Comp. & Maths. with Appls. Vol. 8, No. 4, pp. 273–281, 1982
Printed in Great Britain.
0097–4943/82/040273–09$03.00/0
Pergamon Press Ltd.

SMOOTH INTERPOLATION OF SCATTERED DATA BY LOCAL THIN PLATE SPLINES†

RICHARD FRANKE
Department of Mathematics, Naval Postgraduate School, Monterey, CA 93940, U.S.A.

(*Received March* 1981; *in revised form August* 1981)

Communicated by E. Y. Rodin

Abstract—An algorithm and the corresponding computer program for solution of the scattered data interpolation problem is described. Given points (x_k, y_k, f_k), $k = 1, \ldots, N$ a locally defined function $F(x, y)$ which has the property $F(x_k, y_k) = f_k$, $k = 1, \ldots, N$ is constructed. The algorithm is based on a weighted sum of locally defined thin plate splines, and yields an interpolation function which is differentiable. The program is available from the author.

1. INTRODUCTION

The problem of constructing interpolation functions for sets of scattered data in two independent variables has been treated in many papers. The survey paper on approximations to multivariate functions by Schumaker[1] contains extensive references. Other survey papers are by Barnhill[2] and Sabin[3]. The present author has surveyed and tested a number of algorithms for solution of the problem[4, 5]. Conceptually the problem is quite simple: Given points (x_k, y_k, f_k), $k = 1, \ldots, N$, with distinct (x_k, y_k), construct a function $F(x, y)$ so that $F(x_k, y_k) = f_k$, $k = 1, \ldots, N$. Generally one wants to have a smooth interpolant, $F(x, y)$, in the sense that low order partial derivatives are everywhere continuous. This is complicated for large sets of data by the fact that the interpolant (in a practical sense, to be computable) must be local, so that its value at some point (x, y) depends only on (x_k, y_k, f_k) values for which (x_k, y_k) is "close" to (x, y). A general framework for a class of such methods is given in[6], and we will discuss it briefly in Section 2.

While a large number of ideas have been proposed for solution of the problem, a much smaller number of working computer programs are readily available. These include: (1) a method based on finite element functions defined over triangles, due to Akima[7, 8], a version of which is available in the IMSL† library under the name IQHSCV, (2) a program based on a similar idea, due to Lawson[9], (3) two programs based on weighted local approximations by quadratic functions, due to Franke and Nielson[10]. A program by Little[11], and another by Nielson[12], both based on finite element functions defined over triangles, will probably be available by press time. All of these programs have been tested by the author[4, 5], and most perform adequately in a variety of cases; None of them seems to have a clear edge over all the others, or to be entirely satisfactory. For certain applications, each has its good points. The choice of a method for most users will be based on subjective criteria which vary from person to person and application to application. It is not surprising this is the case in two variables since it is also the case in one variable, although perhaps to a lesser extent.

The purpose of this paper is to document an alternative scheme which performed comparably well in the previously mentioned tests. In this way the computer program will be brought to the attention of potential users, who may request it from the author. The method will supplement the currently available codes in that it is based on a different approach. It is anticipated that it will find application and approval in a variety of areas.

The theoretical background for the method is discussed in Section 2, while details of the program are outlined in Section 3. Section 4 gives information concerning usage of the program. Several examples are given in Section 5, including perspective plots of some surfaces generated by the program. Included is an example of how the variation of a parameter in the method

†Support in part by the Office of Naval Research through the Foundation Research Program.
†International Mathematical and Statistical Libraries, 7500 Bellaire Blvd., Houston, TX 77036.

affects the surface. General guidelines for choice of this parameter are given, even though the suggested value usually leads to satisfactory results. A general discussion of the features of the method is given, along with general guidance for use of the program.

2. THEORETICAL BACKGROUND

The general idea encompassing this scheme and many others is given in [6]. Consider that the points (x_k, y_k, f_k), $k = 1, \dots, N$ are given. Briefly, local approximations to the data are constructed, and these are then blended together by using weighting functions which form a partition of unity on R^2. We pause to give the necessary notation and definitions.

A set of functions, $w_i(x, y)$, $i = 1, \dots, m$ is said to form a partition of unity if each $w_i(x, y) \geq 0$ and $\sum_{i=1}^{m} w_i(x, y) \equiv 1$. The w_i will be called weight functions. Let the support of w_i be denoted by $S_i = \text{closure } \{(x, y): w_i(x, y) \neq 0\}$. Let $I_i = \{k: (x_k, y_k) \in \text{Interior } (S_i)\}$. Now suppose that the functions $Q_i(x, y)$, $i = 1, \dots, m$, are defined on S_i and have the property that they interpolate the data whose (x, y) coordinates are in Interior (S_i), i.e. if $k \in I_i$, then $Q_i(x_k, y_k) = f_k$. These functions Q_i will be called local approximations. We then consider the function $F(x, y) = \sum_{i=1}^{m} w_i(x, y) Q_i(w, y)$. Its properties are summarized in the following.

