

**Assessment of the Adequacy of USCGS Data Tapes for
Transmitter Coverage Area Calculations**

by

Kenneth R. Baker

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of
Master of Science
in
Electrical Engineering

APPROVED:

C. W. Bostian, Chairman

T. Pratt

R. C. Robertson

March, 1987
Blacksburg, Virginia

**Assessment of the Adequacy of USCGS Data Tapes for
Transmitter Coverage Area Calculations**

by

Kenneth R. Baker

C. W. Bostian, Chairman

Electrical Engineering

(ABSTRACT)

This thesis describes the findings of research conducted as to the feasibility of using the U.S. Coast and Geodetic Survey (USCGS) digital topographic data base for the automated prediction of radio transmitter coverage area. The objective was to assess the adequacy of the USCGS data base to support radio-frequency path-loss and terrain clearance calculations. The assessment was made by investigating the basic features of the tapes and of the coverage calculation process. From this investigation, a prototype computer program was developed which calculated coverage areas from sample data tapes. The computer code for the program is included.

Acknowledgements

The author wishes to thank the members of his committee for taking the time out from their busy schedules to serve in that capacity. In particular, Dr. Charles Bostian for this opportunity to grow, his helpful discussions and for showing me that it is possible to be an engineer and a musician. I must thank Robin Rogers for lending her immense talent to create figures 16 and 17. I would also like to thank Robert Porter for his programming help and his timely moments of abstract conversation. Most of all, I thank Wendy Maroney for her tireless efforts in typing, pasting, and literally putting this thesis together. I must also thank each member of the Satellite Communications Group for their inspiration and friendship. Last but not least, I thank my wife, Nancy, for her support and understanding. Without her help and sense of humor, nothing would be possible. Mostly, I give thanks that she believes in me.

Table of Contents

1. Introduction:	1
2. The USCGS Data Tapes	2
3. Propagation Model	4
4. Coverage Area Calculations	7
5. Accuracy Considerations	9
6. Tape Accuracy [1]	11
7. Overview of Computer Method	13
8. Accessing the Data	15

9. Program FRSTPASS.PAS	18
10. The Coordinate System	21
11. Program TERPGEN.PAS	27
11.1 Introduction	27
11.2 Initialization and Operator Input	28
11.3 Databank Manipulation	33
11.4 Step Size	36
11.5 The Map Slice	38
11.6 Determining Elevation	41
11.7 Checking for Blockage	45
12. Coverage Area Program	50
13. Comparison to Paper Map	56
14. Conclusion	66
Appendix I	68
Appendix II	72
Appendix III	78
Computer Code for COVAREA.PAS	92

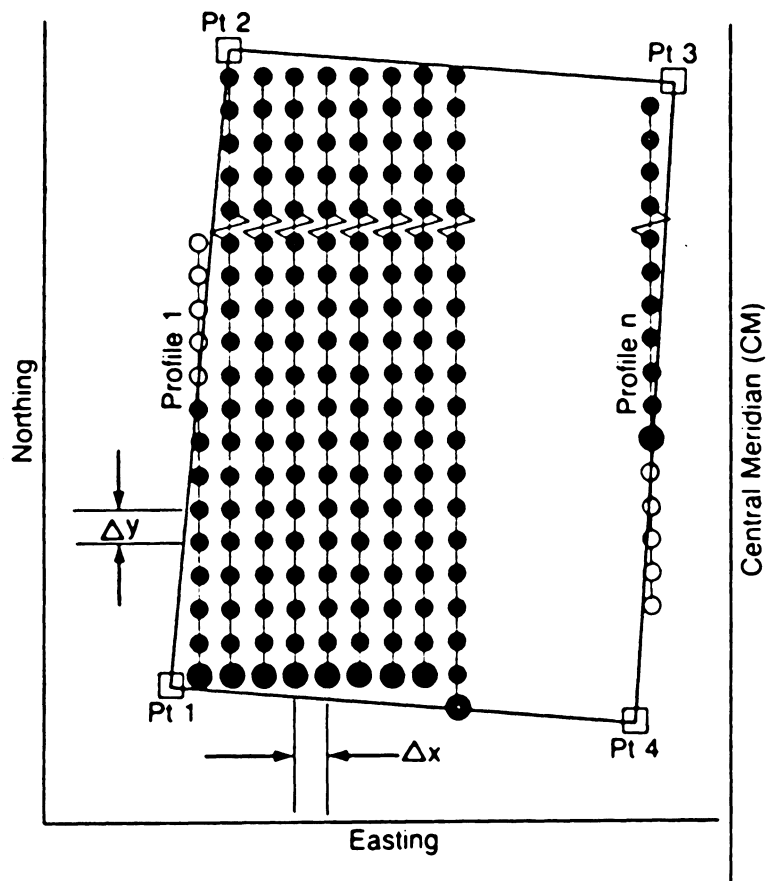
Appendix IV	100
References	109
VITA	110

1. Introduction:

This report describes the findings of the first phase of research into the feasibility of using the U.S. Coast and Geodetic Survey (USCGS) digital topographic data base for the automated prediction of radio transmitter coverage area. The objective was to assess the adequacy of the USCGS data base to support radio-frequency path-loss and terrain clearance calculations. We made this assessment by first investigating the basic features of the tapes and of the coverage calculation process. We then developed a prototype computer program to calculate coverage areas from sample data tapes. A discussion of these procedures follows.

2. The USCGS Data Tapes

Each of the USCGS data tapes used in this study contained elevation data corresponding to a standard 7.5- x 7.5-minute 1:24 000 scale topographic map. The neat lines of the map coincide with the latitude and longitude of the particular quadrangle. The data are organized around the Universal Traverse Mercator (UTM) grid system and consist of a regular array of elevations. The file structure allows elevations randomly located in an irregular array but only regular arrays have been produced to date [1]. In this investigation, we have only considered regular arrays. The data are ordered from south to north in profiles that are ordered from west to east. The spacing of the elevations along and between profiles is 30 m (see Figure 1). It should be noted that the profiles do not have the same number of elevations due to the variable angle between true north and the north of the UTM coordinate grid. Note also that the data base contains no information on ground cover that might be required for use in more refined propagation models.



- Δx = 30 meters (Easting)
- Δy = 30 meters (Northing)
- = Elevation point in adjacent quadrangle
- = Elevation point
- = First point along profile
- = Corner of DEM polygon (7.5-minute quadrangle corners)

(Example is a quadrangle west of central meridian of UTM zone.)

Figure 1. Structure of a 7.5-minute digital elevation model. (Reprinted from Allassal, Atef A. and Vincent M Caruso USGS Digital Cartographic Data Standards. Digital Elevation Models, U.S. Geological Survey circular 895-B, 1983)

3. Propagation Model

The propagation model used in this study was simple line of sight, including atmospheric refraction with a uniform refractivity gradient. The method of calculation follows that found in Chapter 4 of The Physics of Microwave Propagation by D.C. Livingston. [2] See particularly his discussion of parabolic-arc diagrams. Our method, as would be applied to a paper map, is described briefly below. How this method can be and was adapted for use by the computer using the USCGS data tape will be described when we discuss the software that was developed.

Given the transmitter and receiver locations, the transmit and receive antenna heights and the refractivity gradient, one can determine if there is a line of sight path between transmitter and receiver. Using a topographic map (or maps), the profile of the terrain between the transmit and receive locations can be determined. This is done by plotting the map elevations found along the straight line between the transmitter and receiver. This plot reflects the terrain along a flat earth. In a uniform non-homogenous atmosphere, a radio wave will travel a curved path. The most common graphical technique is to apply a correction to

the radius of the earth that will cause the ray from transmitter to receiver to appear as a straight line.

This is accomplished by finding the ratio of the effective earth radius to the actual earth radius. This ratio, called K, can be derived from the refractivity gradient using the formulas given by Livingston [2]. Examination of the atmosphere over the United States reveals a strong tendency for K to lie in the neighborhood of 1.3 or 1.4 over much of the country, particularly in the winter months. It has therefore become customary to adopt $K = 4/3$ as a standard working value. This yields what is known as the "4/3-earth atmosphere".

Once a value of K has been determined, the correction to the elevations on the flat-earth profile plot are applied using the following formula:

$$y = \frac{\ell_1 \ell_2}{2Ka} \quad (1)$$

where y = the effective change in elevation at a particular point
in the profile

ℓ_1 = the distance from the transmitter to the point

ℓ_2 = the distance from the point in question to the receiver

K = ratio of effective earth radius to true earth radius

a = true earth radius

Once the plot has been corrected to incorporate the effective curvature of the earth, the antenna heights are added to the plot. Then drawing a straight line from the top of the transmit antenna to the top of the receive antenna will indicate if the radio wave will collide with the ground.

If the ray has remained above the ground then that receiver location is assumed to be in the coverage area. If the ray has intersected with the ground then this type of plot can readily indicate what changes in antenna heights or location might make reception possible. Note that any change in receiver location makes necessary a

recalculation of the effective earth curvature as this depends upon the distance between the transmitter and receiver.

4. Coverage Area Calculations

The previous section provided a brief description of the graphical process used for years to determine if reception is possible from a given transmitter to a given receiver. In order to determine an area of coverage around a particular transmitter it becomes necessary to step out from the transmitter while recording the terrain profile along a particular ray. For each point along the ray, the effective earth curvature must be calculated and the plot checked to determine if the ray remained above ground. This process is then repeated and a yes or no decision made for each point of interest along the radial. Stepping around the transmitter with a sufficient number of radials will (after much effort) yield a plot of the locations with which the transmitter can communicate.

A computer could be used to make this type of calculation. What is required is only the ability to have the terrain profile of each radial digitized to some suitable quantization interval. With this information the computer can easily calculate the earth curvature, test whether the intermediate points of the profile are above or below a line drawn between the end points and thus decide if the end point is or is not able to "see"

the transmitter. The question then becomes: "To what extent can the USCGS data tapes provide a suitable terrain profile along any given radial?"

5. Accuracy Considerations

The digital tapes corresponding to the 7.5- x 7.5-minute 1:24 000 quadrangle provide, as stated earlier, elevations every 30m on an orthogonal array oriented with the UTM coordinate system. Elevation is given to the nearest meter on these tapes. Linear interpolation between any adjacent north-south or east-west points indicates that the maximum interval between elevations along a digitized profile would be 42.4 meters. This interval should be sufficient in all but the most rugged terrain. Linear interpolation between points will introduce some error, but this will not be excessive except in extremely rugged terrain given the data resolution. Consider, for example, the case of adjacent points, one on top of a cliff and the other at the bottom. Linear interpolation will have a smoothing effect on the terrain contour.

The largest error must occur in the case where adjacent points fall on either side of a peak or ridge. Interpolation will clip the peaks. This could indicate visibility when the receiver was actually shadowed by the peak. By similar reasoning, an error can occur when the receiver is located in a valley between adjacent points which fall on the

surrounding hills. Linear interpolation will cause the receiver to look higher than it is. Thus a path may appear to clear certain obstructions that in reality block it. An analogous error can occur with the transmitter. In this case, if the transmitter is located on a peak between two adjacent data points which fall on the slopes, then the transmitter will look lower than it really is. This could cause some locations to look shadowed which in reality are not.

This last error can be circumvented somewhat by allowing the operator to specify the elevation of the transmitter, if it is known. Specifying the receiver elevation when calculating coverage area is not practical, of course, since the receiver location is constantly changing. Note also that all of these errors are possible with a paper map as well. Paper maps give elevation only on contour lines that are quantized, typically to every 10 feet ($\cong 3$ meters). The elevations at the tops of some hills are given but otherwise the "peak" could be 3 meters in either direction from the last contour line. The true elevation of a receiver between contour lines can only be approximated. A paper map has the advantage though, in that it inherently sets bounds on the range of elevations between contour lines. If the elevation changes sufficiently then a new line will be drawn. If a hole (a well?) occurs in the 30 meters between data points of the digital map, it would never be known.

6. Tape Accuracy [1]

The accuracy of the 7.5-minute digital elevation model (DEM) depends partially on the scale of the source materials from which the data are obtained. The 7.5-minute DEM data originates either from map contour overlays that have been digitized or from automated or manual scanning of photographs usually taken at an average height of 40 000 feet (1:80 000 scale). The vertical accuracy of 7.5-minute DEM's derived from aerial photographs is highly dependant on the presence of natural and manmade features and the abundance of survey control for constraining the stereo-models. The vertical accuracy of 7.5-minute DEM's derived from contours is consistent with the accuracy of the contours on the source map.

The National Digital Cartographic Data Base (NDCDB) is, in reality, two separate data bases. The 7.5-minute DEM's are stored in one of the two depending on the vertical accuracy. Accuracy is either less than 7-meters vertical root mean square error (RMSE) or 7- to 15-meter vertical RMSE. The 7-meter RMSE was determined to be a reasonable

standard for DEM's derived from high-altitude photographs taken at 40 000 feet. This accuracy is attainable under a variety of terrain conditions and instrument constraints.

7. Overview of Computer Method

The method we chose to try to implement was analogous to the paper map method described above. Here, the area around the transmitter is explored one radial at a time. The radials then step around the transmitter until the desired area is plotted. The output is displayed on a CRT screen. A blank representation of the map is marked with the areas visible to the transmitter. Other methods and other outputs are possible. These will be discussed later.

The work was performed on an IBM-PC XT that was available. This machine was chosen for its friendly user environment, low cost of CPU time and readily available graphics. This decision resulted in some special data handling and storage needs that might not be required on a larger machine, the greatest limitation being the amount of map data that could be stored in active memory. Overcoming this occupies a large portion of the programs generated and slows run time because it requires a large amount of disk I/O.

The programming language chosen was PASCAL, specifically TURBO PASCAL ® by BORLAND International. A version utilizing the 8087 Math Co-processor, TURBO-87, was ultimately used in order to reduce run time. TURBO PASCAL was chosen due to its ease of operation on the PC; specifically its very fast compilation time, its data handling ability and graphics capability. The work could, theoretically, be carried out in any number of higher level languages. Little or no reference will be made to the specifics of the PASCAL language in the following description of this work.

8. Accessing the Data

The first thing required is access to the data on the tapes distributed by the USCGS in the standard digital elevation model tape format. The tapes themselves are IBM compatible: 9 tracks, 1600 characters per inch, unlabeled and with a logical record length of 1024. The information is written in FORTRAN format, ASCII characters. The data are organized into three sections, and the last section is optional. The first section is called the type A record. The second is a series of type B records. The third is called a type C record.

The type A record contains information defining the general characteristics of the DEM, including header information describing the DEM's name, boundaries, scale, units of measurement, minimum and maximum data values, number of type B records, and projection parameters. There is only one type A record for each DEM file, and it appears before any other record.

A type B record contains elevation data and associated header information. This header includes the sequence number of the record, the number of elevation points, the

ground planimetric coordinates of the first elevation point, and finally, the minimum and maximum elevations found in this particular record. All type B records of the DEM files currently being distributed by USCGS are made up of data from one-dimensional bands called profiles. Therefore, the number of complete profiles covering the DEM area is synonymous with the number of type B records in the DEM.

The type C record contains accuracy information on the data in the file. The existence of a type C record is indicated in the type A record. If it is non-existent, the accuracy is not known precisely. The accuracy depends on the data source for the DEM as discussed in Section 6 of this report. Neither of the two DEM's available for this study contained type C records.

The tape is easily read using the VA Tech Satellite Communications Group's Harris H-800 mainframe computer. A small program was written which copies the tape to disk storage in the same 1024 character records in which it was written. Transfer of files is possible between the PC environment and the Harris via a commercial terminal emulation program known as KERMIT. KERMIT will transfer data in packets of 96 characters or less per line. This presents a small problem, as the data file on the Harris looks like lines of 1024 characters. This problem is overcome by first rewriting the file on the Harris in lines sixty-four characters long. Sixty-four characters is chosen as it is a divisor of 1024. Once the data are broken into lines sixty-four characters long, they are transferred to the IBM-PC hard disk. Here it was discovered that any trailing blanks before a carriage return/line feed (cr/lf) character are deleted in the transfer process. This causes errors to appear in the data as numbers are run together that were once separated by blanks. This is easily corrected by a program that looks for cr/lf characters throughout the file. If a cr/lf is found at a position that is not a multiple of 64, then blanks are added to fill in the space. At the same time the cr/lf characters are removed from

throughout the file. The result is a duplication of the DEM on the PC hard disk; it is an ASCII data file in FORTRAN format.

9. Program FRSTPASS.PAS

An ASCII file is not the most efficient method of storing numerical data. Also, some way is needed to have random access to the type B records since they cannot all fit into the PC memory. From this need came a program called "First Pass". The computer code for FRSTPASS.PAS is given in Appendix 1. Its function is to read the ASCII data file and write it in binary format. A subsidiary function of First Pass is to read in the FORTRAN formatted real numbers in the double precision D-field specification and convert them to PASCAL format. PASCAL does not recognize the "D" format.

Specifically the ASCII data file was called simply DATA.ext; where .ext was a three letter extension identifying the map area. In this case .JOP was used to signify Joplin, VA and .QNT was used to identify the Quantico, VA, data base. When the program FRSTPASS.PAS is run, the program asks for the extension (.JOP or .QNT) and then proceeds to read through the file DATA .ext and output two files: LOGRECA .ext and PROFILE .ext.

LOGRECA .ext contains the information from the type A record, minus some unneeded information. It is an ASCII type file. Specifically it contains:

- 144 characters of header information
- DEM level code: gives scale of map
- elevation pattern code: regular or random
- ground reference code: indicates geographic, UTM, state plane or any of seventeen other possible reference systems
- zone code: UTM zone or zero if in geographic coordinates
- code for units of ground measure: radians, feet, meters or arc-seconds
- number of sides to the polygon formed by the map data: usually four
- a 4 x 2 array of the ground coordinates for the corners of the map, starting in the southwest corner and proceeding clockwise
- minimum and maximum elevation found in the DEM
- counter-clockwise angle (in radians) from the primary axis of the ground planimetric reference to the primary axis of the DEM local reference system: normally zero
- spatial resolution (X, Y, Z): normally 30, 30, 1 for the 1:24 000 scale map
- number of type B profiles

PROFILE .ext contains the heart of the DEM: the type B records. PASCAL allows this file to be constructed as a series of records. Each record can be accessed at random and each contains the necessary information from a type B record of the ASCII data file. This file is written in a binary format compatible with PASCAL. Each record in PROFILE .ext contains the following:

- sequence number of the profile
- ground coordinates for the first elevation data point
- ground coordinates for the last elevation data point
- number of elevation data points in the profile
- a one dimensional array containing the elevation data of the profile

Once the DEM information is assembled into this more accessible form, then the real work of putting this information to use can be accomplished. The first step is to create a program that can generate a terrain profile given any two points on the map.

10. The Coordinate System

One of the first considerations is to choose a working coordinate system. The options are either the geographic coordinate system, where location is specified via latitude and longitude, or the Universal Transverse Mercator (UTM) grid system, where location is specified via zone, easting and northing. The only real choice is the UTM system, for it has two main advantages over the geographic coordinate system. First, it is orthogonal. This simplifies the mathematics of calculating distances and locations. The second advantage is that the data are organized around the UTM grid. Therefore, once a location has been specified, it is a straightforward procedure to find the elevation data that corresponds to that point.

The UTM grid system is based on the assumption that locations on narrow sections of the earth's nearly spherical surface can be projected on to a flat, orthogonal grid system with very little distortion. In the system, the earth is divided into 60 zones running north and south, each 6° wide. They are numbered 1 to 60 starting at the 180th meridian

near the International Date Line, with zone numbers increasing to the east. Most of the United States is included in zones 10 through 19.

On a map each zone is flattened, and a square grid is superimposed on it. Figure 2 shows a zone with its shape somewhat exaggerated. As can be seen, when drawn on a flat map, the zone's outer edges are curves. These edges follow the meridian lines on the globe which, of course, are farther apart at the equator than at the poles. It is the accepted practice to terminate the UTM grid before it reaches the poles.

Any point in the zone may be referred to by citing its zone number, its distance in meters from the equator ("northing"), and its distance in meters from a north-south reference line ("easting"). Citing the zone number, easting and northing distinguishes one point from any other point on the earth.

Two important features of a zone are the equator, which runs east-west through the center, and the central meridian. The central meridian is a north-south line through the center of the zone. Easting and northing measurements are based on these lines in a manner described below.

By common agreement, there are no negative numbers for the eastings of points west of the central meridian and no negative numbers for northings that are south of the equator. For the eastings, this is accomplished by assigning to the central meridian an arbitrary value of 500 000 meters. As the widest portion of a zone (at the equator) is slightly larger than 600 000 meters, this causes the easting values to range from approximately 200 000 meters to 800 000 meters. Notice that the range of possible easting values narrows as the zone narrows toward the poles.

As for the northings of points, those points north of the equator are measured directly from the equator beginning at a value of zero and increasing. For points below the equator, the equator is assigned an arbitrary value of 10 000 000 meters. Points progressively southward are measured with decreasing but positive northing values.

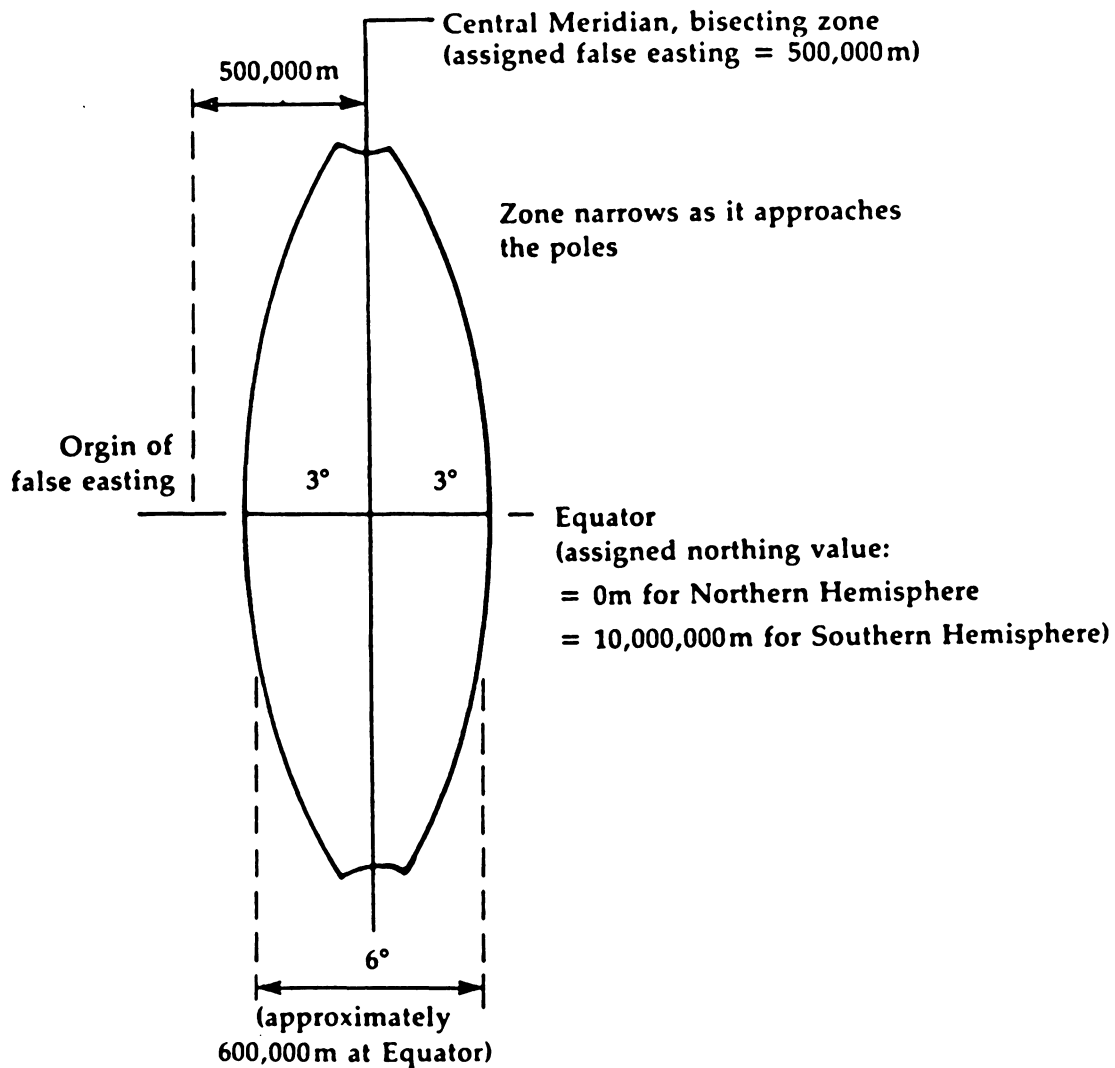


Figure 2. A UTM zone (Shape exaggerated). (Reprinted from Cole, W. P., "Preservation Planning Series," Using the UTM Grid System to Record Historical Sites, US Dept. of Interior, 1980)

For clarity, a minus sign usually proceeds northing values for points south of the equator. This arrangement is summarized in Figure 2.

