TOOLS FOR COMPUTER GRAPHICS APPLICATIONS

R.L. Phillips
The University of Michigan

ABSTRACT

Ten years of extensive research in computer graphics has produced a collection of basic algorithms and procedures whose utility spans many disciplines; they can be regarded as tools. These tools are described in terms of their fundamental aspects, implementations, applications, and availability. Programs which are discussed include basic data plotting, curve smoothing, and depiction of 3-dimensional surfaces. As an aid to potential users of these tools, particular attention is given to discussing their availability and, where applicable, their cost.

INTRODUCTION

Direct computer-produced graphical output, once considered a luxury, is becoming relatively commonplace. The availability of low cost plotters and display terminals is largely responsible for the trend. Increased usage of computer graphics has given rise to a need for application-oriented, non-research, graphical software. It is the goal of this paper to point out and discuss such software. The hope is that duplication of effort can be avoided and that the use of non-general, low quality graphic software will be discouraged.

The software to be described here is of such generality, widespread utility and ready availability so as to be classified as a tool — a tool to be employed to the user's advantage and not encumber him in his work. The paper, then, is a survey of sorts, but a rather limited one. We shall not discuss any software in the research stages, nor any software that is not readily available. Moreover, since device and system independence are also valued attributes, vendor supplied packages, no matter how good, will not be discussed. In what follows we shall discuss basic data presentation techniques, both for two and three dimensions. Then certain data processing and enhancement methods will be described (e.g. clipping and shading).

DATA PRESENTATION (2-Dimensional)

Overview

One of the most useful applications of computer graphics is data presentation, or graphing of data on an axis system. The two-dimensional graph is the most common qualitative and quantitative method of representing relations among data. Several software tools have been developed to facilitate the data presentation process, ranging from automatic axis scale determination to passing smooth curves through the data points.

Automatic Scale Generation

If we impose the reasonable restriction that the scale to be determined is "nice" or readable, the process of automatic scale selection is not at all trivial. A scale obtained by dividing the span of a variable by the corresponding axis length will almost never satisfy this restriction. "Nice" scale intervals will never have values like 0.125 or 1.1 but rather will be more usable values like 0.1 or 2.0. Even where a "nice" interval of, say 5, is used, corresponding axis labels like -1, 4, 9, etc., would not qualify as a readable scale. Tastes may differ on readability but there are some fundamental good practices that should be followed in scale selection.

Several algorithms have been published for automatic production of readable scales. They all produce acceptable results and we shall describe only one in some detail, the algorithm due to Lewart (ref. 1). The rules are simple — the scale intervals must be the product of an integer power of ten and one of a set of "nice" coefficients. Certainly this set should consist of at least 1, 2, and 5 but individual taste may allow perhaps 4 and 8 to be included. The next requirement is that axis labels must be integer multiples of the scale intervals. These requirements result in an axis whose extremes will embrace the data it represents. When the algorithm is applied to each axis, the resulting graph is "efficient" — the data come as close as possible to filling the available plotting area. Figure 1 shows the results of applying this algorithm to a situation where a data zoom is performed on the original graph. The algorithm, being general, can adapt to any situation. Comparable algorithms are described in references 2 and 3.

Labelling with Software Characters

A common goal in the preparation of computer-produced data representations is to make them report-ready, i.e. no subsequent draftsman work should be required. If this goal is to be attained, all text that appears on the graph should be of high quality. The usual stick figure software characters usually will not suffice for this; something more elegant is desired. The nonpareil of all software character fonts are those developed by Hershey (ref. 4). Complete font digitizations as well as several sophisticated typographic subroutines are available for the cost of mailing a tape. A sample of textual output using Hershey's fonts is shown in figure 2; nothing more need be said.

Curve Fitting

We shall make a distinction now between <u>curve fitting</u>, where one attempts to pass a smooth curve through all data points, and <u>curve smoothing</u>, where a smooth curve is passed through a neighborhood of all points according to some least-squares criterion. The latter process is useful where the data is statistical or imprecisely known; this will be discussed in the next section.

In a case where data are precisely known and no smoothing is required, one often wishes to join the data points with a continuous curve. The process is trivial, of course, if many intermediate data points can be calculated so that joining them by a straight line produces a sufficiently smooth curve. This is often not feasible, however. It may be that the data are derived from an expensive computation, as in the solution of a set of nonlinear partial differential equations, or the original computational scheme is not available, such as data obtained from a table of thermophysical properties.