Proposition. The function $F(x, y) = \sum_{i=1}^{m} w_i(x, y) Q_i(x, y)$ has the following properties:

(1) Interpolation; $F(x_k, y_k) = f_k$, $k = 1, \dots, N$.

(2) Smoothness; $F(x, y)$ is at least as smooth as the w_i and Q_i, e.g., if all of the functions w_i, Q_i, $i = 1, \dots, m$ have continuous first derivatives, so does $F(x, y)$.

(3) Local dependence on the data; Let (x, y) be fixed, and let $J_{x,y} = \{i: w_i(x, y) \neq 0\}$, then $F(x, y)$ depends only on the (x_k, y_k, f_k) points for which $k \in (\bigcup_{i \in J_{x,y}} I_i) \cup \{i\text{: some } Q_j,\ j \in J_{x,y}$ depends on $(x_i, y_i, f_i)\}$. In particular, we have $F(x, y) = \sum_{i \in J_{x,y}} W_i(x, y) Q_i(x, y)$.

These properties are essentially observations, but form the basis for construction of appropriate weight functions which will allow easy determination of the set $J_{x,y}$. Our construction yields a set of at most four nonzero terms in the sum defining $F(x, y)$. This provides a considerably faster process during the evaluation of the interpolant than was possible in the choices previously considered [6].

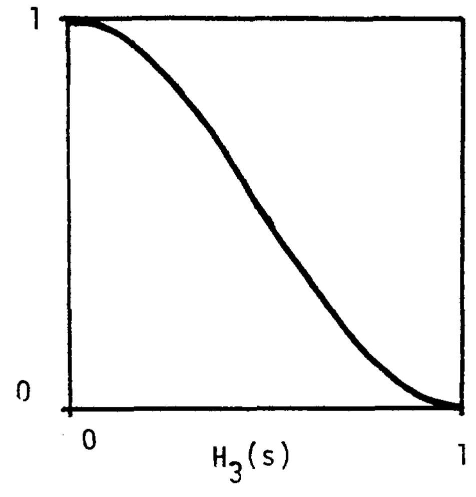
It has been implied, but is not necessary for the proposition to hold, that many of the weight functions, w_i, have finite support. In order for the local approximations $Q_i(x, y)$ to actually be local, this will likely be the case. Therefore, we think of weight functions whose support, S_i, contains relatively few (x_k, y_k) points, and whose support is often local. In order for the interpolant to be defined on all of R^2, some sets S_i must not be finite, and typically would contain points (x_k, y_k) on or near the boundary of the convex hull of the set of points $\{(x_k, y_k)\}$. With this framework and ideas in mind, we are ready to discuss the specific details of the algorithm.

3. DETAILS OF THE ALGORITHM

3.1 *Choice of weight functions*

While the choice of weight functions was allowed to determine the support regions in the discussion of the previous section, it is more convenient to proceed from support regions to weight functions in the actual application. The use of support regions which are rectangles have specific advantages in terms of controlling the number of support regions in which a particular point (x, y) lies, as well as simplifying determination of those regions.

Let n_x and n_y be given positive integers, and let finite values of $\bar{x}_0 < \bar{x}_1 < \bar{x}_2 < \dots < \bar{x}_{n_x} < \bar{x}_{n_x+1}$ and $\bar{y}_0 < \bar{y}_1 < \bar{y}_2 \dots < \bar{y}_{n_y} < \bar{y}_{n_y+1}$ be given. For each $i = 1, 2, \dots, n_x$ and $j = 1, 2, \dots, n_y$ let $r_{i,j} = [\bar{x}_{i-1}, \bar{x}_{i+1}] \times [\bar{y}_{j-1}, \bar{y}_{j+1}]$. Let $H_3(s) = 1 - 3s^2 + 2s^3$, the Hermite cubic satisfying $H_3(0) = 1$, $H_3(1) = H_3'(0) = H_3'(1) = 0$, as shown in the sketch. We then define piecewise cubics with



continuous first derivatives, which are nonzero only on two adjacent intervals, and satisfy

$$v_i(x_j) = \delta_{ij}, \quad i, j = 1, 2, \ldots, n_x$$

$$u_j(y_i) = \delta_{ji} \quad i, j = 1, 2, \ldots, n_y.$$

In particular we take

$$v_1(x) = \begin{cases} 1 & , \quad x < \bar{x}_1 \\ H_3\left(\dfrac{x - \bar{x}_1}{\bar{x}_2 - \bar{x}_1}\right) & , \quad \bar{x}_1 \le x < \bar{x}_2 \\ 0 & , \quad x \ge x_2 \end{cases}$$

$$v_i(x) = \begin{cases} 0 & , \quad x < \bar{x}_{i-1} \\ 1 - v_{i-1}(x) & , \quad \bar{x}_{i-1} \le x < \bar{x}_i \\ H_3\left(\dfrac{x - \bar{x}_i}{\bar{x}_{i+1} - \bar{x}_i}\right) & , \quad \bar{x}_i \le x < \bar{x}_{i+1} \\ 0 & , \quad x \ge \bar{x}_{i+1} \end{cases}$$

for $i = 2, \ldots, n_x - 1$

$$v_{n_x}(x) = \begin{cases} 0 & , \quad x < \bar{x}_{n_x - 1} \\ 1 - v_{n_x - 1}(x) & , \quad \bar{x}_{n_x - 1} \le x < \bar{x}_{n_x} \\ 1 & , \quad x \ge \bar{x}_{n_x} \end{cases}$$