The orthogonality of the UTM system is what should make the concatenation of maps a straightforward procedure. See Figure 3. To the limit that the computer hardware can hold and keep track of the separate DEM's, the data in each DEM will line up and concatenate easily. This is true as long as you remain in the same UTM coordinate zone since the UTM grids of different zones are not orthogonal. This is illustrated in Figure 4. This figure illustrates the extension of zone 18 into zone 17. This would be required in order to refer to all points in state A by a common grid system. This non-orthogonality is predictable and quantifiable and it should be possible to compensate for it. It does, however, seriously complicate the calculations when it is necessary to cross zones. The computer programs developed to date cannot concatenate maps nor cross UTM zone boundaries. If these abilities are required then this is an area for further development.

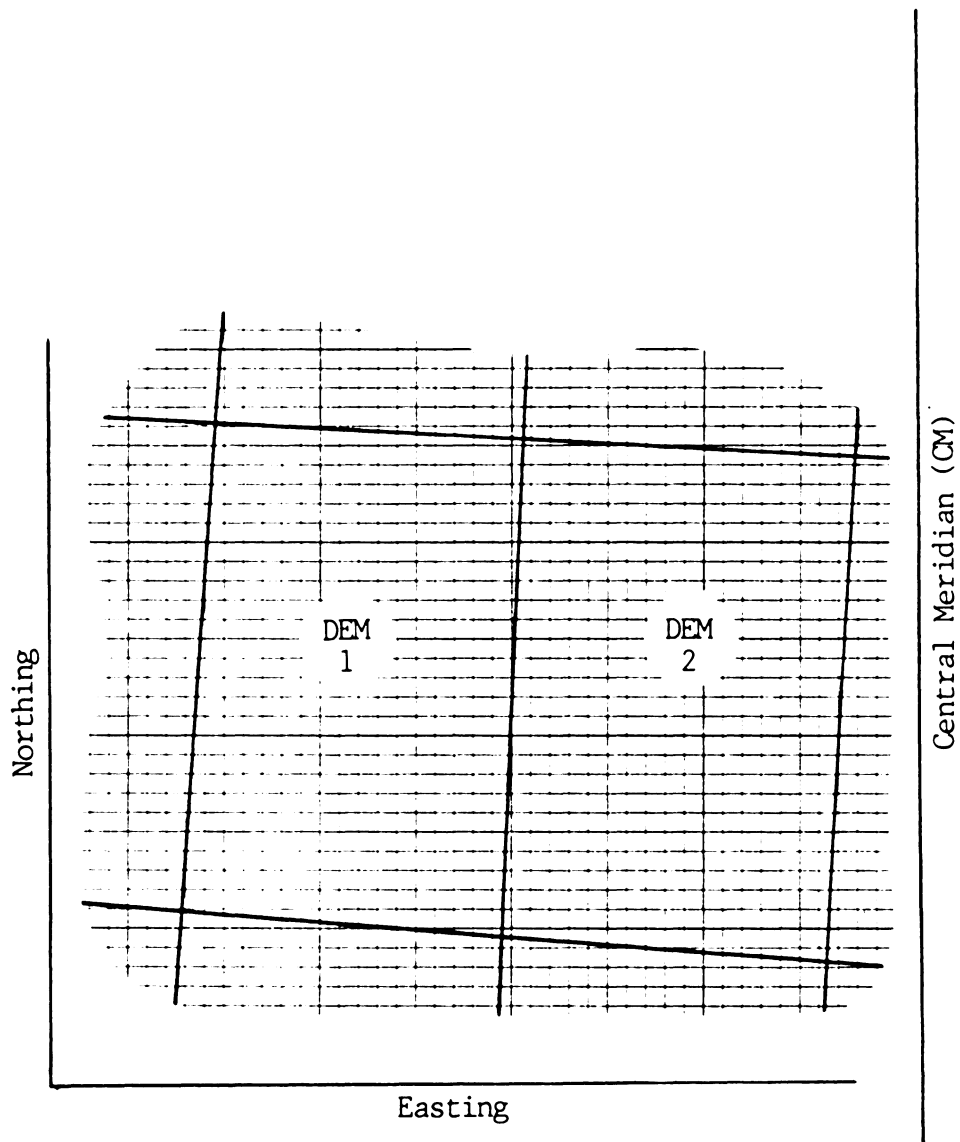


Figure 3. Two DEMs illustrating their common orthogonal grid.

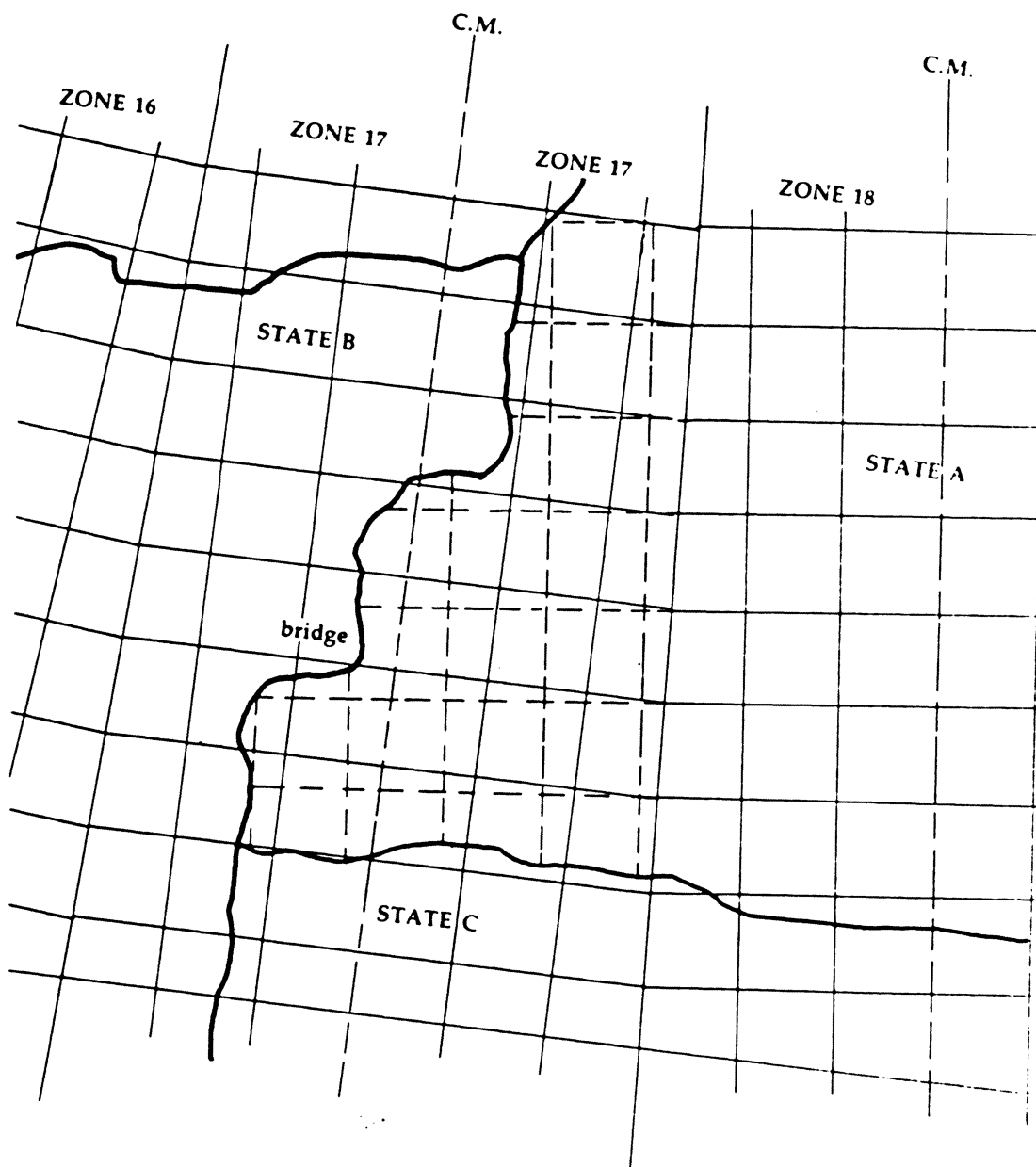


Figure 4. Zone 18 extended into Zone 17. (Reprinted from Cole, W. P., "Preservation Planning Series," Using the UTM Grid System to Record Historical Sites, US Dept. of Interior, 1980)

11. Program TERPGEN.PAS

11.1 Introduction

TERPGEN, short for TERRain Profile GENerator, is a program that takes the DEM as input (in the form of LOGRECA .ext and PROFILE .ext) and generates the terrain profile between any two points on the map. It is the basis for all the coverage area calculation programs produced. Understanding what it does and how it does it is the first step in understanding the methodology of the coverage calculation programs.

Generally TERPGEN, after acquiring the operator input, determines the straight line path between the two given points. Next, it decides on a step size and proceeds to use the data base to determine the elevation at each step along the line. When this is done, the transmit and receive locations, the antenna heights as well as an array containing elevations, earth curvature corrections and step sizes are placed in an output data file. Also in the file are the number of points in the array and an indication of whether the transmitter was visible to the receiver or not. This output file is ready for plotting by a plot program developed for this purpose.

With this background, some of the more pertinent details of how this is accomplished needs to be discussed. This will illustrate the procedures and tradeoffs involved and highlight areas for further development. Items of particular importance are: how the data base was handled, how the step size was determined, the interpolation method used for determining the elevation of a random point and the method used to decide if the ray remained above the ground.

11.2 Initialization and Operator Input

The simplified block diagram of the flow of TERPGEN.PAS is shown in Figure 5 and will be used to discuss the steps to generating a terrain profile from the database. The first block after start indicates the operator input of the three letter extension to tell the program which database will be needed for calculations. In a more automated setting with many possible data sets and particularly when there is need to cross DEM boundaries, it should be possible to have the program determine and call up the databases required without operator input. That was not necessary in this limited study.

The next block indicates the input from disk storage of the required type A record information from the file LOGRECA.ext. From this file the UTM coordinates of the corners of the map are set in memory. Also the number of type B records (profiles) that compose the DEM is stored. The two data sets used in this study both contained 375 profiles.

The other information in the type A record is read and checked at this point and immediately discarded or else it is stored and held for possible use later. Specifically: the header information is stored but not used. The DEM level code is checked to indicate

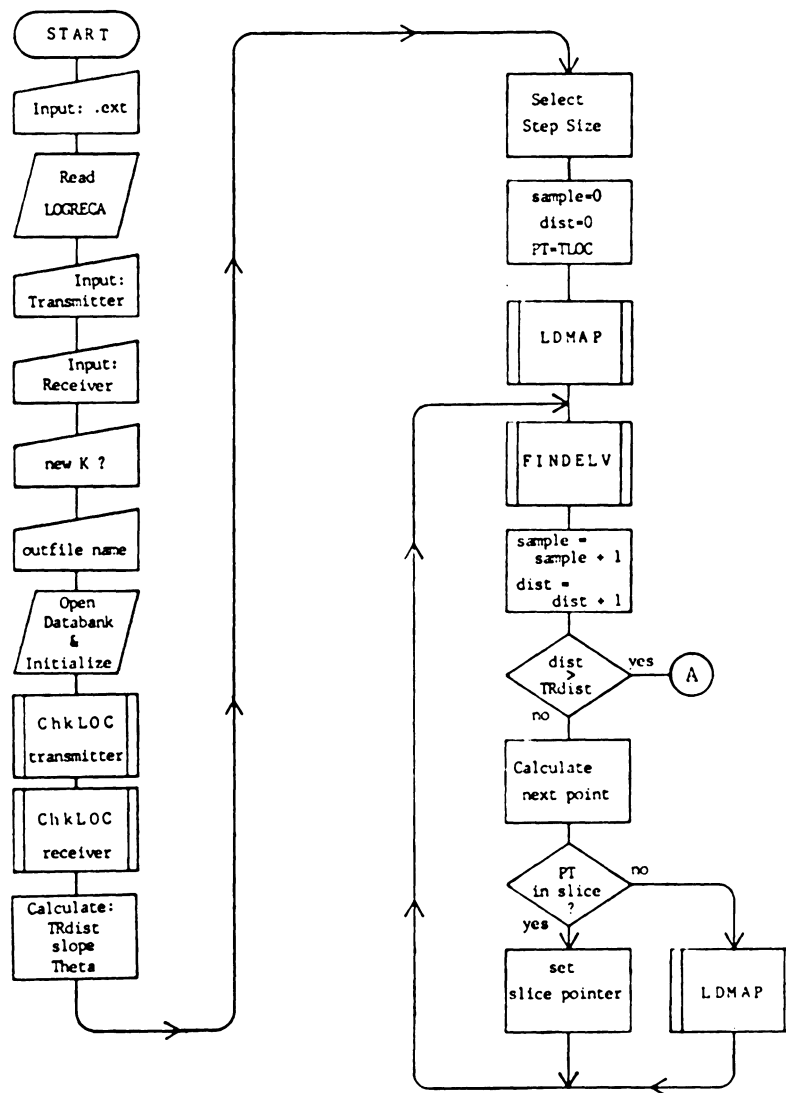


Figure 5. TERPGEN.PAS flow chart

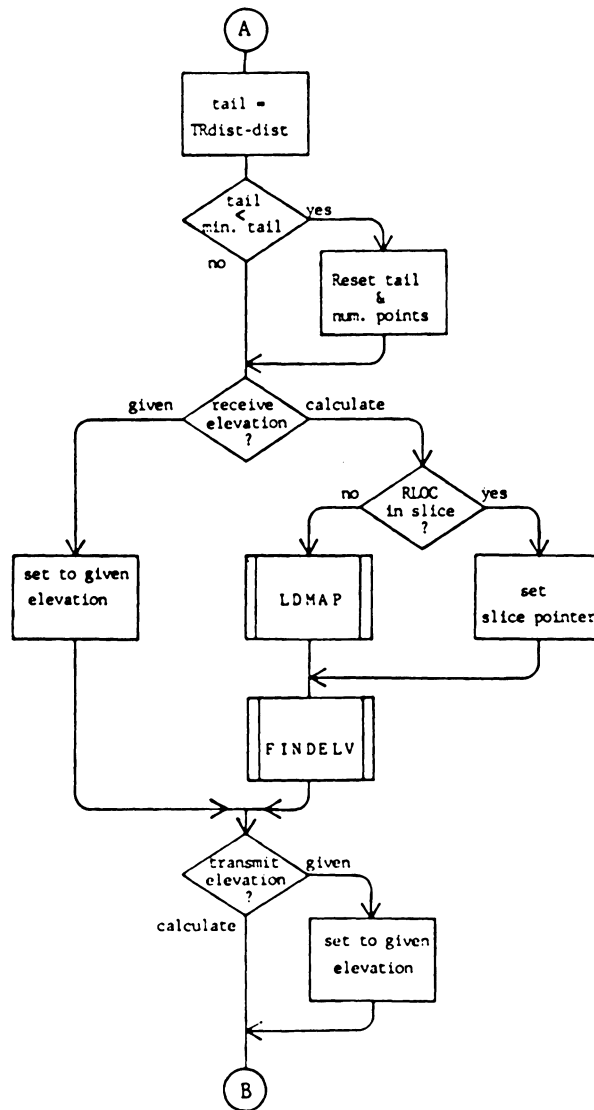


Figure 5. TERPGEN.PAS flow chart continued

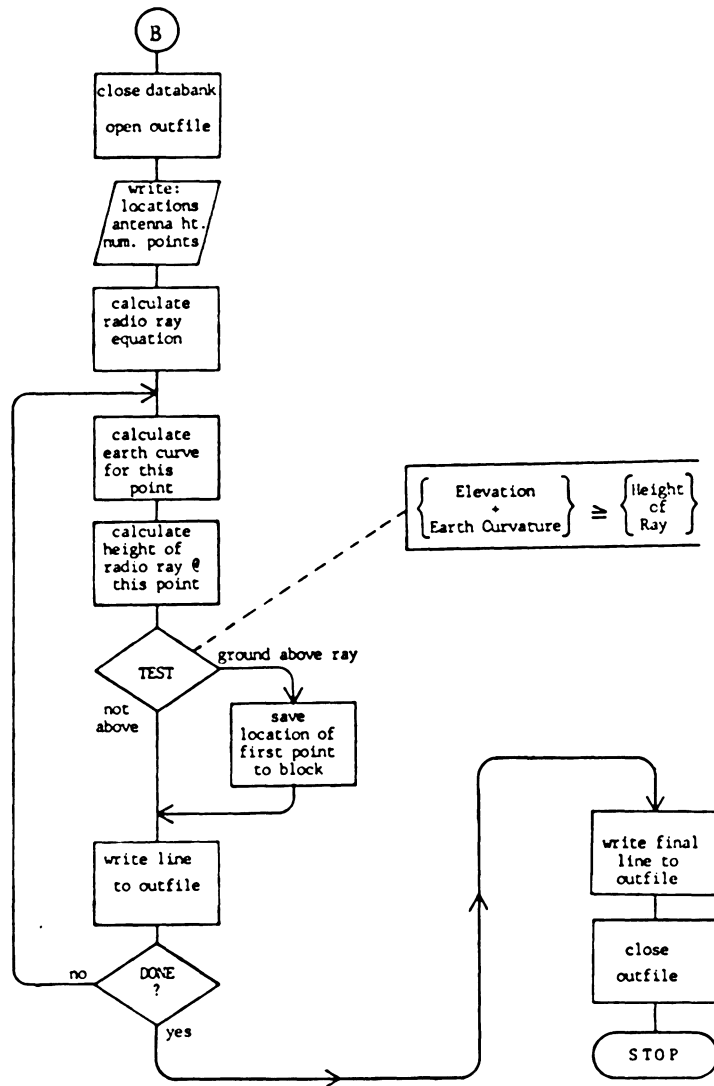


Figure 5. TERPGEN.PAS flow chart continued

a 1:24 000 scale, 7.5-minute DEM and then discarded. The elevation pattern code is checked to indicate the use of the UTM coordinate system and then discarded. (Note: all DEM's for the 1:24 000 scale use the UTM coordinate system and this code is present as a indicator for the situation in which a number of potential coordinate systems exist.) The UTM zone number is stored but not used at this time. The ground units code is checked to verify that distance is given in meters and then discarded. The elevation units code is checked to verify that elevation is given in units of meters and then discarded. The number of sides to the polygon formed by the DEM is checked to be equal to four and then discarded. The minimum and maximum elevations found in the entire DEM are stored but not used at this time. The angle from the primary axis is checked to be zero radians and then discarded. The code is read which indicates the existence of a type C record but is neither checked or stored. The spatial resolution of the X,Y and Z direction data is checked to indicate that there are 30 meters between points in the X direction, 30 meters between points in the Y direction, and that elevation is given to the nearest meter. If any of the above checks proves to be false then, at this time, the program will crash and display on the screen a code number indicating the first check which has not been met.

Failure of a database to pass some of these need not be fatal. They reflect the condition of the DEM's available for this study. The codes discussed provide necessary information so that one program can handle a variety of DEM's. In our situation, the programs were tailored to the DEM's available, making the above information and checks unnecessary. A conversion from feet to meters would be a trivial matter, for example, to incorporate in a more general program. Other conditions, such as a non-four-sided DEM or a random array of elevations, would have to be studied in order to understand the exact occasions under which such a condition arises and how it could be handled. Perhaps non-four-sided DEM's will never occur. Perhaps elevations given

in a random array will never be produced, although the USCGS allowed for the possibility when designing the database. The customer will need to give some thought to these issues when specifying the direction of any further development.

Returning to the flow diagram, the next two blocks indicate input by the operator of the necessary transmitter and receiver information. In order to calculate visibility along a given path, the following information is required for both the transmitter site and receiver site. First, the program requests the coordinates for the location of the site. At this time, the site location can only be entered in UTM coordinates. Next, it requests the antenna height. Finally it gives the operator the option of specifying the elevation at that site.

The next two blocks represent the last required operator input. The first block gives the operator the option of inputting a value of K. K is preset to a value of 4/3. If another value is desired then it is entered here. The next box requests the name of the file where the output will be placed.

11.3 Databank Manipulation

The next box indicates the accessing of the DEM profiles in the form of PROFILE.ext. This collection of profiles will here after be referred to as the databank in an effort to cut down on similar terminology. The records that make up the databank are numbered zero through the number of type B records minus one. Each record can be accessed randomly once the record number is known.