Without benefit of prior experience, one is tempted to try to produce a smooth curve by using either a global high order polynomial fit to all data points or to produce intermediate points by second or higher order interpolation. Neither approach is ever very successful; unwanted oscillations usually result. An excellent overview of these problems is found in Akima's paper (ref. 5) where he proposes a new scheme for curve fitting. His contribution was to devise a new way of locally computing slopes at each data point and using these slopes to construct a series of cubic polynomials, continuous at each join. The program that implements this algorithm appears in reference 6. The author has not been able to find a situation where Akima's method fails. It is inexpensive as well as accurate. Figure 3 demonstrates its capabilities.

In some instances Akima's method produces a curve of greater curvature than may be desired, especially where the original data is sparse. In this case one should consider the tension splines of Cline (ref. 7). Cline develops a rigorous theory for these curves but pragmatically we can imagine them to be flexible wires which are passed through a series of eyelets (the data points) and made as taut as one wishes by pulling on either end. The amount of tension is under control of the user, which is at the same time an advantage and drawback of the approach. It is not clear *a priori* what value of tension is appropriate. Figure 4 shows Cline's method using different amounts of tension on the same set of data.

Curve Smoothing

For data that is imprecisely known or statistical in nature, a curve-fitting approach as described above would be inappropriate. Rather, we wish to obtain some smooth, mean curve that passes through the neighborhood of the data according to some least-squares criterion. Often, the curve obtained is to be used for further computation such as differentiation or interpolation so it is important to perform the smoothing accurately. Variations that are statistically significant must be accounted for; thus, the method must be capable of recognizing trends. There are, by the way, many nonlinear regression techniques that have been developed in the statistical literature (refs. 8 and 9) that treat this problem, but to use them one must usually make some assumptions about the functional form of the data. The method of smoothing splines, however, requires no such assumptions and it is this technique we discuss. Here a series of spline curves are computed which join continuously at *knots*. Knots may or may not coincide with data points; the number of them and their position are selected by the program so as to produce a best fit, subject to a least-squares constraint. The user can supply weighting factors to the original data so that outliers can be eliminated from the smoothing process.

Two smoothing spline algorithms have been published in the literature. The method of Powell (ref. 10) requires somewhat more user judgment than one would like but seems to produce good results. Lyche and Schumaker (ref. 11) have described a method that is based upon local procedures, but it is published in Algol and involves a recursive procedure. Thus, the program cannot be easily transliterated into FORTRAN. Results typical of Powell's algorithms are shown in figure 5. Details on a third smoothing spline algorithm have not been published but the routine that implements it is available from Kahaner (ref. 12). This routine is noteworthy because it allows the user to apply certain boundary conditions to the resulting smooth curve. The other algorithms do not allow this, a fact which may sometimes prove objectionable.

DATA PRESENTATION (3-Dimensional)

Overview

Often one wishes to display bivariate data in the form of a surface, a projection of the three-dimensional representation of the data. While this representation is seldom of any quantitative use, it can provide valuable insight into the behavior of complex datasets. Such a representation is shown in figure 6 where the transfer function of an underwater sound signal is represented (ref. 13). The steps required to obtain a plot such as this can be difficult, depending upon the original form of the data. All possibilities will be discussed below.

Interpolation on a Regular Grid

We are imagining a dataset, functional or tabular, Z(X,Y), where Z is some altitude or third dimension associated with every coordinate pair X,Y. If the data happen to have been derived on a regular lattice, that is, Z is known at all points on a specified X-Y array, intermediate points can be obtained fairly easily. It is not even necessary that the lattice spacing be the same in the X and Y directions; it simply must be regular. The production of intermediate points with the aim of plotting a smooth surface can be approached as a simple bivariate interpolation problem. Unless the function Z(X,Y) is very benign, however, straightforward interpolation schemes produce unreal values in the vicinity of strong local variations. The most successful and generally applicable algorithm for regular grid interpolation is due to Akima (ref. 14). The method is, in fact, the bivariate analog of the successful univariate scheme described in reference 6. The program has a simple interpolation entry point for deriving values from a bivariate table, and a smooth surface mode, where a dense array of interpolated points are returned for subsequent plotting. The author has used these routines in many situations, always with good results.

