Contour tracing for geographical digital data

Tatsuya Ishige

Cogent Geoscience (2017), 3: 1354468
Contour tracing for geographical digital data

Tatsuya Ishige

Abstract: Our purpose is to trace a contour in the form of a polygon. In this research, we use a bicubic spline function for interpolation of the elevation data on a grid covering the area of concern. We construct the polygon as a data consisting of ordered contour points on sides of the grid. The contour enters a cell at an entry point and goes out at an exit point on its sides. The polygon is formed connecting these points. A problem occurs as to which two points should be connected when a cell of the grid has more than three contour points on its sides. As for existing methods of the differential geometry such as discretization using tangential increments, it is difficult to predetermine a suitable step size to arrive at a next contour point correctly if several contour components wind closely to each other within a cell. As a solution, we take an algebraic approach exploiting a simple fact that a bicubic function is viewed as a univariate cubic function with a parameter. From this perspective, we identify the exit point examining the behavior of the real roots of the cubic equation for the contour in terms of the numerical order. Our method enables us to faithfully trace the contour of bicubic spline functions which provide smoother and better fitting curves than bilinear spline functions used by the other authors. Computation time is exhibited in the numerical experiment for an island in Japan.

Subjects: Computer Mathematics; CAD CAE CAM - Computing & Information Technology; Computational Numerical Analysis; Simulation & Modeling; Computer Graphics & Visualization; Computer Science (General); Engineering Mathematics

Keywords: contour trace; polygon; cubic equation; geographical digital data; bicubic spline; Sturm’s theorem

ABOUT THE AUTHOR
Tatsuya Ishige is a graduate student and a member of Management Science Laboratory in Department of Industrial and System Engineering, Faculty of Science and Engineering, Hosei University. In addition to this research, the author studies the geographical properties of the Voronoi territories for the convenience stores in Japan and lectured in APORS 2015.

PUBLIC INTEREST STATEMENT
This research paper is intended for researchers and practitioners who may be interested in making a contour map from geographical digital data. The data-set consists of altitudes at vertices of a rectangular grid. We interpolate these data by a bicubic spline function. We then calculate the locations of contour points on the sides of the grid. These points are connected to form a polygon which represents a contour of the altitude.
1. Introduction
We develop a method to trace a contour for the surface interpolated by a bicubic spline function in the form of a polygon. The principal features of this method are:

- We use the elevation data given on a rectangular grid.
- We interpolate the elevation data by a bicubic spline function which provides fitting and smooth approximation with moderate computational load.
- We detect all the contour points on the sides of the grid and connect them to trace a contour.
- In connecting contour points on the sides of a cell, contour points can be linked correctly even if there are more than three points.
- We can obtain the same polygon no matter which point we choose on the contour as a starting point, different from iterative methods using discretization.

We store contours in the form of polygon data which is easy to handle when applying to various engineering needs such as calculation of the volume of a water reservoir.

The algorithm of tracing and drawing contour is sometimes referred to as “contour tracing” and “contouring.” The pioneering work is Cottafava and Moli (1969). Their method is to compute contour points on a rectangular grid and connect them together to form a contour. Since they use the intermediate value theorem to detect a contour point, the number of contour points on a side of a cell is always presumed to be no more than one. McLain (1971) uses the weighted least square method to approximate the elevation function. This method cannot express the contour winding intricately since the direction of increment vector is presumed to change at an angle of zero or 45 degrees to the previous one. Lopes and Brodlie (1998) use bilinear splines to interpolate the elevation function. By incorporating one or two contour points in the interior of a cell, they enhance the preciseness of contour tracing for the bilinear hyperbola. Maple (2003) uses the method called marching squares which approximates the contour in a rectangular cell by 16 possible patterns.

Recently, in geographical modeling and analysis, the importance of application using discrete altitude data such as geographic information system is widely recognized. In particular, the Triangular Irregular Network (TIN) expressing a terrain surface in a set of triangles is frequently used. The TIN allows the arbitrarily scattered points. For terrain representation, they use only the key points which are less than those aligned regularly in Digital Elevation Model (DEM). Gold, Charters, and Ramsden (1977) propose the method of contour trace using TIN, and Dobkin, Wilks, Levy, and Thurston (1990) extend it to mappings between higher dimensional spaces. Saux, Thibaud, Li, and Kim (2004) extract topographic properties on scattered elevation data model by TIN. Goto and Shimakawa (2015) propose the iterative method of discretization using gradient vectors of an elevation function which is constructed by interpolation with a bicubic spline function.

The preceding works mentioned above do not faithfully represent contours winding intricately in a rectangular cell. In this case, McCullagh (1981) subdivides a grid cell into a series of subcells enabling contours to be laced smoothly through the cell. In our research, however, we presume a fixed rectangular grid and still maintain the accuracy. The outline of our process is illustrated as follows.

