AN ADAPTIVE GRID CONTOURING ALGORITHM

James A. Downing, II
Steven Zoraster
ZYCOR, Inc.
2101 South IH-35
Austin, TX 78744

ABSTRACT

A new algorithm for contouring Digital Terrain Models is presented. This algorithm adapts itself to the terrain so that the number of points required to represent a contour is a function of the roughness of the local terrain and the degree of smoothness requested by the user.

This algorithm avoids two of the most noticeable difficulties encountered in using conventional contouring algorithms, the generating of more points than are required by the terrain, thus using extra computer resources, and the generating of too few, thus creating angular and visually unpleasing contours.

INTRODUCTION

Grid-to-contour (GTOC) is the name of a new computer program developed for Defense Mapping Agency (DMA) which interpolated elevation contours from digital terrain models (DTM). The algorithms used by this program differ from conventional contouring algorithmns in that it adaptively determines the number of digitized contour points required to adequately represent contours as smooth curves when the points are connected by line segments. It avoids two of the most noticeable difficulties encountered in using conventional algorithms; the generation of more points than are required by the terrain, therefore using extra computer resources, and the generation of too few points creating angular and visually unpleasing contours.

DMA has other contour generation programs. They are confined to the mainframe computing systems and are used primarily for data verification and evaluation. The technology and design of GTOC will enable DMA to use the program on many of the mini-computer based systems currently online or planned for DMA. Also, the quality of output suggests the program will be useful in future map production plans for DMA.

A description of the logic and major algorithms used in GTOC are provided in the following sections. Several test outputs based on DTMS are provided as examples of the program outputs.

BACKGROUND

The general contouring problem is to establish all points (x,y) which satisfy the equation

$$E(x,y) - C = 0 \qquad (1)$$

where E represents the elevation as it varies with x and y and C is the contour value.

Cartographers "solve" this equation using a variety of tools, logic, and experience. Their tools range from simple geometric dividers to complex analog photogrammetry systems.

Various digital systems used by the government require elevation data in a form that is easy to handle while maintaining high accuracy. This requirement is satisfied by digital terrain models which are matrices of elevation values.

Although cartographers could apply their classical contouring methods to the tabulated elevations in a DTM, this is unrealistic because of the volume of data and the rate at which DTM's can be produced. Hence computer contouring is a reasonable alternative.

In its simpliest form, computer contouring is a process of connecting dots. If a contour level lies between two DTM values, then the crossing point is on the grid line connecting the values. Other dots can be positioned on grid lines where this contour crosses. Finally, the contour curve can be drawn by connecting the dots with short line segments or curves.

The accuracy and acceptability of contours produced by this simple process depend on several factors. These are

1. accuracy of the DTM values,
2. spacing between DTM values,
3. texture of the area covered by the DTM,
4. procedure for computing dot locations, and
5. procedures for connecting the dots.

Only Factors 4 and 5 are within the control of the contouring logic described above. It assumes that the DTM data are accurate and properly spaced for the topography it represents. Indeed, Factors 4 and 5 are the reasons why computer contouring is not as simple as suggested in the discussion above. There are a variety of techniques for each. Several were investigated and the most effective combination implemented for DMA. These are discussed in the next sections.

CONTOUR PROCESSING

Contour processing starts with a contour level C, initially the first level of interest. Then, a systematic search of the cells is started to locate a cell containing the curve. The search commences at the upper left corner of the matrix and scans down and across. When a curve at level C is detected, points on the curve through the cell are computed as described below. Since the initial cell shares an edge with a cell which also contains the curve, it is possible to move left, right, up, or down in a predictable manner from one cell to the next along the path of the curve. Each time a new cell's elevation values are retrieved from the matrix, points along the curve are computed and added to a string representing the contour.

Tracing the curve from cell-to-cell stops when the last cell processed is on the edge of the matrix (contour exits on a side of the map) or when the next new cell is the same as the initial cell (the curve is closed within the map).

After the curve-tracing stops, the next curve (if any) at level C is sought. The searching starts in the cell just beyond the one where the last curve was initially detected. Searching continues until another curve is located which means that the curve-following logic is evoked again or until the last grid cell in the matrix is analyzed. The entire process repeats for the next contour value or terminates if there are no more contours.

To prevent the detection of a single curve more than once, a logic matrix is constructed. The matrix contains two "YES-NO" flags per grid cell representing the top and left sides of the cell area. Initially, for each contour level, all flags are set to "NO". As a contour is traced through a cell and found to cross the top side, the top flag is changed to "YES". Similarly the left side flag is changed to "YES" if it crosses the left side of the cell. Therefore, all cell areas where the curve intersects the top or left edges will be marked as having been processed. Later when such a cell is checked by the curve detection algorithm, it will be ignored so that the curve will not be retraced a second, third, etc. times. A new curve will be started only if the top or left side flags are "NO". Even if a contour is traced through both the left side and top of a cell, this logic does not prevent a second contour of the same level from crossing the cell since it can be picked up in some other cell and then eventually traced through this one.

