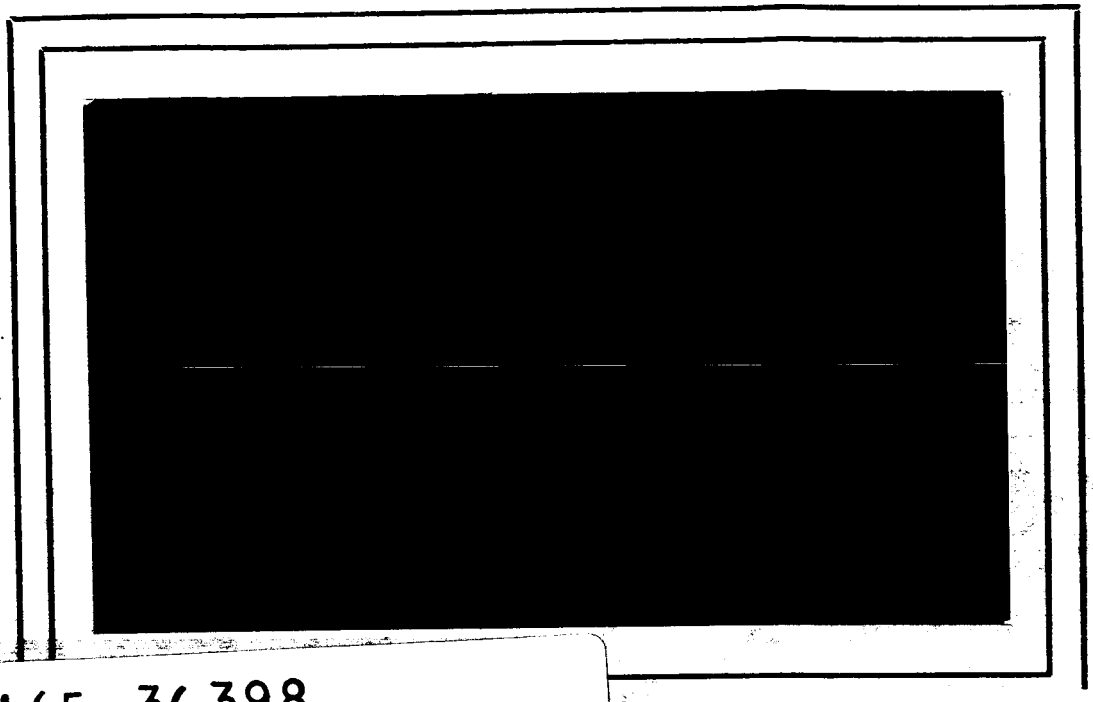


UNIVERSITY OF MARYLAND
COMPUTER SCIENCE CENTER

COLLEGE PARK, MARYLAND

Technical Report TR-65-21 August 1965
NsG-398

Pattern Recognition: V. SAMP - A Computer Program for Estimating Surface Area from Contour Maps

by

Mark S. Monmonier
John L. Pfaltz
Azriel Rosenfeld

ABSTRACT

36398

A FORTRAN program has been written which computes a linear approximation to the surface area of any given portion of a digitized contour map. The program also estimates the slope gradient at each point. It is slower, but appears to be somewhat more accurate, than other proposed methods of estimating average slope from contour maps.

Author

SAMP - A Computer Program for Estimating Surface Area from Contour Maps

Mark S. Monmonier, John L. Pfaltz and Azriel Rosenfeld

Introduction

In recent years there has been considerable progress in the development of methods for quantifying and processing geographical information. In particular, many quantitative parameters have been proposed for the description of terrain. Parameters which describe the terrain geometry in a region (average slope, for example) are usually derived from terrain elevation data such as are provided by a contour map.

This paper describes a FORTRAN program which efficiently approximates the surface area of any given portion of a contour map. The program, designated SAMP (Surface Area Measurement Program), accepts a digitized contour map as input. At each point of a rectangular grid, it estimates the gradient slope of the surface defined by the contour map. This is done by linearly interpolating directional slopes between the contour lines encountered in two orthogonal directions from the given point. The use of this linear approximation is tantamount to approximating the surface by a polyhedron with edges and vertices overlying those of the rectangular grid (see Figure 1). The area of each facet of the polyhedron is a simple trigonometric function of its gradient slope, and the sum of the facet areas approximates the area of the surface.

It should be noted that the linear approximation to the terrain surface made by SAMP is necessarily a lower bound, since a plane facet cannot have greater area than the corresponding portion of the actual surface. However, if the rectangular grid is fine enough, the approximation can be quite accurate. A higher order piecewise approximation (quadratic, exponential, etc.) could be used in place of the linear one, but would require much more computer time for its execution.

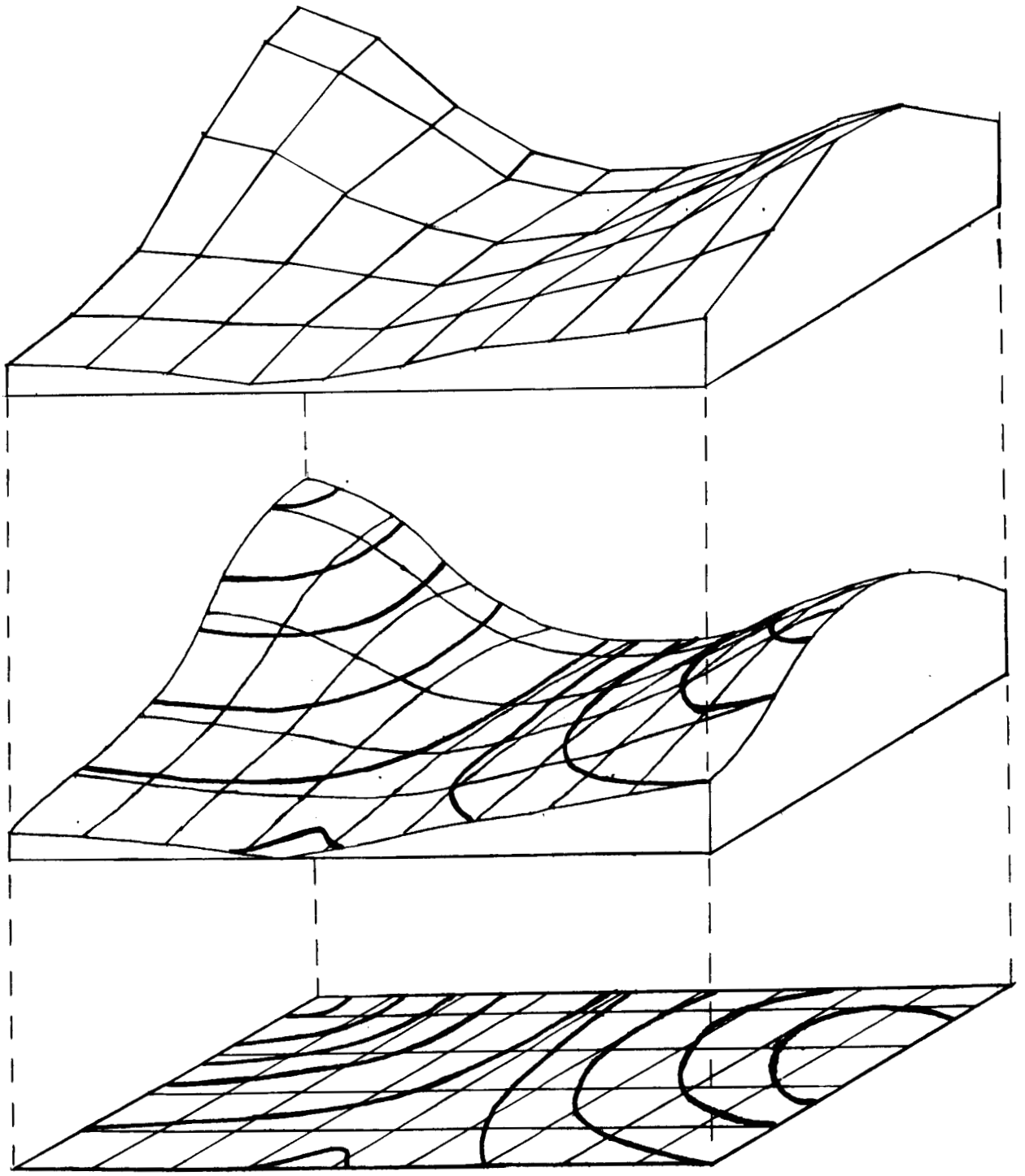


Figure 1

Polyhedral Approximation to the Terrain Surface

Description of the program

The basic input to SAMP is a digitized contour map. In general, this may be thought of as a matrix array in which elements corresponding to points on contour lines have values E_i equal to the elevations of the contours,* as shown in Figure 2, while any element not corresponding to a contour point is given a special value (represented in Figure 2 by a blank) which cannot be mistaken for an elevation. However, it will be shown below that SAMP can operate on much simpler inputs.

For each element of the digitized map matrix, the linear approximation to the gradient slope of the terrain at the corresponding point is obtained as follows: Strings of successive matrix elements are examined on both sides of the given element in each of two orthogonal directions. For each of the two directions, the following cases can occur:

Case 1. No contour line element occurs within a given distance of the starting point, on either side of it. In this case the terrain is assumed to be flat in that direction.

Case 2. The starting point itself is not a contour element, but contour elements occur within the given distance on both sides of it. In this case

a) If the elevations of these contours are the same, the terrain is assumed to be flat in that direction
b) If the elevations are different, they must differ by just one contour interval, so that the linear approximation to the terrain slope in that direction is given by

$$\tan\theta = \frac{I}{(n_1 + n_2)S}$$

where

n_1, n_2 (positive integers) are the number of steps taken on the two sides of the starting point until a contour element is encountered

S is the linear dimension on the terrain corresponding to a single matrix element

---

*A more complex method of labeling contour elements is needed if it is desired to allow for the possibility of sheer cliffs in the terrain. This possibility is not considered in this paper.

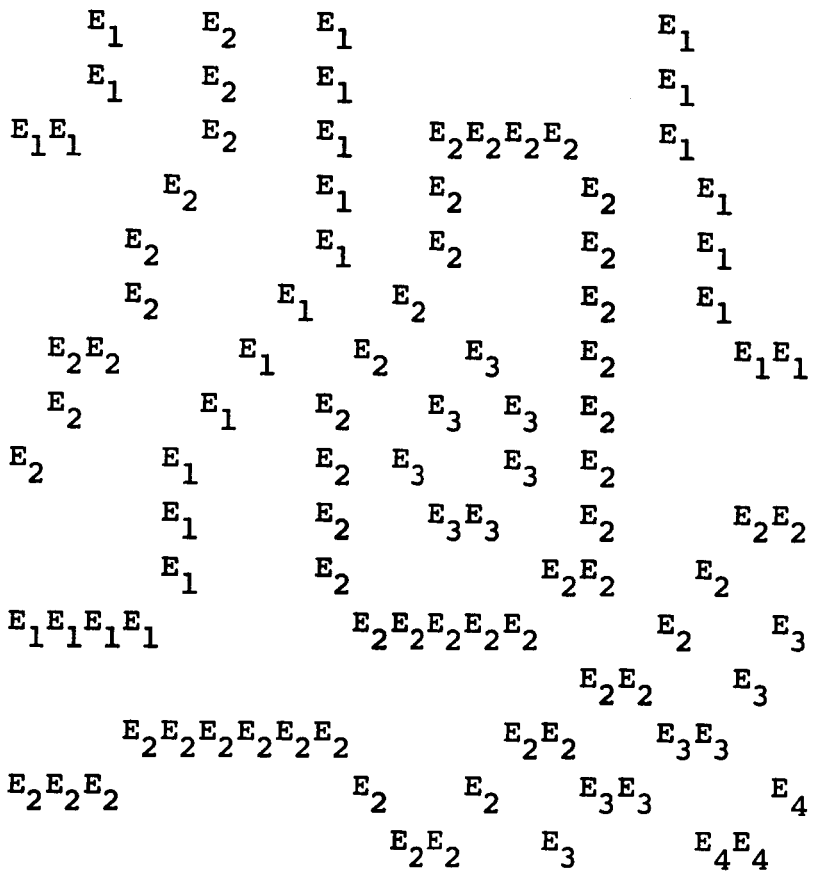


Figure 2. Fictitious Digital Contour Map Matrix

I is the contour interval (measured in the same units as S)

Case 3. The starting point is not a contour element, and no contour element occurs within the given distance of it on one side, but a contour element does occur within the given distance on the other side. In this case the arbitrary assumption is made that a contour element at a different elevation also exists on the other side, at 1.5 times the given distance from the starting point, and Case 2(b) is applied.