The altitude data on a rectangular grid are given in Figure 1. To reproduce the surface, we interpolate the altitude data on the grid by bicubic spline in Figure 2. We then calculate the contour points which are intersections between the rectangular grid and the contour. We finally produce a polygon of the
contour for an altitude as in Figure 3. Although the process is same as that of Lopes and Brodlie (1998), we adopt bicubic spline functions for interpolating elevation data. Taking full advantage of their accuracy, we achieve representation of complicated contours such as having multiple points on a side of a cell shown in Figure 4 connecting A to B, C to D, and E to F. Our key idea which makes it possible is to connect the calculated contour points using the order of real roots for cubic equations.
Figure 3. Polygon of the contour.

Figure 4. Complicated contours in a cell.
2. Preliminaries

2.1. Elevation function

We construct an elevation function over an \( m \times n \) rectangular grid with interval \( \delta \) by interpolating data of altitudes with a bicubic spline function. Although we suppose that a cell of the grid is a square \( \delta \times \delta \) for simplicity, the discussion below can be applied easily to a rectangle \( \delta_x \times \delta_y \). There are various types of spline polynomial functions with different degrees. However, the most commonly used are cubic splines because of their precision and efficiency in computation. In particular, cubic B-splines are preferred. A base function of one-dimensional B-spline is given by

\[
M_k(x) = M_k(x; \xi_{i-1}, \xi_i, \cdots, \xi_{i+k})(i = 1, \cdots, m),
\]

For bicubic spline functions, \( k \) is set to 4 in expression (1). As for the definition of right-hand side of (1), readers should refer to Piegl and Tiller (2012). The parameters \( \xi_{i}^{+}, \xi_{i}^{-}, \cdots, \xi_{i+k}^{-} \) are called knots and used for controlling the smoothness of the base function. In our case, they coincide with consecutive nodes of the grid. If there are no identical knots in expression (1), then the base function is \( C^2 \)-class at each knot. We generate base functions of bicubic spline by taking a tensor product of two base functions of one-dimensional B-spline. Thus, the base functions of two-dimensional B-spline are \( M_{ij}(x) \) \( M_{ij}(y) \) \( (i = 1, \cdots, m; j = 1, \cdots, n) \). Put \((x, y) = ((i-1)\delta, (j-1)\delta)\) for the coordinates of \((i, j)\)-th grid point, set \( \xi_{i}^{+} = x_{i}^{+} \) \( (l = 0, \cdots, k) \) for \( M_{ij}(x) \), and \( \xi_{j}^{+} = y_{j}^{+} \) \( (l = 0, \cdots, k) \) for \( M_{ij}(y) \). The form of \( M_{ij}(x) \) is given by

\[
M_{ij}(x) = \sum_{l=-2}^{2} \frac{\{|x_{i+l} - x| + (x_{i+l} - x)^3\}}{w_i'(x_{i+l})},
\]

where \( w_i(x) = \prod_{l=0}^{2} (x - x_{i+l}) \). The function \( M_{ij}(y) \) is given by replacing \( x \) with \( y \). The graph of \( M_{ij}(x) \) is illustrated in Figure 5. Taking a suitable linear sum, we obtain a bicubic spline function:

\[
h(x, y) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} M_{ij}(x) M_{ij}(y).
\]
2.2. Roots of the cubic equation

The contour of an altitude $h_0$ is given by the equation $h(x, y) = h_0$ which is a cubic equation in one variable while fixing the other. We use some aspects of cubic equations to draw contours. Thus in this section, we consider a general cubic equation

$$ax^3 + bx^2 + cx + d = 0.$$  \hfill (4)

Putting

$$x = x' - b/3a,$$  \hfill (5)

we obtain

$$x'^3 + px' + q = 0,$$  \hfill (6)

Where

$$p = -1/3(b/a)^2 + c/a,$$
$$q = -2/27(b/a)^3 - bc/3a^2 + d.$$  \hfill (7)

Let the cubic and quartic roots of 1 with the positive imaginary part be denoted by $\omega$ and $i$, respectively. According to the Cardano’s method, we put

$$D = -q^2/4 - p^3/27,$$  \hfill (8)

$$U = \begin{cases} 
-q/2 + \sqrt{Di} & (D \geq 0) \\
-q/2 - \sqrt{Di} & (D < 0)
\end{cases},$$  \hfill (9)

$$V = \begin{cases} 
\bar{U} & (D \geq 0) \\
-q/2 - \sqrt{|D|} & (D < 0)
\end{cases}.$$  \hfill (10)

The roots of (6) are given as follows.

$$x'_1 = \omega u + \omega^2 v,$$
$$x'_2 = \omega^2 u + \omega v,$$
$$x'_3 = u + v,$$  \hfill (11)

and $x_i = x'_i - b/3a$ ($i = 1, 2, 3$) for Equation (4).

In the case of $D < 0$, the equation has one real root and two imaginary roots.