As the contour strings are generated, they are transferred directly to disc storage for later processing (display, editing, etc.). The strings are automatically ordered by virtue of the tracing process.

INTERPOLATING CONTOUR POINTS

When the area of a grid cell is very small at map scale, contours can probably be drawn satisfactorily by connecting the points on the cell sides with line segments. However, when the cells are fairly large, it will be necessary to interpolate intermediate points through the cell to produce smooth and realistic contours. The techniques used to locate points along a contour within a cell are described here.

To accommodate both requirements, i.e., defining curves by edge intersections (2 points) alone or by multiple points across each cell, two levels of complexity are required. Ideally the simpliest requirement would be satisfied by Linear Fitting in which the contour intersections with the cell edges are joined by a single straight line. However, since some cells have more than two crossings of a contour level with cell edges, a further refinement is required to prevent ambiguities. Thus for this case a pattern of four triangles is created, each triangle defined by a cell side and the center of the cell. The interpolation is in fact linear over these triangles.

Non-Linear Fitting yields two points where the curve intersects the cell edges plus one or more intermediate points across the cell. When these are closely spaced and connected by straight line segments, the results appear to be a smooth curve through the cell.

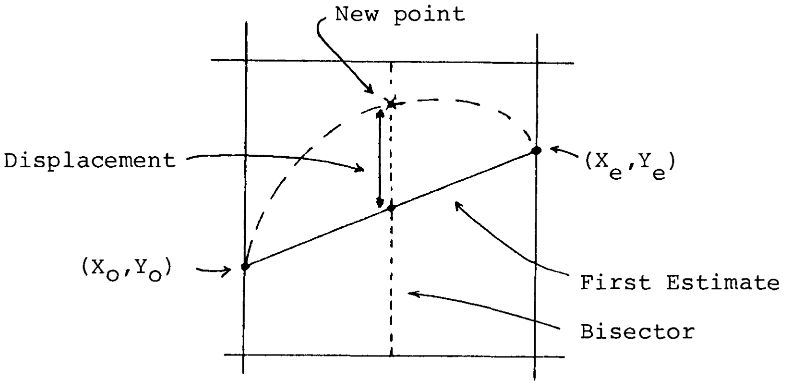
Non-Linear Fitting is much more complex and time-consuming since a local elevation model must be constructed across the grid cell. This model is formulated from 16 elevation values which are at the corners of a 4x4 sub-matrix containing the grid cell at its center.

Two algorithms, called the Adaptive and Stepping Methods, were developed to trace contours across the local elevation models. The Adaptive Method is used whenever possible while the Stepping Method is used only when the Adaptive Method fails.

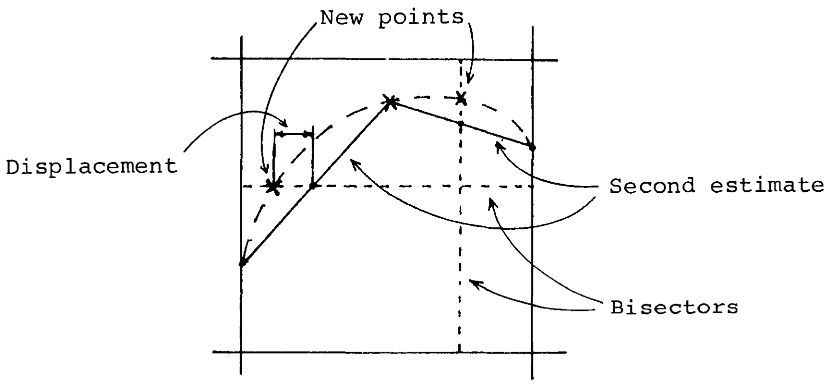
The Adaptive Method starts with the two curve intersections with the cell sides. These are connected by a straight line segment as shown in Figure 1, step 1 where the two curve intersections have coordinates (x_e,y_e) and (x_o,y_o). To decide which additional points are required, the line is bisected to establish a reference point (x,y). Then a vertical or horizontal line is constructed through (x,y) with the line drawn so as to cut the coordinate corresponding to the largest of the two differences $|x_e-x_o|$ and $|y_e-y_o|$. The intersection of the line with the contour is computed by means of interpolation from the local elevation model. The addition of the new point to the original two means the contour can now be drawn with two connected line segments. If the distance from the reference point to the new point is large relative to the cell's width, then the two new line segments can be further bisected to develop 5 points along the curve. This last process is shown in Figure 1, step 2. New line segments are added until the distance from a reference to its new point is "small". Typically, 3 to 5 points are generated; however, up to 50 could be generated if necessary.

