Piecewise-Quadratic Functions and Contouring

Robin Sibson

School of Mathematical Sciences
University of Bath, UK

Contour mapping is a long-established technique for the presentation of two-dimensional variation; meteorological applications are of course very familiar, as are applications in topographical mapping, but there are many other important applications in the earth sciences and elsewhere. In mathematical terms, the contour at level h is the solution of the equation

$$f(x,y) = h \qquad ...(1)$$

(x,y) are the cartesian coordinates of a point in the plane, and f is the function to be contoured. It is a consequence of a theorem in mathematical analysis that if the derivative of f is nonzero, then the solution of (1) is a curve which is at least as differentiable as f is at any point on it, but need not be more so. Thus if f has discontinuities (faults, cliff-edges) then the contour lines may have breaks; if f is continuous but has discontinuities of derivative (V-shaped valleys, sharp ridges, boundaries of scree banks) then the contour lines may have sudden changes of direction. However, such phenomena occupy little of the plane; elsewhere f is at least continuously differentiable; it looks smooth; and its contours have continuously turning tangents except at points of zero derivative such as saddlepoints. The contouring of functions which are, or may acceptably be assumed to be, smooth (that is, continuously differentiable) is the case of interest in meteorological applications, although it is fair to point out that faulting is an important problem in other areas.

Interpolating the data

In some types of empirical study, the function f is known only through observations of various types; commonly these are observations of the value

$$f(x_n,y_n) \qquad ...(2)$$

at a set of distinct points

$$\{(x_n,y_n) : n = 1,...,N\} \qquad ...(3)$$

which are, in general, scattered irregularly. Direct construction of a rainfall map from observations from raingauges is an example of a problem of this kind. In many cases, any error in the values of these observations is normally negligible by comparison with the errors arising from our ignorance of what goes on in the gaps between them; the first stage in the process of data interpretation is accordingly to

choose a function z which interpolates the observations on f, that is,

$$z(x_n,y_n) = f(x_n,y_n) \quad \text{for } n = 1,\ldots,N. \qquad \ldots(4)$$

Except at the points where observations have been taken, there is no guarantee that z coincides with f; the difference is the **interpolation error**. The only way to find the interpolation error at a point is to take an additional observation there. However, one of the criteria for a good interpolation method is that for reasonably well-behaved functions f, the interpolation error should be small and should decrease with the spacing of the observations; and of course the interpolant z must be of the right kind — in the present case, at least continuously differentiable. In many problems there is a significant vertical error in the observed values, and smoothing rather than exact interpolation may be appropriate, the gain in smoothness of the fitted function z being achieved at the expense of the replacement of equality by approximate equality in (4). The function z may be derived from the raw data in many ways, for example by any of a variety of *ad hoc* interpolation or smoothing methods (some of which are mathematically very questionable and can produce results with unacceptable properties), or by more structured and powerful methods such as kriging, or thin-plate or natural-neighbour spline interpolation or smoothing. General-purpose methods for data interpolation and smoothing have some role to play in meteorology, for example in problems like that of rainfall mapping. In weather-forecasting applications, however, the usual situation is that a sophisticated forecasting model takes data of many kinds and from many sources as its input, not just observations on a single scalar field; and outputs a number of scalar and vector fields both for current time and for a succession of future times within the forecast period. Thus the general-purpose interpolation method is replaced by a problem-specific method associated with a model incorporating an understanding of the dynamics of the atmosphere. However, once a scalar field z has been identified, it makes no difference to the subsequent contouring method whether it arose from a general-purpose method or from a model. Thus contouring is seen as a process separate from interpolation/smoothing or modelling.

Approaches to contouring

Some automatic contouring procedures attempt to find directly the contours of z, by locating to sufficient accuracy some points (x,y) for which

$$z(x,y) = h \qquad \ldots(5)$$

and then tracking the contour from those points; it is seldom possible to solve directly the equation (5), even though z (unlike the unknown empirical function f) has some explicit or implicit specific mathematical form. This approach is fraught with difficulty over ensuring that each contour is generated once and once only, and over recognising when a ring contour has closed on itself. It is also extravagant in the number of evaluations of z which are required, and inconvenient in that these evaluations have to be made sequentially, as contouring proceeds, and are not positioned in any regular way. Attempts to economise on this number tend to lead to contour lines crossing one another and other unacceptable effects.