In the case of $D > 0$, we have $\text{Im}(U) > 0$ and $0 < \text{Arg}(U) < \pi$. Then the ranges of the arguments for $u$, $\omega u$, and $\omega^2 u$ are $0 < \text{Arg}(u) < \pi/3$, $2\pi/3 < \text{Arg}(\omega u) < \pi$, and $4\pi/3 < \text{Arg}(\omega^2 u) < 5\pi/3$, respectively, as shown in Figure 6. Since they are separated, we obtain three distinct real roots $x_1 < x_2 < x_3$. In this case, we call the suffix $i$ of $x_i$ the order of the root.

In the case of $D = 0$, we have $U = V = -q/2$ from expression (9). If $q < 0$, $x_1$ is a simple root and $x_2, x_3$ are a multiple root as illustrated in Figure 7. If $q > 0$, $x_1$ is a simple root and $x_2, x_3$ are a multiple root as illustrated in Figure 8.
Figure 6. Three real roots
\( (D > 0) \).

Figure 7. One simple root and one multiple root \( (D = 0, q < 0) \).
2.3. Behavior of the roots

We here generally consider a parameterized cubic equation

\[
 f(s, t) = a(t)s^3 + b(t)s^2 + c(t)s + d(t) = 0. \tag{12}
\]

Each coefficient is a continuous function of \( t \). Solving Equation (12) for \( s \), we obtain the roots \( s_1(t), s_2(t), s_3(t) \) corresponding to \( x_1, x_2, x_3 \) for (4), respectively. The counterparts of \( q \) and \( D \) are parameterized as \( q(t) \) and \( D(t) \). The behavior of the roots is classified as follows.

(i) In the case of \( D(t) < 0 \), one root is real and the other roots are imaginary.

(ii) In the case of \( D(t) > 0 \), we obtain three distinct real roots \( s_1(t), s_2(t), s_3(t) \) corresponding to \( x_1, x_2, x_3 \) for (4), respectively. The counterparts of \( q \) and \( D \) are parameterized as \( q(t) \) and \( D(t) \). The behavior of the roots is classified as follows.

(iii) In the case of \( D(t) = 0 \) for \( t = t_d \), the Equation (12) have a multiple root. We then have \( U(t_d) = V(t_d) = q(t_d)/2 \). If \( q(t_d) < 0 \), then \( s_1(t) \) is a simple root and \( s_2(t), s_3(t) \) are confluent as a multiple root. If \( q(t_d) > 0 \), then \( s_1(t) \) is a simple root and \( s_2(t), s_3(t) \) are confluent as a multiple root.

(iv) In the case of \( a(t) = 0 \) for \( t = t_a \), the Equation (12) degenerates to a quadratic equation. Substituting \( 1/r \) into \( s \) of (12), we obtain an equation

\[
 d(t)r^4 + c(t)r^2 + b(t)r + a(t) = 0. \tag{13}
\]

Since \( a(t) = 0 \), one of the roots for (13) denoted by \( \epsilon(t) \) approaches to zero as \( t \) approaches to \( t_a \). For the roots \( \alpha(t), \beta(t), \epsilon(t) \) of (13) we have relations

\[
 \alpha(t) + \beta(t) + \epsilon(t) = -c(t)/d(t), \tag{14}
\]

\[
 \alpha(t)\beta(t) + \beta(t)\epsilon(t) + \alpha(t)\epsilon(t) = b(t)/d(t), \tag{15}
\]
From (15) and (16), as $0 < |\varepsilon(t)| << 1$, we obtain
\[
\alpha(t)\beta(t) \varepsilon(t) \simeq b(t)/d(t),
\]
\[
\varepsilon(t) \simeq -a(t)/b(t).
\]
If $a(t)b(t) > 0$, then $\varepsilon(t) < 0$ and the smallest of the roots of (12) satisfies $s_1(t) = 1/\varepsilon(t) \to -\infty$ ($t \to t_a$) as shown in Figure 9. On the other hand, if $a(t)b(t) < 0$, then $\varepsilon(t) > 0$ and the largest of the roots of (12) satisfies $s_3(t) = 1/\varepsilon(t) \to +\infty$ ($t \to t_a$).

### 3. Proposed algorithm

We trace the contour points on the sides of a grid cell based on the behavior of the real roots of the cubic equations. To make a polygon, we propose an algorithm composed of the following four phases:

(i) Interpolation with bicubic spline functions.
(ii) Detection of the grid cells with relevant contour points on their sides.
(iii) Computation of the coordinates for the contour points.
(iv) Selection of the next contour point on the sides of a grid cell.

More precisely, (i) we interpolate the altitudes for the grid points with a bicubic spline function. (ii) we count the number of contour points on each side of a grid cell by Sturm’s method (Bourdon, 1853) and detect the grid cells which the contour traverses. (iii) we compute the coordinate values of contour points on the sides of the grid cells detected above using a numerical method. (iv) we determine the exit point corresponding to the entry point and append it to the polygon. In the case that the cell has more than three contour points on the sides, we select the exit point according to the behavior of the real roots of the cubic equations. We elaborate on the first and fourth phases below.

