Information Report FF-X-44

November, 1973

A COMPUTER SOFTWARE PACKAGE TO CONVERT DIGITIZED GEOGRAPHIC BOUNDARY AND CONTOUR DATA TO A UNIFORM GRID STRUCTURE

by

Michael Travis, Gary Elsner,
and Peter Kourtz

F O R E S T F I R E R E S E A R C H I N S T I T U T E

CANADIAN FORESTRY SERVICE

DEPARTMENT OF THE ENVIRONMENT
Nicol Building
331 Cooper Street
Ottawa, Ontario
K1A 0H3

A COMPUTER SOFTWARE PACKAGE TO CONVERT DIGITIZED GEOGRAPHIC BOUNDARY AND CONTOUR DATA TO A UNIFORM GRID STRUCTURE

INTRODUCTION

The Forest Fire Research Institute and the United States Forest Service jointly have developed an operational forest fire spread model that predicts the perimeter location of a specific fire anytime after ignition. Part of this model's input requirements are definitions of fuel type, slope, and aspect for each acre of a uniform grid (a point every 208.7 feet) in the vicinity of the fire's perimeter. Unfortunately, such data in Canada is only available from either maps or photographs and is in a continuous bounded area and contour form. It must be converted to a digital form for use in the fire model.

The method used to make this conversion involves the use of an electronic digitizer. This machine converts tracings of continuous boundaries or contours such as those of a fuel type or topographic map to a digital form by recording an x-y coordinate location at specified distances or time intervals along each line. Computer algorithms then can be used to correctly label each point (or acre) in the grided area under consideration. This paper describes the digitizing procedure, algorithms, and the computer routines that accomplish this task. The algorithms and main computer routines on which this package is based were devised by the Forest Recreation Research Project, U.S. Forest Service, Berkeley, California.

This software package may be useful in other disciplines where geographic data must be translated to digital form. More specifically, users of data systems that involve multiple overlays of different geographic information may find this an efficient procedure.

The description of this package is divided into two sections namely, the fuel type boundary section and the terrain contouring section. Each section is divided further into four subsections describing: (a) the procedure for setting up and use of the digitizer; (b) the method used by the computer for processing the digitized data; (c) the map scale and digitizer increment; and (d) the subroutine structure. The package is written in Fortran and is currently running on a Univac 1108. Card decks of the programs are available from the Forest Fire Research Institute.

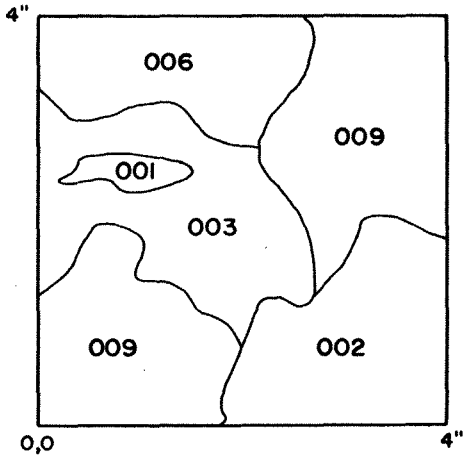
FUEL TYPE SECTION

I. Digitizing Procedure

A. Set the appropriate digitizer output control to produce 5 unsigned digits for each x-y coordinate.

B. Do not use a scaling factor.

C. The program currently assumes an area 4" x 4" will be digitized.



(1 Sq. Mile at a Scale of 4 inches = 1 Mile)

D. Put (0,0) at the lower left corner.

E. Put digitizer in line mode.

F. Trace all fuel type boundary lines:

(1) (a) Enter $$$$$ from keyboard (4 or more accepted by the program). This indicates that next point begins a new line.

(b) Place the cursor on start of a line.

(c) Press down the foot pedal to start recording coordinate data.

(d) Trace along the line. Straight lines can be traced relatively quickly. Slow down in curved sections to get more points recorded.

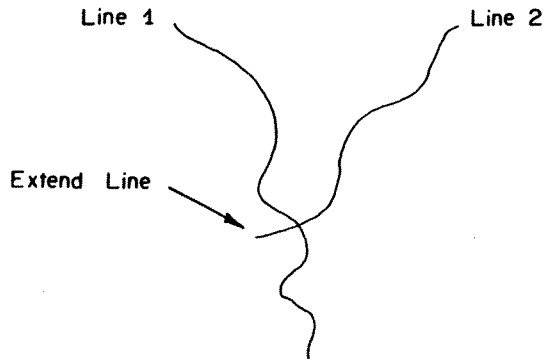
(e) Lift foot from pedal at end of line.

(f) Repeat from (a) for next line.

(2) (a) All lines need to be traced only once in any direction.

(b) There is no need to do the outer boundary -- this is implied in the program.

(c) Try to ensure that there will be no gaps in boundaries. Extend boundaries slightly past crossing points:



(3) Label each region with its fuel type:

(a) Digitizer to point mode.

(b) Place curser on interior point of a region.