The coordinates of the first data point, the southwesternmost point, are also found and stored at this time. This point is used as a reference for calculating the record

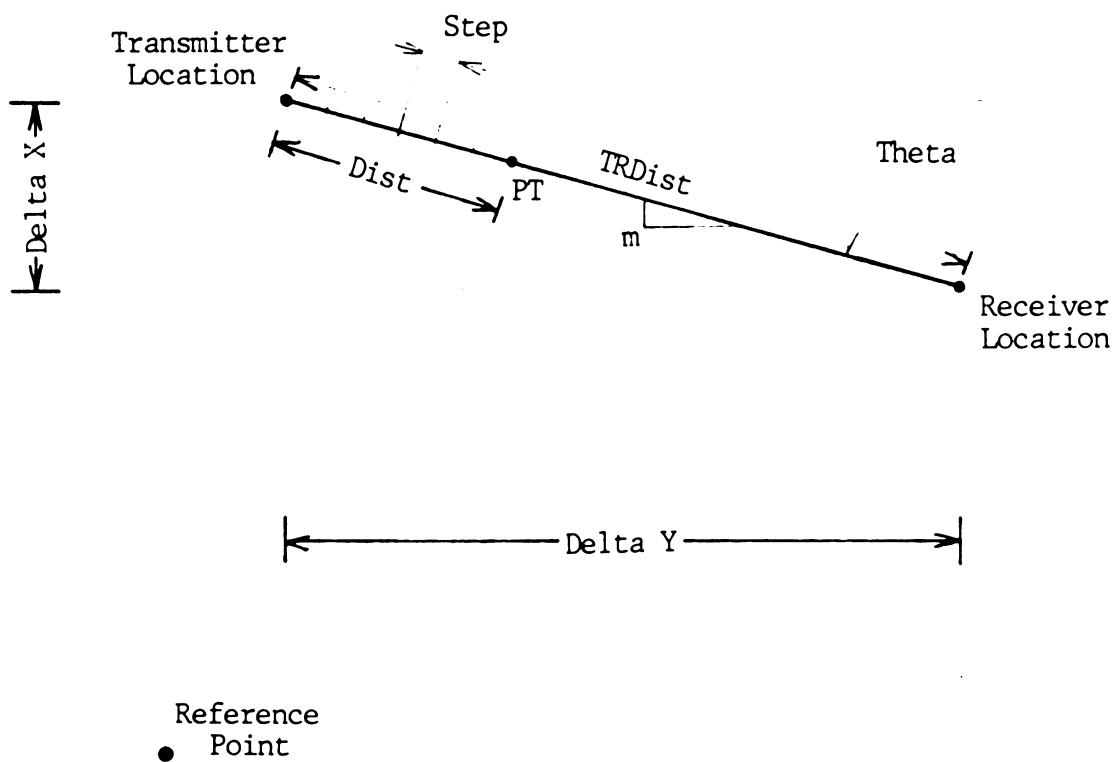


Figure 6. DEM illustrating the point calculation variables.

number of the elevation data profile required during program execution. In other words, all databank searches are keyed from this point. This is done quite simply since the profiles fall every 30 meters to the east of the reference point for a known number of profiles. To find the record number of the profile that lies just to the east of any given point; use the following formula:

$$\text{record number} = \text{truncation of } \left[\frac{\text{Pt}[E] - \text{Ref}[E]}{30\text{meters}} \right]. \quad (2)$$

Here, Pt[E] denotes the easting of the given point. Ref[E] denotes the easting of the reference point and "truncation of" denotes the discard of the fractional portion of the division.

The next two blocks represent two separate calls to a subroutine called ChkLOC. ChkLOC checks to see if the location of the given point is in the area comprising the DEM. If not, then the operator is prompted to reenter the coordinates. The first box indicates the checking of the transmitter location and the second indicates the checking of the receiver location.

The check is performed by first using Formula (2). If the resulting record number is negative or greater than the number of records then the point is not in the DEM. Otherwise the record number is legitimate and so the record is found and read. If the northing of the point is inbetween the northing of the first point and the northing value of the last point of that profile then, of course, the point in question is in the DEM area and the test is complete.

Now that the transmitter and receiver locations are known to be legitimate some basic quantities need to be calculated. This is indicated by the next box in the flow diagram. Three quantities are required: the distance from the transmitter to the receiver; the counter-clockwise angle, theta, of the radial from due east on the UTM grid

and third the slope, m , of the line from transmitter to receiver. Theta and the slope are required in order to calculate the points along the path between the transmitter and receiver. These quantities are illustrated in Figure 6.

11.4 Step Size

Next, the step size is selected. The step size is the incremental distance that the program moves along the radial towards the receiver. This is initially set to 30 meters. This value was a somewhat arbitrary choice and was selected only because the data are given in increments of 30 meters. The step size should be as small as possible, of course, in order to preserve as much of the detail of the terrain features as possible. It was decided that anything less than 30 meters was meaningless given the resolution of the data. The step size is also constrained by the available memory. If the distance from transmitter to receiver is great then the number of points, if taken 30 meters apart, can be enormous. We decided to keep the complete array of elevations comprising the terrain profile in main memory (rather than perhaps partially in off-line storage) in order that run time would not be compromised with additional disk I/O time. This decision necessitated our limiting the number of points allowed in a terrain profile. This number was arbitrarily selected at 200. This means that the longest profile taken in steps of 30 meters is 6 kilometers ($\cong 3.73$ miles). If the distance from transmitter to receiver is greater than this, then the step size is doubled to equal 60 meters. If the distance from transmitter to receiver cannot be covered in 200 points taken every 60 meters ($12\text{km} \cong 7.46$ miles) then the step size is automatically doubled again to equal 120 meters. As the

maximum distance, a diagonal from corner to corner, on a single DEM is approximately 10.2 miles, then this is the maximum step size for a single DEM.

The issue of step size is critical as it is fundamental to the terrain profile resolution. Over long distances, covering perhaps several DEM's, terrain profiles quantized to 30 meter steps could become prohibitively large. What is too large is of course a function of the hardware environment and the time available for computation. Under the present method, if every point along the profile is examined then the computation time is proportional to the factorial of the total number of points possible on a given radial, as will be seen from the description of the calculations. Long distance radials can be handled in at least the following two ways.

One solution would be to increase the step size. The obvious disadvantage here is the loss of definition in the profile. One other problem is a phenomenon we have chosen to call the "lever effect". To a certain degree, the terrain closer to the ends of the radial is more important than the points in the middle. As the distance from transmitter or receiver becomes small, the terrain features become more effective in blocking the signal. This has an analogy with the action of a lever. For example, a 100 ft. hill immediately in front of a transmitter might effectively block the signal while the same 100 ft. hill in the middle of a 5 mile path would probably go unnoticed. Therefore the best way to increase the step size to accomodate long paths would be to do so in the middle and maintain a finer resolution near the ends of the path.

A second possible solution to the long distance radial problem would be to keep track of only the highest peak(s) in the terrain, particularly the peak that can be identified as the one causing blockage to a certain point of reception. As the radial is extended during coverage area calculations, the effect of the peak alone can be checked along with the new terrain information for that portion added to the end of the radial. If a new peak is identified, then it is incorporated into the algorithm as required. The basic concept

here is that, as the radial grows outward, only the part or parts of the previous information is retained that can be identified as limiting transmission. In that way only the important points are examined while the trivial ones are discarded. The disadvantage here is that this discarded information would be necessary if such phenomena as knife-edge refraction were desired to be incorporated into the propagation model. Perhaps some combination of the two methods would yield an optimum method of calculation with respect to accuracy, computation time and the ability to handle long transmission paths.

11.5 The Map Slice

Continuing on with the TERPGEN flow chart, we come to the main loop of the program where the terrain profile along the given radial is calculated. The index of the array of elevations, *sample*, is initialized to zero. Elev[*sample*] is the notation for the array of elevations and Elev[0] is the elevation of the transmitter location. For future reference, Elev[*lastpoint*] will be the elevation of the receiver. Hence all of the elevation points that make up the terrain profile of the radial are numbered 0 to *lastpoint* where *lastpoint* is less than 200 as discussed above. The distance along the radial, *dist*, is initialized to zero. Then the current location variable, PT, is initialized to the location of the first point: the transmitter.

The next box designates the first call to a subroutine called LDMAP; which is a contraction for "load map". This subroutine loads a small section, a slice, of the digital map into memory from the databank on disk storage. A map slice is illustrated in Figure 7. The elevations of this slice are stored in a matrix arrangement in memory. The

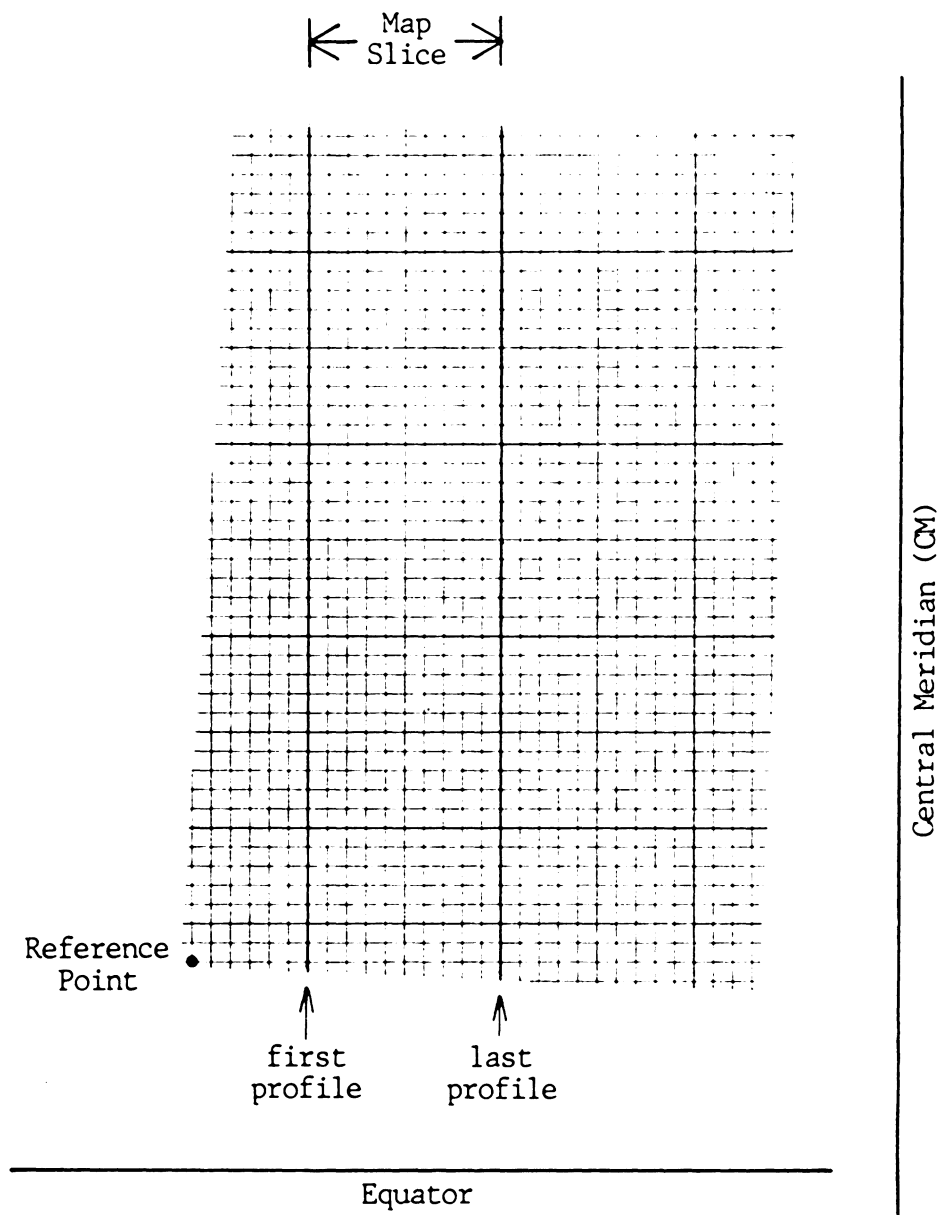


Figure 7. A DEM with map slice.

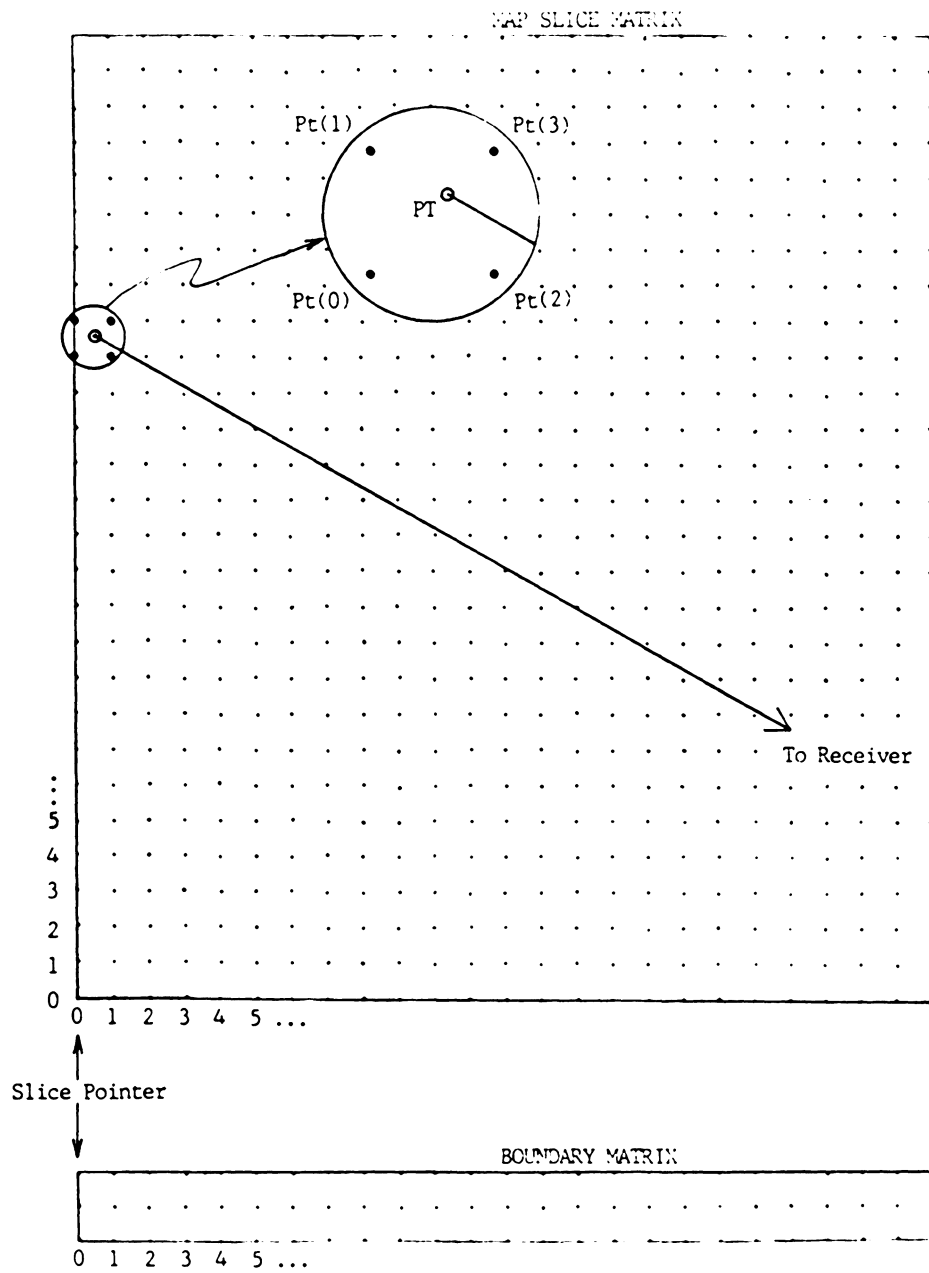


Figure 8. Map slice matrix and boundary matrix.

program loads a particular slice based on the location held by PT. The slice is loaded such that PT, in this case the transmitter location, is either between the first and second map profiles or between the next to last and last map profiles loaded depending on the direction to the next point.

The size of the matrix is limited by the available memory. In the case of the PC using Turbo Pascal, the slice matrix is about 55 to 60 profiles wide. The variance is due to the varying amounts of storage space required for other working variables in the various programs. The width of the slice is also dependant on the number of points in the longest profile. This was 463 in the case of the DEM's available. Once the elevation data are placed in the matrix, they can be recalled instantly by addressing the appropriate row and column. This matrix is illustrated in Figure 8. Also shown is a co-matrix which is loaded simultaneously and has an index which matches that of the elevation matrix. This matrix contains the boundaries of each corresponding map profile. That is, it contains the location of the first and last point of each corresponding map profile. This "boundary matrix" is used to locate points in the slice matrix and to check if the next point, proceeding out the radial, has left the current slice or left the map entirely.

11.6 Determining Elevation

Once the map slice is loaded, the flow proceeds to a call to a subroutine named FINDELV which determines the elevation of the location PT. This is done by interpolating between the four points that surround PT. This is illustrated in Figure 8 also. Figure 9(a) illustrates the manner in which the interpolation is performed. First the elevation at point

A is found by linearly interpolating between elevations at points 0 and 1. Similarly the elevation at point B is found by linearly interpolating between points 2 and 3. Finally, the elevation at PT is found by linear interpolation between points A and B. Once all of the interpolating is complete, the final result is rounded to the nearest meter. This method uses all four corners of the box that surround a given point, and the result will be a combination of the attributes of all four.

An alternate method of interpolating the elevation of a point is illustrated in Figure 9(b). It is yet untried. This method would use the elevations of three corners of the box surrounding the point in order to form a plane. This could be accomplished by first extending the radial on either side of PT until it intersects the sides of the box. Then, as shown in the example of Figure 9(b), the elevation at A would be found by interpolation between points 1 and 3. Next, the elevation at B would be found by interpolation of points 2 and 3. Finally, the elevation of PT would be found by interpolation between points A and B with the result rounded to the nearest meter. This would effectively find the elevation of the point on the plane formed by the elevations at 1, 3 and 2. It would be interesting to compare elevations of random points found from the first method to that of the second to see if there would be significant differences.

Now that we have the elevation of the first point, PT; the index, *sample*, is incremented and the distance traversed along the radial, *dist*, is incremented by the step size. This is shown in the flow chart of Figure 5 by the box following FINDELV. At this point *dist* is tested to see if the distance traveled thus far along the radial is greater than the distance from transmitter to receiver. If it is, then we leave the loop in order to calculate the elevation of the last point (the receiver location) and finish the program. If *dist* is less than the distance from transmitter to receiver then the program flows in the "no" direction and continues the loop.

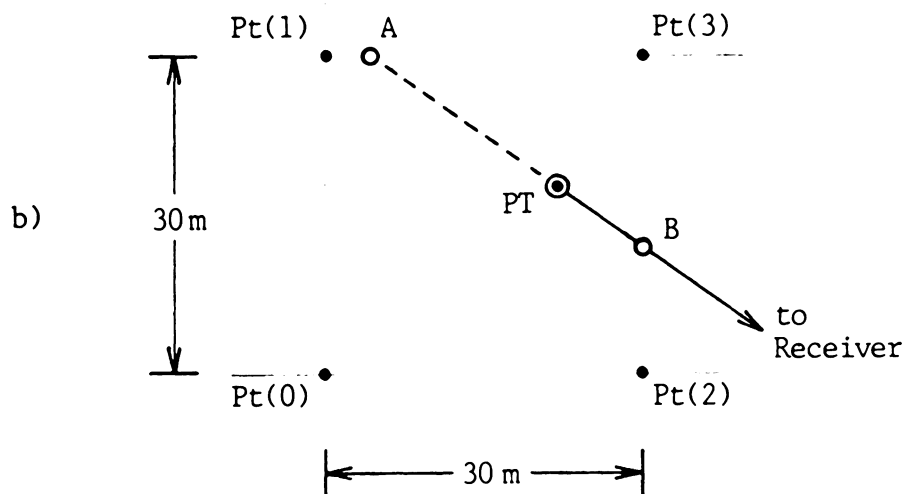
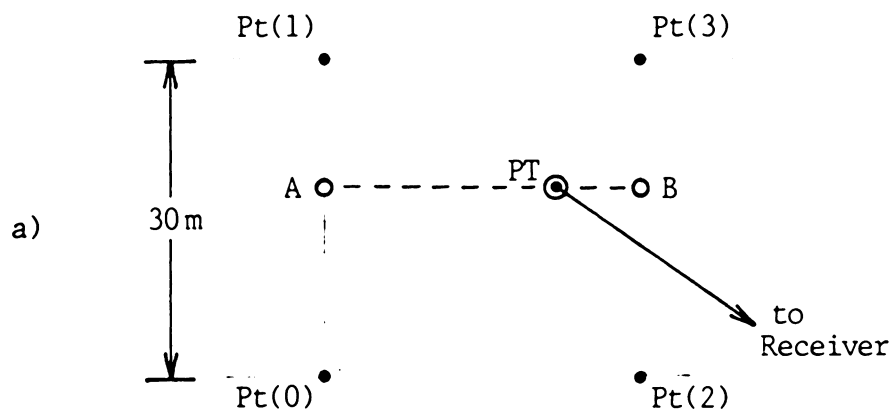


Figure 9 (a & b). Two methods of interpolating elevation from the DEM.

If this is the case, the program proceeds by first calculating the location (easting and northing) of the next point along the radial, which then becomes PT. Next it tests this location to see if it is in the map slice that currently resides in memory. If it is, then an index called the "slice pointer" is incremented as necessary to point to the column of the slice matrix which is just to the west of the new PT. The subroutine FINDELV requires this pointer to be reset in this manner in order to perform its function. If the new PT along the radial is not in the current map slice then the subroutine LDMAP is called in order to load the slice matrix with a fresh slice of elevations from the databank. LDMAP returns with the "slice pointer" set to the proper position.

With the "slice pointer" set, flow passes to the top of the loop. The loop proceeds first by finding the elevation of PT (using FINDELV) and continuing along the radial in the manner described above. The distance along the radial is incremented each pass around the loop, and the elevation at the new point is found and placed in the array *Elev[sample]* until the distance traversed equals less than one step distance from the predetermined transmitter to receiver distance. In other words, the process continues until the next full step increment will place the new PT past the receiver location. This distance from the point of the last full step to the receiver location is called the tail. The first flow box out of the loop shows the calculation of the tail as the difference between the transmitter to receiver distance and the distance traversed along the path so far.

Next, the tail is tested to insure that the length of the tail is not so small as to be negligible. For example, having a tail of 1/2 meter while the distance from transmitter to receiver is 6 kilometers would be silly, for in such a short distance no significant change in elevation or reception could be expected. In order to prevent this, the tail is tested against some minimum tail length, currently set at 5 meters. If the tail is less than the minimum length then the last point calculated by the loop is discarded and the tail

becomes the step size plus the original small tail. Otherwise, the length of the tail will be somewhere between the minimum length and the current step size.

Once the final distance to the receiver location is decided, the next block in the flow chart shows the decision about whether it is necessary to calculate the elevation of the receiver location. If the receiver elevation was entered by the operator, the flow proceeds to the left, and the last element of the array, *Elev[lastpoint]*, is set equal to the entered elevation value. If it is necessary to calculate the receiver elevation, then flow proceeds to the right and the elevation is calculated from the slice matrix as before.

Now that the last elevation has been determined, a similar decision is made concerning the transmitter elevation. Note that the transmitter elevation is always calculated from the databank. Only at this time, if requested by the operator, is *Elev[0]* set equal to the operator's input. At this point the databank is closed to further activity, since it is no longer required. The program has now constructed an array of integers representing the terrain profile between transmitter and receiver. If plotted, they might resemble something like the example shown in Figure 10. The step sizes between all points are equal and constant except for the last which has the variable length *tail*.