### 3.1. Interpolation

We describe below an algorithm for determining coefficients $c_{ij}$ in expression (3). We give matrix representations $A$ and $B$ of the system of base functions for one-dimensional B-spline in $x$ and $y$, respectively, as follows.

\[
A = \begin{bmatrix}
M_{k,1}(x_1) & M_{k,2}(x_1) & \cdots & M_{k,m}(x_1) \\
\vdots & \ddots & \ddots & \vdots \\
M_{k,1}(x_m) & M_{k,2}(x_m) & \cdots & M_{k,m}(x_m)
\end{bmatrix},
\]

\[
B = \begin{bmatrix}
M_{k,1}(y_1) & M_{k,2}(y_1) & \cdots & M_{k,n}(y_1) \\
\vdots & \ddots & \ddots & \vdots \\
M_{k,1}(y_n) & M_{k,2}(y_n) & \cdots & M_{k,n}(y_n)
\end{bmatrix},
\]

where $(x_i, y_j) (i = 1, \ldots, m; j = 1, \ldots, n)$ represent locations of grid points. Let the altitude at a grid point $(x, y)$ be denoted by $h_i$. Since the values of the spline function coincide with the altitudes at the grid points, we have

\[
\alpha(t)\beta(t)\varepsilon(t) = -a(t)/d(t).
\]
We may put \( c = (c_{11}, c_{12}, \ldots, c_{mn})^T \), \( h = (h_{11}, h_{12}, \ldots, h_{mn})^T \) and rewrite (19) as
\[
(A \otimes B)c = h. \tag{20}
\]
Moreover, instead of using the tensor product in the left-hand side of (20), we obtain an equivalent equation
\[
ACB^T = H, \tag{21}
\]
where
\[
C = \begin{bmatrix}
    c_{11} & \cdots & c_{1n} \\
    c_{21} & \cdots & c_{2n} \\
    \vdots & \ddots & \vdots \\
    c_{m1} & \cdots & c_{mn}
\end{bmatrix},
H = \begin{bmatrix}
    h_{11} & \cdots & h_{1n} \\
    h_{21} & \cdots & h_{2n} \\
    \vdots & \ddots & \vdots \\
    h_{m1} & \cdots & h_{mn}
\end{bmatrix}. \tag{22}
\]
The equivalence above is clear when expressed in summations:
\[
\sum_{p=1}^{m} \sum_{q=1}^{n} a_{pq} b_{pq} c_{pq} = h_{ij}. \tag{23}
\]
Putting \( W = CB^T \), the equation of (21) becomes \( AW = H \). The \( j \)th vertical vector of \( W \) is the solution of
\[
A \begin{pmatrix}
    w_{1j} \\
    \vdots \\
    w_{mj}
\end{pmatrix} = \begin{pmatrix}
    h_{1j} \\
    \vdots \\
    h_{mj}
\end{pmatrix}. \tag{24}
\]
We may employ LU-decomposition to solve (24). The computation is made efficient taking account of the fact \( L \) and \( U \) are band matrices in the form of
\[
L = \begin{bmatrix}
    * & 0 & \cdots & \cdots & 0 \\
    * & * & \cdots & \cdots & 0 \\
    \vdots & \vdots & \ddots & \ddots & \vdots \\
    0 & \cdots & 0 & * & * \\
    0 & \cdots & 0 & 0 & *
\end{bmatrix},
U = \begin{bmatrix}
    1 & * & \cdots & 0 \\
    0 & 1 & * & \cdots \\
    \vdots & \vdots & \ddots & \ddots \\
    0 & 0 & \cdots & * \\
    0 & \cdots & 0 & 0 & 1
\end{bmatrix}. \tag{25}
\]
Likewise, we compute \( C \) by solving \( BC^T = W^t \) with LU-decomposition. Thus, we obtain the spline function \( h(x, y) \) in (3).

3.2. Selection of the next contour point
We focus on the fourth phase which is the mainstay of the proposed algorithm. In the course of construction, the polygon of a contour enters a cell at a contour point on a side which we call an entry point. It goes out of the cell at one of the other contour points on the sides which we call an exit point for the entry point. Thus, if the number of the contour points on the sides is two, the exit point is automatically determined. Otherwise, we select the exit point using the following algorithm.
3.2.1. Coordinate transformation

To facilitate the algorithm, each time we enter a new \((i, j)\)-cell, we transform the coordinates from \((x, y)\) to \((s, t)\) as specified in Table 1. Note that the range of the \((i, j)\)-cell is \(\{(x, y) \in \mathbb{R}^2 | (i - 1)\delta \leq x \leq i\delta, (j - 1)\delta \leq y \leq j\delta \}\). As a result, the cell is rescaled as a unit square and the entry point is relocated on the bottom side. By substituting \((s, t)\) into \((x, y)\) in expression (3), the elevation function is formulated by

\[
h_{ij}(s, t) = a_{ij}(t)s^3 + b_{ij}(t)s^2 + c_{ij}(t)s + d_{ij}(t).
\]  

(26)

All the coefficients in expression (26) are polynomials in \(t\) of degree at most 3. The range of the \((i, j)\)-cell is rescaled as \(\{(s, t) \in \mathbb{R}^2 | 0 \leq s \leq 1, 0 \leq t \leq 1\}\). We put \(f_{ij}(s, t) = h_{ij}(s, t) - h_0\). The contour of altitude \(h_0\) within the \((i, j)\)-cell is equal to the solution curve determined by

\[
f_{ij}(s, t) = 0.
\]  

(27)

Thus, we are prepared to proceed to the selection of the exit point. Various cases are described below along with illustration in figures where an entry point is indicated by a triangle and the selected exit point is encircled by a dashed circle.

