A two-dimensional interpolation function for irregularly-spaced data

by DONALD SHEPARD
Harvard College
Cambridge, Massachusetts

INTRODUCTION

In many fields using empirical areal data there arises a need for interpolating from irregularly-spaced data to produce a continuous surface. These irregularly-spaced locations, hence referred to as "data points," may have diverse meanings: in meterology, weather observation stations; in geography, surveyed locations; in city and regional planning, centers of data-collection zones; in biology, observation locations. It is assumed that a unique number (such as rainfall in meteorology, or altitude in geography) is associated with each data point.

In order to display these data in some type of contour map or perspective view, to compare them with data for the same region based on other data points, or to analyze them for extremes, gradients, or other purposes, it is extremely useful, if not essential, to define a continuous function fitting the given values exactly. Interpolated values over a fine grid may then be evaluated. In using such a function it is assumed that the original data are without error, or that compensation for error will be made after interpolation.

In essence, an operational solution to the problem of two-dimensional interpolation from irregularly-spaced data points is desired. It is assumed that a finite number N of triplets (x_i, y_i, z_i) are given, where x_i, y_i are the locational coordinates of the data point D_i, and z_i is the corresponding data value. Data point locations may not be coincident. An interpolation function $z=f(x,y)$ to assign a value to any location $P(x,y)$ in the plane is sought. This two-dimensional interpolation function is to be "smooth" (continuous and once differentiable), to pass through the specified points, (i.e., $f(x_i,y_i)=z_i$), and to meet the user's intuitive expectations about the phenomenon under investigation. Furthermore, the function should be suitable for computer application at reasonable cost.

EXISTING APPROACHES

Although many solutions to related problems in two-dimensional interpolation have been in long use, interpolation functions making an exact fit for irregularly-spaced data are rare. When the data points already form a regular lattice, many solutions are possible. Among the most significant solutions for a rectangular grid are fitting a hyperbolic paraboloid to each four data points by double linear interpolation[1], fitting a polynomial to the surrounding 4, 9, 16 or 25 points using Newton's divided differences formula (discussed in[2] and[3]), or employing bicubic spline interpolation.[4] For a triangular lattice, fitting a plane to each three points is easy and effective. Downing has developed a computer contouring program which interpolates intervening points from a square lattice, making possible planar interpolation over a triangular grid.[5] If irregular data-point locations are allowed and continuity is not required, algorithms such as Tobler's double quadratic surface interpolation[6] are possible. Finally, if the interpolation function need not fit the given values exactly, trend surface fitting[7] may be appropriate to reduce distortion from possible data error.

Even within the stipulations of continuity and exact fit, some of the preceding methods for regular data locations may be extended to irregularly-spaced data, but they tend to be cumbersome or unduly arbitrary. One solution is to connect pairs of data points until the section of the plane containing the data points has been partitioned into triangles, or a mixture of quadrilaterals and triangles. Nordbeck and Bengtsson used triangles, fitting a plane to the three triplets defining each triangle.[8] Fisher used quadrilaterals in the interior of the region and triangles at the boundary.[9] He employed double linear interpolation over the quadrilaterals and planar interpolation over the triangles. These two solutions are attractive since both are computationally simple once the requisite subdivision of the domain has been carried out. They involve fitting a collection of local, easily-evaluated functions to the original data. A serious drawback is the necessity of forming a network among the points to determine how the domain for interpolation is to be partitioned. The resulting interpolation is sensitive to this partition, but no adequate means of establishing it is available. The suggestion (mentioned in[8]) that the network be chosen to minimize the sum of the perimeters of the three-and four-sided sub-domains, has been too complex to follow in practice. Even if such a subdivision were to be found, it might not be unique. The only computer program to use this partitioning method, SYMAP Version III,[9] required the user to specify the partition according to somewhat arbitrary conventions. Although the surface produced by these piecewise interpolation functions was continuous, its derivative was discontinuous at the boundaries of sub-domains.

An alternative approach, easier for the user and more elegant, would be to fit a polynomial or trigonometric function in two variables with enough coefficients so that it assumes exactly all the data values. Berezin suggests a general formula for fitting a polynomial of degree $N-1$ in x and y to N data points.[3] Though it meets all the criteria, the computations became exceedingly long with large numbers of data points. To find an interpolated value given 1000 data points, for example, would require evaluating and multiplying 999 scalar products of two-dimensional vectors.