The user controls the number of points generated in each cell by defining the fraction of a cell size used to decide whether the distance between the new point the reference point is "small". Very roughly this tolerance parameter corresponds to a potential refinement of the input grid to an overlayed subgrid with dimensions corresponding to the value of the parameter. In other words if the tolerance is choosen to be 0.25 then the resulting contours should be similar in appearance to those obtained by contouring a DTM with 4 times as many rows and columns covering the same area. Of course, the difference is that the intermediate points would be created with this new algorithm only if required by the terrain.

The Adaptive Method does not work for all cells. One problem is created by saddles in which the contour crosses all 4 sides of a cell, making the correct exit difficult to determine. In this and other difficult cases, the Stepping Method is used.



Step 1. The first estimate is bisected. The curve intersects the bisector at the new point. The displacement from the first estimate to the curve is shown.



Step 2. The second estimate line segments are bisected and curve intersections computed to obtain two new points. The curve may now be drawn with four line segments.

Figure 1. The Adaptive Method

The Stepping Method traces the contour by taking very small steps along the curve. It starts at an edge point on the curve and then computes elevation values using the local model at the corners of sub-cells. The curve's intersection with the sub-cell grid lines is computed by inverse linear interpolation. Since a curve enters and exits each sub-cell along its path, it is necessary to compute local elevation values for only those sub-cells. When the sub-cells are very small, the algorithm can follow the most complex curve shapes across the cell. Sub-cells may be as small as 1/256 the area of a DTM cell. From 15 to 30 points per cell are typically generated by this method.

The Stepping Method is more time-consuming since the local elevation model is evaluated more times and more points must be computed across the cell. It does not have the freedom to compute only those points necessary to adequately describe the contour. Consequently, for efficiency it is used only when the faster Adaptive Method fails.

DEFINITION OF THE LOCAL ELEVATION MODEL

The contouring method discussed above requires a mathematical representation E(x,y) of the elevation over each grid cell. Moreover, that equation must be simple to solve in both the formulation

$$E(x,y_f) - C = 0 \qquad (2)$$

where y is held fixed and

$$E(x_f,y) - C = 0 \qquad (3)$$

where x is held fixed. In the approach used here, the elevation is locally approximated using 16 adjacent DTM grid values.

These 16 values are obtained from the 4x4 subgrid centered on the current grid cell. This 4x4 subgrid can be partitioned into four 3x3 subgrids, all of which have the center cell in common. A quadratic fit can be made to each of the 3x3 subgrids to produce 4 separate approximations to the elevation over the common center grid cell. These are called $E_1(x,y)$ through $E_4(x,y)$. To produce a composite E(x,y) the four approximations are weighted and averaged using the formula:

$$E(x,y) = \sum w_i(x,y)E_i(x,y) / \sum w_i(x,y) \qquad (4)$$

The weighting function along with this method of obtaining a final estimate is adapted from work done by Jancaitis and Junkins (1973). As the weighting formula is third order, E(x,y) is actually a fifth order polynomial in both x and y. Since solving Equations 2 or 3 directly would be somewhat time-consuming, a further approximation is made. A lattice is defined of the cell being processed with the refinement of the lattice corresponding to the size of the tolerance parameter defined for the Adaptive Contouring Method. E is evaluated at the lattice points as required and further refinement between those points is done by linear interpolation.

EXAMPLE RUNS

Figure 2 presents the results of applying this algorithm to a 20 x 20 DTM. Sixteen contour levels are shown at a separation of 4 feet. A total of 2205 points are used to represent the contours with the tolerance parameter set to 0.25.