3.2.2. The case \(D(t) < 0\) for \(0 \leq t \leq 1\)

For the exit point, we select a point with the smallest \(t\) among the remaining contour points on the sides (Figure 10).

| Side containing the entry point | \(s\)            | \(T\)            |
|--------------------------------|------------------|------------------|
| South                         | \(x\delta - i + 1\) | \(y\delta - j + 1\) |
| West                          | \(j - y\delta\)   | \(x\delta - i + 1\) |
| North                         | \(i - x\delta\)   | \(j - y\delta\)   |
| East                          | \(y\delta - j + 1\) | \(i - x\delta\)   |

Figure 10. The case \(D(t) < 0\) for \((0 \leq t \leq 1)\).
3.2.3. The case $D(t) > 0$ for $0 \leq t \leq 1$

We select the exit point using the numerical order of the three real roots of the cubic equations. The exit point is determined as one with the smallest $t$ among the contour points on the sides which have the same order as the entry point. To obtain the order of a root, no information is required on the other roots. We just use the following fact:

For a monic cubic function

$$f(x) = (x - \alpha)(x - \beta)(x - \gamma), \quad (\alpha < \beta < \gamma),$$

we have

$$f'(x) = (x - \beta)(x - \gamma) + (x - \alpha)(x - \gamma) + (x - \alpha)(x - \beta),$$

and

$$f''(x) = 2(3x - \alpha - \beta - \gamma).$$

Thus, we determine the order according to Table 2.

In the case of Figure 11, if the contour line enters the right-hand cell at the point A which order is 2, then it leaves the cell at the point D which order is also 2. On the other hand, if the contour line enters the right cell at the point B which order is 3, then it goes out of the cell at the point C which order is also 3. Note that the order of a root is determined over the range $-\infty < s < +\infty$.

### Table 2. Sign of the derivatives at the three real roots

| Derivative | $\alpha$ | $\beta$ | $\gamma$ |
|------------|----------|---------|----------|
| $f'(x)$    | +        | -       | +        |
| $f''(x)$   | -        | N/A     | +        |

Figure 11. The case $D(t) > 0$ for $(0 \leq t \leq 1)$. 
3.2.4. The case $D(t_d) = 0$ for some $t_d$ with $0 < t_d < 1$

The process of selecting the exit point varies according to the sign of $D(t_d)$ and $q(t_d)$:

(i) The case $D(t_d) < 0$ and $q(t_d) > 0$

If the entry point is of order 1, among the other contour points, we select a point of order 1 with the smallest $t < t_d$ if any (Figure 12). If there is no such point, we select a point with the smallest $t > t_d$ (Figure 13).

If the entry point is of order 2(3), we select a point of order 2(3) with the smallest $t < t_d$ if any (Figure 14). If there is no such point, we select a point of order 3(2) with the largest $t < t_d$ (Figure 15).

(ii) The case $D(t_d) < 0$ and $q(t_d) < 0$

If the entry point is of order 3, among the other contour points, we select a point of order 3 with the smallest $t < t_d$ if any (Figure 16). If there is no such point, we select a point with the smallest $t > t_d$ (Figure 17).

If the entry point is of order 1(2), we select a point of order 1(2) with the smallest $t < t_d$ if any (Figure 18). If there is no such point, we select a point of order 2(1) with the largest $t < t_d$ (Figure 19).

(iii) The case $D(t_d) > 0$ and $q(t_d) > 0$

Among the contour points other than the entry point, we select a point with the smallest $t < t_d$ if any (Figure 20). If there is no such point, we select a point of order 1 with the smallest $t > t_d$ (Figure 21).
Figure 13. The case \( D(t_j) < 0 \) and \( q(t_j) > 0 \), enter at order 1, exit at \( t > t_j \).

Figure 14. The case \( D(t_j) < 0 \) and \( q(t_j) > 0 \), enter at order 2(3), exit at order 2(3) \( t < t_j \).
Figure 15. The case $D(t_d) < 0$ and $q(t_d) > 0$, enter at order 2(3), exit at order 3(2) ($t < t_d$).

Figure 16. The case $D(t_d) < 0$ and $q(t_d) < 0$, enter at order 3, exit at order 3 ($t < t_d$).
Figure 17. The case $D(t_d) < 0$ and $q(t) < 0$, enter at order 3, exit at $t > t_d$.