11.7 Checking for Blockage

The next phase of the program flow is the output phase. It is during output that the corrections for earth curvature are made and the path checked to see that the ray remains above the ground. These things happen simultaneously as the output file is written. The output phase begins with the opening of the output file. Then the easting and northing of the transmitter location, the transmit antenna height, the easting and

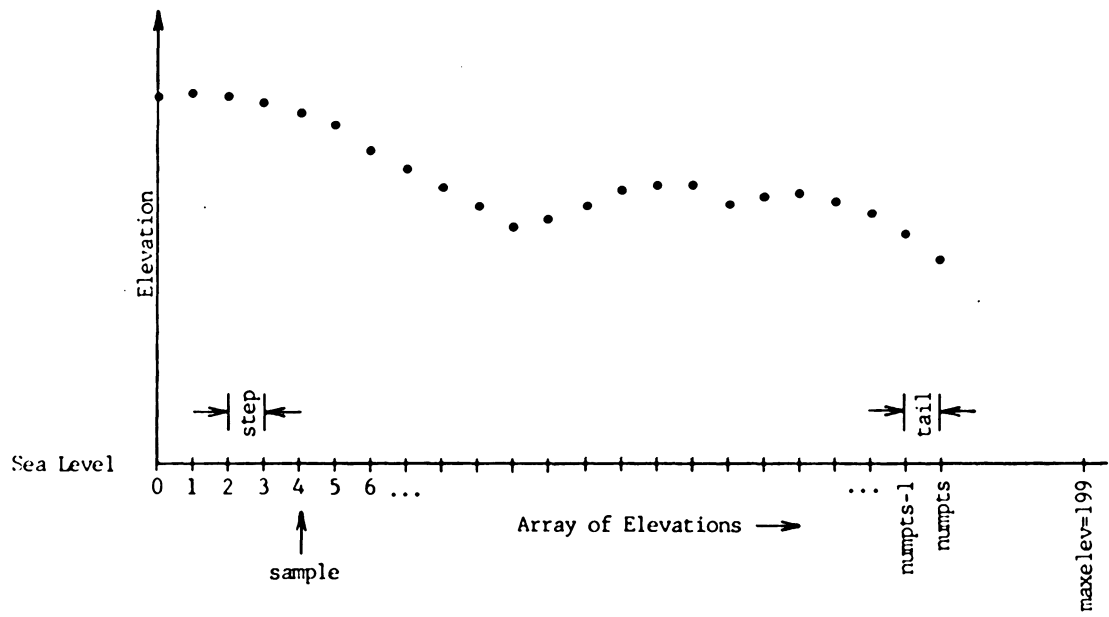


Figure 10. A digitized flat earth profile.

northing of the receiver location, the receive antenna height, and the the number of elevation points are written in that order to the disk.

The next box indicates the construction of the line from the top of the transmit antenna to the top of the receive antenna that represents the actual straight line ray. This is illustrated by Figure 11. What is determined at this point is the slope of the ray, MRAY, and the source elevation. The source elevation is the sum of the transmit antenna height and the transmitter elevation. These two quantities, the slope of the line and the source elevation, specify an equation for the line defining the ray. This allows the calculation of the height of the ray above sea level at any given distance from the transmitter. You will notice that the example shown in Figure 11 illustrates a ray which is blocked by the terrain.

With this information at hand, the flow enters a loop which will check each point of the digitized terrain profile to see if it blocks the ray. This is done by first calculating the earth curvature correction for the point by using (1). Next, the height of the ray above sea-level is calculated by:

$$\text{ray height} = (\text{MRAY}) \times (\text{DIST}) + (\text{source elevation}) \quad (3)$$

Now a comparison is made, as illustrated in the flow diagram. If the elevation of the point plus the earth curvature correction at that point is greater than or equal to the ray height at that point, then the receiver is blocked. The first time this happens the flow proceeds to the right of the decision box. This box indicates the retention of the number of the point of first blockage. The location of the first point to block the ray is saved in order to have the plot routine indicate where blockage has occurred.

From here the program proceeds to write a line to the output file. The line contains the point number (from zero to *lastpoint*), step size, elevation of the point and its earth curvature correction. Flow continues around the loop until all the points, along with their

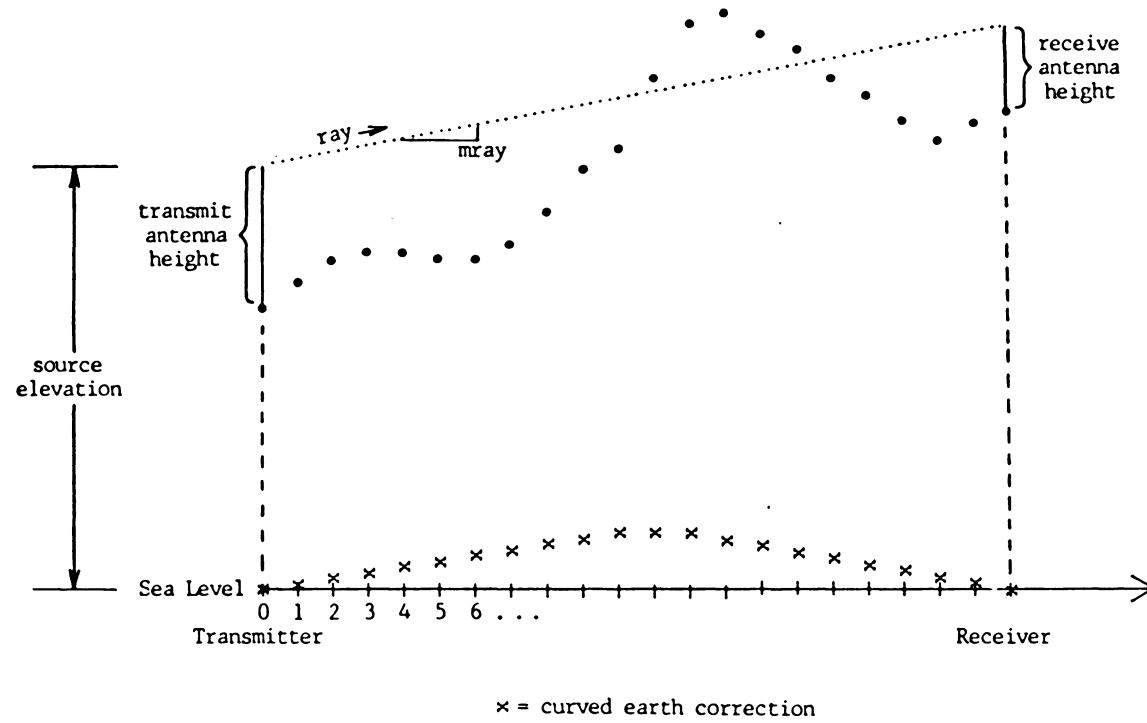


Figure 11. Complete digitized profile corrected for atmospheric refraction.

earth curvature corrections, are written to the file. Once that is complete then the number of the point that blocked the ray is output to the file. If this number is equal to the number of the points in the terrain profile then the receiver location was not blocked.

At this point TERPGEN.PAS terminates. What has been produced is a disk file containing the locations of the transmitter and receiver as well as their respective antenna heights followed by a list of integers and real numbers indicating the terrain elevation profile and earth curvature corrections. Also in the file is an indication of whether the receiver was blocked by the terrain. The computer code for TERPGEN.PAS is given in Appendix 2.

12. Coverage Area Program

TERPGEN.PAS is a program that can take the DEM and from it produce the terrain profile between any two points in the given area. It can also determine if the receiver can or cannot receive the transmitted signal. These qualities allow this code to form the basis for a program that can make a coverage area determination from the DEM. The concept, as described elsewhere in this report, is to extend radials out from the transmitter while checking each point as a potential receiver location. If this process is continued around the transmitter via adjacent radials, then an area of coverage will be described. The program developed to do this is called COVAREA.PAS and it is based on TERPGEN.PAS. An overview of how it works is described in the following paragraphs. The details of the program are given in the Appendix 3 along with a flow chart and the computer code.

COVAREA.PAS begins in a given direction from the transmitter by extending a radial one point at a time. Each new end-point on the radial becomes a potential receiver location. The previous points make up the terrain profile for that receiver location. These

points between the transmitter and the endpoint, are examined in a manner similar to that described in Section 11.7. If no blockage to reception is found, then that endpoint is in the receive area. If a blockage occurs, then it is not.

Once a determination has been made then the radial grows by adding another point to the end. This now becomes the proposed receiver location and the previous points are checked as before. This process continues until the new endpoint reaches the end of the map. When this happens a new radial, at a specified angle from the first, is started. The process continues until a specified arc around the transmitter has been checked or until 360° has been traversed around the transmitter.

There are several issues which arise when developing a scheme such as that described above. For example, when a particular point has been determined to be in the receive area, that is all you know: this "point" is in the receive area. To fill a finite area with points is a task of infinite proportions. Therefore some assumption must be made about the area immediately surrounding a given point. Our assumption was that if a given point is determined to be in the coverage area, so too is the square area around the point whose sides are the length of the current step size. This is illustrated in Figure 12. The sides of the blocks are always oriented north-south and east-west. Squares were chosen for ease of implementation with the current hardware and software. Circles of radius equal to one-half the step size are another possibility.

Ideally some area and geometry needs to be found that is suitable to the reality of the process. This would undoubtedly take the form of some statistical relation. For example, given that a point is in the receive area then at what general distance around that point is there a (for example) 50% probability of still being in the coverage area? Once some handle has been obtained on this matter then coverage area can be calculated more effectively. Knowledge of this "area function" would have bearing on step size and optimum angle between adjacent radials as well.

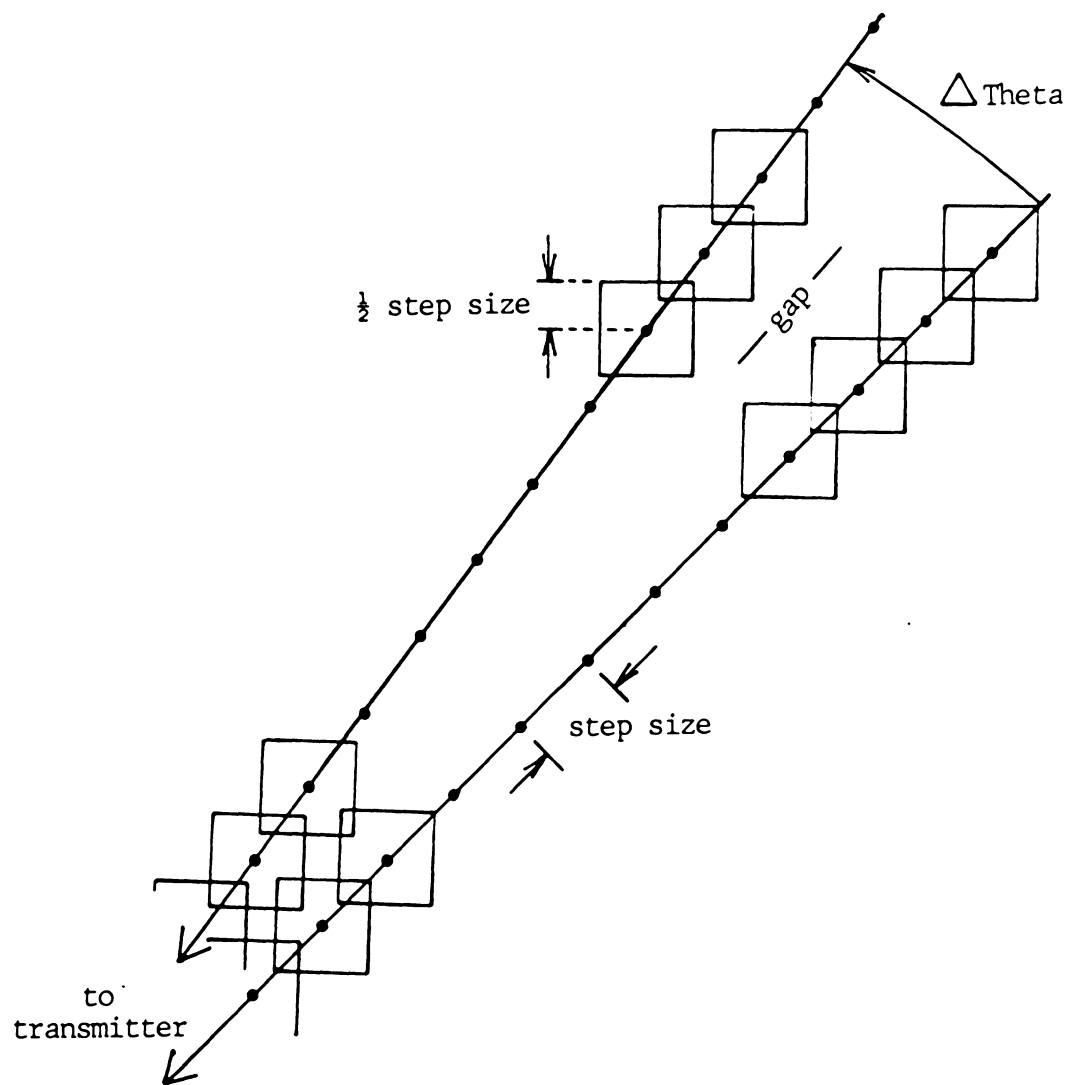


Figure 12. Detail of CRT plotting.

Also seen in Figure 12 is another issue of concern when adopting this method of coverage area determination. That is the question of how far apart should each radial be. The answer depends on several things. As implied above, it depends on the size and shape of the assumed blocks of coverage since it would be ideal for the blocks to meet in some way in order that a contiguous area of coverage be formed, if that is the case. There is also a dependence on the length of the radials since the farther out the radials go the more separated they become. A dependence exists also with the resolution of the output device or method. The radials need only be as close as any gaps can be resolved. The angle between radials used for this study was one degree when the area of the entire DEM was being examined. The reasons for this will become clear after a discussion of the programs output.

Output from this process is to the PC screen. During program initialization, the screen is cleared and the borders of the DEM are drawn as straight lines. When a point is found to be in the reception area of the transmitter, then a block is drawn on the screen that corresponds to the step size as described above and shown in Figure 12. The step size starts at 30 meters and is doubled in a manner similar to that described in Section 11.4. (The process is also described in Appendix 3.) Therefore the areas toward the far ends of the radials are formed from blocks 60 meters on a side. This counteracts the effect of the radials separating with distance away from the transmitter but causes the much weaker assumption to be made that 60 square meters around a given point is also part of the receive area.

When a point is not in the area of coverage, then nothing is drawn. The result, after stepping through several closely spaced radials, is the formation of shapes on the screen inside the map outline that correspond to areas of coverage. An example of this output is shown in Figure 13. The dark areas inside the neat lines correspond to the areas covered by the transmitter. In this figure the transmitter location has been

highlighted by hand for clarity. At present, when the process is complete, only a small white dot is placed at the transmitter location and some prior knowledge is needed to identify it. Certainly this is an area for improvement in the future.

Also visible in Figure 13 are the gaps that appear between adjacent radials at distances far from the transmitter. These become visible approximately 5 kilometers out. One solution might be to interlace radials like these with radials that start plotting 4 to 5 kilometers out from the transmitter. These delayed radials would start to plot in order to fill in these gaps.

One other problem that can not be observed directly from Figure 13 is the effect of screen resolution. The PC screen is made up of rectangular pixels. At this scale the size of a 30 meter square block is on the same order of magnitude as the individual pixels. The result is that as blocks are written to the screen they are not always square. This effect is particularly evident when drawing on radials that are just off vertical or just off horizontal. The problem does tend to compensate itself for when the adjacent radial is drawn adjacent pixels are lit and the result tends toward what is desired.

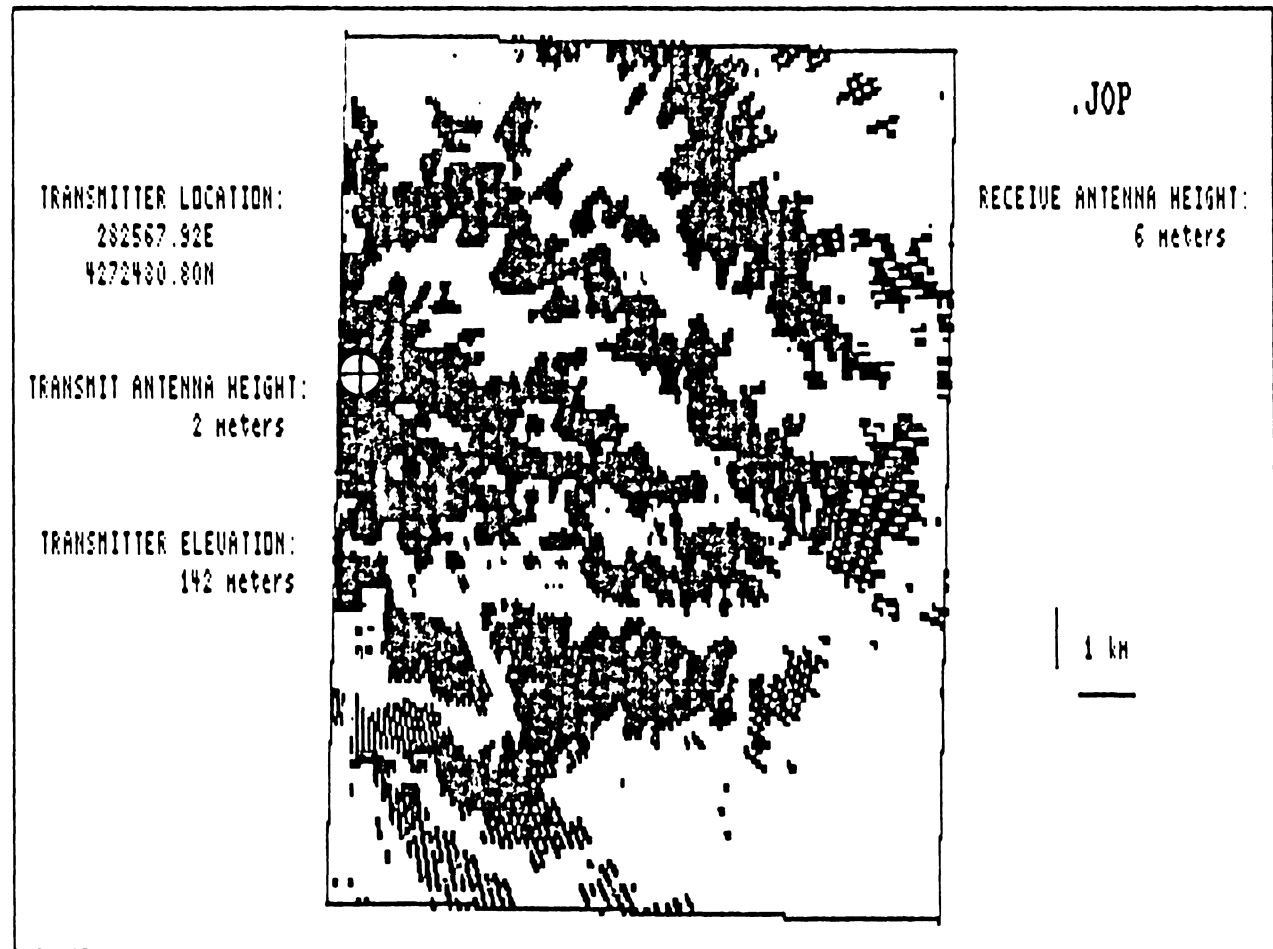


Figure 13. COVAREA.PAS output.

13. Comparison to Paper Map

Developing a program like COVAREA.PAS proves the feasibility of using the DEM to make coverage area calculations. The question now is: "How does this compare to an area of coverage calculated by hand?" Some comparison needs to be made of the output from this automated method to the coverage area as derived from a paper topographic map. Such a comparison will show to what extent the information on coverage area derived the painless automatic way is as good (or better?) as the tedious, time consuming alternative; that of plotting and calculating radials by hand using a paper map. Indeed, rarely are "areas" of coverage determined from a paper map due to the intense human effort involved. Commonly only selected transmitter to receiver locations are checked. In this report we make no attempt to determine how well either of these methods predicts the true coverage area.

In order to compare the computer method to the paper map a new program was developed. It is called ONE2ONE.PAS. It works exactly the same as COVAREA.PAS except that the output is on the same scale as the paper map, hence the name: "one to

one". This program allows the operator to select a smaller area anywhere inside the area of the complete DEM and the output will display the areas of coverage from the selected transmitter sight that fall inside this "window". In other words, it's like selecting a piece of the output shown in Figure 13 and enlarging it to the same scale as the paper map. The computer code for this program is given in Appendix 4.

The ability to make map scale coverage area plots provides a vehicle with which to compare the computer calculated coverage area to the area calculated by hand. Toward that end, a coverage "area" was calculated using the 1967 edition of the Joplin, VA 7.5-minute quadrangle map. The transmitter location selected was 282 567.92E, 4 272 480.80N. This is the same transmitter location used to generate Figure 13. The elevation at this point is given on the paper map as 467 ft. above sea level ($\cong$ 142 meters). This number (in meters) was entered as the transmitter elevation when generating all coverage areas for this location. This location was selected primarily because it seemed to be the highest point in the Joplin, VA, quadrangle.

The area selected to examine at the one to one scale was the area at the top left hand corner of the Joplin map. Figure 14 illustrates the location of this window as well as its relation to the transmitter location. The one to one computer plot of this area is shown in Figure 15. (This figure is not at map scale. It had to be reduced for reproduction.) The spacing of the radials which made this plot was 1/2 degree. The transmit and receive antenna heights used were 2 meters and 6 meters respectively. These antenna heights were used to produce Figures 13, 14 and 15 as well as the plot made by hand.

The hand plot was made by first drawing radials on the map. These were lines that started at the transmitter and moved through the selected window spaced at approximate 3° intervals. This resulted in 27 evenly spaced radials. Next, for each radial the distance from the transmitter to each contour line that the radial crossed or touched

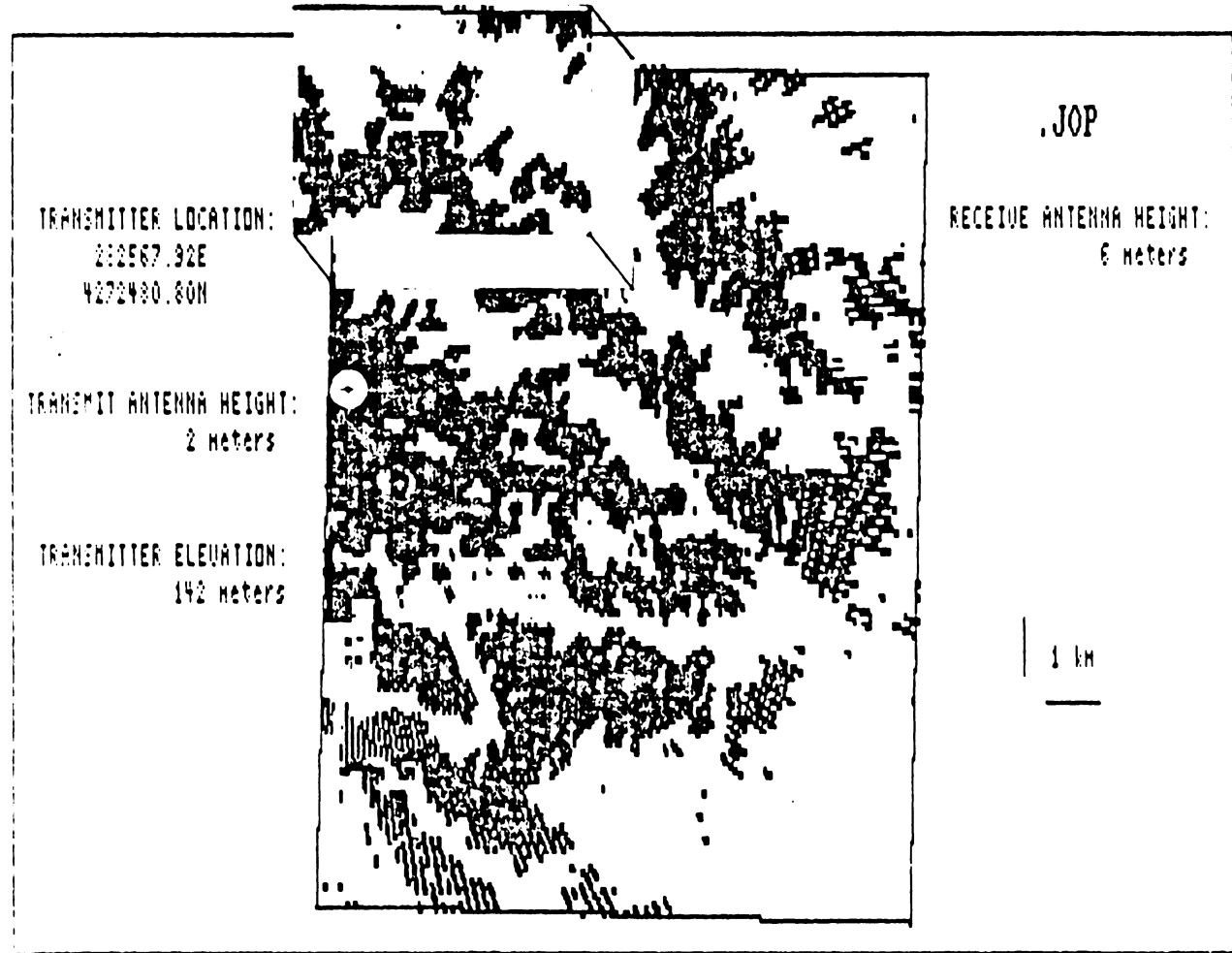


Figure 14. COVAREA.PAS output showing the selected ONE2ONE.PAS area.