Existing methods of two-dimensional exact interpolation are of two types: a single, global function, often of unmanageable complexity; or a suitably-defined collection of simple, local functions which match appropriately at their boundaries. The function developed in this paper is of the latter type, so constructed that the sub-domain for each local function is automatically defined, and the function is continuously differentiable even at the junctions of local functions.

AN APPROACH USING WEIGHTED AVERAGES

It became clear that a surface based on a weighted average of the values at the data points, where the weighting was a function of the distances to those points, would satisfy the criteria. An initial inverse distance function was tested; in it the value at any point P in the plane was a weighted average of the values at the data points D_i.

Let z_i be the value at data point D_i, and $d[P,D_i]$ be the Cartesian distance between P and D . Where the reference point P is understood, $d[P,D_i]$ will be shortened to d_i. The interpolated value at P using this first interpolation function is:

$$f_1(P)=\left[\begin{array}{ll} [\sum_{i=1}^{N}(d_i)^{-u}z_i] / [\sum_{i=1}^{N} (d_i)^{-u}] & \textit{if } d_i \neq 0 \ \textit{ for all } D_i \\ & \qquad (u>o) \\ z_i & \textit{if } d_i = 0 \ \textit{ for some } D_i. \end{array}\right.$$

Notice that as P approaches a data point D_i, $d_i \to 0$, and the i^{th} terms in the numerator and denominator exceed all bounds while the other terms remain bounded. Therefore $\lim_{P\to D_i} f_1(P) = z_i$ as desired, and the function $f_1(P)$ is continuous.

Choice of Exponent

Using coordinates $P(x,y)$ and $D_i(x_i,y_i)$, partial differentiation yields:

$$f_{1x}(x,y) = \frac{\sum_{i=1}^{N} \sum_{\substack{j=1 \\ i\neq j}}^{N} (d_i)^{-u-2} (d_j)^{-u} (x-x_i)z_i(z_i-z_j)}{[\sum_{i=1}^{N} (d_i)^{-u}]^2}$$

Replacing $(x-x_i)$ by $(y-y_i)$ in the above equation gives the corresponding expression for $f_{1y}(x,y)$. The partial derivatives of f_1 exist at all points. For P in a neighborhood of D_i, f_{1x} behaves as $(x-x_i)(d_i)^{u-2}$, or as $\pm(d_i)^{u-1}$. Thus for $u>1$, f_{1x} approaches 0 with $x-x_i$ (or with d_i) as $P \to D_1$. For $u=1$, both left and right-side partial derivatives exist; these in general are non-zero and opposite in sign. For $u<1$, no derivative exists. Thus the requirement that the interpolation function be differentiable necessitates that the exponent exceed 1. Empirical tests showed that higher exponents $(u>2)$ tend to make the surface relatively flat near all data points, with very steep gradients over small intervals between data points. Lower exponents produce a surface relatively flat, with short blips to attain the proper values at data points. An exponent of $u=2$ not only gives seemingly satisfactory empirical results for purposes of general surface mapping and description, but also presents the easiest calculation. In Cartesian coordinates $(d_i)^{-2}=1 \ / \ [(x-x_i)^2+(y-y_i)^2]$.

Shortcomings of Pure Inverse-Distance Weighting

Though the above method was sufficiently simple and general to be quite attractive, it did have several shortcomings.

1) When the number of data points is large, thе calculation of $z=f_1(P)$ becomes proportionately longer. Eventually the method will become inefficient or impractical.
2) Only the distances to P from the data points D_i, and not the direction, are considered. Therefore the following two configurations of co-linear points, for example, would yield identical interpolated values at P:

(*a*) x – – – – x – – – – + – – – – – – – – – – x
D_1 D_2 P D_3

(*b*) x – – – – – – – – – – + – – – – x – – – – x
D_1 P D_2 D_3

If the method is to be intuitively reasonable, the value at P in configuration (a) should be closer to the value at D_3 than in configuration (b) and conversely for the value at D_1, because an intervening data point should be expected to screen the effect of the more distant point.