Figure 18. The case $D(t_d) < 0$ and $q(t) < 0$, enter at order 1(2), exit at order 1(2) ($t < t_d$).
Figure 19. The case $D(t) < 0$ and $q(t) < 0$, enter at order 1(2), exit at order 2(1) ($t < t_d$).

Figure 20. The case $D(t) > 0$ and $q(t) > 0$, enter as a unique real root ($t < t_d$), exit at ($t < t_d$).
(iv) The case \( D'(t_d) > 0 \) and \( q(t_d) < 0 \)

Among the contour points other than the entry point, we select a point with the smallest \( t < t_d \) if any (Figure 22). If there is no such point, we select a point of order 3 with the smallest \( t > t_d \) (Figure 23).
3.2.5. The case $a(t_a) = 0$ for some $t_a$ with $0 < t_a < 1$

The process of selecting the exit point varies according to the sign of $a'(t_a)b(t_a)$ and $D(t)$:

(i) The case $D(t) < 0$ for $0 < t < 1$

Among the contour points other than the entry point, we select a point with the smallest $t < t_a$ (Figure 24).

(ii) The case $a'(t_a)b(t_a) > 0$ and $D(t) > 0$ for $0 < t < 1$

If the entry point is of order 1(2), among the other contour points, we select a point of order 1(2) with the smallest $t < t_a$ if any (Figure 25). If there is no such point, we select a point of order 2(3) with the smallest $t > t_a$ (Figure 26).

If the entry point is of order 3, we select a point of order 3 with the smallest $t < t_a$ (Figure 27).
Figure 25. The case $a'(t_a)b(t_a) > 0$ and $D(t) > 0$, enter at order 1(2), exit at order 1(2) ($t < t_a$).

Figure 26. The case $a'(t_a)b(t_a) > 0$ and $D(t) > 0$, enter at order 1(2), exit at order 2(3) ($t > t_a$).

Figure 27. The case $a'(t_a)b(t_a) > 0$ and $D(t) > 0$, enter at order 3, exit at order 3 ($t < t_a$).
(iii) The case $a'(t_a)b(t_a) < 0$ and $D(t) > 0$ for $0 < t < 1$

If the entry point is of order 2(3), among the other contour points, we select a point of order 2(3) with the smallest $t < t_a$ if any (Figure 28). If there is no such point, we select a point of order 1(2) with the smallest $t > t_a$ (Figure 29).

If the entry point is of order 1, we select a point of order 1 with the smallest $t < t_a$ (Figure 30).

3.2.6. The case $D(t_d) = 0$ and $a(t_a) = 0$ for some $t_d, t_a$ with $0 < t_d, t_a < 1$

We perform the procedures of 3.2.4 and 3.2.5 according to the numerical order of $t_d, t_a$. In the case of more $t_d$ and $t_a$, we iterate the procedures accordingly.
To illustrate the process, we consider a cell of the grid with solutions \( t_a \), \( t_d \) for \( a(t) = 0 \) and \( D(t) = 0 \), respectively, assuming \( a(t) < 0 \) and \( D(t) > 0 \) (\( 0 < t < t_a \)). In this case, we perform a procedure composed of 3.2.5-(ii) and 3.2.4-(ii). We start from an entry point of numerical order 1 shown in Figure 31. Firstly, we perform the process of 3.2.5-(ii) since \( t_a < t < t_d \). No contour points of numerical order 1 are within the range of \( t_a < t < t_d \). Thus, the root curve containing the entry point is connected to the root curve of order 2 within the range of \( t_a < t < t_d \). Secondly, we perform the process of 3.2.4-(ii) considering the intersection between the root curve and line \( t = t_d \) as an entry point of order 2. No contour points of numerical order 2 are within the range of \( t_a < t < t_d \). Thus, we select the contour point of numerical order 1 within the range of \( t_a < t < t_d \) as the exit point.

3.2.7. Selection of the next cell

In generating a polygon for a connected component of contour, we start from the entry point in the initial cell, and link the entry point of the current cell to the correct exit point and add it to the polygon. This process is repeated until returning to the initial point. In this process, we need to select the new cell for the entry point which is also the exit point in the current cell. In other words, we transfer
to the next cell sharing the contour point with the current cell. If the contour point is not a vertex, these cells share a side. In this case, the next cell is clearly determined.

However, if the contour point is a vertex, the next cell is selected by the gradient of the elevation function which is perpendicular to the contour. Namely, if the gradient is in the direction of \(t\)-axis, we select the cell adjacent to the current cell in the direction of \(s\)-axis. If the gradient is in the direction of \(s\)-axis, we select the cell adjacent to the current cell in the direction of \(t\)-axis. If the gradient is neither in the direction of \(s\)-axis nor \(t\)-axis, the next cell is selected diagonal to the current cell. Each time the next cell is determined, we conduct the coordinate transformation in Section 3.2.1.