A more usual procedure is to approximate z by another function g drawn

from some class of functions G, chosen so that the equation

$$g(x,y) = h \qquad ...(6)$$

can be solved directly to give the contour lines. g is determined once-for-all by evaluating z at a number of control points, usually on a regular grid. Because g does not coincide exactly with z, the difference between them represents a second source of error, the **approximation error**. This can show up by a contour line passing on the wrong side of one of the original observations; in practice this effect can usually be made negligible by choosing a fine enough control grid, but the fineness of the grid required is dependent on the order of accuracy of the approximation procedure. It might be thought desirable to eliminate approximation error by ensuring that the interpolant functions themselves were such that (5) could be solved directly; unfortunately no interpolation method is known which satisfies the mathematical requirements for such methods and also has this property. And it would obviously be quite inappropriate to try to impose such a requirement on the output from a forecasting model. Indeed, historically speaking the problem even with the approximating-function approach to contouring has been that of finding a suitable class of functions G even in the context of control points on a grid.

Seamed quadratic elements for contouring

To achieve visually smooth contours without a fine grid spacing, the class G must consist of continuously differentiable functions. Bicubic splines might seem to be a natural choice, and these have indeed been used in contouring (Hutchinson, pers. comm.). Unfortunately there is then no obvious way of solving (6) neatly in parametric form, and this applies in general to functions *more* complicated than quadratics. Conversely, continuous differentiability is certainly not achievable in nontrivial cases with functions *less* complicated than quadratics, and this focusses attention on the possibility that G should consist of piecewise quadratic functions. The piecewise-quadratic approach suggested by Sibson and Thomson (1981), and implemented in the CONICON package, is novel in two ways. First, it uses the value *and gradient* of z on a (rectangular or square) grid as its control information; it seems more natural to do this than to use value-only information to control a continuously differentiable function. It is usually easy to compute the gradient of z at the same time as its value; if this is not possible then in most cases a good gradient estimate can be constructed using adjacent grid values, and the package provides a utility to do this. Secondly, each cell is filled in by using a seamed quadratic finite element of a novel kind. The resultant class of functions G consists of continuously differentiable functions, because the elements are constructed in such a way that continuous differentiability holds across internal seam lines, and across the boundary from one element to another when there is common control information at the two common vertices; thus the contours are continuously differentiable, and because the elements of G are piecewise quadratic the contours are piecewise conic sections (typically ellipse or hyperbola) which can readily be described in parametric form section-by-section. Hence the name of the package: CONICON.