Case 2'-3'. If the starting point is a contour element, similar rules (which will not be described here in detail) are used to provide analogous linear approximations on both sides of the starting point.

This analysis is performed for each of the two orthogonal directions through the given element, yielding two orthogonal components $\tan\theta_1$ and $\tan\theta_2$ of a linear approximation to the terrain slope at that element. From these components, the area of the facet of an approximating polyhedral surface which overlies the given element can be immediately computed as

$$A = S^2 \sqrt{1 + \tan^2\theta_1 + \tan^2\theta_2}$$

The sum of these facet areas, taken over all elements of the digital map matrix, is the desired approximation to the terrain surface area.

In the examples given later in this paper, the search for a contour element extended for twenty elements on each side of the starting point in each of the two directions.

Simplifications

While the above analysis is quite straightforward, it should be noted that in order to resolve the contour lines on the digital contour map matrix, the matrix representing even a relatively small contour map (such as that shown in Figure 3) must contain of the order of a million elements. Such a matrix cannot be handled in the fast-access memories of current digital computers; it must be stored on magnetic tape or the like. Much of the complexity of the SAMP program results from the need to process the matrix efficiently and to perform all the associated bookkeeping.

A major simplification in the digital map matrix used by SAMP can be made if it is realized that SAMP does not require complete knowledge of the contour elevations. In fact, it suffices to store the parities of the elevations - that is, their remainders modulo 2. This is because two consecutive contour lines encountered along any terrain profile can only be at the same or consecutive elevations. For the purposes of the SAMP computation (Case 2 above), it is necessary to know whether the two contour lines are at the same or different elevations, but not what these elevations are. This means that the contour elevations in the digital map matrix can be represented using only two numbers, say 1 and 2 (see Figure 4).

An even more important simplification would result if it were possible to ignore even the contour elevation parities, and to represent all contour line elements in the digital matrix by (say) 1's. This would make it possible to construct the matrix automatically by scanning and digitizing the contour line overlay of an ordinary contour map.* SAMP would then be applied with the arbitrary assumption that Case 2a never occurs. As the results presented later

---

*This ignores the fact that numbers indicating selected contour elevations also occur on this overlay, and that the contour lines may have gaps in them to accommodate these numbers, as in Figure 3. In general, however, digital map matrices required for SAMP and similar programs can be constructed very rapidly by using a computer-controlled scanner to scan and display contour line overlays, and allowing an operator to insert elevations or elevation parities into the computer memory with the aid of a light pencil and suitable pushbutton controls.

in this paper indicate, good approximations to surface area can indeed be obtained from SAMP even if parities are ignored. These results can be made more plausible if it is realized that, as pointed out earlier, the linear approximation used by SAMP must necessarily be lower than the actual surface area, while ignoring Case 2a will tend to give higher than actual results. Thus the errors committed by ignoring Case 2a are in the right direction. In fact, the occurrence of Case 2a only rarely implies that the terrain is flat along the profile between the two equal-elevation contour lines. In general, the terrain will rise (or fall) somewhat, but not enough to reach another contour level, and then fall (or rise) again, along such a profile segment. If it is assumed in Case 2a as a rough approximation that there is a rise of half a contour interval over half the distance between the two equal-elevation contours, followed by an equal fall, it is easily seen that the contribution to surface area is essentially the same as that in Case 2b, where the rise is a full contour interval, but it takes place over the entire distance between the two contours.

Generalizations

The above description of SAMP assumes that surface area is being computed for the entire rectangular piece of terrain represented by the digital map matrix. However, it is also possible to compute surface area for selected region within a given map. The region of interest can be defined by marking each element of the matrix which corresponds to a point of the region with a special symbol. Alternatively, it suffices to delineate the region of interest by simply drawing a curve around it.* The elements within the region can then be distinguished by a computer program, described elsewhere [1], which labels connected components. This was done in one of the examples described in the next section.

Techniques similar to those used in SAMP can be applied to computing other parameters of a given terrain region - for example, its volume or roughness. However, an accurate approximation to volume or roughness requires a knowledge of contour elevation remainders modulo three, not merely modulo two. If it is known only whether two consecutive contour lines along a profile are at the same or different elevations, but not which of them is higher if they are different, it is impossible to distinguish between a smooth ramp surface and a corrugated surface.

---

*If a computer controlled display is used, the curve can be drawn with a light pencil or equivalent device.

Examples

To illustrate the application of SAMP, consider the USGS map of Big Delta, Alaska shown in Figure 3. This map was digitized using an optical scanner, at a resolution such that each element in the digital map matrix corresponds to a portion of the terrain about 100 feet on a side. A 360 by 260 element rectangular portion of the digitized map, the boxed region in the upper right hand corner of Figure 3, was selected, corresponding to a rectangular piece of terrain 5.29 miles by 7.33 miles long. Using a combination of hand and automatic processing techniques, the contour lines in this region were discriminated from the streams; the printed elevation figures were suppressed; and the contour lines were extended through the gaps caused by these figures, thinned to make them only one element wide, and assigned their proper elevation parities. The results of this preprocessing are shown as Figure 4, in which the boxed portion of Figure 3 has been rotated to stand on its left side.

A version of SAMP was applied to Figure 4 in which the two orthogonal directions used were along the rows and columns of the digital map matrix. The computed surface area was 40.96 square miles, as compared to a flat area of 5.29 x 7.33 = 38.77 square miles. As a check on the accuracy of the result, a second version of SAMP was also applied using the two diagonal directions; this yielded a computer surface area of 40.86 square miles.

To illustrate how SAMP can be applied to irregular regions, the drainage basin of a small stream was outlined (see Figure 4). Using a preprocessing routine, the elements outside the outline were given a special label (Figure 5), and SAMP was applied to find the surface area of the unlabeled region. The computed value, in both row-column and diagonal modes, was 5.56 square miles, as compared with a flat area of 5.24 square miles.

