATE THE RESCALED
C     POLYGON Q FROM XPC,YPC WHERE THE AREA OF Q OUTSIDE OF THE LAND MASSES
C     IS CLOSE TO THE ENTIRE AREA OF XPC,YPC. THE PARTS OF
C     Q OUTSIDE OF THE LAND MASSES ARE CALLED THE REMAINDER POLYGONS
C     AND  ARE  RETURNED IN XWO,YWO.  IREM IS THE INDEX OF THE FINAL
C     SET OF REMAINDER POLYGONS. INUM(IREM) IS THE INDEX OF THE LARGEST
C     POLYGON IN SET IREM.
C     NUM(1,J) IS THE OFFSET OF THE INDEX OF THE JTH POLYGON IN XWO,YWO.
C     NUM(2,J) IS THE NUMBER OF VERTICES IN POLYGON J.
C     REM(1,K) IS THE OFFSET OF THE INDEX OF THE KTH REMAINDER SET IN NUM.
C     REM(2,K) IS THE NUMBER OF POLYGONS IN REMAINDER SET K.
C     POINT(I) IS USUALLY THE REMAINDER SET ASSOCIATED WITH WI,I=1 OR 2.
C     A IS THE AREA OF XPC,YPC.
C     AREAS(K) IS THE AREA OF THE LARGEST POLYGON IN REMAINDER SET K.
C     WHEN  -A*CLOSE @ AREAS(K)-A @ A*CLOSE, THE ALGORITHM STOPS WITH THE
C     REMAINDER SET K AND THE REMAINDER POLYGON INUM(K).
C     IN THE INITIALIZING PHASE OF THE ALGORITHM, SCALE=1. AND THE AREAS(1)
C     IS SUCH THAT AREAS(1) < A.
C     FURTHERMORE, IN THIS PHASE THE ALGORITHM MUST FIND A QUANTITY, NEXT,
C     S.T. THE CORRESPONDING AREAS SATISFIES THE STOPPING CRITERIA OR
C     A*CLOSE< AREAS-A.
C     GOLDSEC IS CALLED BY POSINT.
C
      REAL LAX(1),LAY(1),NUSCALE,NEXT,LENGTH,INC,LN(1),LE(1),LS(1),
     $   LW(1)
      INTEGER REM(2,1),POINT(2)
      DIMENSION XPC(1),YPC(1),LIA(1),LIB(1),INUM(1),
     $   XS(1),YS(1),IS(2,1),RX(1),RY(1),IR(2,1),
     $   XW(1),YW(1),IW(2,1),
     $   XWO(1),YWO(1),NUM(2,1),AREAS(1)
      LOGICAL FLAG
      CALL MESSAGA(7HGOLDSEC)
      CALL MESSAGC(7HGOLDSEC,7HCENTRD )
      CALL CENTRD(1,IC1,XPC,YPC,A,AX,AY,AXX,AYY,AXY)
      CALL MESSAGA(7HGOLDSEC)
      ACLOS=A*CLOSE
      XCEN=AX/A
      YCEN=AY/A
      WRITE(6,8) XCEN,YCEN,ACLOS
C     ENDFILE 6
      WRITE(7,8) XCEN,YCEN,ACLOS
C     ENDFILE 7
    8 FORMAT(*XCEN=*,G15.5,*YCEN=*,G15.5,*ACLOS=*,G15.5)
      FLAG=.T.
      SCALE=1.
      IS(1,1)=0.
      EP=0.00001
      KAY=5
C     THE LOWER BOUND OF THE SEARCH INTERVAL IS 1. STOP OR
C     DETERMINE UPPER BOUND OF THE SEARCH INTERVAL SO THAT THE
C     AREA OF THE
C     REMAINDER POLYGON(S),GENERATED BY THE UPPER BOUND, EXCEEDS THE AREA
C     OF XPC,YPC.
C
      LL=0
      CALL MESSAGC(7HGOLDSEC,7HREMAINU)
      LL=LL+1
```

```
      CALL REMAIN(XS,YS,IS,LAX,LAY,LIA,LIB,LAN,XW,YW,IW,RX,RY,IR,
     $   LN,LE,LS,LW,SCALE,XPC,YPC,IC1,XCEN,YCEN,
     $   XWO,YWO,NUM,REM,LL,AREAS,INUM)
      CALL MESSAGA(7HGOLDSEC)
      WRITE(7,13) A,AREAS(LL)
      IF(A-ACLOS.LE.AREAS(LL)) 50,100
   50 CONTINUE
      IREM=LL
      CALL MESSAGR(7HGOLDSEC)
      RETURN
  100 CONTINUE
      DO 1000 K=1,KAY
      NEXT=NUSCALE(SCALE,INC)
      CALL MESSAGC(7HGOLDSEC,7HREMAINU)
C
C
C
      LL=LL+1
      CALL REMAIN(XS,YS,IS,LAX,LAY,LIA,LIB,LAN,XW,YW,IW,RX,RY,IR,
     $   LN,LE,LS,LW,NEXT,XPC,YPC,IC1,XCEN,YCEN,
     $   XWO,YWO,NUM,REM,LL,AREAS,INUM)
      CALL MESSAGA(7HGOLDSEC)
      GW1=A-AREAS(LL)
      WRITE(7,13) A,AREAS(LL),GW1
C     ENDFILE 7
   13 FORMAT(*INPUT POLYGON AREA=*,G15.5,* REMAINDER POLYGON AREA=*,
     $   G15.5,* INPUT AREA - REMAINDER AREA=*,G15.5)
      IF(ABS(GW1).LE.ACLOS) 500,600
  500 CONTINUE
      IREM=LL
      CALL MESSAGR(7HGOLDSEC)
      RETURN
  600 CONTINUE
C
C     CHECK TO SEE IF NEXT IS SUCH THAT METHOD OF GOLDEN SECTIONS CAN BE
C     APPLIED
      IF(A+ACLOS.LE AREAS(LL)) GOTO 1100
      SCALE=NEXT
 1000 CONTINUE
      WRITE(6,15)
C     ENDFILE 6
      WRITE(8,15)
   15 FORMAT(*AN UPPER BOUND OF THE SEARCH INTERVAL HAS NOT BEEN * ,
     $ *DETERMINED. RETURN CONTROL TO FUNCTION SELECTION  (Y OR N) *)
      READ(5,20) Q
      WRITE(6,20) Q
      WRITE(7,20) Q
      WRITE(8,20) Q
   20 FORMAT(A1,I2)
      IF(Q.EQ.1HY) FLAG=.F.
      IF(.NOT.FLAG) RETURN
      WRITE(6,25)
      WRITE(6,27)
C     ENDFILE 6
      WRITE(8,25)
      WRITE(8,27)
   25 FORMAT(*CALCULATION OF THE UPPER BOUND OF THE SEARCH INTERVAL*,
     $ * WILL CONTINUE.*)
   27 FORMAT(*CHANGE SCALE FACTOR INCREMENT  (Y OR N) *,
     $ * FOLLOW Y IMMEDIATELY WITH 2 DIGIT POSITIVE INTEGER.*)
      READ(5,20) Q,NUINC
```

```
      WRITE(6,20) Q,NUINC
      WRITE(7,20) Q,NUINC
      WRITE(8,20) Q,NUINC
      IF(Q.EQ.1HY) INC=NUINC
      GOTO 100
 1100 CONTINUE
C
C     THE METHOD OF GOLDEN SECTIONS APPLIED TO THE SEARCH INTERVAL
C     [SCALE, NEXT] FOLLOWS. SEE ZANGWILL P. 121.
C
      F1=(3.-SQRT(5.))/2.
      F2=(SQRT(5.)-1.)/2.
C
C     A FUNCTION EVALUATION CONSISTS OF THE FOLLOWING STEPS. GENERATE THE
C     RESCALED POLYGON Q(W), WHERE W=W1 OR W2. DETERMINE THE REMAINDER
C     POLYGONS OF Q(W). DETERMINE THE AREA OF THE REMAINDER POLYGONS.
      KK=0
 1150 CONTINUE
      DO 3000 K=1,KAY
      KK=KK+1
      IF(KK.EQ.1) 1200, 1500
C
C     ON THE FIRST ITERATION THE PROGRAM BRANCHES TO THE FOLLOWING,  IN ORDER
C     TO INITIALIZE THE ALGORITHM.
C
 1200 CONTINUE
      GW1=GW2=0.
      FL=F1*(NEXT-SCALE)
      W1=SCALE+FL
      W2=NEXT-FL
      CALL MESSAGC(7HGOLDSEC,7HREMAING)
      LL=LL+1
      CALL REMAIN(XS,YS,IS,LAX,LAY,LIA,LIB,LAN,XW,YW,IW,RX,RY,IR,
     $   LN,LE,LS,LW,W1,XPC,YPC,IC1,XCEN,YCEN,
     $   XWO,YWO,NUM,REM,LL,AREAS,INUM)
      CALL MESSAGA(7HGOLDSEC)
      GW1=ABS(A-AREAS(LL))
      WRITE(7,40)A,AREAS(LL),GW1
C     ENDFILE 7
C
C     CHECK FOR OPTIMALITY.
C
      IF(GW1.LE.ACLOS) 1300,1400
 1300 CONTINUE
C
C     LL IS OPTIMAL
C
      IREM=LL
      CALL MESSAGR(7HGOLDSEC)
      RETURN
 1400 CONTINUE
C
C     LL IS NOT OPTIMAL
C
      POINT(1)=LL
      LL=LL+1
      POINT(2)=LL
      CALL MESSAGC(7HGOLDSEC,7HREMAING)
      CALL REMAIN(XS,YS,IS,LAX,LAY,LIA,LIB,LAN,XW,YW,IW,RX,RY,IR,
     $   LN,LE,LS,LW,W2,XPC,YPC,IC1,XCEN,YCEN,
     $   XWO,YWO,NUM,REM,LL,AREAS,INUM)
```