3) The zero directional derivatives obtained at every data point D_i represent an arbitrary and undesirable constraint on the interpolated surface.
4) Computational error becomes significant in the neighborhood of points D_i, as the predominant term results from the difference of two almost equal numbers.

IMPROVING THE WEIGHTING FUNCTION

As a solution the basic weighting function discussed previously was retained, subject to the following modifications and correction terms.

Selecting Nearby Points

Since the above weighting function means that only nearby data points are significant in computing any interpolated value, a considerable saving in computation could be effected by eliminating calculations with distant data points. As their inclusion tended to make the surface of higher order, extra points of inflection could also be eliminated by interpolating from nearby points only. To select nearby points, either (1) an arbitrary distance criterion (all data points within some radius r of the point P), or (2) an arbitrary number criterion (the nearest n data points) could be employed. The former choice, though computationally easier, allowed the possibilities that no data points, or an unmanageably large number of data points, might be found within the radius r. The latter choice required a more detailed searching and ranking procedure for data points, and presupposed that a single number of interpolating points was best, regardless of the relative location and spacing of the points. A combination of the two criteria combined their advantages. In order that the interpolation work reasonably if the data points were gridded, a minimum of four data points was chosen. A maximum of ten was established to limit the complexity and amount of computation required. Further, an initial search radius r is established according to the overall density of data points. If N is the total number of data points and A is the area of the largest polygon enclosed by the data points, r is defined such that seven data points are included, on the average, in a circle of radius r. That is:

$$\pi r^2 = 7\left(\frac{A}{N}\right).$$

To select and weigh interpolating data points, a collection C' of data points near P and final search radius r' are defined below. First let

$C_P = \{D_i | d_i \leq r\}$ and $n(C_P) =$ the number of elements in C_P. Next consider an ordering of the D_i by increasing distance from P. The subscripts i_j are defined such that $0 \leq d_{i_1} \leq d_{i_2} \leq \ldots \leq d_{i_N}$.

Now define

$$C_P^n = \{D_{i_1}, D_{i_2} \ldots, D_{i_n}\} \qquad (n \leq N),$$

and

$$r'\ (C_P^n) = min\{d_{i_j} \mid D_{i_j} \notin C_P^n\} = d_{i_{n+1}}$$

Finally, let

$$C_P' = \begin{cases} C_P^4 & if\ 0 \leq n(C_P) \leq 4 \\ C_P & if\ 4 < n(C_P) \leq 10 \\ C_P^{10} & if\ 10 < n(C_P) \end{cases}$$

$$and\ r_P' = \begin{cases} r'\ (C_P^4) & if\ n(C) \leq 4 \\ r & if\ 4 < n(C) \leq 10. \\ r'(C_P^{10}) & if\ 10 < n(C) \end{cases}$$

Unless indicated otherwise, the reference point P is to be understood and will be omitted from the notation.

In practice, the distance between successive points P at which the function is to be evaluated is relatively small compared to the average data-point spacing. Most pairs C' and r' can be operationally defined easily in terms of the pair for the preceding point.

For a reference point P, new weighting factors $s_i = s(d_i)$ may be defined from the following function:

$$s(d) = \begin{cases} \frac{1}{d} & if\ 0 < d \leq \frac{r'}{3} \\ \frac{27}{4r'} \quad \left(\frac{d}{r'} - 1\right)^2 & if \frac{r'}{3} < d \leq r' \\ 0 & if\ r' < d \end{cases}$$

This function is defined to be continuously differentiable over all $d > 0$ such that $s(d) = 0$ for $d \geq r'$. Since C' is defined such that $d_i \leq r'$ for every $D_i \epsilon C'$, points outside of C' have a zero weighting and may be excluded without effect. Thus an interpolation function with behavior similar to the original function, but much easier to use for large numbers of data points, is:

$$f_2(P) = \begin{cases} [\sum_{D_1 \epsilon C'} (s_i)^2 z_i] \ / \ [\sum_{D_i \epsilon C'} (s_i)^2] & if\ d_i \neq 0\ for\ all\ D_i \\ z_i & if\ d_i = 0\ for\ some\ D_i \end{cases}.$$