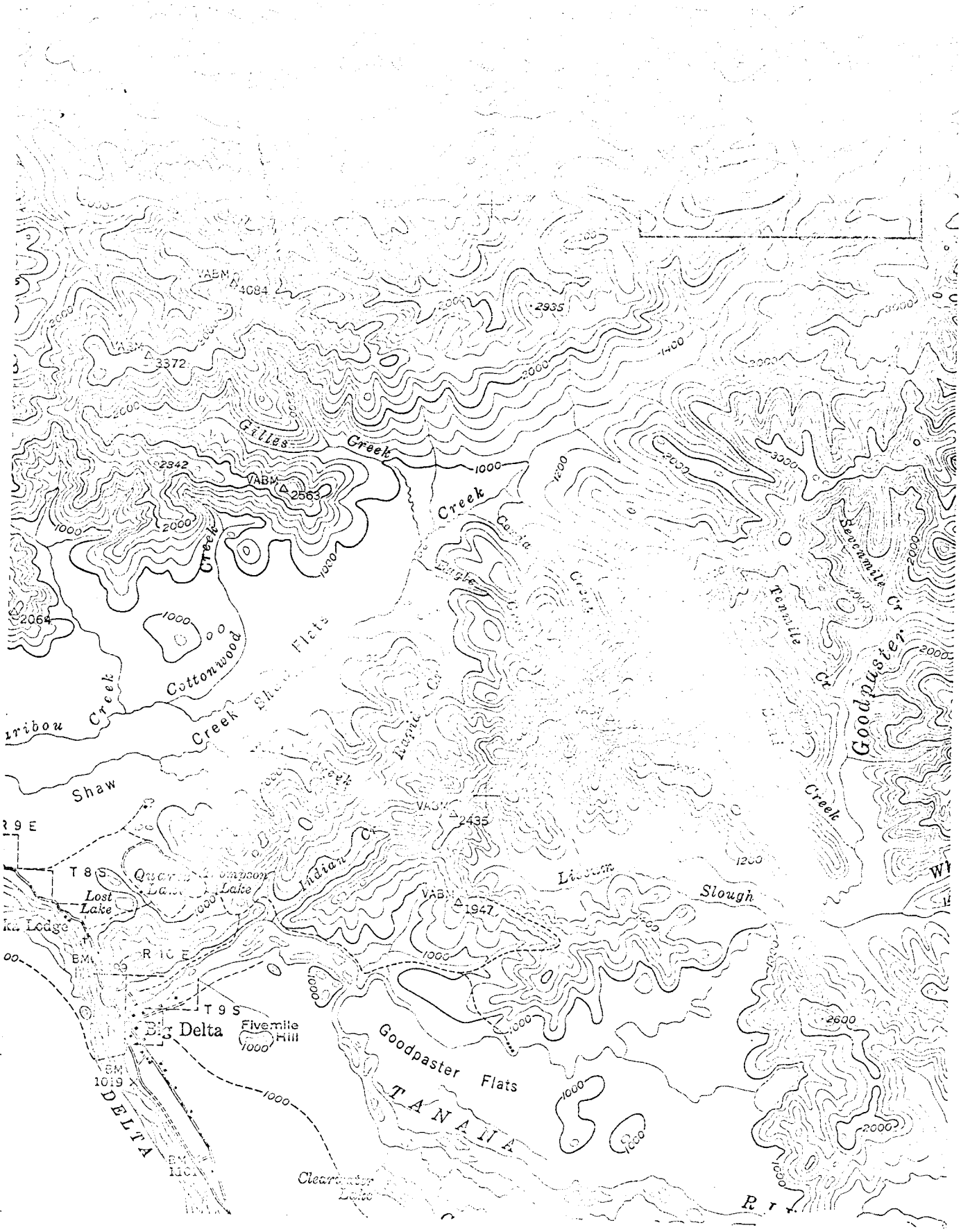


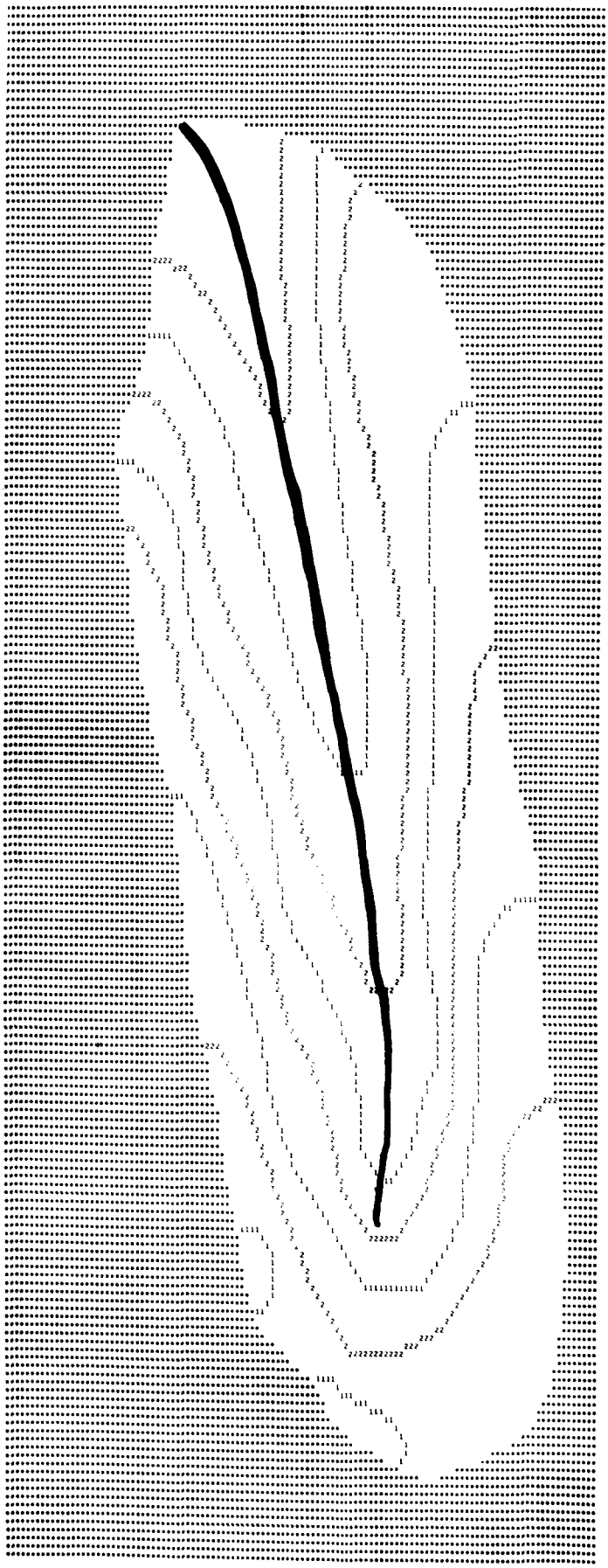
Figure 3. Contour Map (Big Delta, Alaska)

Figure 4

Digital Map Matrix for the Boxed Region
in the Upper Right Hand Corner of Figure 3

Figure 5

Drainage Basin
(Outlined in Figure 4)



Accuracy

In order to test the accuracy of SAMP, digitized contour maps of four geometric surfaces (see Figure 6) of known suface area were prepared. Those surfaces were

(a) A plane inclined at 45^{o} along the N-S direction, but not inclined in the E-W direction
(b) A plane inclined at 45^{o} along both N-S and E-W directions
(c) A right circular cone resting on a flat plane
(d) A hemisphere resting on a flat plane

SAMP was applied to these maps in both the row-column and diagonal modes, both considering and ignoring the contour elevation parities. The results are shown in Table 1.

The very small error in the calculated values for the two planes appears to be entirely due to accumulated round-off. This error seems to be insignificant in comparison with the errors inherent in the linear approximation. It should be noted that the errors when elevation parities are ignored tend to be no greater in these cases then the errors obtained using parities.

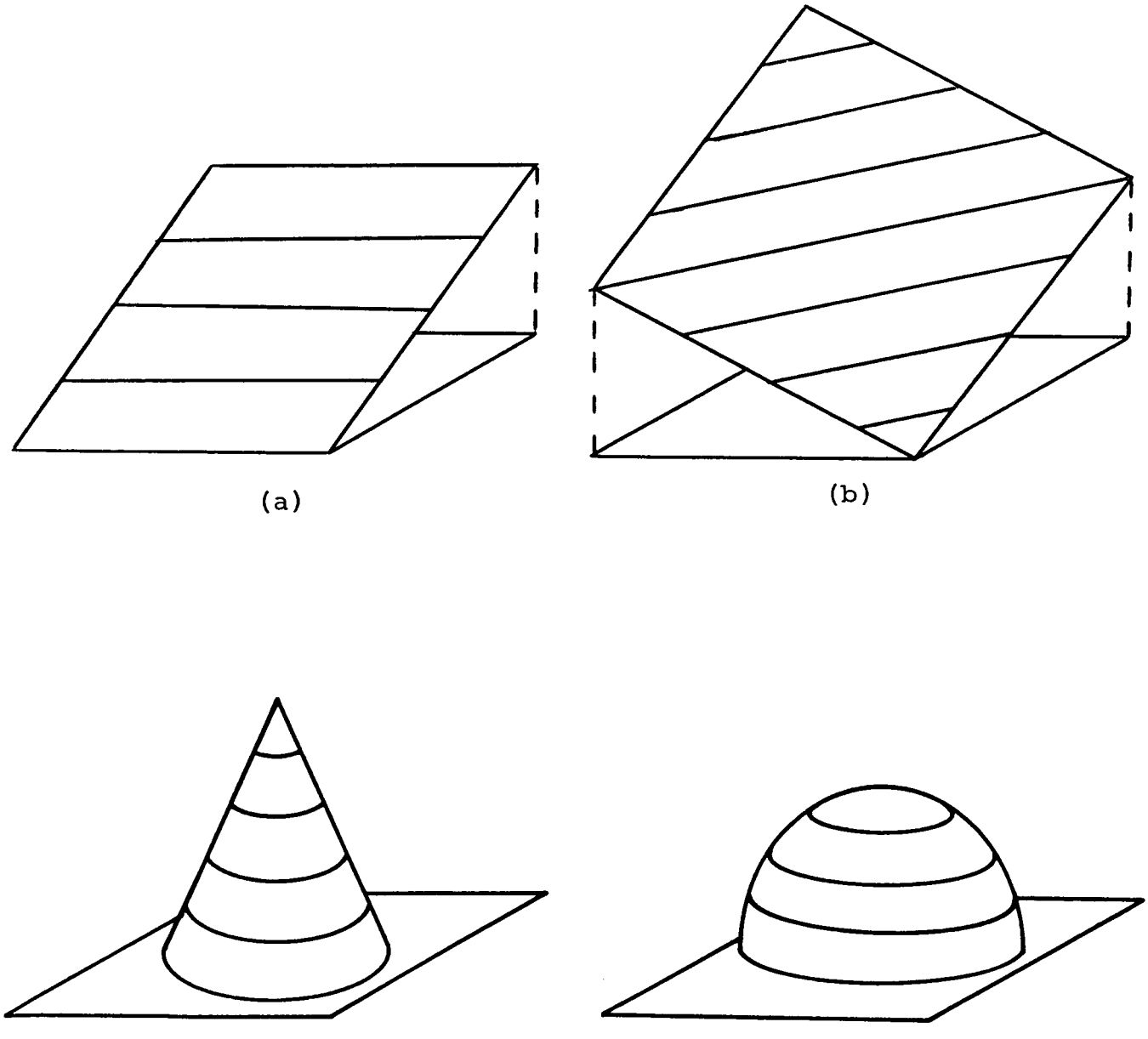


Figure 6

Geometrical Surfaces Analyzed by SAMP

Case	Flat area	True surface area	Areas calculated by SAMP			
			With parity		Without parity	
			Row-column mode	Diagonal mode	Row-column mode	Diagonal mode
Plane, 45-00	1681	2377	2377.3	2377.3	2377.3	2377.3
Plane, 45-45	1681	2911	2911.6	2911.6	2911.6	2911.6
Cone	10000	13125	12962.0	12827.0	13583.4	13472.0
Hemisphere	10000	17543	16109.6	15439.5	18796.4	18328.9

Table 1. Results of applying SAMP to geometrical surfaces

Comparison with related terrain measures

The SAMP technique, which is based on the computation of an approximate gradient slope at each map element, can also provide as output the average slope of the terrain region being analyzed. Conversely, the measures of average terrain slope which have been proposed in the past can be used to estimate terrain surface areas in accordance with the formula

$$A_s = A_f \sqrt{1 + 2\tan^2\alpha}$$

where A_s is the surface area, A_f the flat area, and α the average slope. It is of interest to compare the results obtained using these measures with those obtained using SAMP.

Perhaps the earliest method of estimating average terrain slope was proposed in 1890 by Finsterwalder [2]. He used the formula

$$\tan\alpha = IL/A_f$$

where I is the contour interval and L the total contour line arc length in the region. This technique was applied to the digitized map of Figure 4 in the form of a computer program which counted a unit arc length increment for each pair of horizontally or vertically adjacent contour line elements, and a $\sqrt{2}$ increment for each diagonally adjacent pair. This yielded an average slope of 15.0°. Using the surface area computed by SAMP and the formula given at the beginning of this section, an average slope of 13.3° was obtained. This differs by nearly 13% from the SAMP value, corresponding to about the same percentage difference between the computed areas; it is probably less accurate than the SAMP estimate.

A more recent average slope technique, due to Wentworth [3] and to Wood and Snell [4], uses the formula

$$\tan\alpha = IN/3361$$

where N is the number of contour lines encountered per mile of random traverse of the terrain. In general, this technique has been used only for large scale maps; it requires at least 100 contour intersections to yield a reasonably accurate result. Applying it to the digitized map of Figure

4, using eight random traverses which intersected contour lines 206 times, yielded an average slope of 12.2°. This differs by less than 8½% from the SAMP estimate, corresponding to a difference of about 8½% between the surface areas computed by the two methods; however, it is even lower than the SAMP figure, which is itself on the low side.

It may be concluded from these comparisons that SAMP represents a significant improvement over existing simplified methods of estimating average terrain slope and surface area. Furthermore, these methods can only provide averages over a region, while SAMP yields point by point information. For example, SAMP could be used to produce an approximate slope gradient contour map from the original elevation contour map. The current trend toward storage of map information in digital form will make programs like SAMP increasingly valuable in the future.

References

[1] Rosenfeld, A. and J. L. Pfaltz, Sequential Operations in Digital Picture Processing, J. Assoc. Comp. Mach., in press.

[2] Finsterwalder, S. Über mittleren Böschungswinkel und das wahre Areal einer topographischen Fläche, Sitzber. K. Ak. der Wiss., Math.-Phys. Kl. 20, 35-82, 1890.

[3] Wentworth, C.K., A Simplified Method of Determing the Average Slope of Land Surfaces, Am. J. Sci. 220, 1930, 184-194.

[4] Wood, W. F. and J. B. Snell, A Quantitative System for Classifying Land Forms, U. S. Army Quartermaster Research and Engineering Command Tech. Rept. EP-124, 1960.

APPENDIX A: Detailed Program Description of SAMP

PURPOSE:

To determine minimum surface area given a thinned digitized contour map stored on magnetic tape.

BASIC TERMINOLOGY:

The following terms relating to the storage of and access to the digitized contour map on tape are essential to understanding the description of SAMP. (See the accompanying diagram.)