(c) Enter numeric code for the fuel type by keying in **NNN where NNN is a 3-digit fuel type. Examples: **001, **013 etc.

(d) Two or more *'s may be entered but there must always be 3 digits in fuel type.

(e) Enter the point co-ordinates by pressing the foot pedal once.

(f) If the fuel type entered in (c) was wrong, re-enter fuel type before step (d). Repeat steps (c) and (d) with cursor in same spot if possible.

(g) Traced boundaries, paragraph F(1) above, and interior-point labels, paragraph F(3), can be intermixed in any order.

(4) The program currently accepts fuel types 001 to 015 inclusive. Fuel type 000 is not allowed.

II. Processing of the Digitized Data

A. The digitized fuel type data can be checked for correct format by running program DIGDAT. This prints the data with error messages as required.

B. The actual fuel type determination for each acre is done by executing the main program FTYPE. This performs the following steps:

(1) Initializes arrays.

(2) Calls INSERT to read the digitizer data cards and insert boundary points and fuel type label points into the subgrid (each point in the subgrid represents 1/25th of an acre).

(a) INSERT uses GPOINT to obtain the next data point, and SPAN to do actual insertion.

(b) GPOINT calls GFIELD and both call GCH (see section IV for further details).

(3) Calls SGRID to display the 125 x 125 subgrid.

(4) Calls PATCH to read correction cards. These should be separated from the digitizer cards by an @ EOF card. They contain free-format correction equations in the form (NNNNN,NNNNN) = NNNNN. Where NNNNN is a 1 to 5 digit unsigned integer. In the equation (I,J) = K,I = subgrid X coordinate, J = subgrid Y coordinate and K can be:

(a) 1 to 5 digits to insert a fuel type label.

(b) Letter 'B' to blank out a label or boundary point.

(c) Letter 'X' to install a boundary point.

Multiple patch equations are separated by commas: (1,13) = 1, (23,11) = B, (23,12) = X. PATCH prints a list of corrections, if any.

(5) If corrections were made, calls SGRID again to display corrected subgrid.

(6) Calls INSIDE to fill all labeled areas with their label INSIDE can terminate in 3 ways:

(a) Completion of filling marked areas.

(b) Reaching the pass limit set by FTYPE (currently 12, which seems adequate).

(c) Detecting 'leakage', which means two adjacent cells of different fuel types (not separated by a fuel type boundary).

In cases (b) or (c), FTYPE prints a warning message, calls SGRID to display the subgrid for visual inspection, then stops. Perhaps corrections (step 4) will correct the problem.

(7) Calls SGRID to display the filled subgrid, then:

(8) Calls NOTE to assign to each acre cell in the 25 x 25 grid the predominate fuel type in its 25 corresponding subgrid cells.

(9) Calls AGRID to display the 25 x 25 acre grid.

III. Scale and Digitizer Increment

It is assumed that we are dealing with a 625 acre area, 25 acres on a side. Different map scales, and hence different sized squares on the map corresponding to this 625 acres area, can be accommodated by changing SCALE in subroutine INSERT. INSERT also assumes a digitizer increment of .001 inch. This can be altered by changing DIGINC in INSERT. INSERT passes point corrdinate to SPAN in feet from the origin. SPAN converts these to subcell coordinates, so SPAN needs to know subcell size. This is currently 41.742 feet square (1/25 acre).

IV. Subroutine Structure

FTYPE main program calls:

- PMARG to suppress page skip
- INSERT to insert label and boundary points
  - GPOINT to get next point
    - GFIELD to get a field
      - GCH to get next character
      - ICON to convert a format to an integer
  - SPAN to install cell labels
- SGRID to display labelled subgrid
- PATCH to insert corrections to the subgrid
  - GPAT to read next patch equation
    - GCH for next character

ICON to convert alpha to numeric

RDTO to read to a stop character

GCH to get next character

INSIDE to fill marked areas

VOTE to assign acre-cell fuel types

LETTER to print large title

AGRID to print 25 x 25 acre-cell fuel type map

DIGDAT calls:

GFIELD

GCH

ICON

CONTOURING SECTION

I. Digitizing Procedure

A. Repeat steps (A) through (D) of the fuel typing section.

B. Enter each contour as follows:

(a) Enter an elevation on keyboard by ++NNNNN; two or more +'s followed by a 5-digit elevation field. Example: ++01200 is 1200 feet elevation.

(b) Entering an elevation as in (a) implies that a new line follows. Once an elevation is entered it holds for all lines traced until another elevation is entered. Several lines can be traced by separating them by a keyed-in $$$$$ field.

(c) If an elevation is keyed-in incorrectly, immediately enter another ++NNNNN field with the correct elevation. If a line was traced at an incorrect elevation, go back over it one or more times, following the original path as closely as possible. Or, if possible, correct the original data to the right elevation.

(d) Trace each contour by setting the digitizer in line mode, placing cursor at start of line, depressing foot pedal and moving cursor to end of line. Remember to enter $$$$$ before tracing another line at same elevation, and enter ++NNNNN before tracing a contour at different elevation.