Including Direction

In order to improve the interpolation it was clear that a direction factor, in addition to a distance factor, was needed in defining weightings. Intuitively this represented the "shadowing" of the influence of a data point from P by a nearer one in the same direction. A directional weighting term for each data point D_i near P was defined by

$$t_i = \left[\sum_{D_j \epsilon C'} s_j[1-cos(D_iPD_j)]\right] \Big/ \left[\sum_{D_j \epsilon C'} s_j\right].$$

The cosine of the angle D_iPD_j can be evaluated by the inner product as

$$[(x-x_i)(x-x_j) + (y-y_i)(y-y_j)] \, / \, d_id_j.$$

Since for all angles θ, $-1 \leq cos(\theta) \leq 1$, it follows that $0 \leq t_i < 2$. If other data points D_j are in roughly the same direction from P as D_i, then the (1-cos) factors are near zero and t_i is near zero. If, on the other hand, the other data points are roughly opposite P from D_i, then the (1-cos) factors and hence t_i are near 2.

The cosine function was used as the measure of direction both because of appropriateness and computational ease. The distance weighting factor s_j is included in the numerator and denominator because points near P should be more important in shadowing than distant points. Counting direction, a new weighting function $w_i = (s_i)^2 \times (1+t_i)$ may be defined. The latest version of the interpolation function is:

$$f_3(P) = \begin{cases} [\sum_{D_i \epsilon C'} w_iz_i] \, / \, [\sum_{D_i \epsilon C'} w_i] & if\ d_i \neq 0\ for\ all\ D_i \\ z_i & if\ d_i = 0\ for\ some\ D_i \end{cases}$$

Determining Slope

Despite the foregoing modifications to the weighting function, it should be recalled that for P sufficiently near some data point D_i (d_i is small), s_i will equal d_i^{-1} and w_i will vary as d_i^{-2}, as in the original interpolation function. As before, the interpolated surface will still have zero gradient at every D_i. To correct this undesirable property, increments were added to the function values at nearby data points so that the interpolated surface would achieve desired partial derivatives at D_i.

First, constants A_i and B_i are determined for each data point D_i representing the desired slope in the x and y directions at D_i. A_i and B_i are weighted averages of divided differences of z about D_i. Let $C_i'' = C'_{D_i} - \{D_i\}$. Then

$$A_i = \frac{\sum_{D_j \epsilon C_i''} w_j \frac{(z_j-z_i)(x_j-x_i)}{(d[D_j, D_i])^2}}{\sum_{D_j \epsilon C_i''} w_j}$$

and

$$B_i = \frac{\sum_{D_j \epsilon C_i''} w_j \frac{(z_j-z_i)(y_j-y_i)}{(d[D_j, D_i])^2}}{\sum_{D_j \epsilon C_i''} w_j}$$

Next, a parameter v with the dimension of distance is defined based on the total range of the z_i and the desired slopes:

$$v = 0.1[max\{z_i\} - min\{z_i\}] \, / \, [max\{(A_i^2+B_i^2)\}]^{\frac{1}{2}}.$$

This parameter will bound the maximum effect the slope terms may have on the final interpolated value of z. The factor of 0.1 represents an arbitrary choice for applications to contour mapping with a contour interval of one-fifth the range of z_i. The effect of slope terms will be limited to one-half the contour interval. To include the effect of slope in interpolating a value at $P(x,y)$ an increment Δz_i is computed for each $D_i \epsilon C'_P$ as a function of P.

$$\Delta z_i = [A_i(x-x_i) + B_i(y-y_i)][\frac{v}{v+d_i}]$$

Let P move along a straight line away from D_i and consider the effect on Δz_i. The factor of $\frac{v}{v+d_i}$ was inserted because it decreases monotonically from 1 to 0 as d_i increases from 0 to ∞, behaving like d_i^{-1} for large d_i. Thus, regardless of d_i, for any data point D_i

$$|\Delta z_i| \leq 0.1[max\{z_i\} - min\{z_i\}]$$

as desired. Also, evaluating the partial derivatives at D_i gives

$$\frac{\partial}{\partial x}(\Delta z_i)\Bigg|_{\substack{x=x_i \\ y=y_i}} = A_i \quad and \quad \frac{\partial}{\partial y}(\Delta z_i)\Bigg|_{\substack{x=x_i \\ y=y_i}} = B_i.$$