PICTURE
The entire digitized contour map stored on magnetic tape.

DESIRED REGION
The rectangular part of the picture whose elements are to be inspected and/or have their area determined.

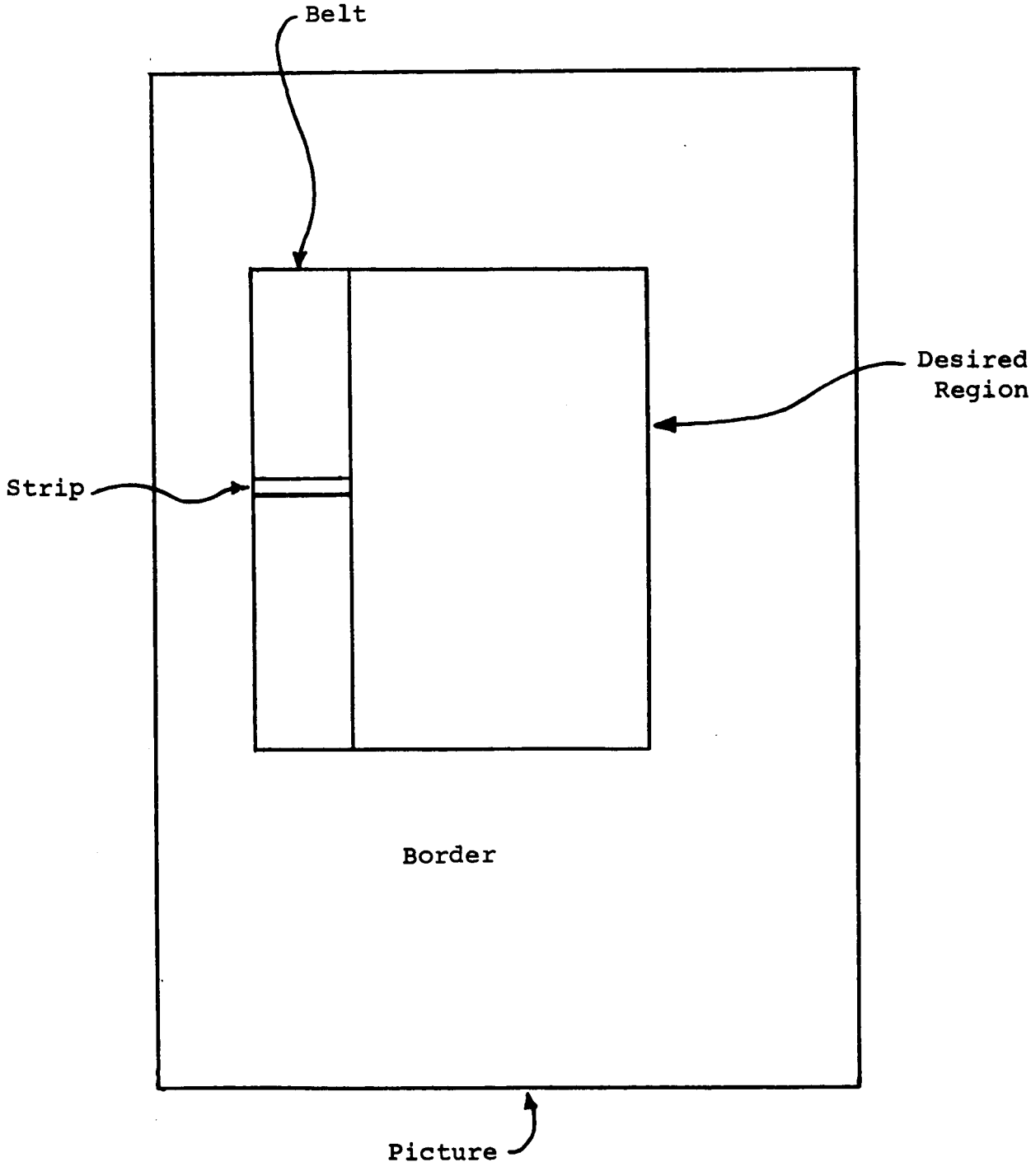
ROWS and COLUMNS
The rows and columns of the picture (or of the desired region). If the picture could be read into an array, these would be the rows and columns of the 2-dimensional array. The picture is actually stored on the tape row by row.

LINE
A row of the picture. LINE1 is the first row of the desired region, and LINE2 is the last row of the desired region.

ELEMENT
An element of a row or of a strip (see PICTURE ELEMENT). ELEM1 is used to indicate the first and ELEM2 the last column of the desired region. These bound and are included in the desired region.

PICTURE ELEMENT
This term is used to indicate a particular element or digit that is to have its surface area determined.

BELT
A section of the desired region that is from 1 to 21 elements in width and extends from the first row (LINE1) to the last row (LINE2) of the desired region.


Belt
Desired
Region
Strip
Border
Picture

STRIP
A row of a belt. A strip may contain from 1 to 21 elements.

BORDER
A frame at least 20 elements thick in the picture that must surround the desired region.

ROW-COLUMN MODE
Indicates that the elements being searched are along rows and columns in the desired region.

DIAGONAL MODE
Indicates that the elements being searched are successively along lines at 45-degree angles to the rows and columns of the desired region.

In summary, the picture is the set of all picture elements on the contour map. The desired region and the border are subsets of the picture whose union is the picture. The desired region is composed of one or more belts that do not overlap. Each belt is divided up into a number of strips which do not overlap. Each strip contains one or more picture elements.

BRIEF DESCRIPTION:

For the picture element whose area is desired, a search is made up to 20 elements out from that picture element to see if a contour is present. The number of elements tested until a contour, if any, is found is recorded along with the digit representing that contour. If no contour is found, a logical variable records that no contour was located in that direction.

The search is made in two orthogonal directions, on each side of the picture element. The row-column mode makes the search along the row and column of the picture which contain the picture element; the diagonal mode makes the search along the diagonals containing the picture element.

In each search direction, a search subroutine determines 1) whether or not a contour was found, 2) the distance to the contour (i. e., the number of picture elements out from the given one) and 3) the parity (if any) of the found contour. This information is used to compute two components of the slope of a plane facet approximating the terrain surface at

the given picture element. From these components, the area of the facet is computed as described in the body of this report. The formulas used are adjusted to the specified scale and contour interval on the digitized map.

The program provides for the use of a check tape in addition to the tape on which the digitized contour map is stored. This check tape represents an overlay for the contour map. If the digit for a particular element on the check tape falls within a specified range, the area for the corresponding picture element on the contour map tape is determined, and 1 is added to the tally of the number of elements whose area has been determined. This makes it possible to determine the surface area of an arbitrary region. If the area of a rectangular section of the map is desired, the check tape option is not employed.

USAGE:

SAMP requires as input a digitized contour map stored on logical tape NTAPE. If the check tape is to be employed, logical tape CTAPE is also required. Both of these tapes must be in standard format, as described at the end of this Appendix. The desired region must be sufficiently small to allow for a border of 20 elements (see RESTRICTIONS).

The user need only call SAMP1 (for the row-column mode) or SAMP2 (for the diagonal mode). He must specify the rectangular boundaries of the desired region; the two values (digits) that represent the contour parities; the scale, contour interval, and linear dimension of a picture (flat map) element; the print-switch number to govern the amount of printed output; the check switch (=1 for use with the check tape, $\neq$1 if this option is not used); and the high and low slice levels for the check tape.

The calling statements are: CALL SAMPn (NTAPE, CTAPE, LINE1, LINE2, ELEM1, ELEM2, PAR1, PAR2, SL, SH, SW, PTSW, SCALE, CI, E, TOTAR), where n = 1 or 2; the arguments are as follows:

1. NTAPE (integer) - logical tape number on which the digitized contour map is stored.

2. CTAPE (integer) - logical tape number on which the check region is stored.

3. LINE1 and LINE2 (integers) - the first and last rows of the desired region on the picture.

4. ELEM1 and ELEM2 (integers) - the first and last columns of the desired region on the picture.

5. PAR1 and PAR2 (integers) - the two parity digits representing contours.

6. SL and SH (integers) - the low and high slice levels. With the check tape option, a picture element's area will be determined only if it lies in the closed interval [SL, SH].

7. SW (integer) - the check switch (=1 for use of check tape; $\neq 1$ if the check tape is not used).

8. PTSW (integer) - the print switch:
   = 0 for total area only
   = 1 for total area and belt area only
   = 2 for total area, belt area, and the area of each individual picture element taken
   (1 and 2 are generally used only for debugging purposes)

9. SCALE (real) - the map scale.

10. CI (real) - the contour interval

11. E (real) - the length of side of a picture element (that is, the resolution of the scanner; must be measured in the same units as CI)

12. TOTAR (real) - the total area for the desired region; transmitted back to the calling routine by SAMP.

RESTRICTIONS:

The numbers in the calling sequence that are specified as integers above must be so declared in a type statement or by using a first letter that indicates an integer.

The desired region must be framed by a border at least 20 elements thick on the digitized contour map.

The compare region must be a picture that fits directly over the desired region with nothing lacking or left over.

The following types of error printout are possible:

1. AREA DESIRED TOO CLOSE TO TOP BOUNDARY. This indicates that an insufficient border has been left at the top of the desired region. The next three error messages are similar.

2. AREA DESIRED TOO CLOSE TO BOTTOM BOUNDARY.

3. AREA DESIRED TOO CLOSE TO LEFT BOUNDARY.

4. AREA DESIRED TOO CLOSE TO RIGHT BOUNDARY.

5. SWITCH LOW EXCEEDS SWITCH HIGH.

6. CLINES DOES NOT EQUAL THE AREA'S NUMBER OF LINES. This means that the number of rows on the check tape and the number of rows in the desired region do not match. The next error message indicates the same for the number of columns.

7. CELEM DOES NOT EQUAL THE NUMBER OF ELEMENTS FOR A ROW OF THE AREA.

8. DESIRED REGION EXTENDS BEYOND PICTURE LIMITS. This error message, which may be printed by either subroutine SPOT or by subroutine CHELM, may indicate that the specified desired region is too large for the picture or that the first row or column of the region is larger than the last row or column of the region. It may be preceded by a statement giving the line or element where this occurs. It seldom appears, as it was used mainly for debugging purposes.

TIMING:

The program should require about .0003 minutes to determine the surface area of one picture element. This value is not constant. The actual time depends on the density of the contours and other factors.

PROGRAM DESCRIPTION:

I. Major constants and arrays.

The only two major constants not already discussed are the logical variable SIDE and the integer variable Z. SIDE causes subroutine WORK to perform the search for contours in the diagonal mode if it is set to TRUE or in the row-column mode if it is set to FALSE. It is so set by SAMP2 and SAMP1, respectively. Z is the integer that indicates the width of a strip. It may range from 1 to 21.

The major arrays and their dimensions are the following:

1. TAKE (21)
A logical array for a strip. If TAKE (I) = .TRUE., the Ith element of the strip will have its surface area determined.

2. BB (21)
An integer array that contains a strip from the check tape.

3. AA (41, 61)
The integer array whose 21 central elements (a strip) in the 21st row may have their area determined.

4. CENT (21)
An integer array with the digits representing the picture elements in a strip

5. BETW (21)
A logical array indicating whether or not a picture element of a strip is on a contour line.

6. PACE (21, 4)
An integer array that contains for each picture element of a strip the number of elements searched (up to 20) before a contour line is found for each of the 4 directions in which the search is made.