Interpolation from Scattered Observations

A more realistic case than that of the above, is where the dataset consists of a table of Z values known only at irregular and arbitrarily spaced X-Y coordinates. Most spatially distributed geographic data is in this category, as is experimentally derived bivariate data. The process of interpolating from scattered observations to produce a regular grid is much more challenging than the corresponding problem for regular data; over 100 papers have been published on the subject over the last 20 years. The problem stems from choosing an interpolation function that will not bias the data thus derived. Two interesting and successful solutions to this problem have recently appeared. One is due to Akima (ref. 15), who we have referenced twice already. He again extends his "local procedure" scheme to handle the case of irregularly spaced initial data. The results are in excellent agreement with test data presented in his paper.

Tobler (ref. 16) has recently produced another approach to this problem, which amounts to an iterative solution of the biharmonic equation in the vicinity of each data point. His program produces results equal to those of Akima, using the same data.

Surface Plotting

Once the dataset has been regularized, one can proceed to produce a plot of the surface it describes. Any method that is to be acceptable for our purposes must satisfy three criteria:

a) user-specified perspective projections of the surface must be obtainable,

b) hidden lines, e.g., the back of the surface must be eliminated,

c) one should be able to display the surface as viewed from any orientation, including from below.

There are dozens of surface plotting packages but only a few satisfy all these criteria. One that does is due to Williamson (ref. 17). It is acceptable in all three of the above respects but it might be criticized for its lack of generality. It is very much plotter oriented, expressing size variables in terms of inches rather than abstract user units. Another system which is acceptable in all respects was developed by Wright (ref. 18) and forms part of the impressive NCAR graphics package (ref. 19). Wright's program has many options for representing a surface, including cross-hatching and the production of stereo pairs. Figure 7 is an example of a surface produced by Wright's program.

DATA PROCESSING AND ENHANCEMENT

Shading and Cross-Hatching

It is often the case that one wishes to automatically shade or cross-hatch a general two-dimensional polygon. This capability is frequently required for architectural applications, engineering drawings, and thematic cartography. The task is, given an n-sided simply connected polygon with no restrictions on concavity or convexity, find the intersection of a family of shading lines with the boundary of the polygon. The lines are then drawn with the proper angle and spacing. For cross-hatching this process is repeated for a different orientation, and perhaps spacing. A general shading routine should also permit variable spacing between shading lines so that arbitrary and unusual patterns can be obtained. Finally, it is desirable to be able to shade a multiply connected region by specifying invisible, coincident cut points that join inner and outer boundaries of a polygon.

All of these desirable properties are displayed in figure 8, which was produced by an algorithm originally suggested by Dwyer (ref. 20) and implemented by Phillips (ref. 21). The algorithm uses vector algebra for the computation of shading line intersections, an operation that is simulated in reference 21 by the complex arithmetic features of FORTRAN. A more elaborate example of shading is shown in figure 9, a thematic map showing the location of water polluting industries in New England. Another shading program has been developed by Ison (ref. 22) which is capable of complex patterns such as bricks, soil patterns, etc. This package, however, seems unnecessarily oriented toward the digital plotter as an output device.

Windowing and Shielding

The process of methodically preventing some part of a graphical display from being plotted is known as <u>clipping</u>. This is often done when the user narrows his field of view in his coordinate space, e.g. zooming in for more detail, and the part of the picture falling outside that area is not to be seen. The boundaries of the narrowed field of view is called a <u>viewport</u> and can generally be formed by any polygon. Usually, however, the viewport is simply a rectangle, making the process of clipping straightforward (refs. 23 and 24). The most general case involves any simply connected polygon, concave or convex. Moreover, the term <u>window</u> implies that the portion of a picture <u>inside</u> the viewport is to be seen, while the viewport acts as a <u>shield</u> if the picture <u>outside</u> it is to be visible. Behler and Zajac (ref. 25) have published an algorithm for treating this general case. The problem differs from the one of general shading in that it does not deal with a family of lines having common characteristics; here every line is a special case. An example of polygonal windowing is shown in figure 10. There the polygon is the lower peninsula of Michigan consisting of 370 points, which windows contour lines that have been computed on a rectangular grid that is much larger than the polygon.

SUMMARY

A reader may find fault with this limited survey for having omitted several of his favorite graphics routines; this is inevitable. I have endeavored to discuss all packages of which I am aware (one could do no more) <u>and</u> with the important stipulations that the software is of proven utility, it can easily be installed on most machines, it is available (from the sources referenced), and the cost, if any, is nominal. Naturally, the author welcomes any revelations of other software that satisfies these constraints.