```
      CALL MESSAGA(7HGOLDSEC)
   40 FORMAT(*INPUT POLYGON AREA=*,G15.5,* REMAINDER POLYGON AREA=*,
     $   G15.5,* ABS(INPUT AREA - REMAINDER AREA)=*,G15.5)
      GW2=ABS(A-AREAS(LL))
      WRITE(7,40) A,AREAS(LL),GW2
C     ENDFILE 7
C     CHECK FOR OPTIMALITY. IF NOT OPTIMAL INCREMENT K COUNTER
      IF(GW2.LE.ACLOS) 1450,2900
 1450 CONTINUE
C
C     LL IS OPTIMAL
C
      IREM=LL
      CALL MESSAGR(7HGOLDSEC)
      RETURN
C
C     ON ALL BUT THE FIRST ITERAION THE PROGRAM BRANCHES TO THE FOLLOWING
C     FROM LABEL 1150.
C
 1500 CONTINUE
C
C     THE FOLLOWING IS THE GENERAL GOLDEN SECTION ITERATION.
C
C     DETERMINE THE NEXT SEARCH INTERVAL.
      IF(GW2-EP.LE.GW1) 1600,2000
 1600 CONTINUE
C     THE NEW INTERVAL IS [W1,NEXT].
C     GENERATE DATA FOR THE NEW W2. THE NEW W1= THE OLD W2, SO THE DATA
C     ASSOCIATED WITH THE NEW W1 = DATA ASSOCIATED WITH OLD W2.
C
      SCALE=W1
      GW1=GW2
      POINT(1)=POINT(2)
      W1=W2
      W2=NEXT-F1*(NEXT-SCALE)
      CALL MESSAGC(7HGOLDSEC,7HREMAING)
      LL=LL+1
      CALL REMAIN(XS,YS,IS,LAX,LAY,LIA,LIB,LAN,XW,YW,IW,RX,RY,IR,
     $   LN,LE,LS,LW,W2,XPC,YPC,IC1,XCEN,YCEN,
     $   XWO,YWO,NUM,REM,LL,AREAS,INUM)
      CALL MESSAGA(7HGOLDSEC)
      GW2=ABS(A-AREAS(LL))
      WRITE(7,40) A,AREAS(LL),GW2
C     ENDFILE 7
C     CHECK FOR OPTIMALITY  IF NOT OPTIMAL, UPDATE POINT AND INCREMENT K.
      IF(GW2.LE.ACLOS) 1700,1800
 1700 CONTINUE
      IREM=LL
      CALL MESSAGR(7HGOLDSEC)
      RETURN
 1800 CONTINUE
      POINT(2)=LL
      GOTO 2900
C
C     WHEN GW2-EP.GT.GW1 THE PROGRAM BRANCHES TO THE FOLLOWING FROM LABEL
C     1500. THE NEW INTERVAL IS [SCALE,W2].
C           GENERATE DATA FOR THE NEW W1. THE NEW W2 = THE OLD W1, SO
C     THE DATA ASSOCIATED WITH THE NEW W2 = DATA ASSOCIATED WITH OLD W1.
C
 2000 CONTINUE
      NEXT=W2
```

```
      GW2=GW1
      POINT(2)=POINT(1)
      W2=W1
      W1=SCALE+F1*(NEXT-SCALE)
      CALL MESSAGC(7HGOLDSEC,7HREMAING)
      LL=LL+1
      CALL REMAIN(XS,YS,IS,LAX,LAY,LIA,LIB,LAN,XW,YW,IW,RX,RY,IR,
     $   LN,LE,LS,LW,W1,XPC,YPC,IC1,XCEN,YCEN,
     $   XWO,YWO,NUM,REM,LL,AREAS,INUM)
      CALL MESSAGA(7HGOLDSEC)
      GW1=ABS(A-AREAS(LL))
      WRITE(7,40) A,AREAS(LL),GW1
C     ENDFILE 7
C     CHECK FOR OPTIMALITY, IF NOT OPTIMAL,UPDATE POINT AND INCREMENT K.
      IF(GW1.LE.ACLOS) 2200,2300
 2200 CONTINUE
      IREM=LL
      CALL MESSAGR(7HGOLDSEC)
      RETURN
 2300 CONTINUE
      POINT(1)=LL
 2900 CONTINUE
 3000 CONTINUE
C     END OF DO LOOP 3000.
      G11=AREAS(POINT(1))/A
      G22=AREAS(POINT(2))/A
      WRITE(6,60) W1,G11,W2,G22
      WRITE(7,60) W1,G11,W2,G22
C     ENDFILE 6
      WRITE(8,60) W1,G11,W2,G22
   60 FORMAT(*AT W1=*,G15.5,* THE REMAINDER AREA IS *,G15.5,* OF A.*,
     $   *AT W2=*,G15.5,* THE REMAINDER AREA IS *,G15.5,* OF A.*)
      WRITE(6,70)
C     ENDFILE 6
      WRITE(8,70)
   70 FORMAT(*ACCEPT W1 OR W2 AS OPTIMAL  (Y OR N). FOLLOW Y*,
     $   *IMMEDIATELY BY 01 FOR W1, 02 FOR W2*)
      READ(5,20) Q,NUINC
      WRITE(6,20) Q,NUINC
      WRITE(7,20) Q,NUINC
      WRITE(8,20) Q,NUINC
      IF(Q.EQ.1HN) GOTO 4000
      IF(NUINC.EQ.01) 3300,3400
 3300 CONTINUE
      IREM=POINT(1)
      CALL MESSAGR(7HGOLDSEC)
      RETURN
 3400 CONTINUE
      IREM=POINT(2)
      CALL MESSAGR(7HGOLDSEC)
      RETURN
 4000 CONTINUE
      WRITE(6,80)
C     ENDFILE 6
      WRITE(8,80)
   80 FORMAT(*AREA OF THE REMAINDER POLYGON(S) IS NOT CLOSE TO *,
     $      *AREA OF THE INPUT POLYGON,XPC,YPC. RETURN CONTROL TO *,
     $ *FUNCTION SELECTION (Y OR N)*)
      READ(5,20) Q
      WRITE(6,20) Q
      WRITE(7,20) Q
```

```
      WRITE(8,20) Q
      IF(Q.EQ.1HY) FLAG=.F.
      IF(.NOT.FLAG) RETURN
      WRITE(6,90)
C     ENDFILE 6
      WRITE(3,90)
   90 FORMAT(*THE GOLDEN SECTION ITERATIONS WILL CONTINUE*)
      GOTO 1150
      RETURN
      END
      SUBROUTINE INTO(XS,YS,IS,JMIN,JMAX,XWO,YWO,NUM,REM,LL,MAXPOL,IP,
     $   IL)
C
C     INTO INSERTS THE CONTENTS OF (XS(IS(1,JO)+I),YS(IS(1,JO)+I)),
C     I=1,IS(2,JO), JO=JMIN,JMAX, IN (XWO(NUM(1,KO)+I),YWO(NUM(1,KO)+I)),
C     I=1,NUM(2,KO), KO=REM(1,LL)+MAXPOL+1,REM(1,LL)+MAXPOL+JMAX-JMIN+1,
C     WHERE NUM(2,KO)=IS(2,JO). SEE COMMENTS OF REMAIN FOR DEFINITIONS OF
C     REM AND NUM. WHEN INTO IS CALLED, MAXPOL IS THE NUMBER OF POLYGONS IN
C     POSITION POLYGONS 1 TO IP-1 BUT NOT IN LAND 1 TO IL.
C     THUS MAXPOL=REM(2,LL) AT ITERATION IP-1.
C
      DIMENSION XS(1),YS(1),IS(2,1),XWO(1),YWO(1),NUM(2,1)
      INTEGER REM(2,1),REM1,REM11
      CALL MESSAGA(7HINTO   )
C
C     A DATA STATEMENT IN MOVMENT INITIALIZES REM(1,1)=NUM(1,1)=0.
C     REM(1,LL+1) IS FIXED WHEN THE FINAL NUMBER OF POLYGONS IN SET LL IS
C     KNOWN. THUS AT THE END OF THIS ROUTINE REM(1,LL+1) IS COMPUTED.
C     REM(2,LL) VARIES AT EACH CALL TO INTO. REM(2,LL) SHOULD BE SET TO 0
C     EACH TIME IL INCREASES. EACH TIME IL INCREASES IP IS RESET TO 1
C     AND MAXPOL IS RESET TO 0, THUS IP=1 IS THE TEST TO SET REM(2,LL) TO 0.
C     AS IL IS CONSTANT, AND EACH TIME IP INCREASES (MAXPOL IS NONZERO HERE)
C     REM(2,LL) INCREASES BY JMAX EACH TIME INTO IS CALLED.
C     NUM(1,REM(1,LL+1)+1) IS FIXED WHEN THE FINAL NUMBER OF VERTICES IN SET
C     LL IS KNOWN. THUS NUM(1,REM(1,LL+1)+1) IS COMPUTED AT THE END OF THIS
C     ROUTINE.
C
C
      REM1=REM(1,LL)
      IF(IP.EQ.1) REM(2,LL)=0
      REM(2,LL)=REM(2,LL)+JMAX-JMIN+1
C
C     KO IS THE NUMBER OF THE POLYGON CURRENTLY BEING TRANSFERRED TO
C     (XWO,YWO).
C
      KO=REM1+MAXPOL
      WRITE(7,30)
   30 FORMAT(*THIS OUTPUT IS FROM ROUTINE INTO.*)
      WRITE(7,40) LL,IL,IP,MAXPOL,JMIN,JMAX,LL,REM(1,LL),LL,REM(2,LL)
C     ENDFILE 7
   40 FORMAT(*LL=*,I3,* IL=*,I3,* IP=*,I3,* MAXPOL=*,I3,* JMIN=*,I3,
     $   * JMAX=*,I3,* REM(1,*,I3,*)=*,I3,* REM(2,*,I3,*)=*,I3)
      DO 3000 JO=JMIN,JMAX
        KO=KO+1
        IS1=IS(1,JO)
C       WRITE(7,45) IS(1,JO)
C  45   FORMAT(*IS1=IS(1,JO)=*,I8)
        NUM1=NUM(1,KO)
        NUM(2,KO)=IS2=IS(2,JO)
C  47   FORMAT(*IS2=IS(2,JO)=*,I8)
C       WRITE(7,47) IS(2,JO)
```