7. PAR (21, 4)
An integer array that stores the parity of the contour found (if any) for each direction for each picture element of the strip.

8. FLAT (21, 4)
A logical array that tells whether or not a contour was

found for each direction for each picture element of the strip.

9. TAN1 (21) and TAN2 (21)
Real arrays containing the orthogonal tangents for each element of the strip.

10. AREA (21)
A real array representing the calculated surface area for each element of the strip.

II. List and Description of Subroutines

1. SAMP1 and SAMP2
These two subroutines are similar, except that SAMP1 executes the contour search around a picture element in the row-column mode and SAMP2 uses the diagonal mode. These subroutines separate the desired region into belts, determining the number and widths of the belts. They then call WORK to run the length of these belts and sum their areas and the number of elements whose areas are taken.

2. WORK
This subroutine calls the remaining subroutines in the following sequence and sums the areas and number of elements taken for every strip in the belt.

3. SPOT
This subroutine, when first called for a belt, reads from the picture tape the array AA(I,J) in which the searches are made. When this subroutine is called for subsequent strips in the same belt, it moves the rows of the array AA up one row (thereby discarding the previous first row) and reads a new last row from the picture tape. Thus, it goes strip-by-strip down a belt giving the array in which the searches are to be made.

4. CHELM
This subroutine reads a row from the check tape to get array BB(I), which corresponds to the strip centered in AA. This subroutine is called only if the check tape option is used.

5. CKDECK
This subroutine checks each element of BB to see if it

lies within the specified slice levels. If the element does not fit in this interval, TAKE(I) for the picture element is set to FALSE. This subroutine is called only if the check tape is used. TAKE(I) had previously been set to TRUE for all elements of the strip.

6. BANDPC and SDSPC

These subroutines are used for the row-column and diagonal modes respectively. They find PACE(I,J), PAR(I,J), BETW (I), CENT(I), and FLAT(I,J) for each picture element of the strip in their respective modes. PACE(I,J) is determined by checking in each J direction for a contour, comparing each successive element with PAR1 and/or PAR2 until a contour is reached. The parity of that contour is then stored in PAR(I,J). If CENT(I) is either PAR1 or PAR2, BETW(I) is set to FALSE, indicating that the Ith picture element of the strip is on a contour. If no contour is sensed after searching 20 elements out from the picture element for the Ith picture element of the strip in the Jth direction, FLAT(I,J) is set to TRUE. In SDSPC an additional check is made to prevent missing a contour by passing between two diagonally adjacent elements of a thinned contour.

7. BELTAN

In this subroutine the configuration of the terrain is determined for each picture element of the strip. This is done by checking arrays PACE, PAR, CENT, BETW, and FLAT to compare the topographic situation of the given picture element with the ten possible general configurations shown in the table below. Unless the terrain is found to be flat in the given direction, the tangent for that direction is determined from the above arrays and from the arguments SCALE, CI, and E.

8. SDBLTN

This subroutine is only called if the search for contour lines is made in the diagonal mode. Here it is necessary to apply a correction before computing the tangents, since PACE in the diagonal mode represents a search along the hypotenuse of, rather than along the side of, a 45-degree right triangle.

9. BELARA

This subroutine uses the tangents and the scale, contour interval, and element side length to determine the surface

area for each picture element of the strip.

Topographic configurations distinguished by SAMP

Case	Given picture element on a contour (elevation X*)	Contour elevations* found within 20 picture elements		Schematic representation of case
		First side	Second side	
A1	NO	X	X	X - - - X**
A2	NO	X	Y	X - - - Y
A3	NO	NONE	X	- - - - X
A4	NO	NONE	NONE	- - - - -**
B1	YES	Y	Y	Y - X - Y
B2	YES	X	Y	X - X - Y
B3	YES	NONE	X	- - X - X**
B4	YES	NONE	Y	- - X - Y
B5	YES	NONE	NONE	- - X - -**
B6	YES	X	X	X - X - X**

* "X" and "Y" represent two different elevation parities

** In these cases the terrain is judged to be flat in the given direction through the given picture element.

STANDARD UNPACKED TAPE FORMAT FOR DIGITAL PICTURES

All digitized pictures must be stored on tape row-wise left to right in FORTRAN binary format at 800 bpi (high density). Each row constitutes one logical record. Each picture element in a row constitutes one word of that record.

Preceding each picture on tape is one logical record of header information consisting of ten words as follows

1: Number of rows (logical records) in the picture
2: Number of picture elements (words) in each row (record)
3: Date picture was written on tape (e.g., 102764)
4: Identifying number of picture
5-10: Any necessary supplemental information