REFERENCES

1. Lewart, C.R., Algorithm 463: Algorithms SCALE1, SCALE2, and SCALE3 for Determination of Scales on Computer Generated Plots. Comm. Acm, 16, October 1973, pp. 639-640.

2. Thayer, R.P. and Storer, R.F., Algorithm AS21: Scale Selection for Computer Plots. Applied Statistics, 18, 1969, pp. 206-208.

3. Herro, J.J., Program 00546B: Optimum Scale for a Graph. HP-65 User's Library, Hewlett-Packard Co., Cupertino, Cal., 1974.

4. Hershey, A.V.: FORTRAN IV Programming for Cartography and Typography. NWL Technical Report TR-2339, U.S. Naval Weapons Laboratory, Dahlgren, Va., Sept. 1969.

5. Akima, H.: A New Method of Interpolation and Smooth Curve Fitting Based Upon Local Procedures. J. ACM, 17, October 1970, pp. 589-602.

6. Akima, H., Algorithm 433: Interpolation and Smooth Curve Fitting Based on Local Procedures. Comm. ACM, 15, October 1972, pp. 914-918.

7. Cline, A.K.: Scalar- and Planar-Valued Curve Fitting Using Splines under Tension. And Algorithm 476: Six Subprograms for Curve Fitting Using Splines Under Tension. Comm ACM, 17, April 1974, pp. 218-223.

8. Draper, N. and Smith, H.: Applied Regression Analysis. Wiley, New York, 1966.

9. Cuthbert, D. and Wood, F.S.: Fitting Equations to Data. Wiley-Interscience, New York, 1971.

10. Powell, M.J.D., Subroutine VC03A, Harwell Subroutine Library, Atomic Energy Research Establishment, Harwell, Berkshire, England, 1967.

11. Lyche, T. and Schumaker, L.L., Algorithm 480: Procedures for Computing Smoothing and Interpolating Natural Splines. Comm. ACM, 17, August 1974, pp. 463-467.

12. Kahaner, D., Program ISPLIN, Los Alamos Scientific Laboratory, Los Alamos, N.M., 1975.

13. Cederquist, G.N.: The Use of Computer-Generated Pictures to Extract Information from Underwater Acoustic Transfer Function Data. Ph.D. Dissertation, The University of Michigan, 1975.

14. Akima, H., Algorithm 474: Bivariate Interpolation and Smooth Surface Fitting Based on Local Procedures. Comm. ACM, 17, January 1974, pp. 26-27.

15. Akima, H.: A Method of Bivariate Interpolation and Smooth Surface Fitting for Values Given at Irregularly Distributed Points. OT Report 75-70, U.S. Dept. of Commerce/Office of Telecommunications, Boulder, Colo., 1975.

16. Tobler, W.: Tuning an Interpolated Lattice. Dept. of Geography, The University of Michigan, Ann Arbor, Mich., 1976.

17. Williamson, H., Algorithm 420: Hidden Line Plotting Program. Comm. ACM, 15, February 1972, pp. 100-103.

18. Wright, T.J.: A Two-Space Solution to the Hidden Line Problem for Plotting Functions of Two Variables. IEEE Transactions on Computers, C22, January 1973, pp. 28-33.

19. NCAR Software Support Library, Volume 3, NCAR-TN/LA-105, National Center for Atmospheric Research, Boulder, Colo., 1975 (contact T.J. Wright).

20. Dwyer, W.C.: Windows, Shields, and Shading. Proc. 6th UAIDE Meeting, Washington, D.C., 1967.

21. Phillips, R.L.: Subroutine SHADE. Department of Aerospace Engineering, The University of Michigan, Ann Arbor, Mich., 1972.

22. Ison, N.T.: An Algorithm to Shade a Plot. Computer Graphics (SIGGRAPH-ACM Quarterly), 7, 1973, pp. 10-23.

23. Newman, W.M. and Sproull, R.F.: Principles of Interactive Computer Graphics. McGraw-Hill, New Yorke 1973, pp. 121-126.