II. Processing the Digitized Data

A. Program ELDAT prints out the data with error messages. This is just to check the data.

B. Main program ZEDVAL processes the data to produce elevation, slope and aspect of each acre cell. The steps of ZEDVAL are:

(1) Initialize arrays.

(2) Calls ZNSERT to read digitizer data cards and insert elevation labels into subcells crossed by contour lines (each subcell represents 1/25th of an acre).

(3) Optionally calls LOOK to print the full subcell map to show the inserted contour lines (set variable YES to .TRUE. to print this out).

(4) Calls NCELLS to determine the percent of cells occupied by contour points, and prints this figure.

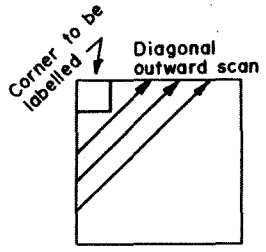
(5) Calls INTERP to perform a planar interpretation to assign elevations to all nonlabelled cells in the subgrid. INTERP proceeds as follows:

(a) Calls KORNER to assign elevations to any map corners not already labelled.

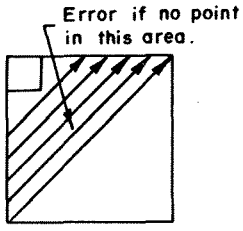
(b) Calls LININT to linearly interpret elevations for all edge cells not labelled.

(c) Planarly interprets elevations for all interior cells not labelled with an elevation. INTERP has two internal error halts:

(i) KORNER tries to label corners by choosing the first elevation found on a diagonal outward scan:

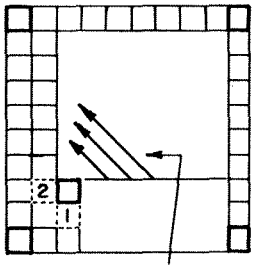


This scan proceeds only to the closest corner outward, so this area must contain at least 1 elevation point.



If not, KORNER prints an error message, calls LOOK to print the map, then stops.

(ii) INTERP scans diagonally outward for its 3rd known point:



Diagonal scans looking for 3 rd. point.

If this cannot be done, INTERP prints an error message, calls LOOK, then stops.

(6) ZEDVAL next optionally (depending on variable YES) prints the complete interpreted subcell map.

(7) Next AMEAN is called to assign to each 1-acre cell the average elevation of its 25 subcells.

(8) LOOK is called to print out the acre-cell elevation map in coded form. Then IAPRNT is called to print the same data in integer format.

(9) SURFN is called to place the average slope of each acre's subcells into array lACRE.

(10) IAPRNT is called to print out slope as a per cent.

(11) SURFN is called to place into each 1-acre cell the average aspect of its subcells {slope and aspect calculations are based on the procedure outlined by Sharpnack and Akin (1969)}.

(12) IAPRNT is called to print out the aspect as an integer, degrees from North, 0^{o} to 359^{o} (a vector averaging procedure is used to calculate average slope and aspect).

III. Map Scale and Digitizer Increment

A. Map scale is known only in ZNSERT. This is currently assumed to be 4" = 1 mile. ZNSERT also knows the digitizer increment. This is currently assumed to be 0.001 inches.

B. Subcell size is needed in two routines:

(1) SPAN, which inserts points (note - SPAN is used by fuel type programs also).

(2) NORMV, which calculates the components of the unit normal vector to each subcell. Subcell size is currently 41.742' on each side (1/25th of an acre).

IV. Subroutine Structure

ZEDVAL main program for elevation, slope, aspect, calls.

- PMARG to suppress skip
- ZNSERT to insert contours: calls
  - ZPOINT for each contour point: calls
    - ZFIELD for data card fields: calls
      - GCH for next character
      - ICON to convert to integer
  - SPAN to insert contour points
- LOOK to print contour map points: calls
  - HED (and TAIL) to print column labels
    - IDIG to convert number to BCD format
- NCELLS to count marked cells
- INTERP to interpolate
  - KORNER to assign elevations to map corners
    - LOOK (if error occurs)
  - LININT to interpolate elevations on map edge
    - LOOK (if error occurs)
- LOOK (if requested) to print interpolated map
- AMEAN to assign average elevation to 1-acre cells
- LOOK to print acre cell elevations
- IAPRNT to print acre cell elevations
- SURFN to calculate average slope of each 1-acre cell
  - NORMV on each subcell to calculate a unit normal vector

```
          CROSSV to find vector cross-product
          UNITV to unitize the vector
     UNITV to unitize the sum normal vector
     ISLOPE to find a slope per cent
IAPRNT to print slope values for 7-acre cells
SURFN to calculate average aspect for each 1-acre cell
     NORMV
          CROSSV
          UNITV
     UNITV
     IASPCT to find an aspect angle
IAPRNT to print aspect values.
```

REFERENCE

Sharpnack, David A. and Sarth Akin, 1969. An Algorithm for Computing Slope and Aspect from Elevation, Photogrammetric Engineering, March 1969, pgs. 247-248.