APPENDIX B: SAMP PROGRAM LISTING

```
$IBFTC SAMP1   LIST,REF,M94,XR7
      SUBROUTINE SAMP1 (NTAPE,CTAPE,LINE1,LINE2,ELEM1,ELEM2,PAR1,PAR2,SL
     1,SH,SW,PTSW,SCALE,CI,E,TOTAR)
C
C     THIS SUBROUTINE COORDINATES THE NECESSARY SUBROUTINES TO EXECUTE
C     THE DETERMINATION OF THE APPROXIMATE SURFACE AREA FROM THE CONTOUR
C     MAP STORED ON TAPE.
C     SAMP1 CONPUTES TANGENTS IN THE UP-DOWN MODE
C
C     MARK S. MONMONIER, 24 FEBRUARY, 1965
C
      LOGICAL SIDE,WRT1,WRT2,WRT3,COMP
      REAL SCALE,CI,E,TOTAR,BELTR
      INTEGER PAR1,PAR2,ELEM1,ELEM2,PTSW,Z,YN1,SL,SH,SW,CLINES,CELEM,CTA
     1PE
      COMP = .FALSE.
      IF (SW.EQ.1) COMP = .TRUE.
      IF (PTSW.EQ.0) GO TO 10
      IF (PTSW.EQ.1) GO TO 11
      IF (PTSW.EQ.2) GO TO 12
   10 WRT1 = .FALSE.
      WRT2 = .FALSE.
      WRT3 = .TRUE.
      GO TO 20
   11 WRT1 = .FALSE.
      WRT2 = .TRUE.
      WRT3 = .TRUE.
      GO TO 20
   12 WRT1 = .TRUE.
      WRT2 = .TRUE.
      WRT3 = .TRUE.
   20 CONTINUE
      IF (SL.GT.SH) WRITE (6,405)
      SIDE = .FALSE.
      REWIND NTAPE
      READ (NTAPE) NLINES,NELEM
      REWIND NTAPE
  300 IF (LINE1.LT.21) WRITE (6,401)
      N = NLINES - 20
      IF (LINE2.GT.N) WRITE (6,402)
      IF (ELEM1.LT.21) WRITE (6,403)
      N = NELEM - 20
      IF (ELEM2.GT.N) WRITE (6,404)
      TOTAR = 0.0
      NELTK = 0
      NV1 = LINE2 - LINE1 + 1
      NV2 = ELEM2 - ELEM1 + 1
      YN1 = ELEM1 - 20
      NV4 = NV2
      IF (.NOT.COMP) GO TO 420
      REWIND CTAPE
      READ (CTAPE) CLINES,CELEM
      REWIND CTAPE
      IF (CLINES.NE.NV1) WRITE (6,406)
      IF (CELEM.NE.NV2) WRITE (6,407)
  420 IF (NV2.GT.21) GO TO 450
      Z = NV2
      CALL WORK (YN1,LINE1,LINE2,Z,SIDE,WRT1,WRT2,PAR1,PAR2,NTAPE,BELTR,
     1CI,E,SCALE,ELEM1,CTAPE,COMP,SL,SH,NTAKEN)
      TOTAR = TOTAR + BELTR
```

```
      NELTK = NELTK + NTAKEN
      GO TO 600
  450 NV3 = NV2 - 21
      Z = 21
      CALL WORK (YN1,LINE1,LINE2,Z,SIDE,WRT1,WRT2,PAR1,PAR2,NTAPE,BELTR,
     1CI,E,SCALE,ELEM1,CTAPE,COMP,SL,SH,NTAKEN)
      TOTAR = TOTAR + BELTR
      NELTK = NELTK + NTAKEN
      NV2 = NV3
      YN1 = YN1 + 21
      IF (NV3.GT.0) GO TO 420
  600 CONTINUE
      NUMEL = NV1 * NV4
      IF (WRT3.AND.COMP) GO TO 700
      IF (WRT3) WRITE (6,411) NUMEL,TOTAR,SCALE,CI,E
      GO TO 705
  700 WRITE (6,412) NUMEL,NELTK,TOTAR,SCALE,CI,E
  705 CONTINUE
      REWIND NTAPE
  401 FORMAT (1H0,40H AREA DESIRED TOO CLOSE TO TOP BOUNDARY.//)
  402 FORMAT (1H0,43H AREA DESIRED TOO CLOSE TO BOTTOM BOUNDARY.//)
  403 FORMAT (1H0,41H AREA DESIRED TOO CLOSE TO LEFT BOUNDARY.//)
  404 FORMAT (1H0,42H AREA DESIRED TOO CLOSE TO RIGHT BOUNDARY.//)
  405 FORMAT (1H0,32H SWITCH LOW EXCEEDS SWITCH HIGH.//)
  406 FORMAT (1H0,50H CLINES DOES NOT EQUAL THE AREA/S NUMBER OF LINES./
     1/)
  407 FORMAT (1H0,67H CELEM DOES NOT EQUAL THE NUMBER OF ELEMENTS FOR A
     1ROW OF THE AREA.//)
  411 FORMAT (1H1,3X,28H ELEMENTS IN DESIRED REGION ,I10/3X,28H TOTAL AR
     1EA IN SQUARE UNITS ,E20.8/3X,9H SCALE = ,F20.9,20H CONTOUR INTERVA
     2L = ,F15.5,26H ELEMENT SIDE DIMENSION = ,F25.15//////)
  412 FORMAT (1H1,3X,18H ELEMENTS SEARCHED,I10/3X,18H ELEMENTS TAKEN   ,
     1I10/3X,28H TOTAL AREA IN SQUARE UNITS ,E20.8/3X, 9H SCALE = ,F20.9
     2,20H CONTOUR INTERVAL = ,F15.5,26H ELEMENT SIDE DIMENSION = ,F25.1
     35//////)
      RETURN
      END
$IBFTC SAMP2   LIST,REF,M94,XR7
      SUBROUTINE SAMP2 (NTAPE,CTAPE,LINE1,LINE2,ELEM1,ELEM2,PAR1,PAR2,SL
     1,SH,SW,PTSW,SCALE,CI,E,TOTAR)
C
C     THIS SUBROUTINE COORDINATES THE NECESSARY SUBROUTINES TO EXECUTE
C     THE DETERMINATION OF THE APPROXIMATE SURFACE AREA FROM THE CONTOUR
C     MAP STORED ON TAPE.
C     SAMP2 CALCULATES THE TANGENTS IN THE DIAGONAL MODE
C
C     MARK S. MONMONIER, 24 FEBRUARY, 1965
C
      LOGICAL SIDE,WRT1,WRT2,WRT3,COMP
      REAL SCALE,CI,E,TOTAR,BELTR
      INTEGER PAR1,PAR2,ELEM1,ELEM2,PTSW,Z,YN1,SL,SH,SW,CLINES,CELEM,CTA
     1PE
      COMP = .FALSE.
      IF (SW.EQ.1) COMP = .TRUE.
      IF (PTSW.EQ.0) GO TO 10
      IF (PTSW.EQ.1) GO TO 11
      IF (PTSW.EQ.2) GO TO 12
   10 WRT1 = .FALSE.
```

```
      WRT2 = .FALSE.
      WRT3 = .TRUE.
      GO TO 20
   11 WRT1 = .FALSE.
      WRT2 = .TRUE.
      WRT3 = .TRUE.
      GO TO 20
   12 WRT1 = .TRUE.
      WRT2 = .TRUE.
      WRT3 = .TRUE.
   20 CONTINUE
      IF (SL.GT.SH) WRITE (6,405)
      SIDE = .TRUE.
      REWIND NTAPE
      READ (NTAPE) NLINES,NELEM
      REWIND NTAPE
  300 IF (LINE1.LT.21) WRITE (6,401)
      N = NLINES - 20
      IF (LINE2.GT.N) WRITE (6,402)
      IF (ELEM1.LT.21) WRITE (6,403)
      N = NELEM - 20
      IF (ELEM2.GT.N) WRITE (6,404)
      TOTAR = 0.0
      NELTK = 0
      NV1 = LINE2 - LINE1 + 1
      NV2 = ELEM2 - ELEM1 + 1
      YN1 = ELEM1 - 20
      NV4 = NV2
      IF (.NOT.COMP) GO TO 420
      REWIND CTAPE
      READ (CTAPE) CLINES,CELEM
      REWIND CTAPE
      IF (CLINES.NE.NV1) WRITE (6,406)
      IF (CELEM.NE.NV2) WRITE (6,407)
  420 IF (NV2.GT.21) GO TO 450
      Z = NV2
      CALL WORK (YN1,LINE1,LINE2,Z,SIDE,WRT1,WRT2,PAR1,PAR2,NTAPE,BELTR,
     1CI,E,SCALE,ELEM1,CTAPE,COMP,SL,SH,NTAKEN)
      TOTAR = TOTAR + BELTR
      NELTK = NELTK + NTAKEN
      GO TO 600
  450 NV3 = NV2 - 21
      Z = 21
      CALL WORK (YN1,LINE1,LINE2,Z,SIDE,WRT1,WRT2,PAR1,PAR2,NTAPE,BELTR,
     1CI,E,SCALE,ELEM1,CTAPE,COMP,SL,SH,NTAKEN)
      TOTAR = TOTAR + BELTR
      NELTK = NELTK + NTAKEN
      NV2 = NV3
      YN1 = YN1 + 21
      IF (NV3.GT.0) GO TO 420
  600 CONTINUE
      NUMEL = NV1 * NV4
      IF (WRT3.AND.COMP) GO TO 700
      IF (WRT3) WRITE (6,411) NUMEL,TOTAR,SCALE,CI,E
      GO TO 705
  700 WRITE (6,412) NUMEL,NELTK,TOTAR,SCALE,CI,E
  705 CONTINUE
      REWIND NTAPE
```

```
  401 FORMAT (1H0,40H AREA DESIRED TOO CLOSE TO TOP BOUNDARY.//)
  402 FORMAT (1H0,43H AREA DESIRED TOO CLOSE TO BOTTOM BOUNDARY.//)
  403 FORMAT (1H0,41H AREA DESIRED TOO CLOSE TO LEFT BOUNDARY.//)
  404 FORMAT (1H0,42H AREA DESIRED TOO CLOSE TO RIGHT BOUNDARY.//)
  405 FORMAT (1H0,32H SWITCH LOW EXCEEDS SWITCH HIGH.//)
  406 FORMAT (1H0,50H CLINES DOES NOT EQUAL THE AREA/S NUMBER OF LINES./
     1/)
  407 FORMAT (1H0,67H CELEM DOES NOT EQUAL THE NUMBER OF ELEMENTS FOR A
     1ROW OF THE AREA.//)
  411 FORMAT (1H1,3X,28H ELEMENTS IN DESIRED REGION ,I10/3X,28H TOTAL AR
     1EA IN SQUARE UNITS ,E20.8/3X,9H SCALE = ,F20.9,20H CONTOUR INTERVA
     2L = ,F15.5,26H ELEMENT SIDE DIMENSION = ,F25.15//////)
  412 FORMAT (1H1,3X,18H ELEMENTS SEARCHED,I10/3X,18H ELEMENTS TAKEN   ,
     1I10/3X,28H TOTAL AREA IN SQUARE UNITS ,E20.8/3X, 9H SCALE = ,F20.9
     2,20H CONTOUR INTERVAL = ,F15.5,26H ELEMENT SIDE DIMENSION = ,F25.1
     35//////)
      RETURN
      END
$IBFTC PCSTP   LIST,REF,M94,XR7
      SUBROUTINE BANDPC (AA,PAR1,PAR2,PACE,PAR,CENT,FLAT,BETW,Z,TAKE)
C
C     THIS SUBROUTINE TESTS IN 4 ORTHOGONAL DIRECTIONS FROM EACH ELEMENT
C     IN THE CENTRAL STRIP OF ARRAY AA FOR A CONTOUR.  PACE(BDELM,AZ)
C     RECORDS THE NUMBER OF ELEMENTS THAT MUST BE SEARCHED TO REACH A
C     CONTOUR FOR EACH OF THE FOUR AZIMUTHS RADIATING FROM EACH ELEMENT
C     IN THE CENTRAL BAND.  CENT(BDELM) RECORDS THE PARITY FOR EACH
C     ELEMENT IN THE CENTRAL BAND.  PAR(BDELM,AZ) RECORDS THE PARITY OF
C     THE CONTOURS HIT ALONG EACH AZIMUTH.  IF NO CONTOUR IS HIT IN THE
C     20 ELEMENTS TESTED ALONG EACH AZIMUTH, PAR = 0, PACE = 30, AND
C     FLAT IS SET TO TRUE.  IF A CONTOUR IS HIT, FLAT IS SET TO FALSE.
C     BETW INDICATES WHETHER WE ARE ON OR ARE BETWEEN CONTOUR(S).  PAR1
C     AND PAR2 ARE THE PARITY FOR CONTOURS.
C
C     Z RANGES FROM 1 TO 21, IT IS THE NUMBER OF ELEMENTS OF THE STRIP
C     FOR WHICH CALCULATIONS ARE DESIRED.
C
C     MARK S. MONMONIER, 24 FEBRUARY, 1965
C
      INTEGER BDELM,AA(41,61),CENT(21),AZ,EXAM,PAR(21,4),PAR1,PAR2,Z
      REAL PACE(21,4)
      LOGICAL FLAT(21,4),BETW(21),TAKE(21)
      DO 500 BDELM = 1,Z
      IF(.NOT.TAKE(BDELM)) GO TO 500
      CENT(BDELM)=AA(21,BDELM + 20)
      DO 273 AZ=1,4
      DO 261 IGG=1,20
      GO TO (201,202,203,204), AZ
  201 INSPT=21-IGG
      EXAM=BDELM + 20