```
C           WRITE(7,48) (XS(IS1+I),YS(IS1+I),I=1,IS2)
C  48       FORMAT(*(XS,YS) IS*,/(5(G13.5,G13.5)))
            NUM(1,KO+1)=NUM(1,KO)+IS2
            DO 2000 I=1,IS2
              XWO(NUM1+I)=XS(IS1+I)
              YWO(NUM1+I)=YS(IS1+I)
 2000       CONTINUE
            WRITE(7,50)  JO,KO,NUM(1,KO),KO,NUM(2,KO)
   50       FORMAT(*JO= *,I3,*NUM(1,*,I3,*)= *,I3,*NUM(2,*,I3,*)= *,I3,
     $        *THE CONTENTS OF XWO,YWO ASSOCIATED WITH THESE INDICES ARE*)
            WRITE(7,60) (XWO(NUM1+I),YWO(NUM1+I),I=1,IS2)
C           ENDFILE 7
   60       FORMAT( 5(G13.6,G13.6))
 3000     CONTINUE
C
C         COMPUTE REM(1,LL+1),NUM(1,REM(1,LL+1)+1),MAXPOL
C
          REM(1,LL+1)=REM11=REM(1,LL)+REM(2,LL)
          NUM(1,REM11+1)=NUM(1,REM11)+NUM(2,REM11)
          MAXPOL=MAXPOL+JMAX-JMIN+1
          CALL MESSAGR(7HINTO    )
          RETURN
          END
        SUBROUTINE MOVECEN(M,N,JA,JB,X,Y,IA,IB,P,SE,KE,KLA,XW,YW,J,ILA,
       $  JLA,XCEN,YCEN,H,LAX,LAY,LIA,LIB,LAN,LAND,LN,LE,LS,LW,INPUT,
       $  COUNT,FLAG)
C       MOVECEN IS CALLED BY POL  THE
C       VELOCITY DISTRIBUTION IS MOVED TO TIME M+1 ACCORDING TO THE POSITION
C       DISTRIBUTION AT TIME M+1 AND MK. LAND INTERACTIONS ARE CONSIDERED. A MOVED
C       VELOCITY POLYGON WHOSE CENTROID IS CONTAINED IN A LAND MASS IS REPLACED BY
C       TWO VELOCITY POLYGONS. MOREOVER THE CENTROIDS OF THE
C       REPLACEMENT POLYGONS ARE CONTAINED IN NO LAND MASS.
        DIMENSION JA(2,5,9),JB(2,5,9),X(1),Y(1),IA(1),IB(1),P(1),
       $  SE(1),H(1),KE(2,5,9),XW(1),YW(1),XCEN(1),YCEN(1),VX(2),VY(2),
       $  CPROB(2),LIA(1),LIB(1)
        REAL LAX(1),LAY(1),LN(1),LE(1),LS(1),LW(1)
        INTEGER COUNT,ENSYD2,ENS
        LOGICAL LAND,INPUT(2,5,9),FLAG,MINIMAX
        CALL MESSAGA(7HMOVECEN)
        WRITE(6,10)
        WRITE(8,10)
    10  FORMAT(*$LIN 2*)
        M1=M+1
C
C       FIND,MK,  THE MOST RECENT INPUT CENTROID DISTRIBUTION IN ORDER TO
C       MAINTAIN VELOCITY TENDENCIES.
C
        MK=M
        FLAG=.T.
        IF((.NOT.LAND).OR.(INPUT(2,M,N))) GOTO 300
        DO 200 ME=2,M
        MK=M-ME+1
        IF(INPUT(2,MK,N)) GOTO 300
   200  CONTINUE
        WRITE(6,15)
        WRITE(7,15)
C       ENDFILE 6
C       ENDFILE 7
        WRITE(8,15)
    15  FORMAT(*VELOCITY DISTRIBUTION IS UNDEFINED*)
        FLAG=.F.
```

```
      IF(.NOT.FLAG) RETURN
  300 CONTINUE
      SE1=-SE(1+KE(1,MK,N))+SE(1+KE(1,M1,N))
      SE2=-SE(2+KE(1,MK,N))+SE(2+KE(1,M1,N))
      JJ=0
      JA2=JA(2,MK,N)
      JB2=JB(2,MK,N)
      DO 4000 J1=JA2,JB2
      IA1=IA(J1)
      IB1=IB(J1)
      IW1=IB1-IA1+1
      IF(.NOT.LAND) GOTO 3100
      XO=XCEN(J1)+SE1
      YO=YCEN(J1)+SE2
      DO 3000 IL=1,LAN
      WRITE(6,20) J1,IL
C     ENDFILE 6
      WRITE(7,20) J1,IL
C     ENDFILE 7
      WRITE(8,20) J1,IL
   20 FORMAT(* MOVECEN CALLS MINIMAX*,2(I3))
      IF(MINIMAX(YO,XO,YO,XO,LN(IL),LE(IL),LS(IL),LW(IL)))
     $  GOTO 2900
C
C     TEST FOR (XO,YO) CONTAINED IN LAND MASS IL,
C     SINCE (XO,YO) IS CONTAINED IN THE RECTANGLE OF LAND MASS IL.
C     IF NOT, INCREASE IL.
C
      L1=LIA(IL)
      L2=LIB(IL)-1
      DO 400 I=L1,L2
      LO=I-L1+1
      XW(LO)=LAX(I)
      YW(LO)=LAY(I)
  400 CONTINUE
C     WRITE(8,25) J1,IL
C     WRITE(6,25) J1,IL
C     ENDFILE 6
      WRITE(7,25) J1,IL
C     ENDFILE 7
   25 FORMAT(* MOVECEN CALLS ENSYD2*,2(I3))
      ENS=ENSYD2(DBLE(XO),DBLE(YO),XW,YW,LO)
      WRITE(7,27) ENS
   27 FORMAT(*ENSYD2=*,I2)
      IF(ENS.EQ.0) GOTO 2900
C     WRITE(8,30) J1,IL
C     WRITE(6,30) J1,IL
C     ENDFILE 6
   30 FORMAT(*AFTER MOTION CENTROID OF *,I3,* IS IN LAND MASS *,
     $  I3)
C     COMPUTE REPLACEMENT POLYGONS.
C     COURSE CHANGE PROBABILITIES ADD TO 1. THESE PROBABILITES
C     DISTRIBUTE THE WEIGHT OF POLYGON J1 AMONG THE TWO VELOCITY POLYGONS
C     COMPUTED BY THE CODE BELOW TO LABEL 1000.
C
      WRITE(6,40)
C     ENDFILE 6
      WRITE(8,40)
   40 FORMAT(*PROBABILITY OF COURSE CHANGE--COUNTERCLOCKWISE  *,
     $ /*  F10.5*)
      READ(5,50) CPROB(1)
```

```
      CPROB(2)=1.-CPROB(1)
   50 FORMAT(F10.5)
      WRITE(6,60) CPROB(1),CPROB(2)
C     ENDFILE 6
      WRITE(8,60) CPROB(1),CPROB(2)
   60 FORMAT(*COUNTERCLOCKWISE COURSE PROBABILITY= *,F10.5,
     $  * ,CLOCKWISE COURSE PROBABILITY= *,F10.5)
C
C     CALCULATE NEW VELOCITY POLYGONS
C
      WRITE(7,60) CPROB(1),CPROB(2)
C     ENDFILE 7
C
C     EACH VELOCITY POLYGON, BEING COMPUTED, HAS CENTROID GIVEN BY
C     NEWCEN. (VX(1),VY(1)) CONTAINS THE COUNTERCLOCKWISE (OR LEFT)
C     RESULTING CENTROID, (VX(2),VY(2)) CONTAINS THE CLOCKWISE (OR RIGHT)
C     RESULTING CENTROID. (VX,VY) ARE COMPUTED TO BE CONTAINED IN NO LAND
C     MASS  AND CLOSE TO (XO,YO).
C
      CALL MESSAGC(7HMOVECEN,7HNEWCEN )
      CALL NEWCEN(XO,YO,LAX,LAY,LIA,LIB,LAN,LN,LE,LS,LW,IL,
     $  VX,VY,SE(1+KE(1,M1,N)),SE(2+KE(1,M1,N)),XW,YW,LO,COUNT,
     $  FLAG)
      CALL MESSAGA(7HMOVECEN)
      IF(.NOT.FLAG) RETURN
      DO 1000 K=1,2
      DO 800 I=IA1,IB1
        IU=I-IA1+1
        XW(IU)=X(I)-XCEN(J1)+VX(K)
        YW(IU)=Y(I)-YCEN(J1)+VY(K)
  800 CONTINUE
      CALL MESSAGC(7HMOVECEN,7HVEC     )
      CALL VEC(1,IW1,XW,YW)
      CALL MESSAGA(7HMOVECEN)
      JJ=JJ+1
      CALL MESSAGC(7HMOVECEN,7HPUT     )
      CALL PUT(2,M1,N,JJ,IW1,XW,YW,IA,IB,JA,JB,J,X,Y,ILA,JLA)
      CALL MESSAGA(7HMOVECEN)
      P(J)=P(J1)*CPROB(K)
 1000 CONTINUE
      GOTO 3900
 2900 CONTINUE
C
C     CENTROID OF NGON J1 IS NOT CONTAINED IN LAND MASS IL.
C
 3000 CONTINUE
C
C     CENTROID OF NGON J1 IS NOT CONTAINED IN ANY LAND MASS
C
 3100 CONTINUE
      DO 3500 I=IA1,IB1
      IU=I-IA1+1
      XW(IU)=X(I)+SE1
      YW(IU)=Y(I)+SE2
 3500 CONTINUE
      CALL MESSAGC(7HMOVECEN,7HVEC     )
      CALL VEC(1,IW1,XW,YW)
      CALL MESSAGA(7HMOVECEN)
      JJ=JJ+1
      CALL MESSAGC(7HMOVECEN,7HPUT     )
      CALL PUT(2,M1,N,JJ,IW1,XW,YW,IA,IB,JA,JB,J,X,Y,ILA,JLA)
```