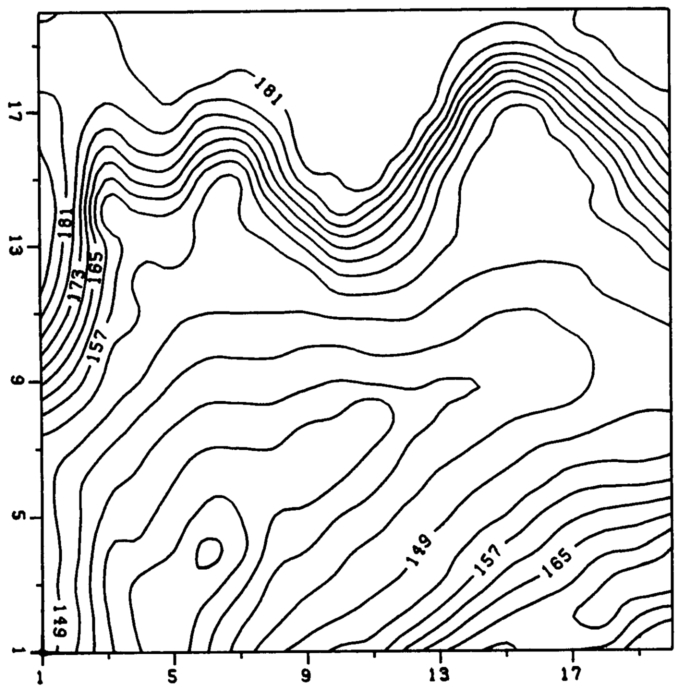


Figure 2. 20x20 DTM, Tolerance = 0.25 - 2205 Contour Points

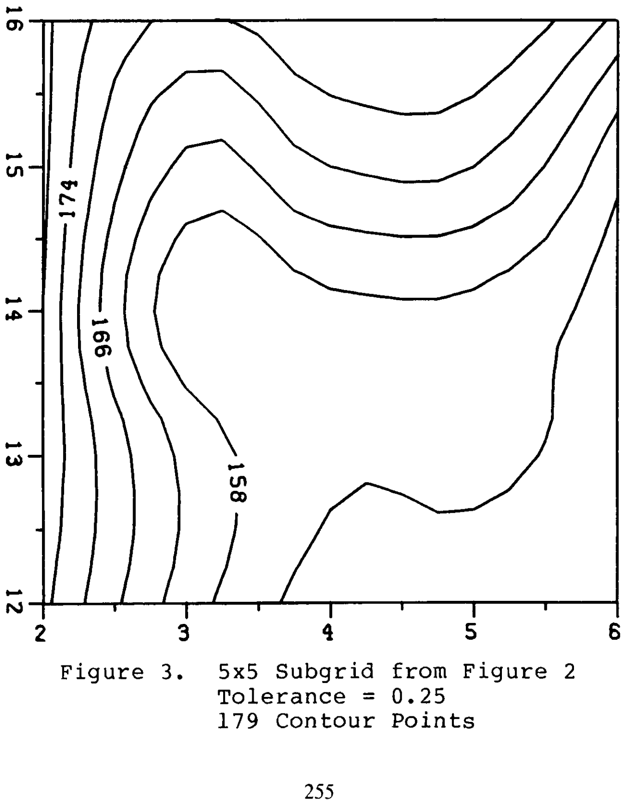


Figure 3. 5x5 Subgrid from Figure 2
Tolerance = 0.25
179 Contour Points

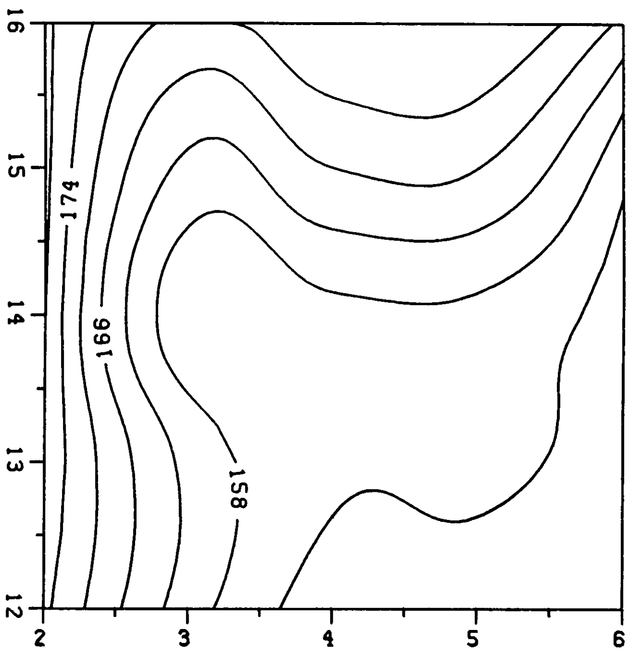


Figure 4. 5x5 Subgrid from Figure 3
Tolerance = 0.0625
665 Contour Points

At the scale shown there would be little improvement in moving to a smaller setting.

A look at a small sub grid in the upper left corner drawn at a different scale shows the effects of varying the tolerance. Figures 3 and 4 are created using tolerances of 0.25 and 0.0625. They require 179 and 665 digitized points respectively. The contours in Figure 4 are noticeably less angular and more pleasing to the eye.

ACKNOWLEDGEMENT

The development of this algorithm was supported by Contract DAAK70-80-C-0248 with the U.S. Army Engineer Topographic Laboratories and the Defense Mapping Agency Hydrographic/ Topographic Center.

REFERENCES

Jancaitis, J.R., and Junkins, J.L., 1973, Mathematical Techniques for Automated Cartography: U.S. Army Engineer Topographic Laboratory, Final Report, Contract DAAK 02-72-C-0256.