Figure 15. ONE2ONE.PAS output (5417.8 m x 3822.8 m)

was measured via a Numonics Corporation digitizer. This instrument allowed the distances to be measured accurately and to be simultaneously entered into a microcomputer. Once the distance to each contour line was measured the corresponding elevation was entered to the computer as well. The result of all this is that a flat earth terrain profile for each radial was now stored on the computer. This is the same information that would be taken from the map and plotted by hand.

Instead of plotting by hand, a small program was written that would determine if each intersection with a contour line could or could not be seen by the transmitter. The algorithm used for this follows closely the algorithm described in Section 11.7. This is natural since this algorithm is an adaptation of hand plotting. The output of the program gave a yes or no decision about each intersection with a contour line. Yes, the point is able to receive the transmitter or no, the point is not in the area of coverage.

With this information in hand, a piece of cellophane was placed over the map. On the cellophane was an outline of the borders of the window. For each point on a radial found to be able to receive the transmitter, a blue dot was placed over that point on the cellophane. Rows of contiguous points were connected by a blue line. The assumption here is that if two points on adjacent contour lines are in the coverage area, so too is the line between these points. Similarly, red dots were placed over points not able to receive the transmitter. Rows of red dots were connected to form lines on a radial that cannot receive the transmitter. Between the blue lines and red lines on the cellophane appear gaps that represent unknown areas. It is known that transmission is possible on one side of the gap and that it is not on the other side. Somewhere between is where transmission fails. This is the best that can be resolved from the information available from a paper map. The result is an overlay that is an approximation to the ONE2ONE.PAS computer plot, only produced by hand from the paper map.

When the overlay is placed over the computer generated coverage calculation, a distinct correspondance can be readily observed. When the overlay is placed on the paper map a correlation to the terrain is also seen. Figure 16 is an attempt to show this given the limitations of a black and white printed page. This composite shows the paper map's contour lines with the "one to one" coverage area and the handmade coverage lines superimposed. Clearly there is good agreement between the two methods of coverage area calculation. Also, both indicate non-reception in the valleys.

Quantitatively comparing the automated and manual coverage areas is a difficult task. In an effort to be able to put a number on how well the two agree a second "one to one" plot of the same area was generated. This time the angular increment between radials was 3° instead of the $1/2^\circ$ that was used to form Figure 15. Recall that the handmade radials were (roughly) 3° apart. Hence, if both the manual and automated methods were to agree completely, the automated and manual outputs should look exactly alike. As can be seen in Figure 17, they do not.

Figure 17 shows the computer plot (the thick lines) and the "blue" handmade lines overlayed. They do not exactly agree but they are very similar. Their disagreement can be attributed to several factors. First, it should be made clear that the hand drawn radials are not precisely 3° apart. At the time the radials were drawn, the need to draw them accurately was not known. They were simply drawn something close to 3° apart. The computer, like the machine that it is, will construct 3° radials to the limit of its capability. The effect of all this can be seen as the radials progressively move out of phase with each other. Ultimately this means that the two methods are not attempting to calculate coverage over precisely the same terrain each time and therefore cannot be expected to arrive at the same conclusion.

Another factor for the increasing difference between the lines is the inability of the computer printer to consistently and accurately reproduce the output. Each printout is

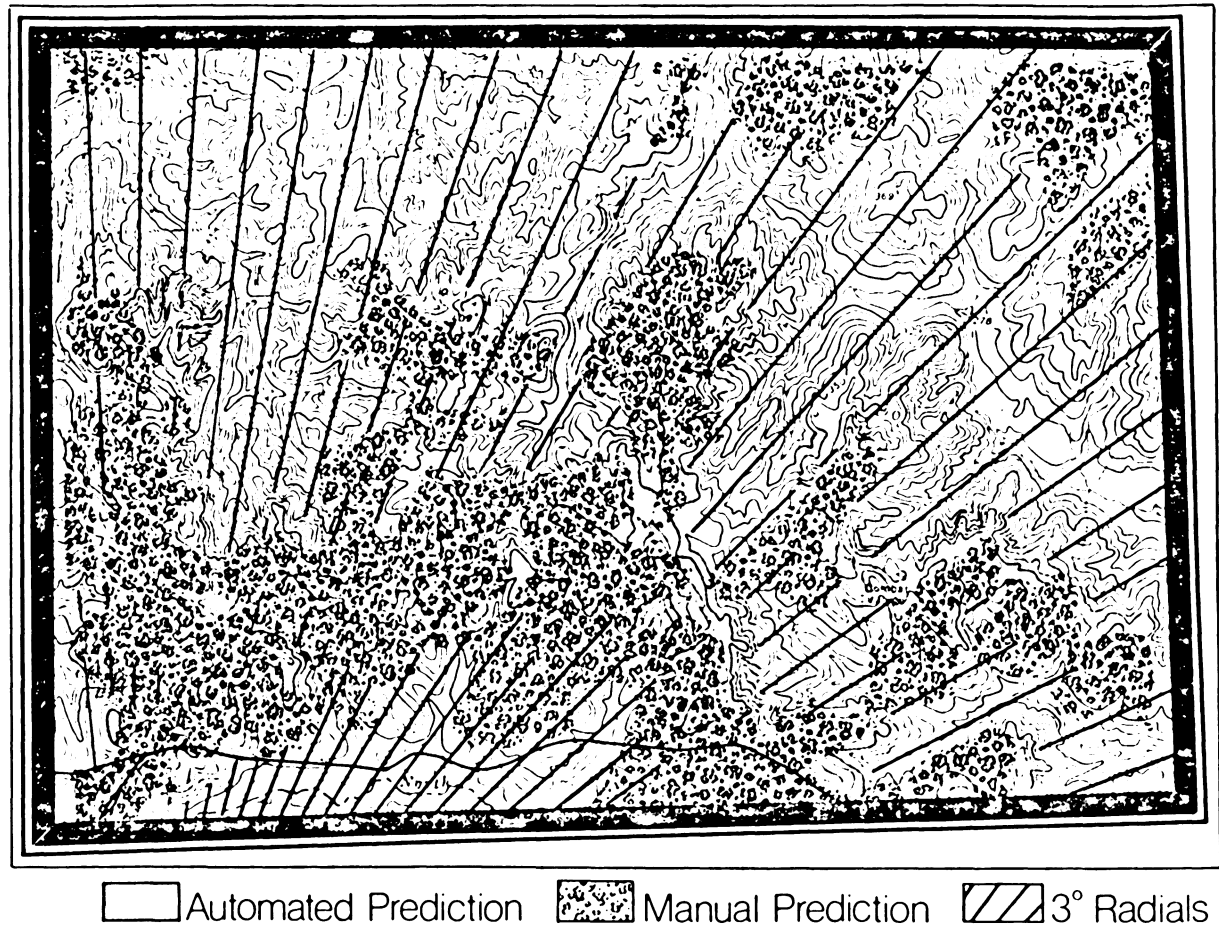


Figure 16. Comparison of automated and manual coverage area including the terrain contour lines.



 Automated Prediction  Manual Prediction

Figure 17. Automated and manual calculation with 3° radials.

slightly different and hence the scale cannot be made precisely to match that of the paper map. Similarly, a distortion arises from the photocopying that was performed in order to produce these documents. These are probably second order effects but should be kept in mind when comparing any of these plots. Obviously, the ability of the output device to produce consistent and accurate plots should be considered when selecting hardware for a working version of this type of program.

Aside from these problems, there is surely another reason attributing to the disagreement. Depending on the direction, the first contour lines crossed when moving away from the transmitter on the paper map were sometimes hundreds of meters out. This means that the paper map plot had little or no information about the terrain closer to the transmitter. If the computer did, then the "lever effect" again will cause error in this type of situation. The computer will have a more highly defined terrain profile close to the transmitter and will more likely find some blockage in this critical area than is possible with the paper map. This will more strongly effect areas far from the transmitter. The situation is modified when the step size is suddenly double to 60m or even 120m as coverage is determined at distances still farther from the transmitter.

One other cause of error would arise from any differences between the paper map and the digitized map. The paper map was last updated in 1967. The digital map is dated March, 1977. How current its information is compared to the paper map is not known. If the two maps do not represent the same information, agreement cannot be expected.

Even with all these things against them some agreement can be observed between the two. One way of comparing them might be to measure the lines of each plot. If you sum all the line lengths on one plot and compare this to the sum of the line lengths from the other plot, you can get a very rough idea about the comparative size between these two coverage "areas". When this was done it was found that the total length of the lines formed by the computer in Figure 17 was approximately 32 626m. The total line length

from the manual method was 31 214m. The difference between the two is 1412m. This is less than 5% of the total line length of either one. The two totals are on the same order of magnitude and would seem to indicate general agreement. Of course, this type of measurement might indicate something about the relative sizes but in no way indicates how well the areas intersect the same location. Visual judgement must suffice for that.

The fact that the computer area is larger is also significant. The computer inherently adds length to the line when it plots the block surrounding a point of coverage. The first and last points of each line segment will tend to produce a lengthening effect. There are approximately 70 separate lines that are measured to produce the sum for the computer generated plot. The 1412m difference would then correspond to about twenty additional meters per line. This number is low by any strict mathematical analysis of the geometry but, given the experimental error involved in resolving the ends of each line with the digitizer, it is at least within reasonable limits.

14. Conclusion

This report has shown that deriving coverage area from the USCGS digital data base is indeed feasible. In other words, the DEM's for the 1:24 000 scale topographic maps have sufficient resolution of the terrain elevation to allow accurate calculation of transmitter coverage area. In addition, the DEM's can be accessed automatically by a computer and used to calculate coverage area. In proving this, we have demonstrated one manner of making this automated calculation. This method is almost a direct extension of the method as would be performed by hand. Other possible methods and/or improvements have also been indicated. Finally, a comparison has been made between the coverage area as derived by hand and the coverage area as derived by the computer.

Further work in this automated process can address several problems as guided by customer need. First, the basic need of concatenating digital maps has been identified. This needs to be defined further by projecting future needs in relation to maximum distances of interest from the transmitter and postulating on the need to cross

UTM zone boundaries. Also related to maximum distances of interest is a study of the best method of determining step size. Doubling of the step size in order to limit the number of points in the digitized terrain profile is not considered the optimal approach. The requirements or projected requirements on the size of the area of interest, that is the limit to the distance from a proposed transmitter, will in some manner dictate the method utilized to determine the area of coverage.

Further development must keep in mind possible refinements to the propagation model used. First Fresnel zone clearance and knife edge refraction, for example, may be two possible refinements desired. Whatever the propagation model, its incorporation will have direct influence on the method of calculation used.

The method of output from any coverage area calculation program needs to be defined. The definition will depend on user application as well as available hardware. Some method of plotting map overlays has been discussed. This would require a plotter able to handle map size paper or acetate. Alternatively small plots of selected area's can be made. These could be concatenated if required. Whether to a CRT screen, as was done here, or directly to a hardcopy form; plotting would seem to be the most desirable form of conveying the information to the user.

As the next step in developing a "production" version of an automated coverage calculation program, we recommend adopting and refining the present system to the designated hardware environment; a Digital Equipment Corporation VAX. By present system we mean essentially one which uses only the present propagation model and is confined to single DEM's. The method of calculation will be defined after examination of the hardware capabilities and the ultimate sophistication desired by the user as discussed above.

Appendix I

This appendix contains the computer code for FRSTPASS.PAS. The operation of this program is discussed in Section 9 of this report.

```
PROGRAM FirstPass;
{
  This program will take the raw USGS tape data (Bigfile) off of
  <Indisk> and will separate the file into two binary files:
    LOGRECA.EXT
    PROFILE.EXT
  These files are written to <Outdisk>.
  A third may have to be added inorder to generate logical
  record C - the accuracy data file. It is not generated here.
}
LABEL  LBLA;
CONST
  Outdisk = 'C: ';
  Indisk  = 'C: ';
  LengthA = 144;
  maxnode = 463;
TYPE
  coord      = (E,N);
  point      = array [coord] of real;
  TypeB      = record
    sequence : integer;
    elnum    : integer;
    Gnot      : point;
    datum     : real;
    minmax    : array [1..2] of real;
    eldat     : array [0..maxnode] of integer;
  end;
```

```

VAR
    chr      : char;
    str1     : string[1];
    str4     : string[4];
    ext      : string[4];
    RECA     : string[lengthA];
    IntB     : Array [1..4] of Integer;
    RealB    : Array [1..5] of real;
    numB,IJunk,recount,elvcnt,chk,logrec  : Integer;
    Bigfile,FileA : Text;
    Count    : Byte;
    RJunk,Min,Max : Real;
    profile  : file of TypeB;
    RecB     : TypeB;

Procedure Crash (code : byte); { Subprogram to exit prematurely }
Begin
    Writeln('! your program has crashed at ',code,' !');
    repeat until keypressed;
    halt;
end;

PROCEDURE ReadRL(VAR RL : Real); { Subprogram to read in the D-format
LABEL  LBL1,LBL2;                      reals }
CONST
    Space = ' ';
VAR
    ST      : STRING[21];
    cnt,LOC : Integer;
    pst,num,CHK : Integer;
    ch      : char;
BEGIN
    st:='';
    loc:=0;
LBL1:  Read(Bigfile,ch);
    While (ch=space) and (loc=0) do read(Bigfile,ch);
    While (ch=space) and (loc=3) do Begin
        delete(st,4,18);
        Val(st,RL,chk);
        If RL=0.0 then Begin
            For cnt:=1 to 18 do Read(Bigfile,ch);
            goto LBL2;
        end
        Else Writeln('Error! space before E/D ',loc);
    end;
    loc:=loc+1;
    If ch='D' then ch:='E';
    If ch='E' then Begin
        Insert(ch,st,loc);
        cnt:=loc+1;
        For loc:=cnt to (cnt+2) do Begin
            Read(Bigfile,ch);
            Insert(ch,st,loc);
        end;
        If loc<21 then Begin
            cnt:=21-loc;
            loc:=loc+1;
            delete(st,loc,cnt);
        end;
        Val(st,RL,chk);
        While CHK<>0 do Begin
            Writeln('This string wont convrt to real: ',ST);
            Writeln('The error is in the ',CHK,' Char. Pos. ');
        End;
    end
end

```

```

        Else Begin
            Insert(ch,st,loc);
            goto LBL1;
        end;
LBL2 : End;

        { Start of MAIN PROGRAM }
BEGIN
    Write(' IS the proper data disk in drive ',outdisk,' ???');
    Readln;

    { Open input and output data files ... }
    Write('Enter the name of the input data file extension: ');
    readln(ext); { This sets the output file extension also }
    assign(Bigfile,concat(Indisk,'DATA',EXT));
    reset(Bigfile);
    Assign(FileA,concat(Outdisk,'LOGRECA',Ext));
    Rewrite(FileA);
    { The following code reads and creates LOGRECA.EXT from
      Logical record Type A }
    read(Bigfile,RecA);
    Writeln(RecA);
    Writeln(FileA,RecA);
    { }
    Read(Bigfile,Ijunk);
    Writeln('The DEM level code is ',Ijunk);
    Writeln(fileA,Ijunk);
    { }
    Read(Bigfile,Ijunk);
    Writeln('The elevation pattern code = ',Ijunk);
    Writeln(FileA,Ijunk);
    { }
    Read (Bigfile,Ijunk);
    Writeln('The grnd Ref. code is: ',Ijunk);
    Writeln(FileA,Ijunk);
    { }
    Read(Bigfile,Ijunk);
    Writeln('Zone code = ',Ijunk);
    Writeln(FileA,Ijunk);
    { }
    Count:=0;
    Repeat
        Count:=Count+1;
        ReadRL(Rjunk);
        If Rjunk<>0.0 then crash(15);
        Writeln(Count,Rjunk);
    UNTIL Count=15 ;
    { }
    Read(Bigfile,Ijunk);
    Writeln('Code for units of grnd measure: ',Ijunk);
    Writeln(FileA,Ijunk);
    { }
    Read(Bigfile,Ijunk);
    Writeln('Code for units of elevation measure: ',Ijunk);
    Writeln(FileA,Ijunk);
    { }
    Read(Bigfile,Ijunk);
    Writeln('The number of sides = ',Ijunk);
    Writeln(FileA,Ijunk);
    { }
    For Count:=0 to 7 do Begin
        ReadRL(Rjunk);
        Writeln(Count,Rjunk);
        Writeln(FileA,Rjunk);
    End;

```

```

{ }
  ReadRL(Min);
  ReadRL(Max);
  Writeln(FileA,min);
  Writeln(FileA,max);
  Writeln('The minimum elevation is: ',Min,' and the Maximum: ',Max);
{ }
  ReadRL(Rjunk);
  Writeln('Angle from primary axis: ',Rjunk,' in Radians');
  Writeln(FileA,Rjunk);
{ }
  Read (Bigfile,chr);
  While (chr=' ') do read(Bigfile,chr);
  insert(chr,str1,1);
  Val(str1,Ijunk,chek);
  Writeln('Accuracy code: ',Ijunk);
  Writeln(FileA,Ijunk);
{ }
  For Count:=1 to 3 do Begin
    ReadRL(Rjunk);
    Writeln('This is 30,30,1: ',Rjunk);
    Writeln(FileA,Rjunk);
  End;
{ }
  Read(Bigfile,Ijunk);
  Writeln('If this Integer = ',Ijunk,' =one');
{ }
  Read(Bigfile,NumB);
  Writeln('...Then this: ',NumB,' = the number of B type records')
  Writeln(FileA,NumB);
  Close(FileA);      { Finished LOGRECA.ext }

{ The following code reads and creates PROFILE.EXT file. }
assign(profile,concat(outdisk,'PROFILE',ext));
rewrite(profile);
Recount:=0; { = record count }

For LogRec:=1 to numb do begin;      { Loop for creating each record
Writeln('Starting record ',recount);{ of PROFILE.ext

For count:=1 to 4 do Begin { Read in integer data in profile header
  Read(Bigfile,IntB[count]);
  Writeln(count,' ',IntB[count]);
end;

For count:=1 to 5 do Begin { Read in real data in profile header }
  ReadRL(realB[count]);
  Writeln(count,' ',RealB[count]);
end;

{ check the data ... }
  If IntB[2]<>(recount+1) then crash(1) ;
  If RealB[3]<>0.0 then crash(2) ;

With RecB do Begin      { Assign values to the record }
  sequence:=IntB[2];
  elnum    :=IntB[3];
  Gnot[E]  :=RealB[1];
  Gnot[N]  :=RealB[2];
  datum    :=RealB[3];
  minmax[1] :=RealB[4];
  minmax[2] :=RealB[5];
end;

```

```

elvcnt:=0;
Repeat      { Loop to read in elevation data for the record }
  read(Bigfile,Ijunk);
  If Ijunk>RealB[5] then Begin
    Writeln('Elvcnt = ',elvcnt);
    crash(3);    { Test each datum < maximum }
  end;
  If Ijunk<RealB[4] then Begin
    Writeln('Elvcnt = ',elvcnt);
    crash(16);   { Test each datum > minimum }
  end;
  RecB.eldat[elvcnt]:=Ijunk;
  elvcnt:=elvcnt+1;
Until elvcnt=(IntB[3]) ;

Seek(profile,recount); { write record to outdisk }
Write(profile,recb);
Recount:=recount+1;    { go do next record }
END;

Close(profile);
Close(bigfile);
END.  { finished }

```

Appendix II

This appendix contains the computer code for TERPGEN.PAS as is discussed in the body of the report. The flow diagram for this program is shown in Figure 5.

```
PROGRAM TERRainProfileGENerator; { FINAL VERSION }
{
  This program will determine the terrain profile between any two
  given points in the selected DEM. The profile data is output to
  a specified output file. The program also corrects for atmospheric
  refraction and checks for terrain blockage of the radio frequency
  ray. The point of blockage, if any, is indicated in the output.
}
LABEL Tale, Next, retry;
Const
  Indisk   = 'C:' ;
  Outdisk  = 'A:' ;
  maxnode  = 463  ;
  slicemax = 64   ;
  maxelev  = 199  ;
  mintail  = 5    ;
  Eradius  = 6370000.0;
Type
  coord = (E,N);
  point = array [coord] of real;
  TypeB = record
    sequence : integer;
    elnum    : integer;
    Gnot     : point;
    GmaxN    : real;
    eldat    : array [0..maxnode] of integer;
  end;
```

```

Var
  DataBank : file of TypeB;
  Profile : TypeB;
  fileA, output : Text;
  {}
  EXT : string[4];
  Header : string[144];
  chr : char;
  outstr : string[14];
  {}
  Pt, start, TLOC, RLOC : Point;
  slice : array [0..slicemax,0..maxnode] of integer;
  bounds : array [0..slicemax,0..2] of Real;
  elev : array [0..maxelev] of integer;
  corners: array [0..3,E..N] of Real;
  {}
  Rayht,mray,screlv,min,max, Tantht, RantHt, TRDist, Dist, Delx,Dely,
  Telv,Relv,K, tail,theta, m, Rjunk, L1, L2, DelH : Real;
  {}
  numB, Ijunk, numpts, node, cnt, sample, Alpha, omega,
  step, endslice, slicnt, maptr, zone, blockpt : Integer;
  {}
  knt : coord;
  calcTelv, calcRelv,RwestT, hidden : Boolean;

Procedure ChkLOC ( var LOC : point ); { Subprogram for checking
Label again ; locations }
Var
  off : boolean;
  Bot,Top : Real;
Begin
again: clrscr;
  off:=false;
  maptr:=trunc((LOC[E]-Start[E])/30);
  If maptr>(numB-1) then off:=true;
  If maptr<0 then off:=true;
  If not off then Begin
    seek(databank,maptr);
    read(databank,profile);
    With profile do Begin
      Top:=GmaxN ;
      Bot:=Gnot[N];
    end;
    If LOC[N]>Top then off:=true;
    If LOC[N]<Bot then off:=true;
  end;
  If off then Begin
    Textbackground(Red);
    writeln('location: ',LOC[E],'E ',LOC[N],'N is not on the map!');
    write(' .....enter new Easting: ');readln(LOC[E]);
    write(' .....enter new Northing: ');readln(LOC[N]);
    goto again;
  end;
  Textbackground(Blue);
  writeln('OK ...');
end;

Procedure Crash (code : byte); { Subprogram for exiting prematurely }
Begin
  Writeln('! your program has crashed at ',code,' !');
  halt;
end;