24. Jarvis, J.F.: Two Simple Windowing Algorithms. Software-Practice and Experience, 5, 1975, pp. 115-122.

25. Behler, B. and Zajac, E.E.: A Generalized Window-Shield Routine. Proc. 8th UAIDE Meeting, Coronado, Cal., 1969.

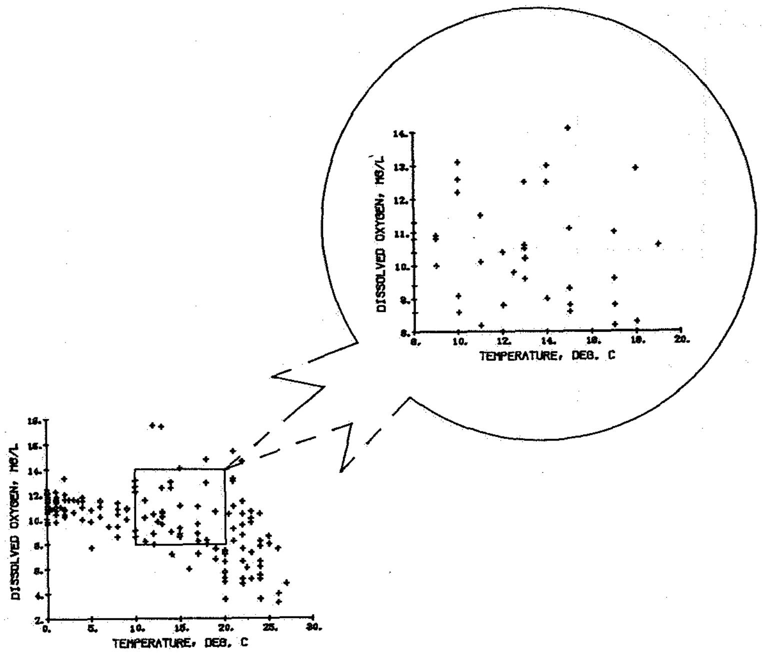

Figure 1.- Automatic scale generation.

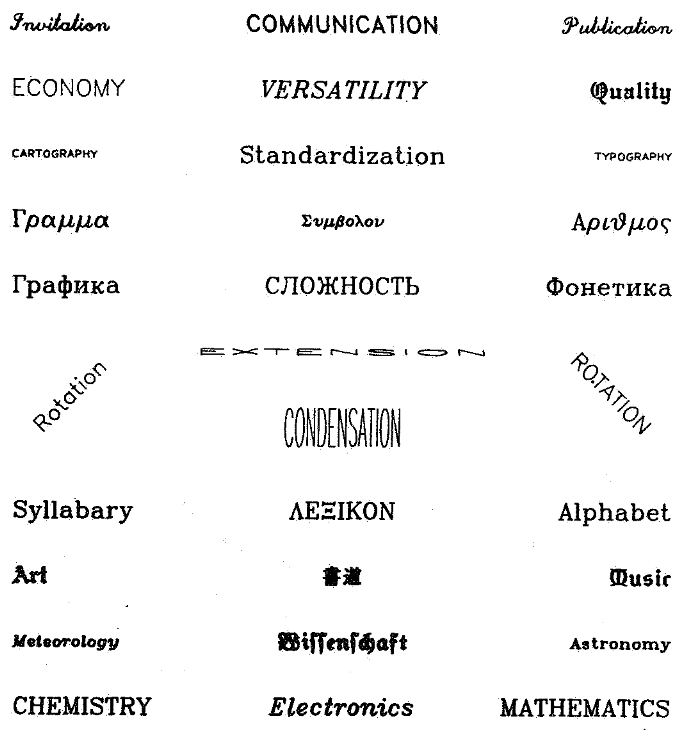

Figure 2.- Hershey's software character fonts.

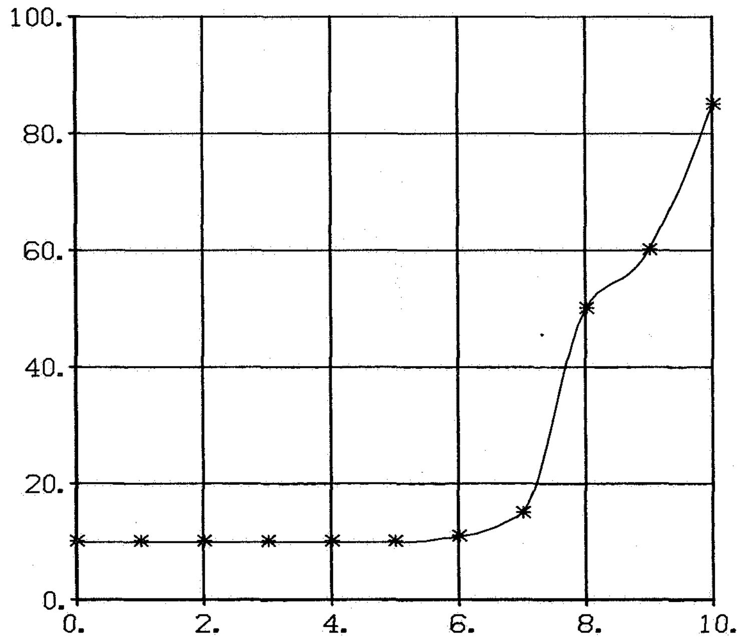

Figure 3.- Curve fitting with Akima's method.