Thus the interpolation function defined below will give the desired partial derivatives A_i and B_i at D_i and behaves otherwise like f_3.

$$f_4(P) = \left[\begin{array}{ll} [\sum\limits_{D_i\epsilon C'} w_i(z_i+\Delta z_i)] \;/\; [\sum\limits_{D_i\epsilon C'} w_i) & \textit{if } d_i\neq 0 \textit{ for all } D_i\epsilon C' \\ z_i & \textit{if } d_i=0 \textit{ for some } D_i\epsilon C' \end{array}\right.$$

Reducing Computational Error

When the foregoing interpolation algorithm is used with a digital computer, rounding and truncation can cause considerable inaccuracy when P is very close to D_i. Since $\lim\limits_{P\to D_i} f(P)=z_i$, the computational problem is avoided by establishing some ϵ, depending on the computer's precision, and defining $f(P)=z_i$ within an ϵ–neighborhood of D_i. If several data points occur within an ϵ– neighborhood $N\epsilon(P)$ *of* P, then their values are averaged.

Although the interpolation function is technically discontinuous, the discontinuity is of the same order as machine error times the magnitude of the gradient. Error created by this ϵ - neighborhood is no greater than that caused elsewhere by machine imprecision. In practice the problem is insignificant because the function is evaluated at discrete points only. The final interpolation function is thus given by:

$$f(P) = \left[\begin{array}{ll} [\sum\limits_{D_i\epsilon C'} w_i(z_i+\Delta z_i)] \;/\; [\sum\limits_{D_i\epsilon C'} w_i)] & \textit{if } d_i>\epsilon \textit{ for all } D_i \\ [\sum\limits_{D_i\epsilon N\epsilon(P)} z_i] \;/\; [\sum\limits_{D_i\epsilon N\epsilon(P)} 1] & \textit{if } d_i<\epsilon \textit{ for some } D_i \end{array}\right.$$

Simulating the Effect of Barriers

When the foregoing interpolation algorithm is used in city and regional planning, where the data points represent the centers of data collection areas (such as census tracts), it is assumed that whenever two areas are adjacent, there is some logical relationship between them. If some physical barrier such as a river, railroad or expressway separates the areas, the logical relation may be attenuated. Through the inclusion of barriers a user may specify discontinuities in the metric space in which d_i is calculated to simulate this attenuation. Because of the distance-dependent interpolation this change is particularly easy.

Suppose a "detour" of length $b[P,D_i]$ perpendicular to the line between P and D_i were required to travel around the barrier between the two points. The quantity $b[P,D_i]$ is considered the strength of the barrier and an effective distance d_i' is given by