```

```

Procedure LDMAP; { Subprogram for loading the map slice }
Begin
  maptr:=trunc((Pt[E]-Start[E])/30);
  If not RwestT then Begin
    Alpha:=maptr;
  If (alpha+slicemax)>numB then omega:=numb-1 else omega:=(alpha+slicemax)
  end;
  If RwestT then Begin
    omega:=maptr+1;
  If (omega-slicemax)<0 then alpha:=0 else alpha:=(omega-slicemax)
  end;
  writeln('loading map slice ...starting at profile:',(alpha+1));
  slicnt:=0 ;
  maptr:=alpha;
  seek(databank,alpha);
  Repeat
    Read(databank,profile);
    If profile.sequence <> (maptr+1) then crash(1);
  with Profile do Begin
    bounds[slicnt,0] := Gnot[E] ;
    bounds[slicnt,1] := Gnot[N] ;
    Bounds[slicnt,2] := GmaxN ;
    For node := 0 to (elnum-1) do slice[slicnt,node]:=eldata[node];
  end; {of with statement}
  slicnt:=slicnt+1;
  maptr:=maptr+1;
  Until maptr=omega+1;
  maptr:=alpha; {map ptr points to beginning of slice}
  endslice:= slicnt-1; {reset end of slice}
  { set slicnt to profile just west of Pt. }
  If RwestT then slicnt:=endslice-1 else slicnt:=0;
end;

```

```

{ Subprogram for finding elevations }
Procedure FINDELV ; {slicnt must be preset to profile west of pt }
{ using method II }

```

```

Var
  elquad : array [0..3] of integer;
  Xup,elA,elB : Real;
Begin
  Rjunk:=(pt[N]-Bounds[slicnt,1])/30;
  node:=trunc(Rjunk);
  Xup:=Frac(Rjunk);
  elquad[0]:=slice[slicnt,node] ;
  elquad[1]:=slice[slicnt,node+1] ;
  elquad[2]:=slice[slicnt+1,node] ;
  elquad[3]:=slice[slicnt+1,node+1];
  elA:=(((elquad[1]-elquad[0])/30)*xup)+elquad[0] ;
  elB:=(((elquad[3]-elquad[2])/30)*xup)+elquad[2] ;
  Rjunk:=(((elB-elA)/30)*(pt[E]-bounds[slicnt,0])) ;
  Elev[sample]:=round(Rjunk + elA);
  writeln('smpl: ',sample,' ',elev[sample]);
end;

```

```

BEGIN { MAIN PROGRAM BEGINS HERE }

```

```

{ Initialize some stuff ... }
  textmode(3);
  textbackground(Blue);
  textcolor(Lightgreen);
  clrscr;
  step:=30;
  K:=4/3;

```

```

{ Begin input of user info ... }
GotoXY(5,12);
write('Input area identifying extension, e.g. ".JOP" or ".QNT" => ');
readln(EXT);

Assign(FileA,concat(Indisk,'LOGRECA',Ext)); { This section reads the }
reset(fileA);                               { type A record and checks }
writeln(' Reading LOGRECA',ext,' ...');     { the codes }
Readln(fileA,header);
readln(fileA,Ijunk); { DEM level code }
If Ijunk<>1 then crash(4);
readln(fileA,IJUnk); If Ijunk<>1 then crash(5); { elev. patrn code }
readln(fileA,Ijunk); If Ijunk<>1 then crash(6); { grnd. ref. code }
readln(FileA,zone); { UTM co-ordinate zone }
readln(FileA,Ijunk); If Ijunk<>2 then crash(7); { grnd. units code }
readln(FileA,Ijunk); If Ijunk<>2 then crash(8); { elev. units code }
readln(FileA,Ijunk); If Ijunk<>4 then crash(9); { # sides map area }
For cnt:=0 to 3 do
  For knt:=E to N do readln(FileA,corners[cnt,knt]);
readln(FileA,min); { minimum and maximum elevations in the }
readln(fileA,max); { entire map area }
readln(FileA,Rjunk); If Rjunk<>0.0 then crash(10); { ang. primary axis }
readln(FileA,Ijunk); { accuacy code <- is there a logical rec. C ? }
readln(FileA,rjunk); If Rjunk<>30.0 then crash(11); { res. code: N }
readln(FileA,Rjunk); If Rjunk<>30.0 then crash(12); { res. code: E }
readln(FileA,Rjunk); If Rjunk<>1.0 then crash(13); { res. code: Up }
readln(fileA,NumB); { # of type B records = # of map profiles }
Close(fileA);

clrscr; { Aquire transmitter information }
gotoXY(5,10);
chr:=' ';
writeln(' TRANSMITTER:');
calcTelv:=true;
writeln('Input transmitter location in UTM co-ordinates (meters) ...');
write(' Easting: '); Readln(TLOC[E]);
write(' Northing: '); Readln(TLOC[N]);
write(' Input Transmit Antenna height in meters: ');
Readln(TantHt);
write(' Input Elevation of transmitter location? (Y/N): ');
readln(chr);
If (chr='Y')or(chr='y') then begin
  calcTelv:=false;
  write('Input transmitter elevation in meters: ');
  Readln(Telv);
end;

```

```

    clrscr;      { Input receiver information }
    gotoXY(5,10);
    chr:=' ';
Writeln(' RECEIVER:');
    calcRelv:=true;
    Writeln('Input receiver loacation in UTM co-ordinates (meters) ...');
    write('      Easting: '); Readln(RLOC[E]);
    write('      Northing: '); Readln(RLOC[N]);
    write('      Input receive antenna height in meters: ');
    Readln(RantHt);
    Write('Input elevation of Receiver location? (Y/N): ');
    readln(chr);
    If (chr='Y')or(chr='y') then begin
        calcRelv:=false;
        write('Input Receiver elevation in meters: ');
        readln(Relv);
    end;

    clrscr;      { New K ? }
    gotoxy(5,10);
    write('K=',k,'      Input new K ? (Y/N): '); readln(chr);
    If (chr='Y')or(chr='y') then begin
        write('Input new K: ');
        readln(K);
    end;

    clrscr;      { Aquire outfile name }
    gotoxy(5,10);
    write('Input output <disk: + > filename + .ext: '); readln(outstr);

{ If TLOC/RLOC are in Lat,Long they will be converted here }

writeln('Loading PROFILE',EXT,' ...');
    Assign(databank,concat(indisk,'profile',ext));
    reset(databank);
    seek(databank,0);
    read(databank,profile);
    If profile.sequence<>1 then crash(14);
    With Profile do begin
        Start[E]:=Gnot[E]; { Set Start to most SW node }
        start[N]:=Gnot[N];
    end;

{ check if TLOC & RLOC are in the map area }
    writeln('Checking transmitter location ... ');
    ChkLOC(TLOC);
    writeln('Checking receiver location ... ');
    ChkLOC(RLOC);

{ All set now to begin calculations ... }
    Writeln('      Working ...');
    DelX:=RLOC[E] - TLOC[E];
    DelY:=RLOC[N] - TLOC[N];
    TRDist:=sqrt(sqr(DelX)+sqr(DelY));
    m:=DelY/DelX;
    Theta:=arctan(m);

retry: numpts:=trunc(TRDist/step)+1;      { Set step size }
    If Numpts>maxelev then Begin
        step:=2*step;
        writeln('Note: Step size reset to: ',step);
        goto retry;
    end;

```

```

{ check whether RLOC is E or W of TLOC }
  IF Delx<0 then Begin
    RwestT:=True;
    Theta:=Theta+Pi;
  end
  Else RwestT:=False;

{ Beginning to find elevations }
  SAMPLE:=0;
  Dist:=0;
  Pt[E]:=TLOC[E];
  Pt[N]:=TLOC[N];
  LDMAP;

NEXT:FindElv;    { start working on next sample }
  sample:=sample+1;
  Dist:=Dist+step;
  If Dist>TRDist then goto Tale;

{ Calculate the next pt }
  delx:=Dist*cos(theta);
  dely:=m*DelX;
  Pt[E]:=Delx + TLOC[E];
  Pt[N]:=Dely + TLOC[N];

{ Is new pt in the current map slice? }
  If Pt[E]>bounds[endslice,0] then LDMAP; { load new map slice }
  If Pt[E]<bounds[0,0] then LDMAP
    Else slicnt:=trunc((Pt[E]-bounds[0,0])/30);
  Goto NEXT;

TALE: Dist:=Dist-step;
  if sample <> numpts then crash(0);
  tail:=TRDist-Dist;

  If tail<mintail then begin
    sample:=sample-1;
    numpts:=sample;
    tail:=tail+step;
  end;

  If calcRelv then Begin
    Pt[E]:=RLOC[E];
    Pt[N]:=RLOC[N];

    { Is the Pt in the current map slice? }
    If Pt[E]>bounds[endslice,0] then LDMAP; { load new
      If Pt[E]<bounds[0,0] then LDMAP          slice }
      Else slicnt:=trunc((Pt[E]-bounds[0,0])/30);

    FINDELV;
  end
  Else elev[sample]:=round(Relv);
Close(databank);

  If not calcTelv then elev[0]:=round(Telv);
  screlv:=TantHt+elev[0];

{ print out to file: ELEV[] Array + tail/calculate curvature correction
  assign(output,outstr);
  rewrite(output);
  writeln(output,TLOC[E],TLOC[N]);
  writeln(output,TantHt);
  writeln(output,RLOC[E],RLOC[N]);
  writeln(output,RantHt);

```

```

writeln(output,numpts);
Dely:=RantHt+elev[numpts]-screlv;
mray:=dely/TRDist;
Hidden:=False;
L1:=0.0 ;

For sample:=0 to (numpts-1) do begin
    L2:=TRDist-L1;
    DelH:=(L1*L2)/(2*Eradius*K);
    Delx:=step*sample;
    RayHt:=(mray*Delx)+screlv;
    If ((elev[sample]+DelH)>=rayHt)and(not hidden) then Begin
        Hidden:=true;
        Blockpt:=sample;
    end;

    writeln(output,sample,' ',step,' ',elev[sample],' ',DelH);
    L1:=L1+step;
end;

If not hidden then Blockpt:=numpts;
writeln(output,numpts,' ',tail,' ',elev[numpts],' ',0.0);
writeln(output,blockpt);

close(output);
textmode;

```

Appendix III

This appendix contains the computer code and flow diagram for the coverage area calculation program COVAREA.PAS. What follows is a detailed description of this code based on the flow diagram shown in Figure A3.1. The discussion assumes that the reader is familiar with the basic philosophy as given in Section 12 and also with the terminology and subroutines of the terrain profile calculating program TERPGEN.PAS as described in Section 11.

The first five blocks illustrate the operator input and program initialization. This is very similar to the input and initialization phase of TERPGEN.PAS as described in Section 11.2 of this report. The first block indicates the operator input of the three letter extension which identifies the DEM information required. At this point LOGRECA.ext is read. From this file, the corners of the map are set in memory and the number of type B records is obtained. Again, just as with TERPGEN.PAS, the other information is checked or unused at this time.

Next, the operator supplies the the information about the transmitter site; i.e. the location, the antenna height and, optionally, the elevation. The next box indicates the specification of the required receiver information: the antenna height. That is the only information that is required of the receiver when making coverage area calculations. Following this, the program requests the angle from due east to start the first radial. This angle is called *theta*. It also requests the arc that will be covered by the radials. This arc is an angle that can be up to 360°. This latter value would cause the program to span completely around the transmitter for a complete area of coverage. By selecting the proper starting angle and the appropriate arc, the coverage in selected directions can be calculated. From this information, *theta* and the arc, the angle at which to stop is calculated. The last of the operator input is the option of designating K to be other than 4/3.

Following this, PROFILE.ext is found and opened. That is to say that the databank of elevation profiles is made ready for reading. Also, the south-westernmost point is set in memory as the reference point for databank searches. As a precaution, the transmitter location is checked via a call to the subroutine ChkTLOC. From this point the program begins to deviate from the flow of TERPGEN.PAS.

At this time the graphics routines are initialized. The CRT screen is cleared and an outline of the DEM is drawn to the screen. This is done using the coordinates of the corners of the map that were obtained from LOGRECA.ext. The graphics routines allow the screen to be accessed via a coordinate system that corresponds directly to meters. There is an offset built in so that the range of possible values will cause the map outline to appear in the middle of the screen. Hence points can be plotted to the screen directly from their UTM coordinates. Using built in commands, straight lines are drawn between these corner points. In reality these are curves that follow the meridians and latitudes of the earth but at this scale (and at map scale) they are indistinguishable from straight

lines. The resolution of the screen prevents precise straight lines from appearing in any case.

Also at this time, identifying information is written on the screen in chosen locations. Such things as the transmitter location, both transmit and receive antenna heights, and the elevation of the transmitter site are recorded to the screen in order to later identify the displayed information.

The next box is the actual beginning to the process of calculating coverage area now that everything is initialized and the screen is ready for plotting. As with TERPGEN.PAS, the elevations for each ray are stored in an array; again, called *Elev[sample]*. *Elev[0]* holds the elevation of the transmitter location. *Sample* is therefore initially set to zero and the current location variable, PT, is made equal to the transmitter location. The next flow box indicates the first call to the subroutine LDMAP which loads memory with a slice of the map. If the transmitter elevation was not given, then we are now ready to calculate it by calling the subroutine FINDELV. If it was given, then *Elev[0]* is simply set equal to the given elevation. The decision box following the LDMAP call indicates this.

Next, the source elevation is calculated. You may recall, that this is the sum of the transmitter elevation and the transmit antenna height. It is used as part of the equation for the line defining the transmitted ray. The source elevation is constant throughout the program.

We now enter the loop whereby the coverage area is calculated. The loop is completed when the angle of the current radial exceeds the stop angle as calculated above. There are actually two intertwined loops here. One that calculates and proceeds out the current radial and the other that begins with the start of each new radial around the transmitter. The latter is entered and begins at the point on the flow diagram labeled "Next Ray". The other begins and is reentered at the point labeled "Next Point".

Upon entering the loop at "Next Ray" certain counters and variables need to be set at the beginning of each new radial. The first is the current distance traveled out the radial from transmitter to receiver, which is called TRDIST. This, of course, is set initially to zero. The next is a variable that holds the number of points from the transmitter to the receiver. This is called NUMPT and it is initially set to two. This is because the first potential receiver location that needs to be checked is two away from the transmitter. The transmitter is at zero, the first point has nothing to block it, and so the first potential receiver is at two. The step size is initially set to 30 meters at the beginning of each radial. As the radial becomes longer the step size is doubled in the same manner described in Section 11.4. Finally, a variable called *hidden* is set to FALSE at the outset of a new radial. The meaning and reason for this variable will be made clear later.

For this new radial, the flow continues on to calculate the first point of the radial. This is the part of the flow diagram labeled "Next Point". Here, the index to the array of elevations, *sample*, is incremented by one. When each new radial is started, *sample* has been reset to zero so that at this point it will indicate the first point out from the transmitter. The TRDIST is incremented by the step size to coincide with this. Now with the index to the array of elevations set to the next point as well as the the corresponding distance, the coordinates for the location of this point can be calculated and becomes the current PT. As before in TERPGEN.PAS, this new point along the radial must be tested to see if it is in the current map slice. If it is, then the "slice pointer" is set and the elevation of this point is found by calling FINDELV.

If it is not in the current map slice several possibilities exist and must be tested. The first is to check if the new point is on the map altogether. If it is, then it is time to load the next map slice and continue on by finding its elevation and then carrying on with the coverage calculation. If the new PT is not in the DEM then it is time to begin a new radial. The angle for the new radial is found by incrementing *theta* by a preset step

angle, *delta theta*. (*Delta theta* set to 1° provides a suitable coverage area plot.) The new *theta* must be tested to see if it is greater than the stop angle as calculated above. If it is greater, then the program is complete once a white point is plotted at the transmitter location in order to identify it. If the new *theta* is not greater than the stop angle, then *sample* is set to zero and flow returns to the "Next Ray" point in order to start a new radial.

Let us assume that this is our first radial and that the PT calculated above is the first point past the transmitter. As should be the case, we'll assume that this point is in the current map slice. After the elevation of the point is found via FINDELV, a test is made to see if *sample* is less than NUMPT. Recall that *sample* equals one since this is the first point out from the transmitter and that the subroutine FINDELV returns a value to the array Elev[*sample*]. Recall also that NUMPT is equal to two because that is the number of points to the proposed receiver location. Therefore the above test will prove true and flow will return to the "Next Point" location of the flow diagram. This, in order to calculate the coordinates of and find the elevation of the next point. This is the only time that this path back to "Next Point" is taken: at the beginning of a radial in order to fill the array of elevations with two points beyond the transmitter.

Upon returning to "Next Point", *sample* is incremented and hence made equal to two. The TRDIST is again incremented by the step size. The second point is calculated and is tested to see if it is in the current map slice. Assuming that it is, the "slice pointer" is set and the subroutine FINDELV is called. It returns with the variable Elev[*sample*] containing the elevation of the second point. Now the value of *sample* and NUMPT are compared again. Since *sample* is equal to NUMPT, flow continues on around the main loop. At this point in our example, Elev[0], Elev[1] and Elev[2] are equal to the transmitter site elevation and the elevation of the first two points out from the transmitter respectively.

Now the testing of the proposed receiver site begins. An additional test has been added to the one described in Section 11.7 which was used with the TERPGEN.PAS program. This additional test cut the run time for this program by a factor of two. The principle behind the test is that if the previous endpoint was found not to be in the coverage area and the current point under consideration has an elevation lower than the previous endpoint, then the current point cannot receive the transmitter either. This fact eliminates the need to test proposed receiver locations when they are at a lower elevation than their prior proposed receiver locations that were not able to see the transmitter. When the need to step through each point of the terrain profile, calculating its earth curvature correction and testing each one to be above or below the ray at that point is eliminated for any proportion of the points along the radial then a good deal of time has been saved.

In order to do this, the program needs to “remember” whether the last receiver point was able to receive the transmitter. This is done by resetting a Boolean type variable called *hidden* at the end of each receiver decision. Recall that *hidden* was set equal to FALSE at the start of each new radial. At the start of a radial, there was no previous receiver so it was *not hidden*. When a receiver is found that could not hear the transmitter then *hidden* is set to TRUE, i.e. the last receiver point was *hidden*.

Returning to the flow diagram we’ve come to the first test utilizing the variable *hidden*. If the elevation of the present proposed receiver location, *Elev[sample]*, is greater than or equal to the elevation of the last receiver location, *Elev[sample-1]*, then *hidden* should be set equal to FALSE. Otherwise it is left alone. This is comparison #1. Flow continues on to the next test: if *Elev[sample]* is less than *Elev[sample-1]* AND *hidden* is TRUE, then there is no need to test each point of the profile. This is comparison #2 on the flow diagram. There is no need to test each point because the last endpoint was unable to receive the transmitter and the elevation of the present endpoint is lower. If

this is the case then flow control jumps to a point in the program flow labeled "No Plot". This is the same location that is jumped to if an end point is found to be blocked during the outright testing of the point for blockage.

Let us return to our working example, where *sample*=2 and this is the first check for reception on the first radial. There was no previous receiver location, hence *hidden*=FALSE, and the flow proceeds to check for blockage. Returning to the flow diagram, observe that the parameters for the line defining the ray are calculated as in TERPGEN.PAS. The counter *sample* is reset to the first point in the terrain profile: one. We now enter a loop equivalent to the one described previously for TERPGEN.PAS. During each pass around the loop, the earth curvature correction is calculated for *Elev[sample]*. The sum of this correction and the elevation are then compared to the ray height at this point. As before, if this sum is greater than or equal to the height of the ray then the transmitter is blocked. When a point is found that will block transmission then flow immediately leaves this loop. If this is the case, the variable *hidden* is set equal to TRUE and the flow jumps to the point called "No Plot".

In our example with a terrain profile of only three points, this first receiver point is only 60 meters from the transmitter and almost always tests able to receive the transmission. Therefore we assume the above test would be false and the loop continues. *Sample* is incremented by one. We then check to see if we have tested all the points along the terrain profile. This is done by testing to see if *sample* is equal to NUMPT. If it is, and it will be in our example, we have tested all the points along the terrain profile between the transmitter and the endpoint (the proposed receiver location) and did not find that any of the intermediate points block reception. Therefore this endpoint is in the coverage area and a white block will be plotted to the screen at this location. Also, *hidden* is set equal to FALSE. Flow then proceeds on into the point in the flow diagram labeled "No Plot".

The first thing to happen here is that NUMPT is incremented by one. This sets the next proposed receiver location to be one point farther out the radial than the last. Next, NUMPT is tested to see if it is equal to 200. If so, then there will be 200 points in the terrain profile and this is the limit. When this happens then the step size is doubled and every other point in the current array of elevations is discarded in order to correspond to the new step size. In either case, flow continues by setting *sample* equal to NUMPT-1. The program now returns to the top of the main loop to the location labeled "Next Point".

To summarize our example; NUMPT is now equal to three, *sample* is equal to two and we have the elevation of the first three points along the radial: the transmitter and two points beyond. We have tested the (trivial) third point for reception and found that the single point between it and the transmitter did not block reception. We have now jumped to the top of the main loop, to "Next Point", and we begin by incrementing *sample* by one. It now equals NUMPT which equals three. Following the flow diagram, the distance in meters from the transmitter to receiver is incremented by the step size. With this, the location of the the next point along the radial, the new proposed receiver location, is calculated. This point is then checked to see if it is in the current map slice. Since it is only the fourth point out the radial, it should be in the current map slice and flow will continue by resetting the "slice pointer" and calling the subroutine FINDELV. FINDELV returns with *Elev[sample]* containing the elevation of the new point. We now have a grand total of four points in our terrain profile, the first being the transmitter location and the last being the new proposed receiver location.

Recall that NUMPT is equal to three and *sample* is equal to three, so flow will continue on around the main loop. *Hidden* is already equal to false so regardless of whether *Elev[sample]* is greater than *Elev[sample-1]*, *hidden* will remain false. Since *hidden* is not TRUE, the next test in the flow (comparison #2) will prove false and flow will procede in the direction that will test the terrain profile for blockage.

Flow enters the loop to check the terrain profile for blockage by first setting the counter, *sample*, equal to one. Each pass around the loop will increment *sample* by one. If each point is checked until *sample* becomes equal to NUMPT, then no blockage has been found and a white block is plotted on the screen. If a blockage is found then the loop is exited immediately and no point is plotted to the screen.