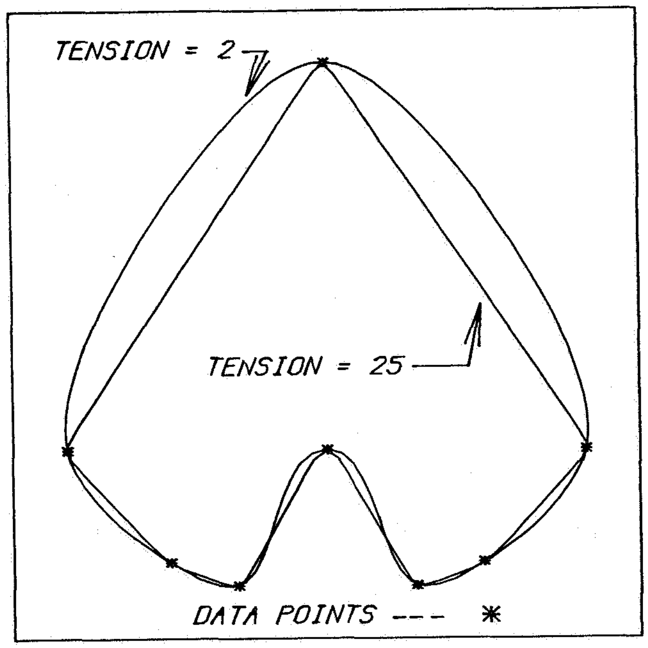

Figure 4.- Application of Cline's tension splines.

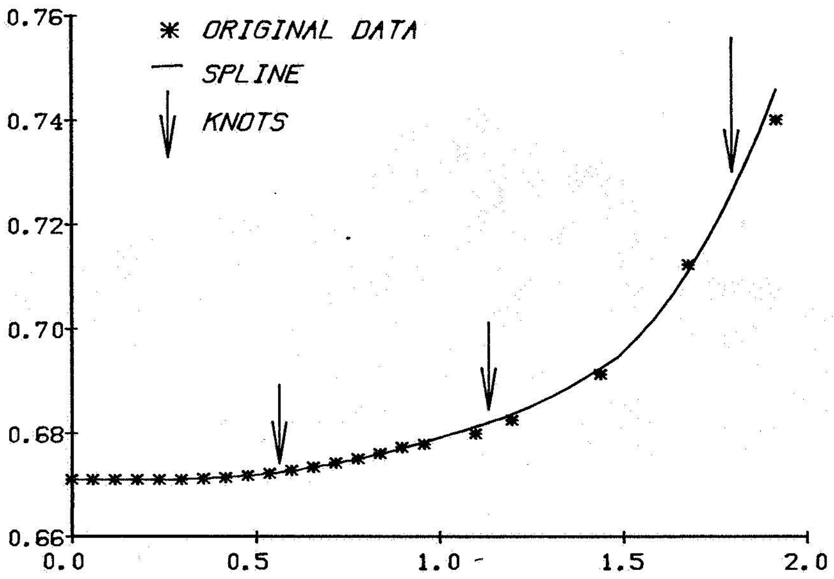

Figure 5.- Application of Powell's smoothing splines.