```
      CALL MESSAGA(7HMOVECEN)
      P(J)=P(J1)
 3900 CONTINUE
 4000 CONTINUE
      CALL MESSAGC(7HMOVECEN,7HUPDATE )
      CALL UPDATE(M1,N,JA,JB,X,Y,IA,IB,P,XCEN,YCEN,J,ILA,JLA,SE,KE,
     $  H,KLA,FLAG,2)
      CALL MESSAGA(7HMOVECEN)
      WRITE(6,97)
C     ENDFILE 6
      WRITE(7,97)
C     ENDFILE 7
      WRITE(8,97)
   97 FORMAT(*$LIN 1*)
      CALL MESSAGR(7HMOVECEN)
      RETURN
      END
      SUBROUTINE NEWCEN(XO,YO,LAX,LAY,LIA,LIB,LAN,LN,LE,LS,LW,IL,VX,VY,
     $  SE1,SE2,XW,YW,LO,COUNT,FLAG)
C
C     NEWCEN ITERATIVELY CALCULATES CANDIDATES FOR NEW VELOCITY CENTROIDS. ALL
C     CANDIDATES ARE OF THE FORM (X(A),Y(A)) WHERE
C     X(A)=SE1+(XO-SE1)COS(A)+(YO-SE2)SIN(A)
C     Y(A)=SE2+(YO-SE2)COS(A)-(XO-SE1)SIN(A).
C     THE ALGORIHM STOPS WHEN A NEGATIVE A=A1 AND A POSITIVE A=A2 ARE FOUND
C     S.T. (X(A1),Y(A1)) AND (X(A2),Y(A2)) ARE CONTAINED IN NO LAND MASS,
C     OR WHEN TIME IS UP.
C     NEWCEN IS CALLED BY MOVECEN.
C
      DIMENSION LIA(1),LIB(1),VX(1),VY(1),XW(1),YW(1),XS(200),YS(200)
      REAL LAX(1),LAY(1),LN(1),LE(1),LS(1),LW(1)
      INTEGER TIME,COUNT,ENSYD2,ENS
      LOGICAL FLAG,FLAG1,MINIMAX
      DATA PI/3.141592653/
      CALL MESSAGA(7HNEWCEN )
      XOS=XO-SE1
      YOS=YO-SE2
      DELTA=PI/18.
      D2=DELTA
      FLAG=.T.
      FLAG1=.F.
      WRITE(7,10) XO,YO,SE1,SE2,XOS,YOS
   10 FORMAT(*XO= *,G15.5,* YO= *,G15.5,* SE1= *,G15.5,* SE2= *,G15.5,
     $* XOS= *,G15.5,* YOS= *,G15.5)
C     ENDFILE 7
      KAY=18
C
C     KK IS THE NEW CENTROID INDEX OF DIRECTION.
C     WHEN KK=1, THE COUNTERCLOCKWISE NEW CENTROID IS COMPUTED AND STORED IN
C     (VX(1),VY(1)). WHEN KK=2, THE CLOCKWISE NEW CENTROID IS COMPUTED AND
C     STORED IN (VX(2),VY(2)).
      DO 3000 KK=1,2
      IO=0
  190 CONTINUE
      TIME=0
C     TIME COUNTS THE ABSOLUTE NUMBER OF CANDIDATES FOR NEW CENTROID
C     GENERATED FOR FIXED KK.
  200 CONTINUE
      DO 2000 ITER=1,KAY
      TIME=TIME+1
      DELI=DELTA*ITER
```

```
      IF(IO.NE.0) DELI=D1+D2*ITER
      IF(KK.EQ.1) DELI=-DELI
      COSDEL=COS(DELI)
      SINDEL=SIN(DELI)
C
C     COMPUTE CANDIDATE FOR NEW CENTROID, (X1,Y1).
C
      X1=SE1+XOS*COSDEL+YOS*SINDEL
      Y1=SE2+YOS*COSDEL-XOS*SINDEL
      WRITE(7,15) TIME,ITER,DELI,X1,Y1
   15 FORMAT(*TIME= *,I3,* ITER= *,I3,* DELI= *,G15.5,* X1= *,G15.5,
     $* Y1= *,G15.5)
      CALL MESSAGC(7HNEWCEN ,7HENSYD2 )
C     ENDFILE 7
C
C     CHECK FOR (X1,Y1) IN LAND IL. IF SO, COMPUTE NEW (X1,Y1)
C
      ENS=ENSYD2(DBLE(X1),DBLE(Y1),XW,YW,LO)
      WRITE(7,17) ENS
   17 FORMAT(*ENSYD2=*,I2)
      IF(ENS.EQ.1) GOTO 1900
      FLAG1=.T.
C
C     (X1,Y1) IS NOT CONTAINED IN LAND MASS IL.
C     CHECK FOR (X1,Y1) CONTAINED IN OTHER LAND MASSES.
C
      DO 1000 L=1,LAN
C
C       WHEN L=IL, THE ANSWER 'S ALREADY KNOWN.
C
        IF(L.EQ.IL) GOTO 900
        CALL MESSAGC(7HNEWCEN ,7HMINIMAX)
C         CHECK FOR (X1,Y1) IN THE RECTANGLE OF LAND L. IF NOT INCREASE L.
          IF(MINIMAX(Y1,X1,Y1,X1,LN(L),LE(L),LS(L),LW(L))) GOTO 900
C         CHECK FOR (X1,Y1) IN LAND L. IF NOT INCREASE L
            L1=LIA(L)
            L2=LIB(L)
            DO 400 I=L1,L2
              IO=I-L1+1
              XS(IO)=LAX(I)
              YS(IO)=LAY(I)
  400       CONTINUE
            IO=L2-L1+1
            CALL MESSAGC(7HNEWCEN ,7HENSYD2 )
            ENS=ENSYD2(DBLE(X1),DBLE(Y1),XS,YS,IO)
            WRITE(7,17) ENS
            IF(ENS.EQ.0) GOTO 900
C
C           (X1,Y1) IS CONTAINED IN LAND MASS L. THE INTERVAL,[DELI-D2,DELI],HAS
C           THE PROPERTY (X(DELI-D2),Y(DELI-D2)) IS IN LAND IL, AND
C           (X(DELI),Y(DELI)) IS IN LAND L. THE INTERVAL IS SEARCHED TO FIND
C           (X1,Y1) CONTAINED IN NO LAND.
C
            IO=L
            D1=DELI-D2
            D2=D2/10.
            IF(TIME.LE.COUNT) GOTO 200
            WRITE(6,20)
C           ENDFILE 6
            WRITE(8,20)
   20       FORMAT(* THE NUMBER OF ALLOWED ITERATIONS IS EXCEEDED.*,
```

```
     $        *RETURN CONTROL TO FUNCTION SELECTION  (Y OR N)*)
              READ(5,25) Q
   25         FORMAT(A1)
              IF(Q.EQ.1HY) FLAG=.F.
              IF(.NOT.FLAG) RETURN
              WRITE(6,30)
C             ENDFILE 6
              WRITE(8,30)
   30         FORMAT(* THE NEWCEN INTERATIONS WILL CONTINUE.*)
              GOTO 190
  900         CONTINUE
C
C             (X1,Y1) IS NOT CONTAINED IN LAND MASS IL, NOR IN THE RECTANGLE OF
C             LAND MASS L, NOR IN LAND MASS L.
C
 1000       CONTINUE
C
C           (X1,Y1) IS A FEASIBLE NEW CENTROID, SINCE IT IS CONTAINED IN NO
C           LAND MASS.
C
            VX(KK)=X1
            VY(KK)=Y1
            GOTO 2900
 1900       CONTINUE
C
C           (X1,Y1) IS CONTAINED IN LAND MASS IL
C
 2000     CONTINUE
 2900     CONTINUE
 3000  CONTINUE
       WRITE(7,49)
   49  FORMAT(*COUNTERCLOCKWISE NEW CENTROID WHEN KK=1, CLOCKWISE NEW *,
      $   *CENTROID WHEN KK=2.*)
       DO 4000 KK=1,2
         WRITE(7,50) KK,VX(KK),KK,VY(KK)
 4000  CONTINUE
   50  FORMAT(*VX(*,I1,*)= *,G15.5,* VY(*,I1,*)= *,G15.5)
       CALL MESSAGR(7HNEWCEN )
       RETURN
       END
       SUBROUTINE REMAIN(XS,YS,IS,LAX,LAY,LIA,LIB,LAN,XW,YW,IW,RX,RY,IR,
      $   LN,LE,LS,LW,ALPHA,XPC,YPC,IC1,XCEN,YCEN,XWO,YWO,NUM,REM,LL,
      $   AREAS,INUM)
C
C      REMAIN COMPUTES THE REMAINDER POLYGONS. THAT IS A POLYGON Q IS COMPUTED
C      FROM XPC,YPC, XCEN,YCEN, AND THE SCALE FACTOR ALPHA. REMAIN CALCULATES
C      THOSE PARTS OF Q INTERSECTING NO LAND MASS. EACH SUCH PART OF Q, CALLED
C      A REMAINDER POLYGON, IS A MEMBER OF REMAINDER SET LL.
C      ALL REMAINDER SETS ARE STORED IN XWO,YWO.
C      REM(1,LL) IS THE OFFSET OF THE INDEX OF REMAINDER SET LL, ALSO CALLED
C      REM1, IN NUM.
C      REM(2,LL) IS THE NUMBER OF POLYGONS IN REMAINDER SET LL, ALSO CALLED
C      MAXPOL, IN NUM.
C      NUM(1,J) IS THE OFFSET OF THE INDEX OF THE JTH POLYGON IN XWO,YWO,
C      ALSO CALLED NUM1.
C      NUM(2,J) IS THE NUMBER OF VERTICES IN THE JTH POLYGON IN XWO,YWO,
C      ALSO CALLED NUM2.
C      THE OFFSET OF POLYGON K IN REMAINDER SET LL IS REM(1,LL)+K
C      THE OFFSET OF THE INDEX OF POLYGON K IN REMAINDER SET LL IN XWO,YWO IS
C      NUM(1,REM(1,LL)+K).
C      THE VERTICES OF POLYGON K IN REMAINDER SET LL ARE GIVEN BY
```