When this has happened the counter for the number of points to the current proposed receiver location, NUMPT, is again incremented by one. Again, if it becomes equal to 200 then the step size is adjusted and the terrain profile array, *Elev[sample]*, is made to coincide with this change. Regardless, *sample* is set equal to one less than NUMPT in preparation for adding a new point to the end and repeating the process again. As stated above, the process continues until the next point along the present radial is off the map. Then the next radial is started. Then the next and the next and so on until the specified arc or 360° around the transmitter has been traversed.

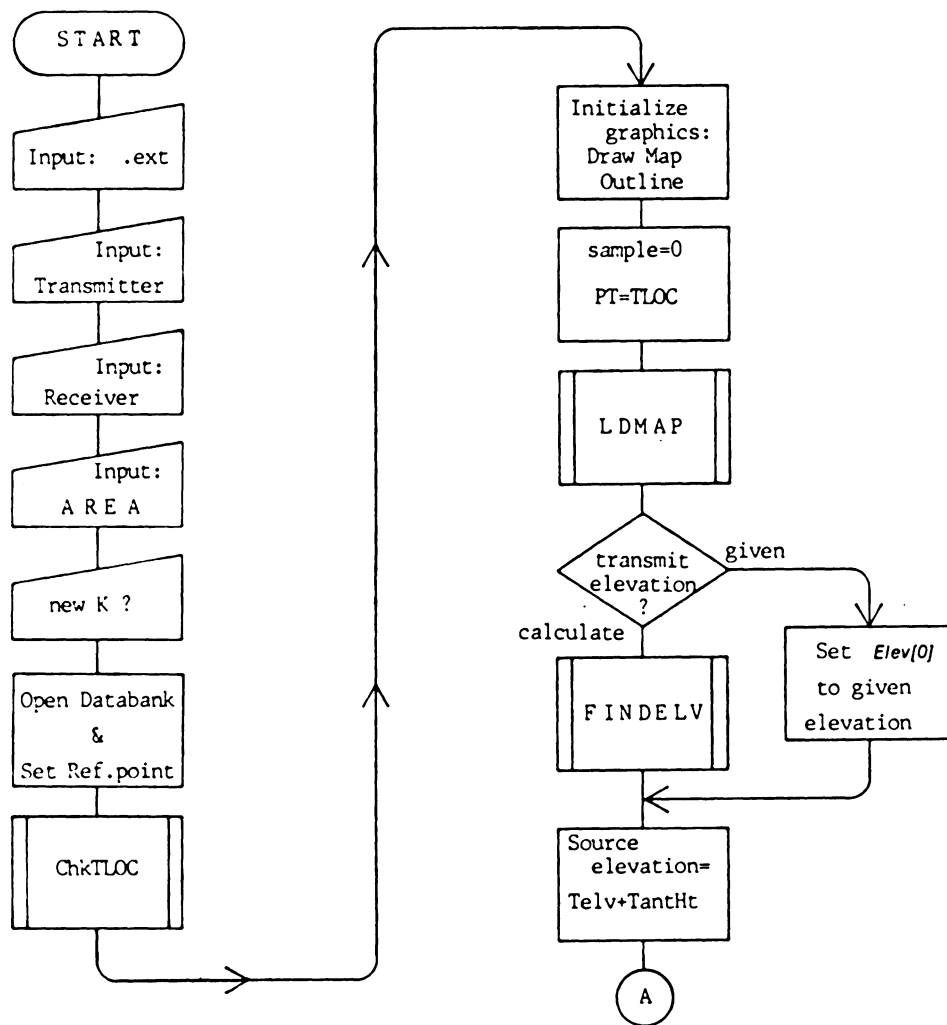


Figure A3.1 COVAREA.PAS flow chart.

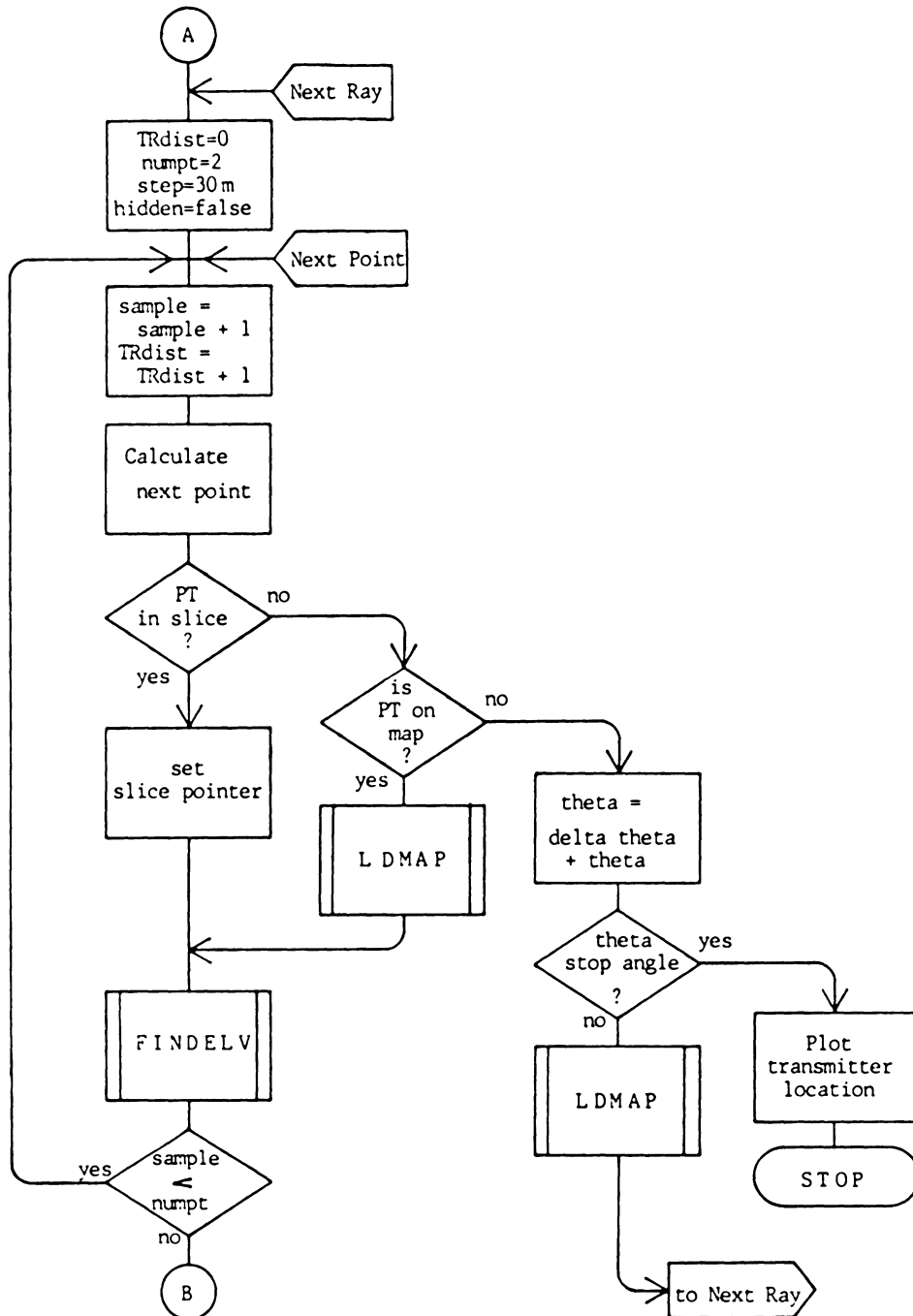


Figure A3.1 COVEREA.PAS flow chart continued.

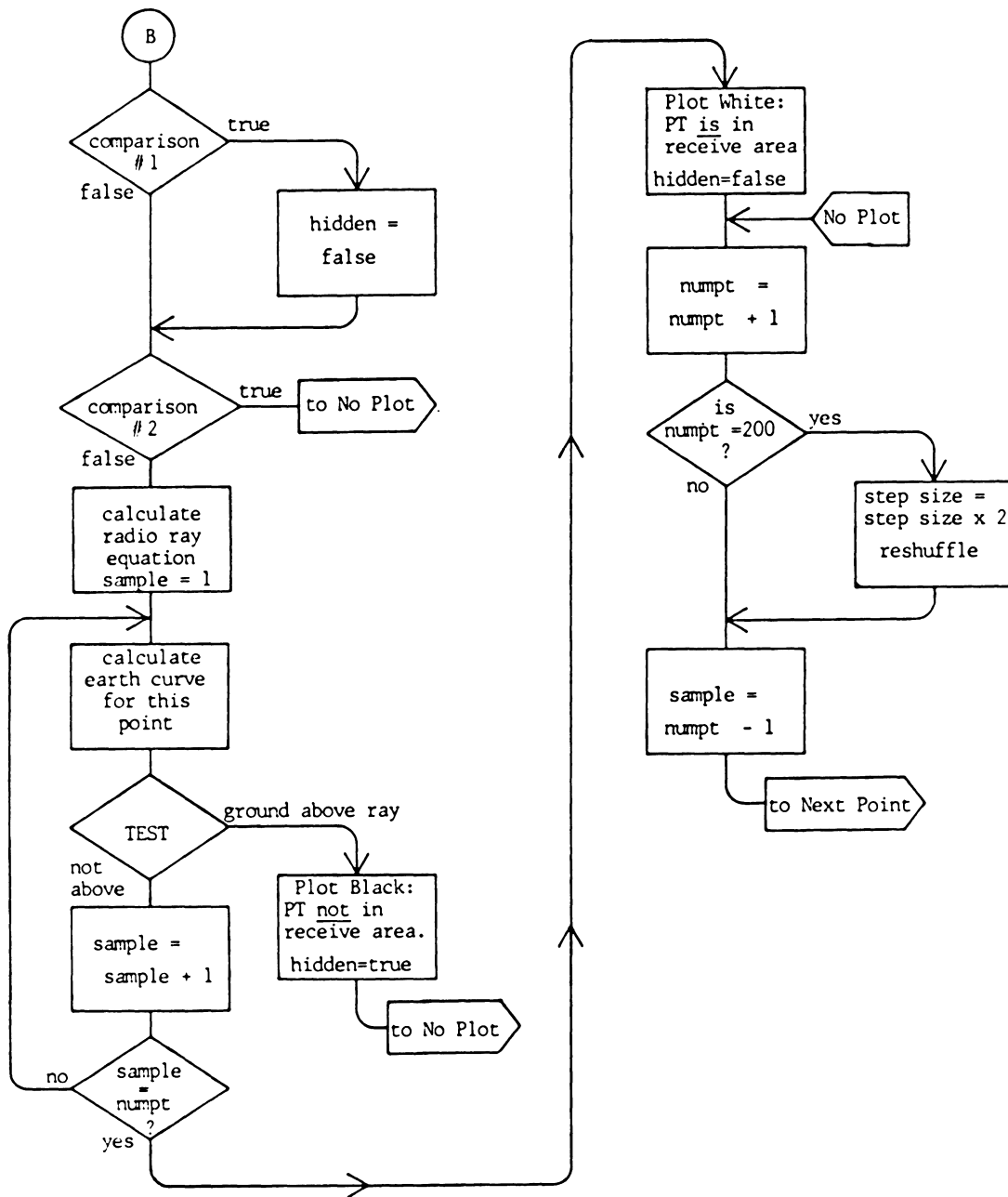


Figure A3.1 COVAREA.PAS flow chart continued.

Computer Code for COVAREA.PAS

```

PROGRAM CoverageAREAcaculation;
{$C-}                                { FINAL VERSION  }
{$I TYPEDEF.SYS}
{$I GRAPHIX.SYS}
{$I KERNEL.SYS}
{
  This program will determine the coverage area from a selected
  transmitter for the entire DEM.  Output is to the screen, which
  can be saved and recalled later.
}
LABEL Nextray, nextpoint, retry, wayt, noplot;
Const
  Indisk   = 'C:' ;      esc   = #27;
  Outdisk  = 'A:' ;      scale = 1;
  Eradius  = 6370000.0;
  maxelev  = 199 ;
  maxnode  = 463 ;      { angstp = 0.01745329252; = 1 degree }
  slicemax = 58 ;      { angstp = 0.03490658504; { = 2 degrees }
Type
  coord = (E,N);
  point = array [coord] of real;
  TypeB = record
    sequence : integer;
    elnum    : integer;
    Gnot     : point;
    GmaxN    : real;
    eldat    : array [0..maxnode] of integer;
  end;
Var
  DataBank : file of TypeB;
  Profile  : TypeB;
  fileA, output : Text;
  {}
  EXT      : string[4];
  str4     : string[4];
  Header   : string[144];
  outstr   : string[14];
  chr      : char;
  {}
  Pt, start, TL0C : Point;
  slice : array [0..slicemax,0..maxnode] of integer;
  bounds : array [0..slicemax,0..2] of Real;
  elev   : array [0..maxelev] of integer;
  corners: array [0..3] of point;
  cnr     : array [0..3] of point;
  {}
  min, max, Tantht, RantHt, TRDist, Delx, DelY, rayHt,
  srcelv, Telv, K, theta, sintheta, costheta, Rjunk,
  orgX, orgY, l1, l2, DelH, mray, stpang : Real;
  {}
  numB, Ijunk, numpts, node, cnt, sample, Alpha, omega,
  sze, step, endslice, slicnt, maptr, zone : Integer;
  {}
  knt : coord;
  hidden, off, calcTelv, RwestT : Boolean;

```

```

Procedure ChkTLOC ( var LOC : point ); { Subprogram for checking }
Label again ; { the transmitter location }
Var
    Bot,Top : Real;
Begin
again:  off:=false;
        maptr:=trunc((LOC[E]-Start[E])/30);
        If maptr>(numB-1) then off:=true;
        If maptr<0 then off:=true;
        If not off then Begin
            seek(databank,maptr);
            read(databank,profile);
            With profile do Begin
                Top:=GmaxN ;
                Bot:=Gnot[N];
            end;
            If LOC[N]>Top then off:=true;
            If LOC[N]<Bot then off:=true;
        end;
        If off then Begin
            clrscr;
            gotoxy(5,10);
            writeln('transmitter location: ',LOC[E],'E ',LOC[N],'N');
            writeln(' is not on the map!');
            write(' .....enter new Easting: ');readln(LOC[E]);
            write(' .....enter new Northing: ');readln(LOC[N]);
            goto again;
        end;
end;

Procedure Crash (code : byte); { Subprogram for exiting prematurely }
Begin
    Writeln('! your program has crashed at ',code,' !');
    repeat until keypressed;
    leavegraphic;
    halt;
end;

Procedure LDMAP; { Subprogram for loading the map slice }
label return;
Begin
    maptr:=trunc((Pt[E]-Start[E])/30);
    { check if maptr is pointing on the map }
    If maptr>(numB-1) then off:=true;
    If maptr<0 then off:=true;
    If off then goto return; { If not then return }
    If not RwestT then Begin
        Alpha:=maptr;
    If (alpha+slicemax)>=numB then omega:=numB-1 else omega:=(alpha+slicemax)
    end;
    If RwestT then Begin
        omega:=maptr+1;
    If (omega-slicemax)<=0 then alpha:=0 else alpha:=(omega-slicemax)
    end;
    slicnt :=0 ;
    maptr :=alpha;
    seek(databank,alpha);

```

```

Repeat
  Read(databank,profile);
  If profile.sequence <> (maptr+1) then crash(1);
  with Profile do Begin
    bounds[slicnt,0] := Gnot[E] ;
    bounds[slicnt,1] := Gnot[N] ;
    Bounds[slicnt,2] := GmaxN ;
    For node := 0 to (elnum-1) do slice[slicnt,node]:=eldat[node];
  end; {of with statement}
  slicnt:=slicnt+1;
  maptr:=maptr+1;
  Until maptr=omega+1;
  maptr:=alpha;      {map ptr points to beginning of slice}
  endslice:= slicnt-1; {reset end of slice}
  { set slicnt to profile just west of Pt. }
  If RwestT then slicnt:=endslice-1 else slicnt:=0;
return: randomize;
end;

{ Subprogram for finding the elevation ... }
Procedure FINDELV ; {slicnt must be preset to profile west of PT }
{ using method II }
Var
  elqua: array [0..3] of integer;
  Xup,elA,elB : Real;
Begin
  Rjunk:=(pt[N]-Bounds[slicnt,1])/30;
  node:=trunc(Rjunk);
  Xup:=Frac(Rjunk);
  elquad[0]:=slice[slicnt,node] ;
  elquad[1]:=slice[slicnt,node+1] ;
  elquad[2]:=slice[slicnt+1,node] ;
  elquad[3]:=slice[slicnt+1,node+1];
  elA:=(((elquad[1]-elquad[0])/30)*xup)+elquad[0] ;
  elB:=(((elquad[3]-elquad[2])/30)*xup)+elquad[2] ;
  Rjunk:=(((elB-elA)/30)*(pt[E]-bounds[slicnt,0])) ;
  Elev[sample]:=round(Rjunk + elA);
end;

BEGIN { Start of MAIN PROGRAM }
{ Initialize some stuff ... }
  textmode(3);
  textbackground(Blue);
  textcolor(Lightgreen);
  clrscr;
  K:=4/3;

  { Begin input of user info ... }
  GotoXY(5,12);
  write('Input area identifying extension, e.g."JOP" or "QNT" => ');
  readln(str4);
  for cnt:=1 to 4 do insert(upcase(copy(str4,cnt,1)),EXT,cnt);
Assign(FileA,concat(Indisk,'LOGRECA',Ext));
reset(fileA);
Writeln(' Reading LOGRECA',ext,' ...'); { This section reads the }
  Readln(fileA,header); { type A record and checks }
  readln(fileA,Ijunk); { DEM level code } { the codes }
  If Ijunk<>1 then crash(4);
  readln(fileA,IJunk); If Ijunk<>1 then crash(5); { elev. patrn code }
  readln(fileA,Ijunk); If Ijunk<>1 then crash(6); { grnd. ref. code }
  readln(FileA,zone); { UTM co-ordinate zone }
  readln(FileA,Ijunk); If Ijunk<>2 then crash(7); { grnd. units code }
  readln(FileA,Ijunk); If Ijunk<>2 then crash(8); { elev. units code }
  readln(FileA,Ijunk); If Ijunk<>4 then crash(9); { # sides map area }

```

```

        For cnt:=0 to 3 do
            For knt:=E to N do readln(FileA, corners[cnt, knt]);
        readln(FileA, min); { minimum and maximum elevations in the }
        readln(fileA, max); { entire map area }
        readln(FileA, Rjunk); If Rjunk<>0.0 then crash(10); { ang. primary axis }
        readln(FileA, Ijunk); { accuracy code <- is there a logical rec. C ? }
        readln(FileA, rjunk); If Rjunk<>30.0 then crash(11); { res. code: N }
        readln(FileA, Rjunk); If Rjunk<>30.0 then crash(12); { res. code: E }
        readln(FileA, Rjunk); If Rjunk<>1.0 then crash(13); { res. code: Up }
        readln(fileA, NumB); { # of type B records = # of map profiles }
    Close(fileA);

    clrscr;          { Aquire transmitter information }
    gotoXY(5,10);
    chr:=' ';
writeln(' TRANSMITTER:');
    calcTelv:=true;
    WRITELN('Input transmitter location in UTM co-ordinates (meters) ...');
    write('          Easting: '); Readln(TLOC[E]);
    write('          Northing: '); Readln(TLOC[N]);
    write('    Input Transmit Antenna height in meters: ');
    Readln(TantHt);
    write('    Input Elevation of transmitter location? (Y/N): ');
    readln(chr);
    If (chr='Y')or(chr='y') then begin
        calcTelv:=false;
        write('Input transmitter elevation in meters: ');
        Readln(Telv);
    end;

    clrscr;          { Aquire the receiver information }
    gotoXY(5,10);
Writeln(' RECEIVER:');
    write('    Input receive antenna height in meters: ');
    Readln(RantHt);

    clrscr;          { Aquire area information }
    gotoxy(5,10);
Writeln(' AREA:');
writeln('Input angle from transmitter (degrees) to begin area calculations,');
    write('    note: 0 degrees = due east in UTM coordinates: ');
    readln(Rjunk);
    theta:=(pi/180)*Rjunk;
    write('    Input arc to be considered (degrees): ');
    readln(Rjunk);
    Stpang:=theta+(Rjunk*(pi/180)); { Calculate stop angle }

```

```

clrscr;          { New K ? }
gotoxy(5,10);
write('K=',k,'   Input new K ? (Y/N): '); readln(chr);
  If (chr='Y')or(chr='y') then begin
    write('Input new K: ');
    readln(K);
    end;

{ If TL0C is in Lat,Long it will be converted here }

writeln('Loading PROFILE',EXT,' ...');
  Assign(databank,concat(indisk,'profile',ext)); { Open Databank }

  reset(databank); { Set reference point: Start[E,N] }
  seek(databank,0);
  read(databank,profile);
  If profile.sequence<>1 then crash(14);
    With Profile do begin
      Start[E]:=Gnot[E]; { Set Start to most SW node }
      start[N]:=Gnot[N];
    end;

{ check if TL0C is in the map area }
  writeln('Checking transmitter location ... ');
  ChkTL0C(TL0C);

{ calculate screen coordinate offsets for the graphix ... }
  orgX:=corners[1,E]-6000.0 ;
  orgY:=corners[1,N]+420.0 ;

{ start graphix }
  Initgraphic;
  drawborder;
  Defineworld(1,0,0,22500,15000);
  selectworld(1);
  selectwindow(1);          { Set up screen coordinates }

  for cnt:=0 to 3 do begin
    cnr[cnt,E]:=(corners[cnt,E]-orgx);
    cnr[cnt,N]:=(orgy-corners[cnt,N]);
  end;

  drawline(cnr[0,E],cnr[0,N],cnr[1,E],cnr[1,N]); { Draw neatlines of
  drawline(cnr[1,E],cnr[1,N],cnr[2,E],cnr[2,N]);          DEM }
  drawline(cnr[2,E],cnr[2,N],cnr[3,E],cnr[3,N]);
  drawline(cnr[3,E],cnr[3,N],cnr[0,E],cnr[0,N]);

  drawpoint((TL0C[E]-orgX),(orgY-TL0C[N])); { Place dot at Transmitter }

  drawtextw(500,3000,scale,'TRANSMITTER LOCATION:');
  STR(TL0C[E]:14:2,OUTSTR);
  drawtextw(500,3600,scale,OUTSTR+'E');
  str(TL0C[N]:14:2,outstr);
  drawtextw(500,4200,scale,outstr+'N');

  drawtextw(300,6000,scale,'TRANSMIT ANTENNA HEIGHT:');
  STR(TantHt:14:0,outstr);
  drawtextw(500,6600,scale,outstr+' meters');

  drawtextw(500,8500,scale,'TRANSMITTER ELEVATION:');

  drawtextw(17300,3000,scale,'RECEIVE ANTENNA HEIGHT:');
  STR(RantHt:14:0,outstr);
  drawtextw(17300,3600,scale,outstr+' meters');