      GO TO 250
  202 INSPT=IGG + 21
      EXAM=BDELM + 20
      GO TO 250
  203 INSPT=21
      EXAM = BDELM + IGG + 20
      GO TO 250
  204 INSPT=21
      EXAM = BDELM - IGG + 20
```

```
      GO TO 250
  250 IF (AA(INSPT,EXAM) - PAR1) 260,270,260
  260 IF (AA(INSPT,EXAM) - PAR2) 261,272,261
  261 CONTINUE
      PACE(BDELM,AZ) = 30.
      FLAT(BDELM,AZ)=.TRUE.
      PAR(BDELM,AZ)=0
      GO TO 273
  270 PACE(BDELM,AZ)=IGG
      PAR(BDELM,AZ)=PAR1
      FLAT(BDELM,AZ)=.FALSE.
      GO TO 273
  272 PACE(BDELM,AZ)=IGG
      PAR(BDELM,AZ)=PAR2
      FLAT(BDELM,AZ)=.FALSE.
      GO TO 273
  273 CONTINUE
  280 IF (CENT(BDELM) - PAR1) 281,284,281
  281 IF (CENT(BDELM) - PAR2) 283,284,283
  283 BETW(BDELM)=.TRUE.
      GO TO 500
  284 BETW(BDELM)=.FALSE.
  500 CONTINUE
      RETURN
      END
$IBFTC SDSPC   LIST,REF,M94,XR7
      SUBROUTINE SDSPC (AA,PAR1,PAR2,PACE,PAR,CENT,FLAT,BETW,Z,TAKE)
C
C     THIS SUBROUTINE IS SIMILAR TO BANDPC. IT CALCULATES PACE IN A
C     SIDE-STEPPING MANNER, THAT IS IN DIRECTIONS NORTHWEST, SOUTHEAST,
C     NORTHEAST, AND SOUTHWEST INSTEAD OF NORTH, SOUTH, EAST, AND WEST
C     AS IN BANDPC. IN DETERMINING TAN1 AND TAN2 AND ANYOTHER VALUES
C     USING PACE AS A DIRECT MEASUREMENT, PACE VALUES SHOULD BE
C     MULTIPLIED BY THE SQUARE ROOT OF TWO. THIN SHOULD BE USED WITH
C     CARE WITH THIS SUBROUTINE.
C
C     Z RANGES FROM 1 TO 21, IT IS THE NUMBER OF ELEMENTS OF THE STRIP
C     FOR WHICH CALCULATIONS ARE DESIRED.
C
C     MARK S. MONMONIER, 24 FEBRUARY, 1965
C
      INTEGER BDELM,AA(41,61),CENT(21),AZ,EXAM,PAR(21,4),TRY1,TRY2
      INTEGER PAR1,PAR2,Z
      LOGICAL FLAT(21,4),BETW(21),HALF,TAKE(21)
      REAL PACE(21,4)
      DO 500 BDELM = 1,Z
      IF (.NOT.TAKE(BDELM)) GO TO 500
      CENT(BDELM)=AA(21,BDELM + 20)
      DO 273 AZ=1,4
      DO 269 IGG=1,20
      GO TO (201,202,203,204), AZ
  201 INSPT=21-IGG
      EXAM = BDELM + 20 - IGG
      TRY1 = 1
      TRY2 = 1
      GO TO 240
  202 INSPT=IGG + 21
      EXAM = BDELM + 20 + IGG
```

```
      TRY1 = -1
      TRY2 = -1
      GO TO 240
  203 INSPT = 21 + IGG
      EXAM = BDELM - IGG + 20
      TRY1 = 1
      TRY2 = -1
      GO TO 240
  204 INSPT = 21 - IGG
      EXAM = BDELM + IGG + 20
      TRY1 = -1
      TRY2 = 1
  240 HALF = .FALSE.
  250 IF (AA(INSPT,EXAM) - PAR1) 260,270,260
  260 IF (AA(INSPT,EXAM) - PAR2) 261,272,261
  261 JOY = INSPT + TRY2
      JAY = EXAM + TRY1
      IF (AA(JOY,EXAM) - PAR1) 264,262,264
  262 IF (AA(INSPT,JAY) - PAR1) 264,263,264
  263 HALF = .TRUE.
      GO TO 270
  264 IF (AA(JOY,EXAM) - PAR2) 269,265,269
  265 IF (AA(INSPT,JAY) - PAR2) 269,268,269
  268 HALF = .TRUE.
      GO TO 272
  269 CONTINUE
      PACE(BDELM,AZ) = 30.
      FLAT(BDELM,AZ)=.TRUE.
      PAR(BDELM,AZ)=0
      GO TO 273
  270 PACE(BDELM,AZ)=IGG
      PAR(BDELM,AZ)=PAR1
      FLAT(BDELM,AZ)=.FALSE.
      IF (HALF) PACE(BDELM,AZ) = PACE(BDELM,AZ) - 0.5
      GO TO 273
  272 PACE(BDELM,AZ)=IGG
      PAR(BDELM,AZ)=PAR2
      FLAT(BDELM,AZ)=.FALSE.
      IF (HALF) PACE(BDELM,AZ) = PACE(BDELM,AZ) - 0.5
  273 CONTINUE
  280 IF (CENT(BDELM) - PAR1) 281,284,281
  281 IF (CENT(BDELM) - PAR2) 283,284,283
  283 BETW(BDELM)=.TRUE.
      GO TO 500
  284 BETW(BDELM)=.FALSE.
  500 CONTINUE
      RETURN
      END
$IBFTC BELARA  LIST,REF,M94,XR7
      SUBROUTINE BELARA (TAN1,TAN2,Z,E,SCALE,AREA,STPAR,WRT1,TAKE)
C     MARK S. MONMONIER, 24 FEBRUARY, 1965
      REAL TAN1(21),TAN2(21),E,SCALE,FACTOR(21),AREA(21),STPAR,COR
      LOGICAL WRT1,TAKE(21)
      INTEGER X,Z
      STPAR = 0.0
      COR = E**2./SCALE**2.
      DO 700 X=1,Z
      IF (.NOT.TAKE(X)) GO TO 700
```

```
      FACTOR(X) = SQRT(TAN1(X)**2. + TAN2(X)**2. + 1.)
      AREA(X) = COR * FACTOR(X)
      STPAR = STPAR + AREA(X)
      IF (WRT1) WRITE (6,2121) AREA(X)
 2121 FORMAT (1H ,3X,5H AREA,F15.5)
  700 CONTINUE
      RETURN
      END
$IBFTC BELTAN  LIST,REF,M94,XR7
      SUBROUTINE BELTAN (PACE,PAR,CENT,FLAT,BETW,CI,E,SCALE,TAN1,TAN2,Z,
     1TAKE)
C
C     THIS SUBROUTINE CALCULATES THE DIRECTED TANGENTS
C     FOR UP TO 21 (X=1,21) OF THE ELEMENTS OF A BELT STRIP. THE RE/
C     QUIRED INFORMATION IS RECEIVED FROM THE CALLING ROUTINE.
C     MARK S. MONMONIER, 24 FEBRUARY, 1965
C
      INTEGER X,PAR(21,4),CENT(21),Z
      REAL CI,E,SCALE,TAN1(21),TAN2(21),TANT(4),PACE(21,4)
      LOGICAL FLAT(21,4),BETW(21),TAKE(21)
      DO 1000 X=1,Z
      IF (.NOT.TAKE(X)) GO TO 1000
      DO 900 I=1,3,2
      IF(.NOT.BETW(X))GO TO 400
C
C     THE A CASES
C
      IF (.NOT.FLAT(X,I+1)) GO TO 100
      IF (.NOT.FLAT(X,I)) GO TO 100
C
C     CASE A-4, COMPLETELY LEVEL SINCE NOTHING FOUND
C
      TANT(I)=0.0
      GO TO 900
  100 IF (FLAT(X,I)) GO TO 140
      IF (FLAT(X,I+1)) GO TO 140
      IF (PAR(X,I).EQ.PAR(X,I+1)) GO TO 160
C
C     CASE A-2, BOTH WITHIN, NOT LEVEL
C
      ALAL = PACE(X,I) + PACE(X,I+1)
      TANT(I) = ((CI * SCALE)/(ALAL * E))
      GO TO 900
C
C     CASE A-3,ONE OF TWO SENSES DOES NOT HIT CONTOUR
C     30 IS SUBSTITUTED FOR THE UNFOUND PACE VALUE (IN PACE ITSELF)
C
  140 ALAL = PACE(X,I) + PACE(X,I+1)
      TANT(I) = (CI * SCALE)/(ALAL * E)
      GO TO 900
C
C     CASE A-1,SAME CONTOUR HIT IN BOTH SENSES
C
  160 TANT(I) = 0.0
      GO TO 900
C
C     THE B CASES
C
```

```
  400 IF (.NOT.FLAT(X,I)) GO TO 450
      IF (.NOT.FLAT(X,I+1)) GO TO 450
C
C     CASE B-5, COMPLETELY LEVEL SINCE NOTHING FOUND
C
      TANT(I) = 0.0
      GO TO 900
  450 IF (FLAT(X,I)) GO TO 480
      IF (FLAT(X,I+1)) GO TO 490
      IF (PAR(X,I).EQ.PAR(X,I+1)) GO TO 550
C
      IF (PAR(X,I).EQ.CENT(X)) GO TO 501
      IF (PAR(X,I+1).EQ.CENT(X)) GO TO 502
      WRITE (6,222)
  222 FORMAT(1H1,15HERROR,CHECK 222)
C
C     CASE B-2,BOTH IN, ONE DIFFERS FROM CENTER
C
  501 BROP = PACE(X,I+1)
      GO TO 505
  502 BROP = PACE(X,I)
  505 OB = .5 * E
      TANX = (CI * SCALE)/(BROP * E)
      THETA=ATAN(TANX)
      COSX = COS(THETA)
      OA =(.5 * E)/COSX
      AOB = OA + OB
      COSY = AOB/E
      SINY = SQRT(ABS(1. - COSY**2.))
      TANT(I) = SINY/COSY
      GO TO 900
  480 IF (PAR(X,I+1).EQ.CENT(X)) GO TO 530
      GO TO 510
  490 IF (PAR(X,I).EQ.CENT(X)) GO TO 530
      GO TO 520
C
C     CASE B-4,ONE OUT ONLY,,OTHER NOT EQUAL TO CENT
C
  510 BROP = PACE(X,I+1)
      GO TO 521
  520 BROP = PACE(X,I)
  521 TANX = (CI * SCALE)/(BROP * E)
      THETA = ATAN(TANX)
      COSX = COS(THETA)
      OA = (.5 * E)/COSX
      AOB = OA + .5*OA + .5*E
      COSY = AOB/E
      SINY = SQRT(ABS(1. - COSY**2.))
      TANT(I) = SINY/COSY
      GO TO 900
C
C
C     CASE B-3,ONE IN,EQUALS CENTER
C
  530 TANT(I)=0.0
      GO TO 900
  550 IF (PAR(X,I).EQ.CENT(X)) GO TO 600
C
```

```
C      CASE B-1,BOTH IN,ABA ARRANGEMENT
C
       GEO = PACE(X,I) + PACE(X,I+1)
       TANT(I) = ((CI*2.*SCALE)/(GEO*E))
       GO TO 900
C
C      CASE B-6,BOTH IN, ALL LEVEL
C
  600  TANT(I)=0.0
       GO TO 900
  900  CONTINUE
       TAN1(X)=TANT(1)
       TAN2(X)=TANT(3)
 1000  CONTINUE
       RETURN
       END
$IBFTC BELTEL  LIST,REF,M94,XR7
       SUBROUTINE SPOT (INDR,INDC,NTAPE,BEGIN,FIRST,AA,Z)
C      THIS SUBROUTINE FETCHES THE BELT ELEMENT FOR THE DETERMINATION OF
C      THE AREA OF A STRIP.
C      THE DIMENSION FOR AA IN THE CALLING ROUTINE IS AA(41,61).
C      BEGIN MUST BE SET TO TRUE FOR THE TOP OF THE VERTICAL STRIP.
C      THIS SUBROUTINE IS DERIVED FROM SUBROUTINE GET3X3 (WRITTEN BY JOHN
C      PFALTZ).
C      MARK S. MONMONIER, 7 JANUARY 1965
C      INITIALLY SET FIRST TO TRUE.
       INTEGER AA(41,61),TEMP(2600),FSTROW,DIFF,FSTEL,Z
       LOGICAL BEGIN,BEEN,FIRST
       COMMON TEMP
       LOT = 40 + Z
       IF (.NOT.BEGIN)GO TO 700
       IF (.NOT.FIRST) GO TO 10
       BEGIN = .FALSE.
       FIRST = .FALSE.
       BEEN = .FALSE.
       REWIND NTAPE
       NXTROW = 1
       READ (NTAPE) NLINES,NELEM,IDATE,ID
C
C      COMPARE TAPE POSITION WITH FIRST DESIRED ROW
C
    10 FSTROW=INDR
   600 LSTROW=INDR + 40
       IF(LSTROW.GT.NLINES) GO TO 61
       IF(FSTROW.LT.1) GO TO 60
    11 DIFF=FSTROW-NXTROW
       IF(DIFF)12,18,16
    12 IF(DIFF + 20)13,14,14