$$d_i' = \{(d[P,D_i])^2 + (b[P,D_i])^2\}^{\frac{1}{2}}$$

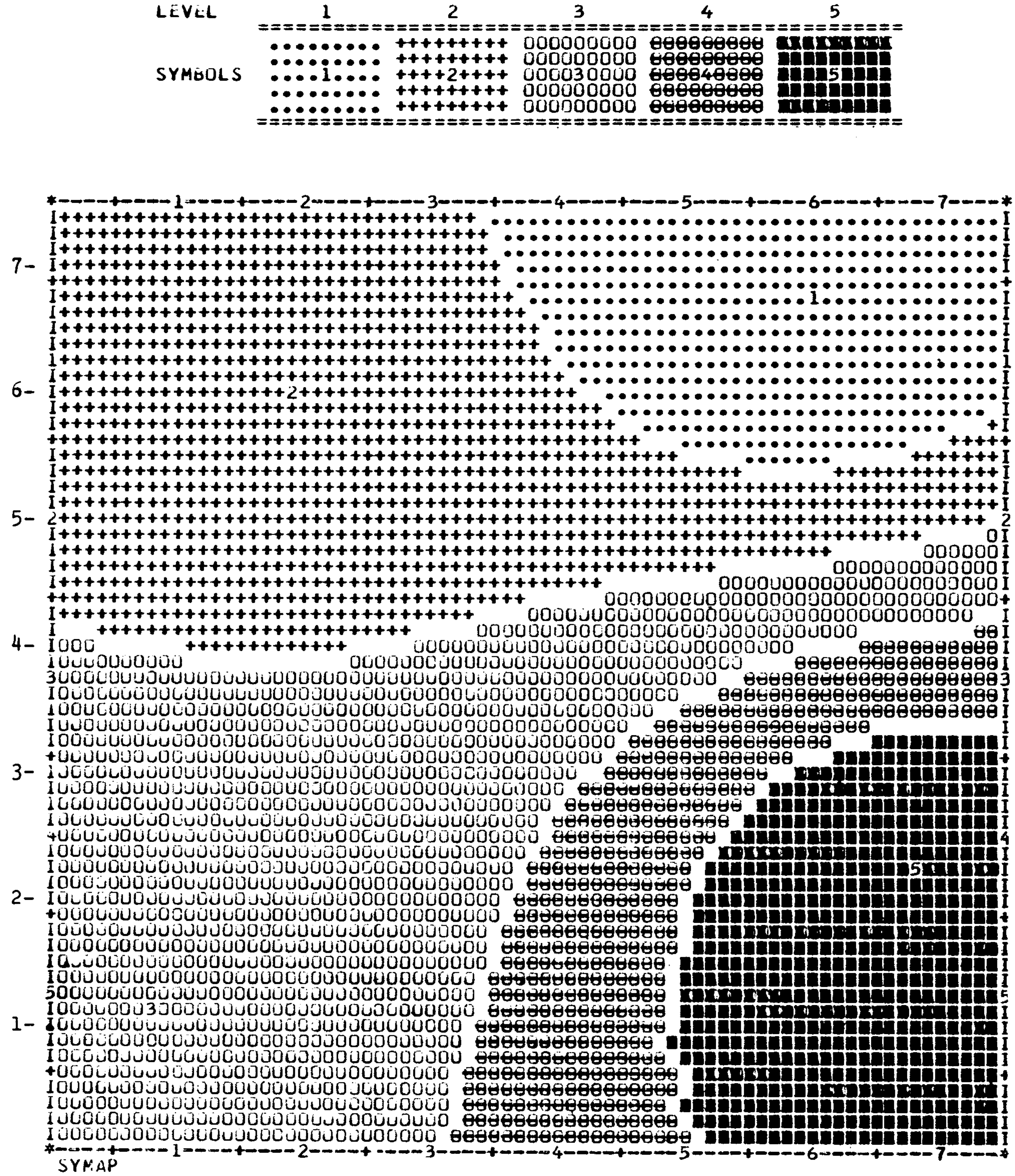
This definition is general so that if no barrier separates D_i and P, $b[P,D_i] = 0$ and $d_i' = d_i$. When barriers are included d_i' should replace d_i in all preceding expressions. In general the inclusion of barriers will, as expected, result in the selection of a different set of "nearby" data points for interpolation, and in different weightings and slopes being established. Because of the discontinuity in effective distance as P crosses a barrier, the interpolated surface will be discontinuous at a barrier.

Properties of f(P)

The interpolation function $f(P)$ defined above meets all the explicit requirements initially postulated. The surface defined by $f(P)$ is continuously differentiable everywhere except at barriers where discontinuities are deliberately specified. The surface assumes the required value at all data points. On a digital computer the evaluation of interpolated values is reasonably straightforward. Partial deprivatives at data points are based on a notion of what they "should" be, an average of first divided differences from nearby known values. The final interpolation function may have extrema anywhere, not just at data points. The range of the interpolation function is up to 10% greater at either end than the range of the initial values z_i. The interpolation function is defined at every point in the plane. It thus has the ability to extrapolate naturally outside the smallest polygon whose closure contains all the data points. At large distances from this polygon the interpolated value will be approximately the average of the values at the four nearest data points.

APPLICATIONS

The principal application to date of the foregoing interpolation function has been its programming in FORTRAN IV and its incorporation into a mapping program, SYMAP Versions IV and V.[10] Following is an example of a contour map produced on the IBM 7094 in 0.2 minutes using SYMAP Version V. Four data points were specified with the coordinates and values below. The range of the original data point values has been divided into five equal levels, and the lowest and highest levels extended to include all interpolated values. The special symbols were produced on a high-speed printer by superimposing standard alphanumeric characters.