```
C     XWO(NUM(1,REM(1,LL)+K)+I),YWO(NUM(1,REM(1,LL)+K)+I), I=1,NUM(2,REM(1,LL)
C     +K)
C     AREAS(LL) IS THE AREA CONTAINED THE LARGEST POLYGON IN REMAINDER SET LL.
C     INUM(LL) IS THE INDEX OF THE LARGEST POLYGON IN SET LL.
C     REMAIN IS CALLED BY GOLDSEC AND POSINT.
C
      REAL LAX(1),LAY(1),NORTH,LN(1),LE(1),LS(1),LW(1)
      INTEGER REM(2,1),REM1
      DIMENSION XPC(1),YPC(1),LIA(1),LIB(1),XW(1),YW(1),IW(2,1),
     $  RX(1),RY(1),IR(2,1),INUM(1),XWO(1),YWO(1),
     $  NUM(2,1),AREAS(1),XS(1),YS(1),IS(2,1),XT(200),YT(200),WK(300)
      CALL MESSAGA(7HREMAIN )
      IR(1,1)=IS(1,1)=IW(1,1)=0
      REM1=0
      IF(LL.GT.1) REM1=REM(1,LL)
      NUMR=NUM(1,REM1+1)
      DO 1000 I=1,IC1
        RX(I)=ALPHA*XPC(I)+(1.-ALPHA)*XCEN
        RY(I)=ALPHA*YPC(I)+(1.-ALPHA)*YCEN
 1000 CONTINUE
      IR(2,1)=IC1
      JMAX=1
      MAXPOL=1
C
C     MAXPOL KEEPS A RUNNING TOTAL OF THE NUMBER OF POLYGONS IN POSITION
C     POLYGONS 1 TO IP-1 BUT NOT IN LAND 1 TO IL. MAXPOL IS UPDATED
C     IMMEDIATELY BEFORE IP INCREASES AND AT THAT POINT IT IS THE
C     NUMBER OF POLYGONS IN POSITION POLYGONS 1 TO IP BUT NOT IN LAND
C     1 TO IL.
C
C     MAXPOL COUNTS THE NUMBER OF POLYGONS STORED IN XWO,YWO FOR REMAINDER
C     SET LL. LAN IS THE NUMBER OF LAND MASSES.
C
      DO 4000 IL=1,LAN
        WRITE(7,20) IL
   20   FORMAT(*IL=*,I3)
C
C       TEST EACH CHAIN IN RX,RY FOR INTERSECTION WITH LAND MASS IL
C
        NO=MAXPOL
        MAXPOL=0
        L1=LIA(IL)
        L2=LIB(IL)-1
        DO 3000 IP=1,NO
          CALL MESSAGC(7HREMAIN ,7HRECTAN )
          I1=IR(1,IP)+1
          I2=IR(1,IP)+IR(2,IP)
          CALL RECTAN(I1,I2,RX,RY,NORTH,EAST,SOUTH,WEST)
          CALL MESSAGC(7HREMAIN ,7HMINIMAX)
          IF(MINIMAX(NORTH,EAST,SOUTH,WEST,LN(IL),LE(IL),LS(IL),
     $      LW(IL))) 1500,1700
 1500     CONTINUE
C
C         POLYGON IP DOES NOT INTERSECT LAND MASS IL.
C         STORE POLYGON IP IN (XWO,YWO).
C
          CALL MESSAGC(7HREMAIN ,7HINTO    )
          JMIN=JMAX=IP
          CALL INTO(RX,RY,IR,JMIN,JMAX,XWO,YWO,NUM,REM,LL,MAXPOL,IP,
     $      IL)
C
```

```
C          INCREMENT IP COUNTER
C
           GOTO 2900
 1700      CONTINUE
C
C          A NONEMPTY INTERSECTION BETWEEN IL AND IP IS POSSIBLE.
C          GENERATE AND STORE THE POLYGON(S) IN IP BUT NOT IN IL
C          IN (XWO,YWO). KALC=3.
           DO 1800 I=I1,I2
             IO=I-I1+1
             XW(IO)=RX(I)
             YW(IO)=RY(I)
 1800      CONTINUE
           DO 1900 I=L1,L2
             LO=I-L1+1
             XT(LO)=LAX(I)
             YT(LO)=LAY(I)
 1900      CONTINUE
           CALL MESSAGC(7HREMAIN ,7HIUCALC )
           CALL IUCALC(XW,YW,IO,XT,YT,LO,3,WK,300,JMAX,IS,20,XS,YS,200)
           CALL MESSAGA(7HREMAIN )
           IF(JMAX.LE.0) 1950,2000
 1950      WRITE(8,35) JMAX
           WRITE(6,35) JMAX
C          ENDFILE 6
           WRITE(7,35) JMAX
C          ENDFILE 7
   35      FORMAT(*IUCALC ERROR IN REMAIN.JMAX= *,I3)
           STOP 10
 2000      CONTINUE
           WRITE(8,40) IL,IP,JMAX
           WRITE(6,40) IL,IP,JMAX
C          ENDFILE 6
           WRITE(7,40) IL,IP,JMAX
C          ENDFILE 7
   40      FORMAT(*IL=*,I3,* IP=*,I3,* THE NUMBER OF RESULTS CHAINS *,
     $       *=JMAX= *,I3)
           JMIN=1
           CALL MESSAGC(7HREMAIN ,7HINTO    )
           CALL INTO(XS,YS,IS,JMIN,JMAX,XWO,YWO,NUM,REM,LL,MAXPOL,IP,
     $       IL)
           CALL MESSAGA(7HREMAIN )
 2900      CONTINUE
 3000    CONTINUE
C
C          THE FOLLOWING CODE TRANSFERS THE DEVELOPING REMAINDER RESULT, IN
C          (XWO,YWO) TO (RX,RY). THIS TRANSFER IS UNNECESSARY WHEN IL=LAN.
           IF(IL.EQ.LAN) GOTO 3999
C
C          MAXPOL=REM(2,LL)
C          TRANSFER TO (RX,RY)      THE REMAINDER POLYGONS COMPUTED FROM THE
C          INTERSECTION OF LAND MASS IL AND THE CURRENT CONTENTS OF (RX,RY).
C          NUMR=THE OFFSET OF THE INDEX OF POLYGON 1 IN SET LL.
C          NUM1=THE OFFSET OF THE INDEX OF POLYGON IP IN SET LL.
C          THEREFORE NUM1=NUMR+NUM(2,REM1+1)+...+NUM(2,REM1+IP-1), FOR IP>1,
C          WHENCE NUM1-NUMR=NUM(2,REM1+1)+...+NUM(2,REM1+IP-1) IS THE
C          APPROPRIATE OFFSET IN IR.
C
C
C
           WRITE(6,45) IL,MAXPOL
```

```
C        ENDFILE 6
         WRITE(7,45) IL,MAXPOL
         WRITE(7,50)
         WRITE(8,45) IL,MAXPOL
   45    FORMAT(* AT THE END OF ITERATION *,I3,* MAXPOL= *,I3)
   50    FORMAT(*THE CONTENTS OF (RX,RY) ARE *)
         DO 3500 IP=1,MAXPOL
           JO=REM1+IP
           NUM1=NUM(1,JO)
           IR1=IR(1,IP)=NUM1-NUMR
           IR2=IR(2,IP)=NUM(2,JO)
           DO 3400 I=1,IR2
             RX(IR1+I)=XWO(NUM1+I)
             RY(IR1+I)=YWO(NUM1+I)
 3400      CONTINUE
           WRITE(7,55) IP
   55      FORMAT(*IP= *,I3)
           WRITE(7,80) (RX(IR1+I),RY(IR1+I),I=1,IR2)
C          ENDFILE 7
 3500    CONTINUE
 3999    CONTINUE
 4000  CONTINUE
C
C      DETERMINE AREAS(LL)
C
       CALL MESSAGC(7HREMAIN ,7HAREA    )
C
C      COMPUTE AREA OF LARGEST POLYGON.
C
       INUM(LL)=1
       AREAS(LL)=AREA(XWO,YWO,NUM,REM1+1)
       EP=.0001
       IF(MAXPOL.EQ.1) GOTO 5050
       DO 5000 IP=2,MAXPOL
       AR=AREA(XWO,YWO,NUM,REM1+IP)
       IF(AREAS(LL).GE.AR-EP) GOTO 4900
       AREAS(LL)=AR
       INUM(LL)=IP
 4900  CONTINUE
 5000  CONTINUE
 5050  CONTINUE
       CALL MESSAGA(7HREMAIN )
       WRITE(7,65) LL
   65  FORMAT(*REMAINDER SET *,I3)
       WRITE(7,70) LL,REM(1,LL),LL,REM(2,LL),MAXPOL
   70  FORMAT(*REM(1,*,I3,*)=*,I3,* REM(2,*,I3,*)=*,I3,* MAXPOL=*,I3)
       DO 6000 IP=1,MAXPOL
         JO=REM1+IP
         NUM1=NUM(1,JO)
         NUM2=NUM(2,JO)
         WRITE(7,75) IP,JO,NUM1,JO,NUM2
         WRITE(7,80) (XWO(NUM1+I),YWO(NUM1+I),I=1,NUM2)
C        ENDFILE 7
 6000  CONTINUE
   75  FORMAT(*IP=*,I3,* NUM(1,*,I3,*)=*,I3,* NUM(2,*,I3,*)=*,I3,* THE*
     $   ,*CONTENTS OF XWO,YWO ASSOCIATED WITH THESE INDICES ARE *)
   80  FORMAT(5(G13.6,G13.6))
       RETURN
       END
```

Appendix B

PACKAGE IUCALC SOURCE CODE

```
      SUBROUTINE IUCALC(APX,APY,NOAP,BCX,BCY,NOBC,KALC,
     &  WORK,WRKMAX,NORC,INORC,INOMAX,RCX,RCY,NRCMAX)
C
C  THIS ROUTINE (1) DETERMINES THE POLYGONS DEFINING THE UNION,
C  INTERSECTION, OR RELATIVE DIFFERENCE (I.E. THE INTERIOR OF ONE
C  THAT IS NOT INTERIOR TO THE OTHER) OF TWO GIVEN POLYGONS, OR
C  (2) DETERMINES WHAT PORTIONS OF A CHAIN OF LINE SEGMENTS LIE ON
C  THE BOUNDARY, INTERIOR, OR EXTERIOR TO A GIVEN POLYGON.
C
      INTEGER NOAP,NOBC,KALC
      REAL APX(NOAP),APY(NOAP),BCX(NOBC),BCY(NOBC)
C  ******************** USER SPECIFIED VARIABLES **********************
C  APX  A REAL ARRAY OF THE X COORDINATES OF A, THE FIRST POLYGON
C  APY  A REAL ARRAY OF Y COORDINATES OF THE A POLYGON
C  NOAP AN INTEGER COUNT OF THE NUMBER OF POINTS DEFINING THE A POLYGON.
C     THE POLYGON CONSISTS OF LINE SEGMENTS FROM APX(1),APY(1) TO
C      APX(2),APY(2) TO ... TO APX(NOAP),APY(NOAP) TO APX(1),APY(1)
C
C  IF KALC IS FOUR OR LESS:
C  BCX  A REAL ARRAY OF THE X COORDINATES OF B, THE SECOND POLYGON
C  BCY  A REAL ARRAY OF Y COORDINATES OF THE B POLYGON
C  NOBC AN INTEGER COUNT OF THE NUMBER OF POINTS DEFINING THE B POLYGON
C  KALC  AN INTEGER SPECIFYING THE DESIRED TYPE OF OVERLAY
C       KALC=1 UNION OF A AND B
C       KALC=2  INTERSECTION OF A AND B
C       KALC=3  RELATIVE DIFFERENCE OF A TO B (A INTERSECT NOT B)
C       KALC=4  RELATIVE DIFFERENCE OF B TO A (B INTERSECT NOT A)
C
C  IF KALC IS GREATER THAN FOUR, BCX,BCY IS AN OPEN CHAIN OF SEGMENTS
C  (RATHER THAN A CLOSED POLYGON) FROM BCX(1),BCY(1) TO BCX(2),BCY(2)
C  TO ... TO BCX(NOBC),BCY(NOBC)
C       KALC=5 SUBCHAINS OF B ON THE BOUNDARY OF A
C       KALC=6 SUBCHAINS OF B EXTERIOR TO A
C       KALC=7 SUBCHAINS OF B INTERIOR TO A
C       KALC=8 SUBCHAINS OF B EXTERIOR TO OR ON THE BOUNDARY OF A
C       KALC=9 SUBCHAINS OF B INTERIOR TO OR ON THE BOUNDARY OF A
C
C
      INTEGER WRKMAX,WORK(WRKMAX)
C ********************* WORK SPACE **********************
C
C  WORK  IS A REAL*4 ARRAY OF LENGTH NOWORK.  THE AMOUNT OF WORK SPACE
C     REQUIRED CAN BE ESTIMATED BY THE FORMULA NOAP+NOBC+6*K+2 WHERE K
C     IS THE MAXIMUM NUMBER OF INTERSECTIONS EXPECTED.
C  WRKMAX  THE INTEGER LENGTH OF THE WORK ARRAY.
C
C
C ********************* RESULT VARIABLES ********************
      INTEGER NORC,INORC(2,NORC)
      REAL RCX(1),RCY(1)
C  NORC  AN INTEGER COUNT OF THE NUMBER OF CHAINS IN THE OVERLAY.
C     NORC=0 MEANS NO RESULT CHAINS HAVE BEEN CALCULATED. EITH 1) NO
C     RESULT EXISTS OR 2) THE RESULT IS THE SAME AS THE ORIGINAL DATA.
C     NORC IS ALSO AN ERROR INDICATOR FOR THE FOLLOWING TYPES OF ERRORS:
C        NORC=-1     THE TWO DATA CHAINS ARE NOT SIMPLY CONNECTED AT
C                    A POINT OF INTERSECTION.
C        NORC<-10  WORK SPACE IS OF INSUFFICIENT SIZE.
C
C  INORC  AN INTEGER MATRIX DIMENSIONED  (2,NORC) WHERE NORC IS THE
C     NUMBER OF CHAINS IN THE RESULT.  INORC(1,I) IS THE OFFSET OF THE
C     INDEX FOR THE ITH CHAIN OF THE RESULT IN THE RCX,RCY ARRAYS.
```