    13 REWIND NTAPE
       NXTROW=1
       READ (NTAPE)
       GO TO 11
    14 DO 15 K=1,DIFF
       BACKSPACE NTAPE
    15 CONTINUE
       GO TO 18
    16 DO 17 K=1,DIFF
       READ (NTAPE)
```

```
C      CASE B-1,BOTH IN,ABA ARRANGEMENT
C
       GEO = PACE(X,I) + PACE(X,I+1)
       TANT(I) = ((CI*2.*SCALE)/(GEO*E))
       GO TO 900
C
C      CASE B-6,BOTH IN, ALL LEVEL
C
  600  TANT(I)=0.0
       GO TO 900
  900  CONTINUE
       TAN1(X)=TANT(1)
       TAN2(X)=TANT(3)
 1000  CONTINUE
       RETURN
       END
$IBFTC BELTEL  LIST,REF,M94,XR7
       SUBROUTINE SPOT (INDR,INDC,NTAPE,BEGIN,FIRST,AA,Z)
C      THIS SUBROUTINE FETCHES THE BELT ELEMENT FOR THE DETERMINATION OF
C      THE AREA OF A STRIP.
C      THE DIMENSION FOR AA IN THE CALLING ROUTINE IS AA(41,61).
C      BEGIN MUST BE SET TO TRUE FOR THE TOP OF THE VERTICAL STRIP.
C      THIS SUBROUTINE IS DERIVED FROM SUBROUTINE GET3X3 (WRITTEN BY JOHN
C      PFALTZ).
C      MARK S. MONMONIER, 7 JANUARY 1965
C      INITIALLY SET FIRST TO TRUE.
       INTEGER AA(41,61),TEMP(2600),FSTROW,DIFF,FSTEL,Z
       LOGICAL BEGIN,BEEN,FIRST
       COMMON TEMP
       LOT = 40 + Z
       IF (.NOT.BEGIN)GO TO 700
       IF (.NOT.FIRST) GO TO 10
       BEGIN = .FALSE.
       FIRST = .FALSE.
       BEEN = .FALSE.
       REWIND NTAPE
       NXTROW = 1
       READ (NTAPE) NLINES,NELEM,IDATE,ID
C
C      COMPARE TAPE POSITION WITH FIRST DESIRED ROW
C
    10 FSTROW=INDR
   600 LSTROW=INDR + 40
       IF(LSTROW.GT.NLINES) GO TO 61
       IF(FSTROW.LT.1) GO TO 60
    11 DIFF=FSTROW-NXTROW
       IF(DIFF)12,18,16
    12 IF(DIFF + 20)13,14,14
    13 REWIND NTAPE
       NXTROW=1
       READ (NTAPE)
       GO TO 11
    14 DO 15 K=1,DIFF
       BACKSPACE NTAPE
    15 CONTINUE
       GO TO 18
    16 DO 17 K=1,DIFF
       READ (NTAPE)
```

```
   17 CONTINUE
C
C     READY TO READ FIRST ROW
   18 IF (BEEN) GO TO 610
      FSTEL = INDC
      LSTEL = INDC + LOT - 1
      IF (FSTEL.LT.1) GO TO 62
      IF (LSTEL.GT.NELEM) GO TO 63
      DO 20 K=1,41
      READ (NTAPE) (TEMP(N),N=1,NELEM)
      DO 19 L=1,LOT
      INDEX=FSTEL + L -1
      AA(K,L)=TEMP(INDEX)
   19 CONTINUE
   20 CONTINUE
      NXTROW=LSTROW + 1
      RETURN
C
C     THE ARRAY IS UPDATED BY MOVING ALL ROWS UP ONCE AND REPLACING THE
C     LAST ROW IN THE ARRAY BY THE NEXT ROW OF THE MAP FROM THE TAPE.
C
  700 DO 701 I=1,40
      DO 701 J=1,LOT
      AA(I,J)=AA(I+1,J)
  701 CONTINUE
      FSTROW=INDR + 40
      BEEN=.TRUE.
      GO TO 600
  610 FSTEL = INDC
      LSTEL = INDC + LOT - 1
      IF (FSTEL.LT.1) GO TO 62
      IF (FSTEL.GT.NELEM) GO TO 63
      READ (NTAPE) (TEMP(N),N=1,NELEM)
      DO 29 L=1,LOT
      INDEX=FSTEL + L -1
      AA(41,L)=TEMP(INDEX)
   29 CONTINUE
      NXTROW=LSTROW + 1
      RETURN
C
C     ERROR ROUTINES
   50 FORMAT (1H ,71HDESIRED REGION EXTENDS BEYOND PICTURE LIMITS, REF.(
     1SUBROUTINE SPOT).    )
   51 FORMAT (1H0,7HFSTROW=,I4)
   52 FORMAT (1H0,7HLSTROW=,I4,4X,7HNLINES=,I4)
   53 FORMAT (1H0,6HFSTEL=,I4)
   54 FORMAT (1H0,6HLSTEL=,I4,4X,6HNELEM=,I4)
   60 WRITE (6,51) FSTROW
      GO TO 64
   61 WRITE (6,52) LSTROW,NLINES
      GO TO 64
   62 WRITE (6,53) FSTEL
      GO TO 64
   63 WRITE (6,54) LSTEL,NELEM
   64 WRITE (6,50)
      RETURN
      END
$IBFTC WORK     LIST,REF,M94,XR7
```

```
      SUBROUTINE WORK (YN1,LINE1,LINE2,Z,SIDE,WRT1,WRT2,PAR1,PAR2,NTAPE,
     1BELTR,CI,E,SCALE,ELEM1,CTAPE,COMP,SL,SH,NTAKEN)
      INTEGER YN1,X1,X2,Z,AA(41,61),PAR1,PAR2,CENT(21),PAR(21,4),XX,X3,X
     14,S,A,COL,RO,BB(21),TAKEN,CTAPE,ELEM1,SL,SH
      LOGICAL SIDE,FIRST,BEGIN,FLAT(21,4),BETW(21),WRT1,WRT2,TAKE(21),CO
     1MP,CBEGIN,CFIRST
      REAL TAN1(21),TAN2(21),AREA(21),PACE(21,4)
      FIRST = .TRUE.
      CFIRST = .TRUE.
      X3 = LINE1 - 20
      X4 = LINE2 - 20
      NTAKEN = 0
      A = 21 - ELEM1
      S = 1 - X3
      COL = YN1 + A
      BEGIN = .TRUE.
      CBEGIN = .TRUE.
      NUMEL = (X4 - X3 +1) * Z
      IF (.NOT.COMP) NTAKEN = NUMEL
      IF (COMP) GO TO 80
      DO 75 I=1,Z
      TAKE(I) = .TRUE.
   75 CONTINUE
   80 CONTINUE
      BELTR = 0.0
      DO 200 XX=X3,X4
      CALL SPOT (XX,YN1,NTAPE,BEGIN,FIRST,AA,Z)
      RO = XX + S
      IF (COMP) CALL CHELM (RO,COL,BB,CTAPE,CBEGIN,CFIRST,Z)
      IF (COMP) CALL CKDECD (Z,BB,SL,SH,TAKE,TAKEN)
      IF (COMP) NTAKEN = NTAKEN + TAKEN
      IF (.NOT.SIDE) GO TO 100
      CALL SDSPC (AA,PAR1,PAR2,PACE,PAR,CENT,FLAT,BETW,Z,TAKE)
      GO TO 105
  100 CALL BANDPC (AA,PAR1,PAR2,PACE,PAR,CENT,FLAT,BETW,Z,TAKE)
  105 CALL BELTAN (PACE,PAR,CENT,FLAT,BETW,CI,E,SCALE,TAN1,TAN2,Z,TAKE)
      IF (SIDE) CALL SDBLTN (Z,TAN1,TAN2,TAKE)
      CALL BELARA (TAN1,TAN2,Z,E,SCALE,AREA,STPAR,WRT1,TAKE)
      BELTR = BELTR + STPAR
  200 CONTINUE
      IF (WRT2) WRITE (6,70) BELTR,YN1,LINE1,LINE2,Z,NTAKEN,NUMEL
   70 FORMAT (1H0,3X,10HTHE AREA  ,F25.10,34H SQUARE UNITS IS FOR AA OF
     1COLUMN ,I6,2H ./3X,22HTHE BELT RAN FROM ROW ,I6,8H TO ROW ,I6, 9H
     2AND WAS ,I2,23H PICTURE ELEMENTS WIDE./3X,11HTHERE WERE ,I7,23H EL
     3EMENTS TAKEN OUT OF ,I7,32H POSSIBLE ELEMENTS IN THIS BELT.//)
      RETURN
      END
$IBFTC SDBLTN  LIST,REF,M94,XR7
      SUBROUTINE SDBLTN (Z,TAN1,TAN2,TAKE)
C
C     FOR USE WITH SUBROUTINE SDSPC THIS SUBROUTINE TAKES Z TAN1)S AND
C     TAN2)S FROM SUBROUTINE BELTAN AND CORRECTS THEM FOR THE PACE
C     VALUES THAT WERE TAKEN IN DIRECTIONS NW, SE, NE,SW TO PERMIT AN
C     ACCURATE DETERMINATION OF SURFACE AREA.
C     MARK S. MONMONIER, 4 FEBRUARY,1965
C
      INTEGER Z,X
      REAL TAN1(21),TAN2(21)
```

```
      LOGICAL TAKE(21)
      AFAC = 1./SQRT(2.)
      DO 100 X=1,Z
      IF (.NOT.TAKE(X)) GO TO 100
      TAN1(X) = TAN1(X) * AFAC
      TAN2(X) = TAN2(X) * AFAC
  100 CONTINUE
      RETURN
      END
$IBFTC CKDECD  LIST,REF,M94,XR7
      SUBROUTINE CKDECD (Z,BB,SL,SH,TAKE,NTAKEN)
      LOGICAL TAKE(21)
      INTEGER Z,BB(21),SL,SH
      NTAKEN = 0
      DO 70 I=1,Z
      IF (BB(I).LT.SL) GO TO 50
      IF (BB(I).GT.SH) GO TO 50
      TAKE(I) = .TRUE.
      NTAKEN = NTAKEN + 1
      GO TO 70
   50 TAKE(I) = .FALSE.
   70 CONTINUE
      RETURN
      END
$IBFTC CHELM   LIST,REF,M94,XR7
      SUBROUTINE CHELM (ROW,COL,BB,CTAPE,BEGIN,FIRST,Z)
      INTEGER FSTEL,TEMP(1200)
      INTEGER ROW,COL,BB(21),Z,CTAPE,FSTROW,DIFF
      COMMON TEMP
      LOGICAL BEGIN,BEEN,FIRST
      IF (.NOT.BEGIN)GO TO 700
      IF (.NOT.FIRST) GO TO 10
      BEGIN = .FALSE.
      FIRST = .FALSE.
      BEEN = .FALSE.
      REWIND CTAPE
      NXTROW = 1
      READ (CTAPE) NLINES,NELEM,IDATE,ID
   10 FSTROW = ROW
  600 LSTROW = ROW
      IF(LSTROW.GT.NLINES) GO TO 61
      IF(FSTROW.LT.1) GO TO 60
   11 DIFF=FSTROW-NXTROW
      IF(DIFF)12,18,16
   12 IF(DIFF + 20)13,14,14
   13 REWIND CTAPE
      NXTROW=1
      READ (CTAPE)
      GO TO 11
   14 DO 15 K=1,DIFF
      BACKSPACE CTAPE
   15 CONTINUE
      GO TO 18
   16 DO 17 K=1,DIFF
      READ (CTAPE)
   17 CONTINUE
   18 IF (BEEN) GO TO 610
      FSTEL = COL
```

```
      LSTEL = COL + Z - 1
      IF (FSTEL.LT.1) GO TO 62
      IF (LSTEL.GT.NELEM) GO TO 63
      READ (CTAPE) (TEMP(N),N=1,NELEM)
      DO 19 L=1,Z
      INDEX=FSTEL + L -1
      BB(4) = TEMP(INDEX)
   19 CONTINUE
   20 CONTINUE
      NXTROW=LSTROW + 1
      RETURN
  700 FSTROW = ROW
      BEEN=.TRUE.
      GO TO 600
  610 FSTEL = COL
      LSTEL = COL + Z - 1
      IF (FSTEL.LT.1) GO TO 62
      IF (FSTEL.GT.NELEM) GO TO 63
      READ (CTAPE) (TEMP(N),N=1,NELEM)
      DO 29 L=1,Z
      INDEX=FSTEL + L -1
      BB(L)= TEMP(INDEX)
   29 CONTINUE
      NXTROW=LSTROW + 1
      RETURN
C
C     ERROR ROUTINES
   50 FORMAT (1H ,71HDESIRED REGION EXTENDS BEYOND PICTURE LIMITS, REF.(
     1SUBROUTINE CHELM).  )
   51 FORMAT (1H0,7HFSTROW=,I4)
   52 FORMAT (1H0,7HLSTROW=,I4,4X,7HNLINES=,I4)
   53 FORMAT (1H0,6HFSTEL=,I4)
   54 FORMAT (1H0,6HLSTEL=,I4,4X,6HNELEM=,I4)
   60 WRITE (6,51) FSTROW
      GO TO 64
   61 WRITE (6,52) LSTROW,NLINES
      GO TO 64
   62 WRITE (6,53) FSTEL
      GO TO 64
   63 WRITE (6,54) LSTEL,NELEM
   64 WRITE (6,50)
      RETURN
      END
```