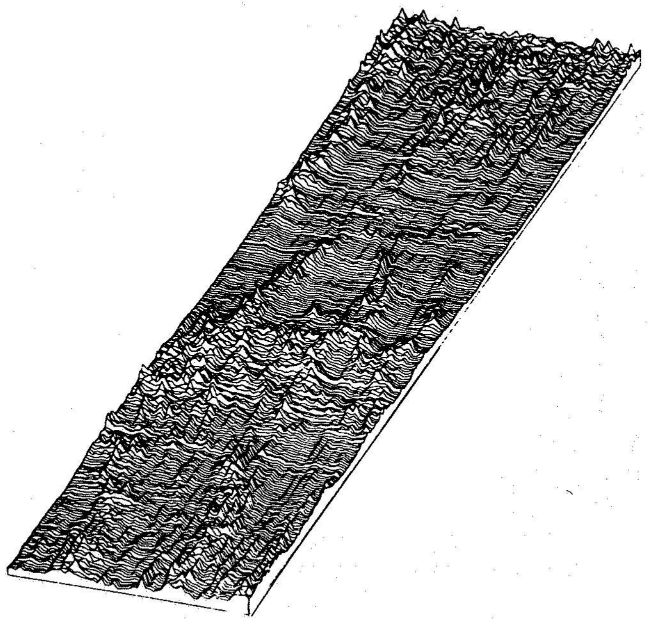

Figure 6.- Representation of a complex dataset as a three-dimensional surface.

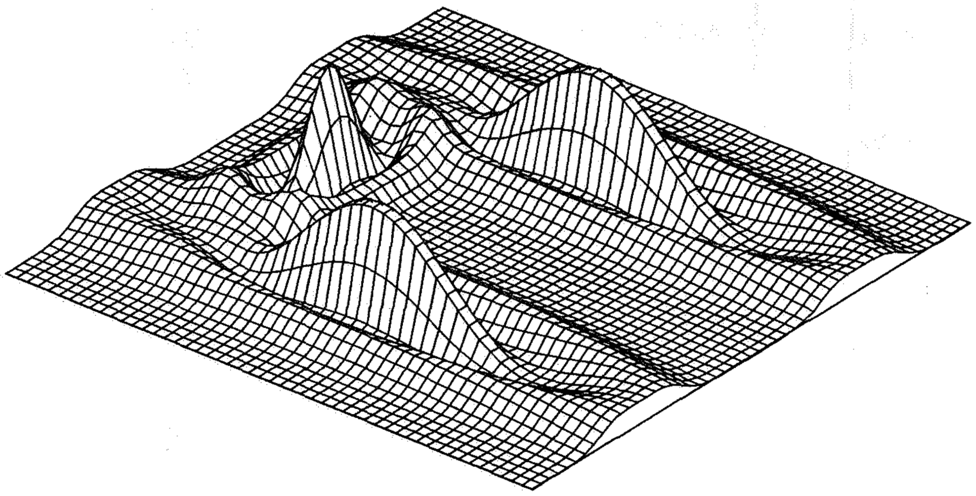

Figure 7.- Three-dimensional representation of a mathematical function.

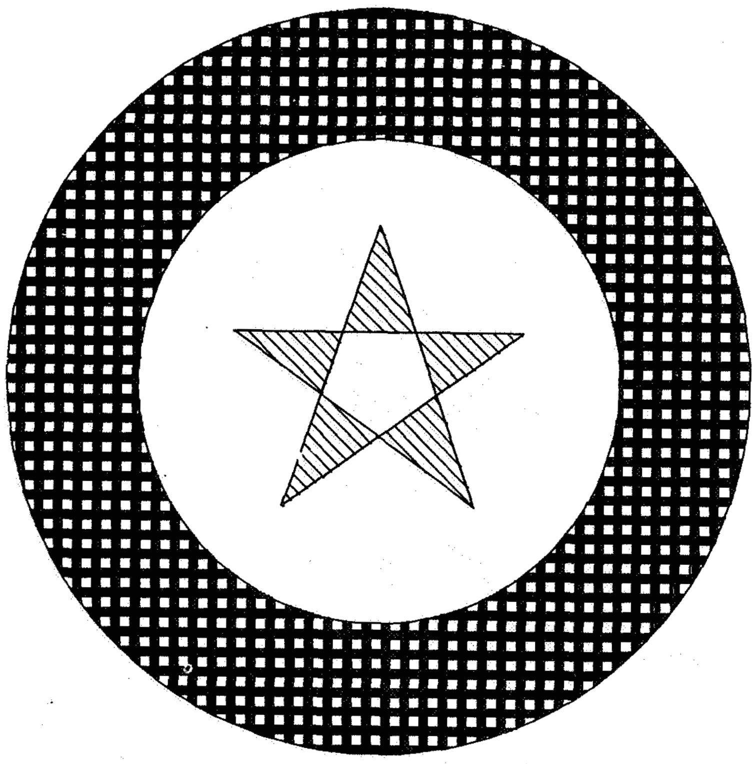

Figure 8.- Shading of compound polygons.