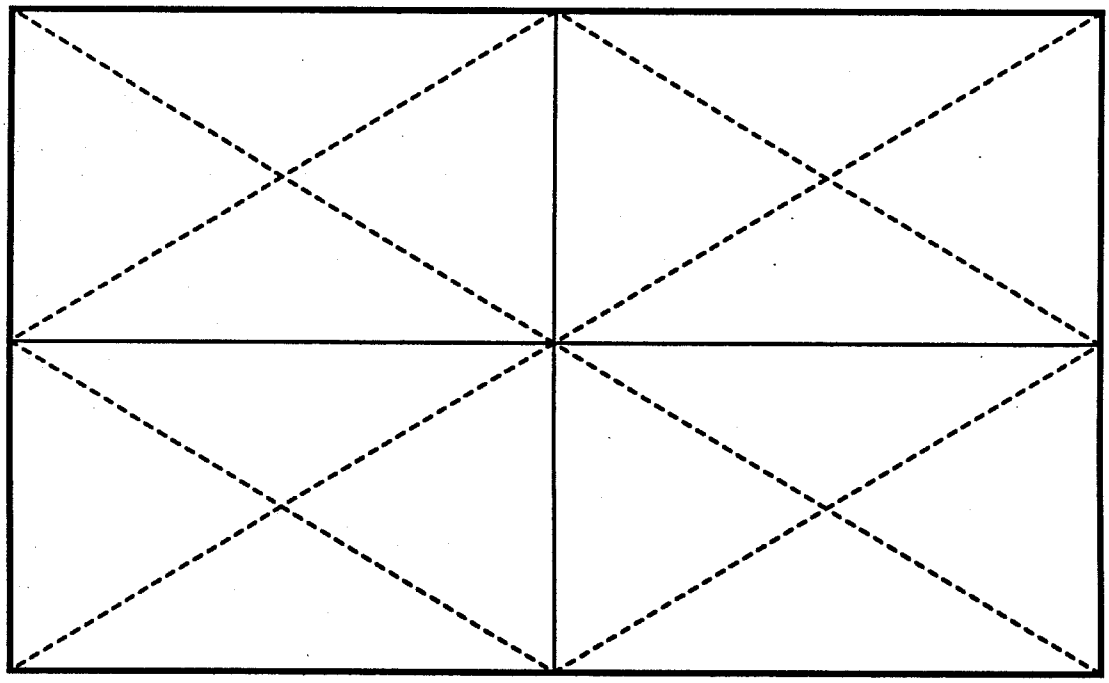

Figure 1: Structure of the seamed quadratic element

The use of continuously differentiable piecewise quadratic functions (but of a different kind not actually forming a finite element scheme) for contouring was proposed by Powell (1974), and Marlow and Powell (1976) proposed a tracking algorithm for the separate contour sections. Powell and Sabin (1977) proposed some C^1-conforming seamed quadratic elements on triangles, and Ritchie (1978) developed a rectangular element from this, a construction which divided the cell into 40 triangles, compared with the 16 used in the Sibson-Thomson construction shown in Figure 1. Although even the 16-triangle element looks at first sight like a complicated construction, it proves to offer in practice a good tradeoff between flexibility and simplicity. Its control information consists of twelve quantities: the value and two components of the gradient of z at each of its four vertices. If z is quadratic, the element matches it exactly. Thus approximation error is controlled by the magnitude of the third derivative of z and decreases as the cube of the grid spacing, which means that the grid spacing can in practice be considerably larger than with lower-order methods for a given size of approximation error. The construction of the element is parsimonious, in the sense that once the conditions of internal continuous differentiability and conformability of value and gradient on the boundary are satisfied, there are exactly enough remaining degrees of freedom to match the control data. The determination of the quadratic on each triangle in homogeneous coordinates relative to that triangle is very efficient, and good bounds are available for the rapid elimination of cells, and then of triangles, which any given contour avoids. Because the function as a whole is continuously differentiable the contour lines are always smooth, whatever the grid spacing, and are easily parametrised so that the **tracking error** (the third source of error, arising from the fact that on any actual vector graphics device, curves are represented by short straight-line segments) is negligible — a good adaptive step

length algorithm is easy to devise. Thus the apparent complexity of the element is no barrier to its use. Further details of its mathematical properties may be found in Thomson (1984).

Sources of error

In the course of the above discussion, three different sources or components of error in contouring have been identified, at least in the context of a method which uses the approximating-function approach. These are: the interpolation error, which is the difference between the empirical function being contoured, a function knowable only through empirical observation, and the interpolant which provides values and gradients at any point in the plane in a manner consistent with the observations taken on the empirical function; the approximation error, which is the difference between the interpolant function and the function whose contours are actually drawn, in the present case a continuously differentiable piecewise quadratic function; and the tracking error, which is the error in following the contours of this last function arising from the finite length of the straight-line steps which approximate its (generally) curved contours. It is easy to understand the distinction between these different components of error by considering how they might be reduced. First, with a fixed interpolation method, interpolation error can only be reduced by the input of more data about the function; thus further field observation is required to reduce this component of error. Secondly, approximation error is reduced by reducing the grid spacing on which the interpolant is evaluated to provide the input data to the (grid-based) contouring method; thus nothing can be done to reduce this error *within* the contouring method, but it can be made arbitrarily small by refining the grid — no additional field observation is required, but the data processing workload is increased at both the interpolation and contouring stages. Thirdly, tracking error is reduced by altering the step-length control parameter within the contouring method, and can be made arbitrarily small for a fixed grid at the cost only of increasing the data processing workload within the contouring method. Obviously it is important to reduce the approximation and tracking errors to a level which allows full value to be extracted from the (commonly expensive) raw data, but pointless to reduce them further once they are swamped by the interpolation error, although tracking error is normally also chosen small enough to ensure that the contours are visually smooth.