```

```

gotoxy(65,10);
write('ANGLE:');

gotoxy(68,3);
write(ext);

drawtextw(19100,10150,scale,'1 km');
drawline(19000,11000,20000,11000);
drawline(18600,9600,18600,10600);

{ All set now to begin calculations ... }
SAMPLE:=0;
Pt[E]:=TLOC[E];
Pt[N]:=TLOC[N];
If ((theta>1.570796327)and(theta<4.71238898)) then RwestT:=true
  Else RwestT:=false;
LDMAP;
If Calctelv then Findelv
  Else elev[0]:=round(Telv);
srcelv:=tantHt+elev[0];
STR(elev[0],outstr);
drawtextw(3000,9100,scale,outstr+' meters');

Nexttray:TRDist:=0.0;
gotoxy(1,1);
writeln('numpts=  ');
numpts:=2;
Ijunk:=round((180/Pi)*theta);
GOTOxy(70,11);
write(Ijunk,'  ');
sintheta:=sin(theta);
costheta:=cos(theta);
step:=30;
size:=15;
hidden:=false;

Nextpoint: sample:=sample+1;
gotoxy(1,1);
writeln('numpts=',numpts);
TRdist:=TRDist+step;

{ Calculate the next pt }
delx:=TRDist*costheta;
dely:=TRDist*sintheta;
Pt[E]:=Delx + TLOC[E];
Pt[N]:=Dely + TLOC[N];

{ Is new pt in databank? }
{ Is new pt in the current map slice? }
off:=false;
If Pt[E]>bounds[endslice,0] then LDMAP; { then load new map slice }
  If Pt[E]<bounds[0,0] then LDMAP
    Else slicnt:=trunc((Pt[E]-bounds[0,0])/30);

If not off then begin
  If Pt[N]>Bounds[slicnt,2] then off:=true;
  If Pt[N]<Bounds[slicnt,1] then off:=true; { then PT is off map
    end;                                     { to North or South

If off then begin          { Calculate next theta }
  theta:=theta+angstp;
  If theta>stpang then goto wayt; { If true, leave program }
  If theta>(2*Pi)then begin
    theta:=theta-(2*Pi);
    stpang:=stpang-(2*Pi);
  end;

```

```

    sample:=0;
    If ((theta>1.570796327)and(theta<4.71238898)) then RwestT:=true
      Else RwestT:=false;
    goto Nextray; { start next radial }
  end;

{ If got here then PT must be on map so ... }
Findelv;
If sample<numpts then goto Nextpoint;
If (elev[sample]>=elev[sample-1]) then hidden:=false;
If (elev[sample]<elev[sample-1])and(hidden) then goto noplot;

Dely:=RantHt+elev[numpts]-srcelv; { calculate ray equation }
mray:=Dely/TRDist;

L1:=step;
sample:=1;

Repeat { Loop for checking blockage }
  L2:=TRDist-L1;
  DelH:=(L1*L2)/(2*Eradius*K);
  DelX:=step*sample;
  RayHt:=(mray*DelX)+srcelv;
  If ((elev[sample]+DelH)>=rayHt) then hidden:=true;
  If hidden then goto noplot;
  L1:=L1+step;
  sample:=sample+1;
Until sample=numpts;
  { plot white : point is in coverage area }
  hidden:=false;
Drawsquare((Pt[E]-orgX-sze),orgY-Pt[N]-sze,Pt[E]-orgX+sze,orgY-Pt[N]
+sze,true);
Noplot: numpts:=numpts+1; { Jump to here if PT not in coverage
area }

If numpts>maxelev then begin { Code to reset step size }
  step:=step*2;
  sze:=sze*2;
  sample:=1;
  Ijunk:=2;
  Repeat
    elev[sample]:=elev[Ijunk];
    Ijunk:=Ijunk+2;
    sample:=sample+1;
  Until Ijunk>maxelev;
  numpts:=sample;
end;

sample:=numpts-1;
goto nextpoint; { Go do it again ... }

Wayt: Close(databank); { Jump here when program is over }
gotoxy(65,10);
write(' ');
gotoxy(70,11);
write(' ');
setcolorblack;
drawpoint((TLOC[E]-orgX),(orgY-TLOC[N])); { Plots a dot at TLOC }
setcolorwhite;
setforegroundcolor(lightgreen);

```

```

gotoxy(1,1);
writeln('SAVE SCREEN?');
write(' (Y/N): ');
readln(chr);
gotoxy(1,1);
writeln(' ');
write(' ');
  If (chr='Y')or(chr='y') then begin
    gotoxy(1,1);
    writeln('FILENAME?(+EXT): ');
    READln(outstr);
    gotoxy(1,1);
    writeln(' ');
    write(' ');
    drawborder;
    savescreen(outstr);
    end;

drawborder;
repeat until keypressed; { holds output on screen }
leavegraphic;
END.

```

Appendix IV

This appendix contains the computer code for the program which calculates coverage area from a transmitter to a selected area of the DEM. This area is output to be at approximately the same scale as the paper map.

```

PROGRAM ONEtoONEcoverageareacalculation;
{$I TYPEDEF.SYS}
{$I GRAPHIX.SYS}
{$I KERNEL.SYS}
{
    This program will determine the coverage area from a selected
    transmitter site to a (maxX)by(maxY) square meter area. (the box)
    The output is on the same scale as the paper map. The box is
    selected by specifying the location of the top right hand corner.
}
LABEL  Nextheta, Checkpnt, Nextray, nextpoint, retry, wayt, noplot;
Const
    maxnode = 463 ;      maxX = 5417.8;
    slicemax = 57 ;      maxY = 3822.8;
    maxelev = 199 ;      angstp= 0.00872664626; { = .5 degree in
    Indisk = 'C:' ;                      radians }
    Outdisk = 'A:' ;
    Eradius = 6370000.0;
Type
    coord = (E,N);
    point = array [coord] of real;
    TypeB = record
        sequence : integer;
        elnum : integer;
        Gnot : point;
        datum : real;
        minmax : array [0..1] of real;
        eldat : array [0..maxnode] of integer;
    end;

```

```

Var
  DataBank : file of TypeB;
  Profile : TypeB;
  fileA, output : Text;
  {}
  EXT : string[4];
  Header : string[144];
  chr : char;
  str1,str2,str3,str4 : string[14] ;
  str47 : string[47];
  {}
  BLOC,ELOC, CLOC, Pt, start, TLOC : Point;
  slice : array [0..slicemax,0..maxnode] of integer;
  bounds : array [0..slicemax,0..2] of Real;
  elev : array [0..maxelev] of integer;
  corners: array [0..3] of point;
  {}
  Dist, m,
  min,max, Tantht, RantHt, TRDist, Delx, DelY, rayHt, srcelv, Telv,
  K, theta,sintheta,costheta, Rjunk, L1, L2, DelH, mray, stpang:Real
  {}
  numB, Ijunk, numpts, node, cnt, sample, Alpha, omega,
  sze, step, endslice, slicnt, maptr, zone : Integer;
  {}
  knt : coord;
  midE, midN, north, south , east, west,
  Atbox, Tinside,inbox, hidden, off, calcTelv,RwestT : Boolean;

Procedure ChkTLOC ( var LOC : point ); { Subprogram for checking that }
Label again ; { the transmitter location is }
Var { is in the DEM as specified }
  Bot,Top : Real;
Begin
again: off:=false;
  maptr:=trunc((LOC[E]-Start[E])/30);
  If maptr>(numB-1) then off:=true;
  If maptr<0 then off:=true;
  If not off then Begin
    seek(databank,maptr);
    read(databank,profile);
    With profile do Begin
      Top:=Gnot[N]+((elnum-1)*30);
      Bot:=Gnot[N];
      end;
      If LOC[N]>Top then off:=true;
      If LOC[N]<Bot then off:=true;
      end;
    If off then Begin
      clrscr;
      gotoxy(5,10);
      writeln('transmitter location: ',LOC[E],'E ',LOC[N],'N');
      writeln(' is not on the map!');
      write(' .....enter new Easting: ');readln(LOC[E]);
      write(' .....enter new Northing: ');readln(LOC[N]);
      goto again;
      end;
    end;

Procedure Crash (code : byte); { Subprogram for exiting prematurely }
Begin
  Writeln('! your program has crashed at ',code,' !');
  repeat until keypressed;
  leavegraphic;
  halt;
end;

```

```

Procedure LDMAP; { Subprogram for loading the map slice }
label return;
Begin
  maptr:=trunc((Pt[E]-Start[E])/30);
  { check if maptr is pointing on the map }
  If maptr>(numB-1) then off:=true;
  If maptr<0 then off:=true;
  If off then goto return;      { If not then return }
  If not RwestT then Begin
    Alpha:=maptr;
  If (alpha+slicemax)>=numB then omega:=numB-1 else omega:=(alpha+slicemax);
  end;
  If RwestT then Begin
    omega:=maptr+1;
  If (omega-slicemax)<=0 then alpha:=0 else alpha:=(omega-slicemax)
  end;
  slicnt:=0;
  maptr:=alpha;
  seek(databank,alpha);
Repeat
  Read(databank,profile);
  If profile.sequence <> (maptr+1) then crash(1);
  If profile.datum <> 0.0 then crash(2);
  with Profile do Begin
    bounds[slicnt,0] := Gnot[E];
    bounds[slicnt,1] := Gnot[N];
    Bounds[slicnt,2] := Gnot[N]+((elnum-1)*30);
    For node := 0 to (elnum-1) do slice[slicnt,node]:=eldata[node];
  end; {of with statement}
  slicnt:=slicnt+1;
  maptr:=maptr+1;
Until maptr=omega+1;
  maptr:=alpha;      {map ptr points to beginning of slice}
  endslice:= slicnt-1; {reset end of slice}
  { set slicnt to profile just west of Pt. }
  If RwestT then slicnt:=endslice-1 else slicnt:=0;
return: randomize;
end;

{ Subprogram for finding elevations }
Procedure FINDELV ; {slicnt must be preset to profile west of PT }
{ using method II }
Var
  elquad : array [0..3] of integer;
  Xup,elA,elB : Real;
Begin
  Rjunk:=(pt[N]-Bounds[slicnt,1])/30;
  node:=trunc(Rjunk);
  Xup:=Frac(Rjunk);
  elquad[0]:=slice[slicnt,node];
  elquad[1]:=slice[slicnt,node+1];
  elquad[2]:=slice[slicnt+1,node];
  elquad[3]:=slice[slicnt+1,node+1];
  elA:=(((elquad[1]-elquad[0])/30)*xup)+elquad[0];
  elB:=(((elquad[3]-elquad[2])/30)*xup)+elquad[2];
  Rjunk:=(((elB-elA)/30)*(pt[E]-bounds[slicnt,0]));
  Elev[sample]:=round(Rjunk + elA);
end;

```

```

BEGIN { Start MAIN PROGRAM }
{ Initialize some stuff ... }
    textmode(3);
    textbackground(Blue);
    textcolor(Lightgreen);
    clrscr;
    K:=4/3;

    { Begin input of user info ... }
    GotoXY(5,12);
    write('Input area identifying extension, e.g. ".JOP" or ".QNT" => ');
    readln(EXT);

Assign(FileA,concat(Indisk,'LOGRECA',Ext)); { This section reads the }
reset(fileA);                               { type A record and checks }
Writeln(' Reading LOGRECA',ext,' ...');      { the codes. }
    Readln(fileA,header);
    readln(fileA,Ijunk); { DEM level code }
    If ijunk<>1 then crash(4);
    readln(fileA,Ijunk); If Ijunk<>1 then crash(5); { elev. patrn code }
    readln(fileA,Ijunk); If Ijunk<>1 then crash(6); { grnd. ref. code }
    readln(FileA,zone); { UTM co-ordinate zone }
    readln(FileA,Ijunk); If Ijunk<>2 then crash(7); { grnd. units code }
    readln(FileA,Ijunk); If Ijunk<>2 then crash(8); { elev. units code }
    readln(FileA,Ijunk); If Ijunk<>4 then crash(9); { # sides map area }
    For cnt:=0 to 3 do
        For knt:=E to N do readln(FileA,cornerRadius[cnt,knt]);
    readln(FileA,min); { minimum and maximum elevations in the }
    readln(fileA,max); { entire map area }
    readln(FileA,Rjunk); If Rjunk<>0.0 then crash(10); { ang. primary axis }
    readln(FileA,Ijunk); { accuacy code <- is there a logical rec. C ? }
    readln(FileA,rjunk); If Rjunk<>30.0 then crash(11); { res. code: N }
    readln(FileA,Rjunk); If Rjunk<>30.0 then crash(12); { res. code: E }
    readln(FileA,Rjunk); If Rjunk<>1.0 then crash(13); { res. code: Up }
    readln(fileA,NumB); { # of type B records = # of map profiles }
Close(fileA);

    clrscr; { Aquire transmitter information }
    gotoXY(5,10);
    chr:=' ';
    writeln(' TRANSMITTER:');
    calcTelv:=true;
    WRITELN('Input transmitter location in UTM co-ordinates (meters) ...');
    write(' Easting: '); Readln(TLOC[E]);
    write(' Northing: '); Readln(TLOC[N]);
    write(' Input Transmit Antenna height in meters: ');
    Readln(TantHt);
    write(' Input Elevation of transmitter location? (Y/N): ');
    readln(chr);
    If (chr='Y')or(chr='y') then begin
        calcTelv:=false;
        write('Input transmitter elevation in meters: ');
        Readln(Telv);
    end;
    chr:=' ';

    clrscr; { Aquire receiver information }
    gotoXY(5,10);
    Writeln(' RECEIVER:');
    write(' Input receive antenna height in meters: ');
    Readln(RantHt);

```

```

    clrscr;      { Aquire BOX location }
    gotoxy(5,10);
    Writeln(' AREA:');
    WRITELN('Input NW corner location in UTM co-ordinates (meters) ');
    write('      Easting: '); Readln(CLOC[E]);
    write('      Northing: '); Readln(CLOC[N]);

    clrscr;      { New K ? }
    gotoxy(5,10);
    write('K=',k,'      Input new K ? (Y/N): '); readln(chr);
    If (chr='Y')or(chr='y') then begin
        write('Input new K: ');
        readln(K);
    end;

{ If TLOC is in Lat,Long it will be converted here }

writeln('Loading PROFILE',EXT,' ...');
Assign(databank,concat(indisk,'profile',ext)); { Open databank }
reset(databank);

seek(databank,0);      { Set reference point: Start[E,N] }
read(databank,profile);
If profile.sequence<>1 then crash(14);
    With Profile do begin
        Start[E]:=Gnot[E]; { Set Start to most SW node }
        start[N]:=Gnot[N];
    end;

{ check if TLOC is in the map area }
writeln('Checking transmitter location ... ');
ChkTLOC(TLOC);

{ start graphix }
Initgraphic;
str(CLOC[E]:9:1,STR1); { Start putting screen header together }
STR(CLOC[N]:9:1,STR2);
STR(TLOC[E]:9:1,STR3);
STR(TLOC[N]:9:1,STR4);
str47:=(STR1+E,'+STR2+'N T:'+STR3+'E,'+STR4+'N');
STR(TantHT:4:1,str1);
str(RantHt:4:1,str2);
STR(k:4:3,str3);
STR(Telv:4:1,str4);
Defineheader(1,str47+' Telv:'+str4+' T:'+str1+' R:'+str2+' K: '
+str3);
setheaderon;
drawborder;

Defineworld(1,0,0,maxX,maxY);
selectworld(1);      { Set up screen coordinates }
selectwindow(1);

{ set theta and stop angle ... }
{ Initialize ... }
midE:=false; midN:=false; north:=false;
south:=false; east:=false; west:=false;
Tinside:=false;

{ test ... }
If TLOC[N]>CLOC[N] then NORTH:=TRUE;
If TLOC[N]<(CLOC[N]-maxY) then south:=true;
If (not North)and(not South) then midN:=true;
If TLOC[E]<CLOC[E] then West:=true;
If TLOC[E]>(CLOC[E]+maxX) then East:=true;
If (not East)and(not West) then midE:=true;

```

```

{ assign values based on above tests ... }
If north then Begin
    BLOC[E]:=CLOC[E]+maxX;
    ELOC[E]:=CLOC[E];
    If west then begin
        BLOC[N]:=CLOC[N];
        ELOC[N]:=CLOC[N]-maxY;
    end;
    If east then begin
        BLOC[N]:=CLOC[N]-maxY;
        Eloc[N]:=CLOC[N];
    end;
    If midE then begin
        BLOC[N]:=CLOC[N];
        ELOC[N]:=CLOC[N];
    end;
end;
If south then Begin
    BLOC[E]:=CLOC[E];
    ELOC[E]:=CLOC[E]+maxX;
    If west then begin
        BLOC[N]:=CLOC[N];
        ELOC[N]:=CLOC[N]-maxY;
    end;
    If east then begin
        BLOC[N]:=CLOC[N]-maxY;
        Eloc[N]:=CLOC[N];
    end;
    If midE then begin
        BLOC[N]:=CLOC[N]-maxY;
        ELOC[N]:=CLOC[N]-maxY;
    end;
end;
If midN then begin
    If west then begin
        BLOC[E]:=CLOC[E];
        BLOC[N]:=CLOC[N];
        ELOC[E]:=CLOC[E];
        ELOC[N]:=CLOC[N]-maxY;
    end;
    If east then begin
        BLOC[E]:=CLOC[E]+maxX;
        BLOC[N]:=CLOC[N]-maxY;
        ELOC[E]:=CLOC[E]+maxX;
        Eloc[N]:=CLOC[N];
    end;
    If midE then begin
        Tinside:=true;
        BLOC[E]:=CLOC[E];
        BLOC[N]:=CLOC[N];
    end;
end;
end;

{ All these tests determine the
{ appropriate start and stop angles }
{ based on the relative position }
{ of the transmitter location to }
{ the box. }

{ All set now to begin calculations ... }
Delx:=BLOC[E]-TLOC[E];
delY:=BLOC[N]-TLOC[N];
Dist:=sqrt(sqr(delx)+sqr(delY));
m:=delY/DelX;
theta:=arctan(m);

    If Delx<0 then Begin
        RwestT:=true;
        theta:=theta+Pi;
    end
    Else RwestT:=false;

```

```

    If not Tinside then Begin
        Delx:=ELOC[E]-TLOC[E];
        Dely:=ELOC[N]-TLOC[N];
        m:=Dely/Delx;
        stpang:=arctan(m);
        If delx<0 then stpang:=stpang+Pi;
        end;
    If Tinside then stpang:=theta-(2*Pi);

    gotoxy(60,1);
    Ijunk:=round((180/Pi)*stpang);
    write('stpang=',Ijunk);
    SAMPLE:=0;

    Pt[E]:=TLOC[E];
    Pt[N]:=TLOC[N];
    LDMAP;

    If Calctelv then Findelv
        Else elev[0]:=round(Telv);
    srcelv:=tantHt+elev[0];

Nextray: TRDist:=0.0;
    Atbox:=false;
    numpts:=2;
    gotoxy(1,1);
    write('numpts=    ');
    Rjunk:=((180/Pi)*Theta);
    gotoxy(40,1);
    write('Theta=',Rjunk:4:1,' ');
    sintheta:=sin(theta);
    costheta:=cos(theta);
    step:=30;
    size:=15;
    hidden:=false;

Nextpoint: sample:=sample+1;
    TRdist:=TRDist+step;
    gotoxy(1,1);
    write('numpts=',numpts);

    Calculate the next pt )
    delx:=TRDist*costheta;
    dely:=TRDist*sintheta;
    Pt[E]:=Delx + TLOC[E];
    Pt[N]:=Dely + TLOC[N];

Is new pt in databank? )
Is new pt in the current map slice? )
    off:=false;
    If Pt[E]>bounds[endslice,0] then LDMAP; { then load new map slice }
    If Pt[E]<bounds[0,0] then LDMAP
        Else slicnt:=trunc((Pt[E]-bounds[0,0])/30);

    If not off then begin
        If Pt[N]>Bounds[slicnt,2] then off:=true;
        If Pt[N]<Bounds[slicnt,1] then off:=true; { then PT is off map
            end;                                { to North or South

    If not off then goto checkpnt; { PT is therefore not past the BOX }

```

```

    { If off map then next theta ... }
Nexttheta: theta:=theta-angstp;
           If theta<stpang then goto wayt;
           sample:=0;
           If ((theta>1.570796327)and(theta<4.71238898)) then RwestT:=true
           Else RwestT:=false;
           goto Nexttray;
    { must be on so ... }
checkpnt: Findelv;
           If sample<numpts then goto Nextpoint;
           If (elev[sample]>=elev[sample-1]) then hidden:=false;
           If (elev[sample]<elev[sample-1])and(hidden) then goto noplot;

{ Is new pt in Box? }
Inbox:=false;
If (Pt[E]>CLOC[E])and(Pt[E]<(CLOC[E]+maxX)) then begin
    If (Pt[N]<CLOC[N])and(Pt[N]>(CLOC[N]-maxY)) then inbox:=true;
    end;
If (inbox)and(not atbox) then atbox:=true; { just got to BOX }
gotoxy(25,1);
write('atbox=',atbox);

If atbox then begin { to next radial if finished here }
    If (not inbox) then goto nextheta;
    end;

If not inbox then goto noplot; { haven't reached box yet }

Dely:=RantHt+elev[numpts]-srcelv; { calculate ray line }
mray:=Dely/TRDist;
l1:=step;
sample:=1;
Repeat { Loop for checking for blockage }
    l2:=TRDist-l1;
    DelH:=(l1*l2)/(2*Eradius*K);
    DelX:=step*sample;
    RayHt:=(mray*DelX)+srcelv;
    If ((elev[sample]+DelH)>=rayHt) then hidden:=true;
    If hidden then goto noplot;
    l1:=l1+step;
    sample:=sample+1;
Until sample=numpts;

{ plot white : point is in coverage area }
hidden:=false;
Rjunk:=Pt[E]-CLOC[E];
Drawsquare(Rjunk-sze,(CLOC[N]-Pt[N]-sze),Rjunk+sze,(CLOC[N]-Pt[N]+sze),true);

```

```

Noplot:  numpts:=numpts+1; { jump to here if point not to be plotted }

      If numpts>maxelev then begin { reset step size as required }
        step:=step*2;
        size:=size*2;
        sample:=1;
        Ijunk:=2;
        Repeat
          elev[sample]:=elev[Ijunk];
          Ijunk:=Ijunk+2;
          sample:=sample+1;
        Until Ijunk>maxelev;
        numpts:=sample;
      end;

      sample:=numpts-1;
      goto nextpoint; { Go do it again }

Wayt:  Close(databank); { Jump to here to leave the program }

      If Tinside then begin { Draw dot at TLOC if it is inside BOX }
        setcolorblack;
        drawpoint((TLOC[E]-CLOC[E]),(CLOC[N]-TLOC[N]));
        setcolorwhite;
      end;

      drawborder;
      setforegroundcolor(lightgreen);

      gotoxy(1,1);
      write('Save screen? (Y/N): '); readln(chr);
      drawborder;
      If (chr='Y')or(chr='y') then begin { Save Screen ? }
        gotoxy(1,1);
        write('Enter filename<+EXT>: '); readln(str1);
        drawborder;
        savescreen(str1);
      end;

      repeat until keypressed; { Hold output on screen }
      leavegraphic;
END.

```

References

1. Elassal, Atef A. and Vincent M. Caruso USCGS Digital Cartographic Data Standards. Digital Elevation Models, U.S. Geological Survey Circular 895-B, 1983
2. Livingston, Donald C., The Physics of Microwave Propagation, Prentice-Hall, 1970, pp. 59-86.

**The vita has been removed from
the scanned document**