```
C      INORC(2,I) IS THE NUMBER OF POINTS IN THE ITH CHAIN OF THE
C      RESULT.
C
C   RCX  A REAL ARRAY FOR THE X COORDINATES OF THE CHAIN(S) OF THE
C      RESULT.  THE X COORDINATES OF THE ITH CHAIN ARE RCX(K+1),RCX(K+2),
C      RCX(K+3), ... ,RCX(K+L) WHERE K=INORC(1,I) AND L=INORC(2,I).
C   RCY  A REAL ARRAY FOR THE Y COORDINATES OF THE RESULT CHAIN(S).
C
C      A DOUBLE PRECISION TO REAL FUNCTION, ROUND(X), IS CALLED BY
C       THE SUBROUTINE IUCALC
C
     1 NORC=0
       CALL IUSUB1(APX,APY,NOAP,BCX,BCY,NOBC,KALC,WORK(1),WORK(1),WRKMAX,
      &  NONXT,NORC)
       IF(NORC .LT. 0)RETURN
       CALL IUSUB2(APX,APY,IABS(NOAP),BCX,BCY,IABS(NOBC),KALC,
      &  WORK(1),WORK(1),WRKMAX,NONXT,WORK(2*NONXT+1),
      &  NORC,INORC,INOMAX,RCX,RCY,NRCMAX)
       RETURN
       END
       SUBROUTINE IUSUB1(X,Y,NAS,U,V,NBS,OTYPE,NXT,XYT,NWORK,NONXT,ERROR)
       DIMENSION X(1),Y(1),U(1),V(1)
       INTEGER OTYPE
C   XYT  A REAL MATRIX DIMENSIONED BY (2,MAX1) FOR STORAGE OF THE X AND Y
C      COORDINATES OF THE INTERSECTION POINTS NOT OCCURING AT THE
C      DEFINING POINTS.
C
       REAL XYT(2,1)
C
C   NXT  AN INTEGER MATRIX DIMENSIONED BY (2,MAX3) WHERE MAX3 EQUALS
C      (NA+NB+TWICE THE NUMBER OF INTERSECTIONS NOT OCCURING AT THE
C      DEFINING POINTS).
       INTEGER NXT(2,1)
       INTEGER ERROR
C       ERROR=-11      XT,YT OVERRUN
C       ERROR=-12        SEG OVER-RUN
C       ERROR=-13        NXT OVER-RUN
C
       LOGICAL ASENSE,BSENSE,SENSE,INTER,BSTEMP
       DOUBLE PRECISION ZERO,ONE
       DOUBLE PRECISION AX,AY,BX,BY,B2,DX,DY,
      $  AXB,AXD,BXD,D,Q,DMIN,DOTJM,DOTJNM,DOTJN,DOTJNN
       DATA ZERO/0.D0/,ONE/1.D0/
       DATA ASENSE/.TRUE./,BSENSE/.TRUE./
       NA=IABS(NAS)
       NB=IABS(NBS)
       NXYEND=NWORK/2+1
       BSTEMP=BSENSE
       IF(OTYPE .LE. 4 .AND. NAS .GT. 0)ASENSE=SENSE(X,Y,NA)
       IF(OTYPE .LE. 4 .AND. NBS .GT. 0)BSENSE=SENSE(U,V,NB)
       NBT=NB
       IF(OTYPE .GT. 4)NBT=NB-1
       IF(OTYPE .EQ. 3)BSENSE=.NOT.BSENSE
       IF(OTYPE .EQ. 4)ASENSE=.NOT.ASENSE
       IF(OTYPE .GT. 4)BSENSE=.TRUE.
       NONXT=0
       NXYT=0
       DO 5 I=1,NA
       IF(.NOT.ASENSE)NXT(2,I)=MOD(I+NA-2,NA)+1
       IF(ASENSE)NXT(2,I)=MOD(I,NA)+1
       IF(X(I) .EQ. X(NXT(2,I))) .AND.
```

```
     &  Y(I) .EQ. Y(NXT(2,I)))GO TO 403
    5 NXT(1,I)=I
      DO 10 I=1,NB
      IF(BSENSE)NXT(2,I+NA)=MOD(I,NB)+NA+1
      IF(.NOT.BSENSE)NXT(2,I+NA)=MOD(I+NB-2,NB)+NA+1
      IF(U(I) .EQ. U(NXT(2,I+NA)-NA) .AND.
     &  V(I) .EQ. V(NXT(2,I+NA)-NA))GO TO 403
   10 NXT(1,I+NA)=I+NA
      BSENSE=BSTEMP
      NONXT=NA+NB
      J=1
      XJN=X(1)
      YJN=Y(1)
      DO 65 I=1,NBT
      N=NXT(2,I+NA)
      UOLD=U(I)
      VOLD=V(I)
      NOLD=I+NA
C          START OF LOOP ON J
   11 BX=U(N-NA)-UOLD
      BY=V(N-NA)-VOLD
      B2=BX**2+BY**2
      INTER=.FALSE.
   14 XJ=XJN
      YJ=YJN
      JN=NXT(2,J)
      IF(JN .LE. NA)GO TO 20
      JZ=-NXT(1,JN)-NA-NB
      IF(JZ .LE. 0)GO TO 16
      XJN=XYT(1,NXYEND-JZ)
      YJN=XYT(2,NXYEND-JZ)
      GO TO 22
   16 JZ=JZ+NB
      XJN=U(JZ)
      YJN=V(JZ)
      GO TO 22
   20 XJN=X(JN)
      YJN=Y(JN)
   22 DX=XJ-UOLD
      DY=YJ-VOLD
      AX=XJN-XJ
      AY=YJN-YJ
      AXB=AX*BY-AY*BX
      BXD=BX*DY-BY*DX
      IF(AXB .EQ. ZERO)GO TO 30
      Q=BXD/AXB
      IF(Q .LE. ZERO .OR. Q .GT. ONE)GO TO 45
      AXD=AX*DY-AY*DX
      D=AXD/AXB
      IF(D .LE. ZERO  OR. D .GT. ONE)GO TO 45
      K=0
      L=0
      XI=ROUND(UOLD+D*BX)
      YI=ROUND(VOLD+D*BY)
      IF(XI .NE. UOLD .OR. YI .NE. VOLD)GO TO 24
      K=NOLD
   24 IF(XI .NE. U(N-NA) .OR. YI .NE. V(N-NA))GO TO 26
      K=N
   26 XP=ROUND(XJ+Q*AX)
      YP=ROUND(YJ+Q*AY)
      IF(XP .NE. XJ .OR. YP .NE. YJ)GO TO 28
```

```
      L=J
   28 IF(XP .NE. XJN .OR. YP .NE. YJN)GO TO 40
      L=JN
      GO TO 40
   30 IF(BXD .NE. ZERO)GO TO 45
      K=0
      L=0
      DOTJM=DX*BX+DY*BY
      DOTJNM=(XJN-UOLD)*BX+(YJN-VOLD)*BY
      DOTJN=(XJ-U(N-NA))*BX+(YJ-V(N-NA))*BY
      DOTJNN=(XJN-U(N-NA))*BX+(YJN-V(N-NA))*BY
C        IS J OUTSIDE M-N AND JN OUTSIDE M-N AND
C        N OUTSIDE J-JN AND M OUTSIDE J-JN
      IF(DOTJN*DOTJM .GT. ZERO .AND. DOTJNM*DOTJNN .GT. ZERO .AND.
     $  DOTJN*DOTJNN .GT. ZERO .AND. DOTJM*DOTJNM .GT. ZERO)GO TO 45
C        IS N INSIDE J-JN AND M INSIDE J-JN
      IF(DOTJN*DOTJNN .LE. ZERO .AND. DOTJM*DOTJNM .LE. ZERO)GO TO 36
C        IS J OUTSIDE M-N AND JN INSIDE M-N
C        OR J INSIDE M-N AND JN INSIDE M-N
C        AND JN RATHER THAN J FURTHER FROM M
      IF(DOTJM*DOTJN .GE. ZERO .AND. DOTJNM*DOTJNN .LE. ZERO .OR.
     $  DOTJM*DOTJN .LE. ZERO .AND. DOTJNM*DOTJNN .LE. ZERO .AND.
     $  DOTJNM .GT. DOTJM)GO TO 38
      IF(DOTJM .EQ. ZERO)GO TO 45
      D=DOTJM/B2
      L=J
      IF(DOTJN .EQ. ZERO)K=N
      GO TO 40
   36 D=ONE
      K=N
      IF(DOTJN .EQ. ZERO)L=J
      IF(DOTJNN .EQ. ZERO)L=JN
      GO TO 40
   38 IF(DOTJNM .EQ. ZERO)GO TO 45
      D=DOTJNM/B2
      L=JN
      IF(DOTJNN .EQ. ZERO)K=N
   40 IF(INTER .AND. D .GT. DMIN)GO TO 45
      IF(K .NE. 0 .AND. NXT(1,K) .LT. 0 .OR.
     *  L .NE. 0 .AND. NXT(1,L) .LT. 0)GO TO 45
      DMIN=D
      XMIN=XI
      YMIN=YI
      KMIN=K
      LMIN=L
      JI=J
      INTER=.TRUE.
   45 J=NXT(2,J)
      IF(J .NE  1)GO TO 14
C
C          END OF LOOP ON J
C
C
      IF(.NOT.INTER)GO TO 65
C
C          LINK THE NEW POINT INTO THE U,V POLYGON
      IF(KMIN .NE. 0)GO TO 54
      NONXT=NONXT+1
      IF(2*NONXT .LE. NWORK-2*NXYT)GO TO 46
      ERROR=-13
      GO TO 402
```