In most mathematical and statistical contexts it is **vertical** error — for example the difference between f and z — which is of importance, although there are statistical methods (*errors-in-variables analysis*) for handling the case where there are errors or uncertainties in the values of the explanatory (independent) variables as well as in the dependent variable. **Horizontal** errors are also of importance in contouring, however, because the solution of an equation of the form (6) turns a vertical error into a horizontal one, namely the extent to which the contour line as generated deviates from where it should be (in terms of f, or of z, or of g). There is a simple approximate relationship between vertical and horizontal error in contouring, namely that the ratio of the two is the local value of the magnitude of the function gradient. Thus horizontal error can be large because vertical error is large, or because gradient is low; nothing can be done about the latter, and users of contour maps are generally well aware of the need to

interpret with caution contour lines across nearly flat regions. Only in rather special problems, such as the determination of reservoir shorelines, is horizontal error critical in itself.

Contour linkage

A good contouring method has to manipulate contours as mathematical objects, not just as marks on paper. This requires the linkage of the separate conic sections of which each contour is composed. For this linkage procedure always to work, a case analysis of the incidence of a conic on a triangle is necessary together with an association rule to orient contours and resolve 'flat-spot' ambiguities. In CONICON 3, the current version of the package, the rule used (the *psalmist's rule*) is that contours are drawn with strictly higher ground (to whatever is the lowest order available) immediately on the left as the contour is traversed. The case analysis is complete, in that it classifies and handles correctly all possible cases including all forms of degeneracy; and exact, in that it avoids numerical 'fudge factors' and also avoids assumptions about how floating-point arithmetic behaves — a small number of tolerance values are set to permit the recognition of degenerate cases without reliance on conformity to IEEE floating point standards. The parametrisation of contours and the incidence/linkage analysis depend crucially on the mathematical properties of quadratic functions and conic sections, and the code which implements the relevant algorithms in CONICON 3 is extremely intricate although quite efficient. There is no obvious way to carry out these steps of the process for functions any more complicated than piecewise quadratics. Once properly linked, contours can be annotated, drawn in fancy line styles, and even fed to polygon-fill firmware on advanced devices; it is also possible to devise cross-hatching procedures as a substitute for colour-fill on pure vector devices. Figures 2 and 3 illustrate some of the capabilities of the CONICON 3 package.

REFERENCES

Marlow, S., and Powell, M.J.D. (1976). A Fortran subroutine for plotting the part of a conic that is inside a given triangle. *UKAEA Harwell Paper AERE-8336*, HMSO, London.

Powell, M.J.D. (1974). Piecewise quadratic surface fitting for contour plotting. In *Software for Numerical Mathematics* (D.J. Evans, ed.), Chapter 14, pp. 253-271, Academic Press, London.

Powell, M.J.D., and Sabin, M.A. (1977). Piecewise quadratic approximation on triangles. *ACM Transactions on Numerical Software*, **3**, pp. 316-325.

Ritchie, S. (1978). *Representation of surfaces by finite elements.* MSc Thesis, University of Calgary.

Sibson, R., and Thomson, G.D. (1981). A seamed quadratic element for contouring. *The Computer Journal*, **24**, pp. 378-382.

Thomson, G.D. (1984). *Automatic Contouring by Piecewise Quadratic Approximation.* PhD Thesis, University of Bath.