3.3. Case study of Yakushima island

In this section, we use the JMC50 m mesh data which is a proprietary product of Japan Map Center. These data are a remake of the altitude data on 50 m mesh of Geospatial Information Authority of Japan currently out of print. In this data, the altitudes of oceanic area are expressed by \(-999.9\) m. We change it to \(-1\) m for the purpose of interpolation following the precedent of Goto and Shimakawa (2015).

As a case study, we apply the algorithm described above to draw the contour map of Yakushima island which is located at the south part of Kagoshima prefecture. We use an altitude data on 498 \(\times\) 598 grid. The grid interval is 50 m \(\times\) 50 m. The altitude of each grid point is measured by meters. The highest point of Yakushima is at an altitude of 1,935 m.

We set \(m = 498\), \(n = 598\), and \(\delta = 50\) in expression (3) and obtain the spline function which is shown in Figure 32. As a result of the entire algorithm, the contour map is completed as shown in Figure 33 where the contours are drawn every 100 m from 0 to 1,900 m. To see the polygonal structure of the

![Figure 32. Topographical illustration of Yakushima island.](image-url)
contours, we focus on area A in Figure 33, and enlarge it in Figure 34, where we add the contours for every 10 m from 0 to 100 m. In this scale, the piecewise linear structure of the polygons is recognizable. Although the discussion about the superiority of bicubic splines over bilinear splines is out of scope of our research, we compare these two splines by applying them to an area in Yakushima island in Figure 35. The contour of the elevation function by bicubic spline is colored black, while the one by bilinear spline is colored gray. The central cell of Figure 35 is magnified in Figure 36. The polygons constructed by our method are delineated by black solid lines, while the one by the method of Lopes and Brodlie (1998) is drawn by gray dashed lines in Figure 37 and Figure 38. Both methods faithfully trace the contour of spline functions, respectively. As a consequence, the advantage of bicubic spline interpolation is reflected on our polygon.
We measure the time of computation for constructing each polygon as shown in Figure 39 where the level axis represents the number of points in a polygon and the vertical axis the computational time. The maximum computational time is measured 242.39 s for a polygon of 500 m consisting of 5,000 points. To compare the computational time between a cell with two contour points on its sides and a cell with more than two, let $n_1$ be the number of the former and $n_2$ be the number of the latter. The regression of computational time $t$ on $n_1$ and $n_2$ is $t = 0.046n_1 + 0.108n_2 - 10.68$ with $R^2 = 0.993$. Although our algorithm of selecting the exit points seems complicated, the computational time is not so affected by the number of the contour points on the sides of a cell: The ratio of the regression coefficients for $n_1$ and $n_2$ above is about 1:2, while the ratios of $n_1$ and $n_2$ is around 95:5.
Figure 35. Contours for bilinear and bicubic splines.

Figure 36. Contours for bilinear and bicubic splines magnified.
The experiment above is performed in the following environment:

Machine: Aspire M3970,

CPU: Inter Core i7–3770 3.40 GHz,

OS: Windows 7 Enterprise,

Memory: 8.00 GB,

Program: Wolfram Mathematica 10.3.
Figure 38. Polygons by our method and Lopes-Brodie's magnified.

Figure 39. Computational time and number of points in a polygon.

\[ t = 0.046n - 0.228 \]
4. Conclusion
The method of contour tracing developed in this research makes full use of the fact that a bicubic function is regarded as a univariate cubic function with a parameter. The behavior of the real roots within a rectangular cell is examined to link an entry point with the corresponding exit point. This algorithm detects and traces all the connected components of a target contour which intersect the grid and gives the same polygon independent of the choice of a starting point on a connected component. In the case of complex terrain, this method shows clear benefits over the alternative methods using data on a rectangular grid in providing smooth and fitting contour representation with moderate load of implementation and computation. Tracing a closed contour contained in the interior of a cell remains as a future work.

Acknowledgments
The author is grateful to Prof Hiroyuki Goto in the Department of Industrial and System Engineering, Hosei University, Japan for introducing the author to the literature of preceding works about contour tracing including his own.

Funding
The author received no direct funding for this research.

Author details
Tatsuya Ishige
E-mail: tatsuya.ishige0809@gmail.com
1 Department of Industrial and System Engineering, Hosei University, Tokyo, Japan.

Citation information
Cite this article as: Contour tracing for geographical digital data, Tatsuya Ishige, Cogent Geoscience (2017), 3: 1354468.

Cover image
Source: Author.