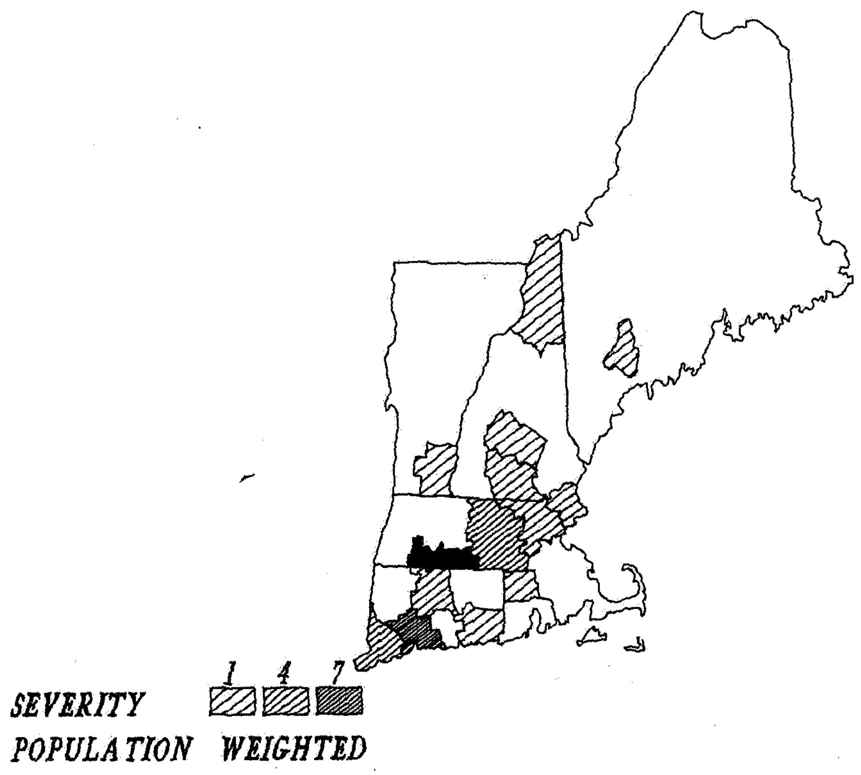

Figure 9.- Use of shading to depict pollution patterns.

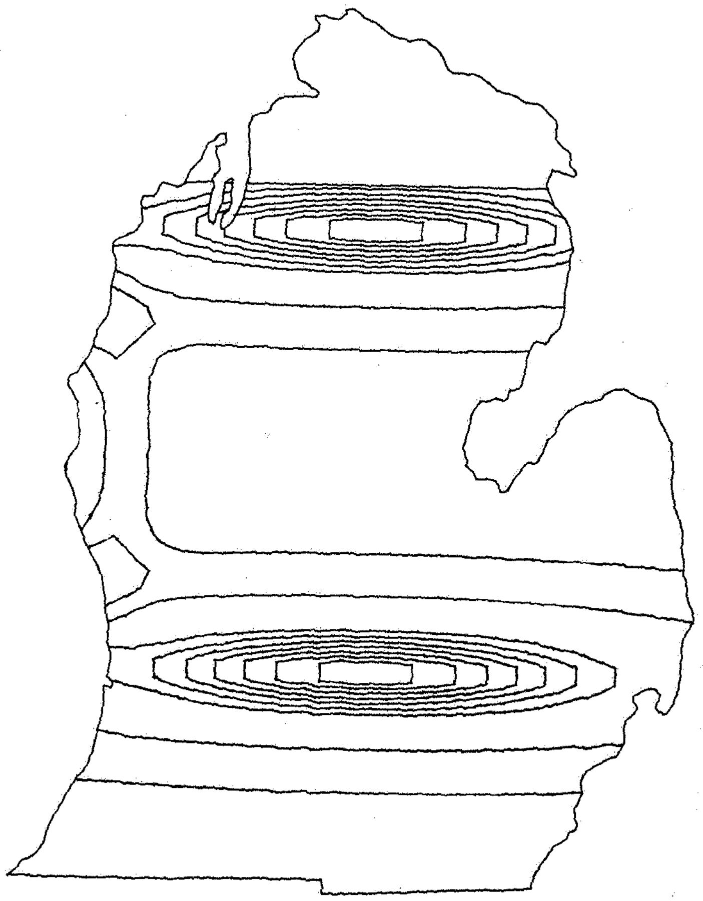

Figure 10.- General polygonal windowing.