DEMONSTRATION OF INTERPOLATION FROM FOUR IRREGULARLY-SPACED DATA POINTS. NUMBERS SHOW DATA POINTS LOCATIONS AND LEVELS. CONTOUR MAP PRODUCED BY SYMAP, VERSION 5 AT HARVARD UNIVERSITY.

DEMONSTRATION OF INTERPOLATION FROM FOUR IRREGULARLY-SPACED DATA POINTS

Location and Value at Data Points

data point number	*locational coordinates*		*data value*	*level*
i	x_i	y_i	z_i	
1	6.00	6.75	0.0	1
2	6.80	2.25	5.0	5
3	0.80	1.13	2.5	3
4	1.90	6.00	1.5	2

Value Range Corresponding to Each Level of Symbolism

level	*range*
1	[—0.5, 1.0)
2	[1.0, 2.0)
3	[2.0, 3.0)
4	[3.0, 4.0)
5	[4.0, 5.5]

(Numbers on map indicate data-point levels
and locations.)

Computing time for map-making under SYMAP Version V program is roughly proportional to the map area, and rises gradually as the number of data points increases. Preparing a map 10" by 10" square on the IBM 7094 requires about .35 minute with four data points, .70 minute with 200 data points, and 1.3 minutes with 600 data points. Version V has been adapted to run also on the IBM 360/50.[10]

To prepare the map in Figure 1, as is normally the case in SYMAP Version IV and V, two levels of interpolation were actually used. First, every third value across, and every second value down was computed by the preceding algorithm. Then double linear interpolation was used at the intervening locations. Barely altering the appearance of the map, this two-level procedure reduces computing time by 70%. Although Versions IV and V are still slightly slower than Version III, their flexibility and convenience are compensating advantages.

CONCLUSIONS

The foregoing algorithm is generalizable to interpolation from data points in three-or higher-dimensional space with few modifications. In higher-dimensional space, the Cartesian metric replaces the two-dimensional distance. Also, expressions involving scalar products or slope increments require another term for each additional dimension. Finally, the radius r is evaluated from the volume of the minimal polyhedron, instead of the area of the minimal polygon, containing all the data points.

In general no method of interpolation is precisely accurate except at the points through which the function has been fitted. The foregoing function was developed with particular consideration for variables such as population density, housing conditions, and others from the fields of planning and geography. For such work, little is known about the exact properties of the variable involved; thus the investigator's intuitive judgment, in the absence of an objective measure, may represent an acceptable evaluation of the appropriateness of any interpolation function used. Though the foregoing interpolation function contains arbitrary choices, it has at least provided a workable tool assisting in the analysis of areal data via computer mapping. It is hoped that the method will aid in the derivation of more refined two- and higher-dimensional interpolation functions.

REFERENCES

1 P SWITZER C M MOHR R E HEITMAN
Statistical analyses of ocean terrain and contour plotting procedures
Project Trident Report 1440464 Arthur D Little Inc April 1964

2 J F STEFFENSEN
Interpolation
Williams and Wilkins Baltimore 1927 Chap 19

3 I S BEREZIN N P ZHIDKOV
Computing methods
Addison-Wesley Publishing Company Inc Reading Mass 1965 Vol I Chap 2

4 C deBOOR
Bicubic spline interpolation
J Math and Phys 41 212-218 1962

5 J A DOWNING
The automatic construction of contour plots with applications to numerical analysis research
Defense Documentation Center Report AD 649811 January 1966

6 W R TOBLER
An interpolation algorithm for geographical data
University of Michigan undated

7 W C KRUMBEIN
Trend surface analysis of contour-type maps with irregular control point spacing
J Geophys Re 64 823-834 1959

8 B E BENGTSSON S NORDBECK
Construction of isarithms and isarithmic maps by computers
University of Lund Sweden BIT 4 87-105 1964

9 A H SCHMIDT
SYMAP A user's manual
. Tri-County Regional Planning Commission Lansing Michigan May 1966

10 *User's referemce manual for synagraphic computer mapping 'SYMAP' version V*
Laboratory for Computer Graphics Harvard Univ June 1968