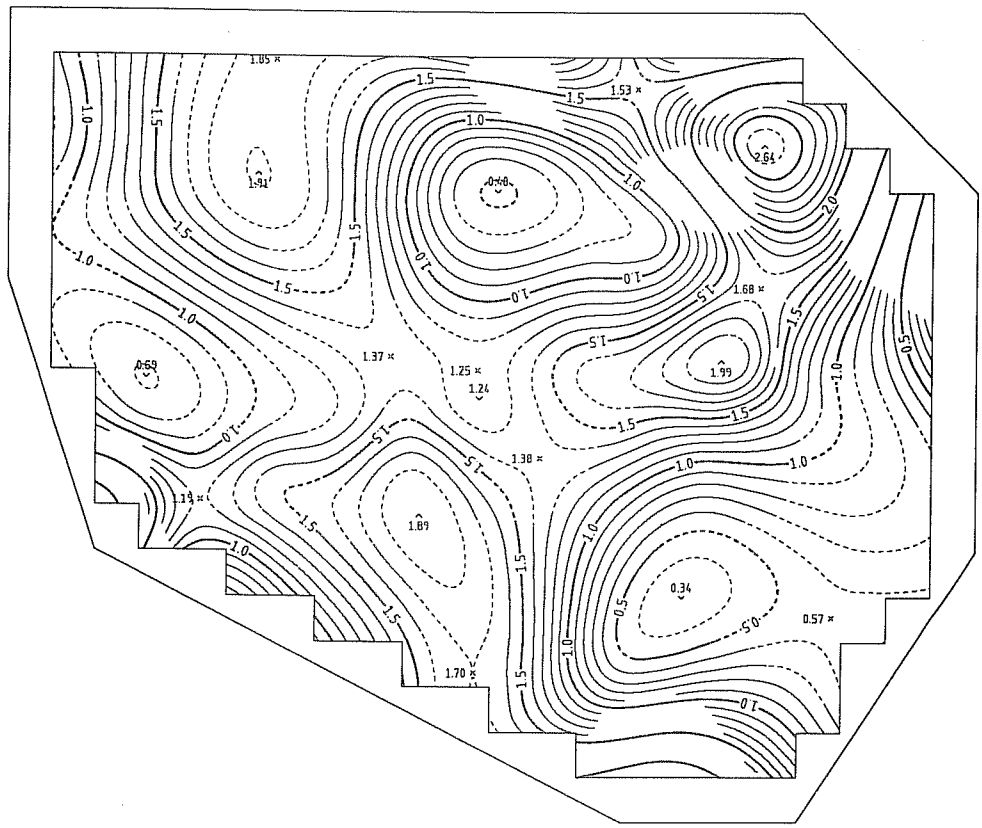

Figure 2: Thinning, low-gradient warning, and stationary points

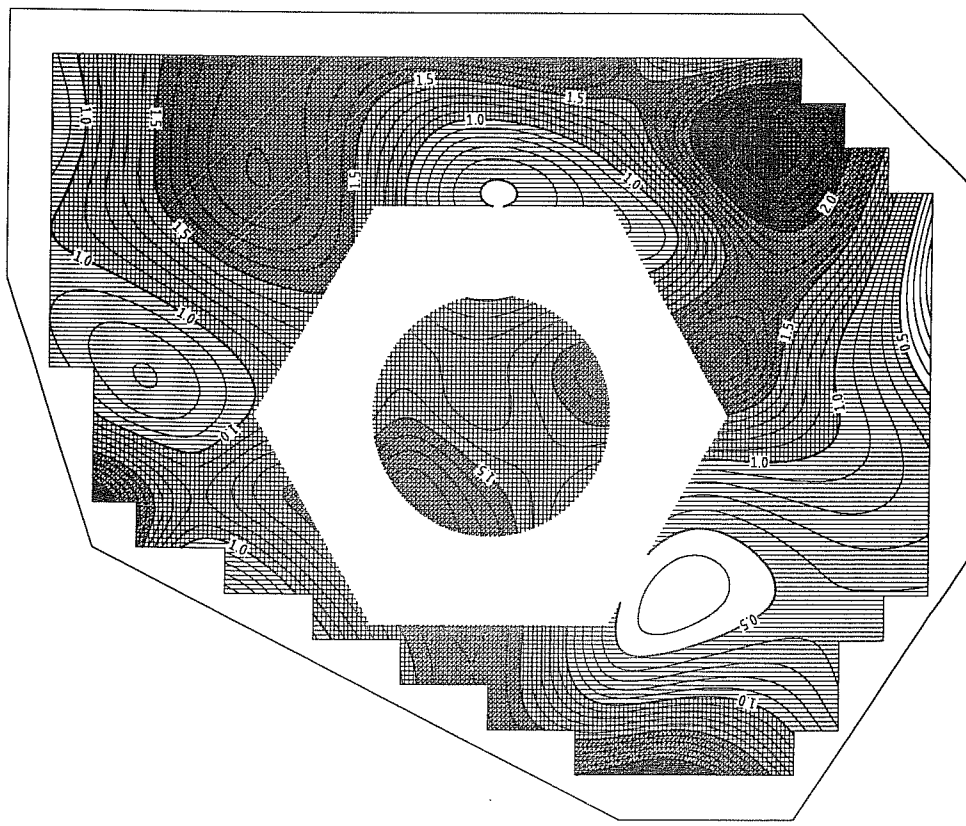

Figure 3: Stencilling to arbitrary boundary and crosshatching