```
   46 IF(LMIN .GT. 0)GO TO 52
      NXYT=NXYT+1
      IF(NXYT .LE. (NWORK-2*NONXT)/2)GO TO 47
      ERROR=-11
      GO TO 402
   47 XYT(1,NXYEND-NXYT)=XMIN
      XYT(2,NXYEND-NXYT)=YMIN
      NXT(1,NONXT)=-NXYT-NA-NB
      UOLD=XMIN
      VOLD=YMIN
      GO TO 53
   52 NXT(1,NONXT)=-LMIN
      UOLD=X(LMIN)
      VOLD=Y(LMIN)
   53 NXT(2,NONXT)=NXT(2,NOLD)
      NXT(2,NOLD)=NONXT
      NOLD=NONXT
      GO TO 55
   54 NXT(1,KMIN)=-KMIN
      IF(LMIN .GT. 0)NXT(1,KMIN)=-LMIN
C
C          LINK THE NEW POINT INTO THE X,Y POLYGON
   55 IF(LMIN .GT. 0)GO TO 59
      NONXT=NONXT+1
      IF(2*NONXT .LE. NWORK-2*NXYT)GO TO 56
      ERROR=-13
      GO TO 402
   56 IF(KMIN .GT. 0)GO TO 57
      NXT(1,NONXT)=NXT(1,NONXT-1)
      GO TO 58
   57 NXT(1,NONXT)=-KMIN
   58 NXT(2,NONXT)=NXT(2,JI)
      NXT(2,JI)=NONXT
      GO TO 60
   59 NXT(1,LMIN)=-NXT(1,LMIN)
   60 IF(KMIN .NE. N)GO TO 11
   65 CONTINUE
C
  402 CONTINUE
      RETURN
  403 ERROR=-1
      GO TO 402
      END
      SUBROUTINE IUSUB2(X,Y,NA,U,V,NB,OTYPE,NXT,XYT,NWORK,
     $  NONXT,SEG,NONIU,NIUS,NIUMAX,XIU,YIU,NXYMAX)
      DIMENSION X(NA),Y(NA),U(NB),V(NB)
      INTEGER OTYPE
C
C  SEG  AN INTEGER MATRIX DIMENSIONED (3,MAX2) WHERE MAX2-2 EQUALS THE
C     NUMBER OF INTERSECTIONS OF THE TWO FIGURES.
      INTEGER NXT(2,1),SEG(3,1)
      REAL XYT(2,1),XIU(1),YIU(1)
      INTEGER NIUS(2,1)
      INTEGER NIUMAX
      LOGICAL IDENT
      LOGICAL PTS
      NSEG=0
      NXYT=NONXT-NA-NB
      IF(OTYPE .LE. 4)GO TO 68
      IF(NXT(1,NA+1) .GT. 0)NXT(1,NA+1)=-NXT(1,NA+1)
      IF(NXT(1,NA+NB) .GT. 0)NXT(1,NA+NB)=-NXT(1,NA+NB)
```

```
      GO TO 69
   68 CALL SEGDEF(U,V,NB,X,Y,NA,0,OTYPE,
     $  SEG,NSEG,NXT,NONXT,XYT,NWORK)
      IF(NSEG .LT. 0)GO TO 400
   69 CALL SEGDEF(X,Y,NA,U,V,NB,NA,OTYPE,
     $  SEG,NSEG,NXT,NONXT,XYT,NWORK)
      IF(NSEG .LT. 0)GO TO 400
      IDENT=.TRUE.
      DO 70 I=1,NSEG
      IF(SEG(1,I) .NE. 4)IDENT=.FALSE.
      IZ=SEG(2,I)
      IF(NXT(1,IZ) .LT. 0)NXT(1,IZ)=-NXT(1,IZ)
   70 CONTINUE
      IZ=SEG(3,NSEG)
      IF(NXT(1,IZ) .LT. 0)NXT(1,IZ)=-NXT(1,IZ)
C          TAKE CARE OF UNION AND INTERSECTION OF IDENTICAL POLYGONS
C          AS A SPECIAL CASE
      IF(.NOT.IDENT .OR. OTYPE .GT. 2)GO TO 80
      NONIU=1
      NIUS(2,NONIU)=NB
      NIUS(1,NONIU)=0
      DO 71 I=1,NB
      XIU(I)=U(I)
   71 YIU(I)=V(I)
      RETURN
C
C       LINK THE SEGMENTS AND GENERATE THE RESULT VARIABLES
C
   80 CONTINUE
   89 NIU=1
      NONIU=0
C    SEG(1,I) = 1  SEGMENT I OF A IS ON THE BOUNDARY OF B
C    SEG(1,I) = 2  SEGMENT I OF A IS OUTSIDE B
C    SEG(1,I) = 3  SEGMENT I OF A IS INSIDE B
C    SEG(1,I) = 4  SEGMENT I OF B IS ON THE BOUNDARY OF A
C    SEG(1,I) = 5  SEGMENT I OF B IS OUTSIDE A
C    SEG(1,I) = 6  SEGMENT I OF B IS INSIDE A
   90 DO 100 I=1,NSEG
      GO TO (91,92,93,94,95,96,97,98,99),OTYPE
   91 IF(SEG(1,I) .EQ. 2 .OR. SEG(1,I) .EQ. 5)GO TO 110
      GO TO 100
   92 IF(SEG(1,I) .EQ. 3 .OR. SEG(1,I) .EQ. 6)GO TO 110
      GO TO 100
   93 IF(SEG(1,I) .EQ. 2 .OR. SEG(1,I) .EQ. 6)GO TO 110
      GO TO 100
   94 IF(SEG(1,I) .EQ. 3 .OR. SEG(1,I) .EQ. 5)GO TO 110
      GO TO 100
   95 IF(SEG(1,I) .EQ. 4)GO TO 110
      GO TO 100
   96 IF(SEG(1,I) .EQ. 5)GO TO 110
      GO TO 100
   97 IF(SEG(1,I) .EQ. 6)GO TO 110
      GO TO 100
   98 IF(SEG(1,I) .EQ. 4 .OR. SEG(1,I) .EQ. 5)GO TO 110
      GO TO 100
   99 IF(SEG(1,I) .EQ. 4 .OR. SEG(1,I) .EQ. 6)GO TO 110
  100 CONTINUE
      RETURN
  110 IP=SEG(2,I)
      II=NXT(1,IP)
      NONIU=NONIU+1
```

```
      IF(NONIU .GT. NIUMAX)GO TO 400
  111 PTS=.FALSE.
      NIUS(1,NONIU)=NIU-1
      NIUS(2,NONIU)=0
  120 IF(IP .EQ. SEG(3,I) .AND. NXT(1,IP) .EQ. I1 .AND. PTS)GO TO 310
      NIUS(2,NONIU)=NIUS(2,NONIU)+1
  121 IF(NIU .GT. NXYMAX)GO TO 400
      CALL PNTGET(X,Y,NA,U,V,NB,XYT,NWORK,NXT(1,IP),
     &  XIU(NIU),YIU(NIU))
      NIU=NIU+1
      PTS=.TRUE.
      IF(IP .EQ. SEG(3,I) .AND. SEG(2,I) .NE. SEG(3,I))GO TO 125
  122 IP=NXT(2,IP)
      GO TO 120
  125 IF(OTYPE .GT. 4)GO TO 310
      DO 160 J=1,NSEG
      JP=SEG(2,J)
      IF(J .NE. I .AND. SEG(1,I) .EQ. SEG(1,J) .AND.
     $   NXT(1,IP) .EQ. NXT(1,JP))GO TO 300
  160 CONTINUE
      DO 180 J=1,NSEG
      JP=SEG(2,J)
      IF(NXT(1,IP) .NE. NXT(1,JP))GO TO 180
      GO TO (171,172,173,174),OTYPE
  171 IF(SEG(1,J) .EQ. 2 .OR. SEG(1,J) .EQ. 5)GO TO 300
      GO TO 180
  172 IF(SEG(1,J) .EQ. 3 .OR. SEG(1,J) .EQ. 6)GO TO 300
      GO TO 180
  173 IF(SEG(1,J) .EQ. 2 .OR. SEG(1,J) .EQ. 6)GO TO 300
      GO TO 180
  174 IF(SEG(1,J) .EQ. 3 .OR. SEG(1,J) .EQ. 5)GO TO 300
  180 CONTINUE
      DO 190 J=1,NSEG
      JP=SEG(2,J)
      IF(NXT(1,IP) .EQ. NXT(1,JP) .AND. SEG(1,J) .GT. 0)GO TO 300
  190 CONTINUE
      NONIU=-2
      GO TO 402
  300 SEG(1,I)=-SEG(1,I)
      IP=SEG(2,J)
      I=J
      GO TO 122
  310 SEG(1,I)=-SEG(1,I)
      GO TO 90
  400 NONIU=-12
  402 RETURN
      END
      LOGICAL FUNCTION SENSE(X,Y,N)
C
C      SENSE IS TRUE IF THE X,Y POLYGON CLOSES CLOCKWISE, FALSE IF
C     IT CLOSES COUNTERCLOCKWISE
C
      INTEGER N
      REAL X(N),Y(N)
      DOUBLE PRECISION TSUM
      IF(N .LT. 3)RETURN
      TSUM=0.D0
      AY=Y(2)-Y(1)
      AX=X(2)-X(1)
      DO 2 J=3,N
      BY=Y(J)-Y(1)
```