References
Bourdon, M. (1853). Elements of algebra (pp. 354–366). New York, NY: A. S. Barnes & .
Cottafava, G., & Moli, G. L. (1969). Automatic contour map. Communications of the ACM, 12, 386–391. doi:10.1145/363156.363172
Dobkin, D. P., Wilks, A. R., Levy, S. V. F., & Thurston, W. P. (1990). Contour tracing by piecewise linear approximations. ACM Transactions on Graphics, 9, 389–423. doi:10.1145/88560.88575
Gold, C. M., Charters, T. D., & Ramsden, J. (1977). Automated contour mapping using triangular element data structure and an interpolant over each irregular triangular domain. In ACM SIGGRAPH Computer Graphics, San Jose (pp. 170–175). doi:10.1145/563858.563887
Goto, H., & Shimakawa, Y. (2015). Storage-efficient method for generating contours focusing on roundness. International Journal of Geographical Information Science, 30, 200–220. doi:10.1080/13658816.2015.1080830
Lopes, A., & Brodlie, K. (1998). Accuracy in contour drawing. In Eurographics UK Conference Proceedings (pp. 301–311), Leeds.
Maple, C. (2003). Geometric design and space planning using the marching squares and marching cube algorithms. In 2003 International conference on geometric modelling and graphics (pp. 90–95). IEEE Computer Society. https://doi.org/10.1109/GMAG.2003.1219671
McCullagh, M. J. (1981). Creation of smooth contours over irregularly distributed data using local surface patches. Geographical Analysis, 13, 51–63. https://doi.org/10.1111/j.1538-4632.1981.tb00714.x
McLain, D. H. (1971). Drawing contours from arbitrary data points. The Computer Journal, 17, 318–324. doi:10.1093/comjnl/17.4.318
Piegl, L., & Tiller, W. (2012). The NURBS book (pp. 47–138). Berlin: Springer.
Saux, E., Thibaud, R., Li, K.-J., & Kim, M.-H. (2006). A new approach for topographic feature-based characterization of digital elevation data. In Workshop on Advances in Geographic Information System (pp. 73–81). ACM Press. doi:10.1145/1032222.1032235

4. Conclusion
The method of contour tracing developed in this research makes full use of the fact that a bicubic function is regarded as a univariate cubic function with a parameter. The behavior of the real roots within a rectangular cell is examined to link an entry point with the corresponding exit point. This algorithm detects and traces all the connected components of a target contour which intersect the grid and gives the same polygon independent of the choice of a starting point on a connected component. In the case of complex terrain, this method shows clear benefits over the alternative methods using data on a rectangular grid in providing smooth and fitting contour representation with moderate load of implementation and computation. Tracing a closed contour contained in the interior of a cell remains as a future work.

Acknowledgments
The author is grateful to Prof Hiroyuki Goto in the Department of Industrial and System Engineering, Hosei University, Japan for introducing the author to the literature of preceding works about contour tracing including his own.

Funding
The author received no direct funding for this research.

Author details
Tatsuya Ishige
E-mail: tatsuya.ishige0809@gmail.com
1 Department of Industrial and System Engineering, Hosei University, Tokyo, Japan.

Citation information
Cite this article as: Contour tracing for geographical digital data, Tatsuya Ishige, Cogent Geoscience (2017), 3: 1354468.

Cover image
Source: Author.

References
Bourdon, M. (1853). Elements of algebra (pp. 354–366). New York, NY: A. S. Barnes & .
Cottafava, G., & Moli, G. L. (1969). Automatic contour map. Communications of the ACM, 12, 386–391. doi:10.1145/363156.363172
Dobkin, D. P., Wilks, A. R., Levy, S. V. F., & Thurston, W. P. (1990). Contour tracing by piecewise linear approximations. ACM Transactions on Graphics, 9, 389–423. doi:10.1145/88560.88575
Gold, C. M., Charters, T. D., & Ramsden, J. (1977). Automated contour mapping using triangular element data structure and an interpolant over each irregular triangular domain. In ACM SIGGRAPH Computer Graphics, San Jose (pp. 170–175). doi:10.1145/563858.563887
Goto, H., & Shimakawa, Y. (2015). Storage-efficient method for generating contours focusing on roundness. International Journal of Geographical Information Science, 30, 200–220. doi:10.1080/13658816.2015.1080830
Lopes, A., & Brodlie, K. (1998). Accuracy in contour drawing. In Eurographics UK Conference Proceedings (pp. 301–311), Leeds.
Maple, C. (2003). Geometric design and space planning using the marching squares and marching cube algorithms. In 2003 International conference on geometric modelling and graphics (pp. 90–95). IEEE Computer Society. https://doi.org/10.1109/GMAG.2003.1219671
McCullagh, M. J. (1981). Creation of smooth contours over irregularly distributed data using local surface patches. Geographical Analysis, 13, 51–63. https://doi.org/10.1111/j.1538-4632.1981.tb00714.x
McLain, D. H. (1971). Drawing contours from arbitrary data points. The Computer Journal, 17, 318–324. doi:10.1093/comjnl/17.4.318
Piegl, L., & Tiller, W. (2012). The NURBS book (pp. 47–138). Berlin: Springer.
Saux, E., Thibaud, R., Li, K.-J., & Kim, M.-H. (2006). A new approach for topographic feature-based characterization of digital elevation data. In Workshop on Advances in Geographic Information System (pp. 73–81). ACM Press. doi:10.1145/1032222.1032235