```
      BX=X(J)-X(1)
      TSUM=TSUM+BY*AX-AY*BX
      AX=BX
    2 AY=BY
      SENSE=.FALSE.
      IF(TSUM .LT. 0.DO)SENSE=.TRUE.
      RETURN
      END
      SUBROUTINE SEGDEF(X,Y,NXY,U,V,NUV,OFFSET,OTYPE,
     $  SEG,NSEG,NXT,NONXT,XYT,NWORK)
C
C     SEGDEF GENERATES THE SEG MATRIX.  A SEGMENT IS A SERIES OF
C     SUCCESSIVE SIDES OF ONE OF THE POLYGONS ALL LYING ALL INSIDE OR
C     ALL OUTSIDE THE OTHER POLYGON, OR A SIDE OF ONE POLYGON LYING ON
C     THE BOUNDARY OF THE OTHER. IF THE POLYGONS INTERSECT, THE END
C     POINTS OF THE SEGMENTS LIE OF THE BOUNDARY OF THE OTHER POLYGON.
C     IF THE POLYGONS DON'T INTERSECT OR ARE IDENTICAL, EACH POLYGON IS
C     A SINGLE SEGMENT. EACH TRIPLET OF THE SEG MATRIX DEFINES A
C     SEGMENT. THE FIRST NUMBER IN THE ITH TRIPLET CONTAINS A CODE:
C   SEG(1,I) = 1  SEGMENT I OF A IS ON THE BOUNDARY OF B
C   SEG(1,I) = 2  SEGMENT I OF A IS OUTSIDE B
C   SEG(1,I) = 3  SEGMENT I OF A IS INSIDE B
C   SEG(1,I) = 4  SEGMENT I OF B IS ON THE BOUNDARY OF A
C   SEG(1,I) = 5  SEGMENT I OF B IS OUTSIDE A
C   SEG(1,I) = 6  SEGMENT I OF B IS INSIDE A
C     SEG(2,I) IS THE INDEX IN THE NXT ARRAY OF THE FIRST POINT OF
C     SEGMENT I. SEG(3,I) IS THE INDEX IN THE NXT ARRAY OF THE LAST
C     POINT OF SEGMENT I.
C
      INTEGER NXY,NUV,OFFSET,NSEG,NXYT,OTYPE
      INTEGER ENSYD2
      REAL X(NXY),Y(NXY),U(NUV),V(NUV)
      INTEGER NXT(2,1),SEG(3,1)
      REAL XYT(2,1)
      LOGICAL SUMLNK
      INTEGER SOLD
      MAX2=(NWORK-2*NONXT)*2/3
      KK=2
      IF(OFFSET .NE. 0)KK=5
      SUMLNK=.FALSE.
      IOLD=OFFSET+1
      MSTART=IOLD
      NSTART=1
      IF(OFFSET .EQ. 0)NSTART=NUV+1
      IOF=-1
      IF(NXT(1,IOLD) .GT. 0)GO TO 40
      SUMLNK=.TRUE.
      SOLD=IOLD
      MSTART=IOLD
      IF(OTYPE .GT. 4)MSTART=NXY+NUV
   40 INEW=NXT(2,IOLD)
      IF(NXT(1,INEW) .LT. 0)GO TO 42
      IF(.NOT.SUMLNK .OR. IOF .GE. 0)GO TO 49
      IOF=ENSYD2(DBLE(U(INEW-OFFSET)),DBLE(V(INEW-OFFSET)),X,Y,NXY)
      GO TO 49
   42 IF(SUMLNK)GO TO 43
      MSTART=INEW
      SUMLNK=.TRUE.
      SOLD=INEW
      GO TO 60
   43 NSEG=NSEG+1
```

```
      IF(NSEG .GT. MAX2)GO TO 51
      SEG(2,NSEG)=SOLD
      SEG(3,NSEG)=INEW
      SOLD=INEW
      IF(IOF .LT. 0)GO TO 45
      SEG(1,NSEG)=KK+IOF
      IOF=-1
      GO TO 49
   45 IF(OFFSET .EQ. 0)GO TO 46
      CALL PNTGET(X,Y,NXY,U,V,NUV,XYT,NWORK,
     $   -NXT(1,IOLD),XOLD,YOLD)
      CALL PNTGET(X,Y,NXY,U,V,NUV,XYT,NWORK,
     $   -NXT(1,INEW),XNEW,YNEW)
      GO TO 47
   46 CALL PNTGET(U,V,NUV,X,Y,NXY,XYT,NWORK,
     $   -NXT(1,IOLD),XOLD,YOLD)
      CALL PNTGET(U,V,NUV,X,Y,NXY,XYT,NWORK,
     $   -NXT(1,INEW),XNEW,YNEW)
   47 IOFMID=ENSYD2((XOLD+XNEW)/2D0,(YOLD+YNEW)/2D0,X,Y,NXY)
  473 IF(OTYPE .GT. 4)GO TO 484
      ILS=NSTART
      I=NXT(2,NSTART)
  481 IF(NXT(1,ILS) .EQ. NXT(1,IOLD) .AND. NXT(1,I) .EQ. NXT(1,INEW)
     & .OR. NXT(1,I) .EQ. NXT(1,IOLD) .AND. NXT(1,ILS) .EQ. NXT(1,INEW))
     &  GO TO 482
        ILS=I
      I=NXT(2,I)
      IF(ILS .EQ. NSTART)GO TO 484
      GO TO 481
  482 IOFMID=-1
  484 SEG(1,NSEG)=KK+IOFMID
   49 IF(MSTART .NE. INEW)GO TO 60
      IF(SUMLNK)RETURN
      IOF=ENSYD2(DBLE(U(1)),DBLE(V(1)),X,Y,NXY)
      NSEG=NSEG+1
      IF(NSEG .LE. MAX2)GO TO 52
   51 NSEG=-12
      GO TO 53
   52 SEG(1,NSEG)=KK+IOF
      SEG(2,NSEG)=OFFSET+1
      SEG(3,NSEG)=SEG(2,NSEG)
   53 RETURN
   60 IOLD=INEW
      GO TO 40
      END
      INTEGER FUNCTION ENSYD2(U,V,X,Y,N)
C
C     THE FUNCTION ENSYD2 IS 1 IF THE POINT (U,V) IS INTERIOR TO THE
C     X,Y POLYGON. ENSYD2 IS 0 OTHERWISE. THE POINT (U,V) IS ASSUMED
C     NOT ON THE BOUNDARY OF THE X,Y POLYGON.
C
      INTEGER N
      REAL X(N),Y(N)
      DOUBLE PRECISION U,V
      LOGICAL BOOL
      ENSYD2=0
      BOOL=.FALSE.
      DO 10 I=1,N
      IN=I+1
      IF(IN .GT. N)IN=1
      IF(Y(IN) .EQ. Y(I) .AND. X(IN) .EQ. X(I))GO TO 10
```

```
      IF(V .LE. Y(IN) .OR. V .GT. Y(I))GO TO 5
      IF((U-X(I))*(Y(IN)-Y(I))-(V-Y(I))*(X(IN)-X(I)) .LT. 0.D0)
     $  BOOL=.NOT.BOOL
      GO TO 10
    5 IF(V .GT. Y(IN) .OR. V .LE. Y(I))GO TO 10
      IF((U-X(I))*(Y(IN)-Y(I))-(V-Y(I))*(X(IN)-X(I)) .GT. 0.D0)
     $  BOOL=.NOT.BOOL
   10 CONTINUE
      IF(BOOL)ENSYD2=1
      RETURN
      END
      SUBROUTINE PNTGET(X,Y,NA,U,V,NB,XYT,NWORK,K,A,B)
      REAL X(1),Y(1),U(1),V(1),XYT(2,1)
      INTEGER K
      IF(K .GT. NA+NB)GO TO 20
      IF(K .GT. NA)GO TO 10
      A=X(K)
      B=Y(K)
      GO TO 30
   10 A=U(K-NA)
      B=V(K-NA)
      GO TO 30
   20 A=XYT(1,NWORK/2+1-K+NA+NB)
      B=XYT(2,NWORK/2+1-K+NA+NB)
   30 RETURN
      END
      REAL FUNCTION ROUND(X1)
      DOUBLE PRECISION A,Y,X,X1
      REAL B(2),C(2)
      EQUIVALENCE (B(1),Y),(A,C(1))
      X=DABS(X1)
      ROUND=0.0
      IF(X.EQ.0.0D0) RETURN
      Y=X
      C(1)=B(1).AND.7777000000000000000B
      C(2)=B(2).AND.7777000000000000000B
      C(2)= C(2).OR.0000400000000000000B
      ROUND=X+A
      IF(X1.LT.0.D0) ROUND=-ROUND
      RETURN
      END
```

DISTRIBUTION LIST

Name	Number of Copies
Defense Technical Information Center Cameron Station Alexandria, VA 22314	12
Center for Naval Analyses 2000 North Beauregard Street Alexandria, VA 22311	1
Defense Advanced Research Projects Agency Tactical Technology Office 1400 Wilson Boulevard Arlington, VA 22209	1
Office of Naval Research Code 431 Arlington, VA 22217	2
Dr. Thomas E. Fortmann Bolt Beranek and Newman Inc. 50 Moulton Street Cambridge, MA 02138	1
Orincon Corporation 3366 North Torrey Pines Court Suite 322 La Jolla, CA 92037	1
DARPA Research Center Unit 1, Building 301-A Naval Air Station Moffett Field, CA 94035	1
Naval Postgraduate School Technical Library Monterey, CA 93940	1
Naval Underwater Systems Center New London Laboratory Code 313 New London, CT 06320	1

Distribution List (Continued)

Name	Number of Copies
Naval Underwater Systems Center Code 352 Code 3502 Newport, RI 02840	1
Dr. J. Anton Systems Control, Inc. 1801 Page Mill Road Palo Alto, CA 94304	1
Office of Naval Research Western Regional Office 1030 East Green Street Pasadena, CA 91106	1
Naval Ocean Systems Center Code 16 Code 724 Code 824 San Diego, CA 92152	1
Naval Surface Weapons Center White Oak Laboratory Code U-20 Silver Spring, MD 20910	1
Dr. Yaakov Bar-Shalom The University of Connecticut Department of Electrical Engineering and Computer Science Box U-157 Storrs, CT 06268	1
Mr. Conrad Naval Intelligence Support Center Code 20 Suitland, MD 20390	1
Naval Air Development Center Warminster, PA 18974	1
Naval Electronics Systems Command Washington, DC 20360 Code 320 Code 330 Code 350 PME-108	1