The $u_i(y)$ are dual. Then if we define

$$W_{i,j}(x, y) = v_i(x) u_i(y), \quad i = 1, \ldots, n_x, \quad j = 1, \ldots, n_y,$$

it is easily observed that $W_{i,j}$ has support $r_{i,j}$, except for when $i = 1$ or n_x, or $j = 1$ or n_y, when the support extends to ∞ in one or both variables. We denote the support of $W_{i,j}$ by $R_{i,j}$. Further, we note that the $W_{i,j}$ form a partition of unity on R^2.

Since the $\bar{x}_i$ and $\bar{y}_j$ values give rise to a grid on the plane, we call them grid values. The choice of grid values $\bar{x}_i$ and $\bar{y}_j$ depend on the data. They may be specified by the user, but the default option is for them to be determined by the program. In the default mode the user specifies a parameter, NPPR (for number of points per region), the suggested value being about 10. The program will then determine the grid values so that the anticipated number of (x_k, y_k) points in each rectangle $R_{i,j}$ will be approximately NPPR. For data which is not somewhat uniformly distributed the actual numbers may vary considerably. However, for most situations we have encountered, the process is quite adequate.

An equal number of grid values is taken in each direction, i.e. $n_x = n_y$. Because we want NPPR points per rectangle, each subrectangle $(x_i, x_{i+1}) \times (y_j, y_{j+1})$ should have 1/4 NPPR points. Since $n_x = n_y$, we want $(n_x + 1)^2(1/4)$ NPPR $= N$, leading us to take n_x to be the nearest integer to $(4N/\text{NPPR})^{1/2} - 1$.

The grid values $\bar{x}_i$, are now chosen so that there are approximately equal numbers of points from the values x_k, $k = 1, \ldots, N$ in each interval $(\bar{x}_i, \bar{x}_{i+1})$. Specifically, let $\hat{x}_k$ denote the x_k values arranged in nondecreasing order. Consider the points $(0, \hat{x}_1), (1, \hat{x}_2), \ldots, (N-1, \hat{x}_N)$, and let $g(t)$ be the piecewise linear interpolant for them. Divide the interval $(0, N-1)$ into $n_x + 1$ subintervals of length $\Delta = (N - 1/n_x + 1)$. The values of $\bar{x}_i$ are determined by taking them to be the values of g at the endpoints of the subintervals, i.e. $\bar{x}_i = g(i)$, $i = 0, 1, 2, \ldots, n_x + 1$. The $\bar{y}_j$ are determined in dual fashion. This process results in the grid values and hence the rectangles, being symmetric if the (x_k, y_k) points are symmetric. In addition, the relative positions of grid values are unchanged by linear displacements and stretching in each variable.

When chosen in the above fashion, the location of the lines is not a local process in the sense that insertion of an additional point will change the boundaries of all of the rectangles. While one could argue that the scheme is not local, we take the view that the idea of local determination of the interpolant is most important in the evaluation phase. The determination of parameters in the scheme (here, the $\bar{x}_i$ and $\bar{y}_j$) may be a global process. Of course, if the user specifies the grid values, he will likely be using a global process to choose them.

3.2 *Choice of local approximations*

The only constraint on the local approximations is that they interpolate the appropriate points, and that they have continuous first derivatives (at least) to assure a smooth interpolant. In the previously mentioned tests conducted by the author, a number of global interpolation schemes for scattered data were considered. In principle, any of these might be used. The choice here was made for two reasons, (1) the method scored very well in the tests, and (2) the method has an elegant and well developed mathematical theory which also has direct application to some engineering problems.

The local approximations used in this algorithm are the thin plate splines first mentioned by Harder and Desmairis[13], with theoretical developments by Duchon[14–17] and Meinguet[18, 19]. It was first developed as the solution to the problem of a thin plate which is forced to pass through certain points (the interpolation points) by application of point loads. For our purposes it is sufficient to know that the approximation is of the form

$$Q(x, y) = \sum_{k \in I} A_k d_k^2 \log d_k + a + bx + cy,$$

where $I = \{k\colon Q$ is to take on the value f_k at $(x_k, y_k)\}$, and $d_k^2 = (x - x_k)^2 + (y - y_k)^2$. The coefficients A_k, and a, b, and c are determined by the linear system of equations

$$\sum_{k \in I} A_k d_k^2 \log d_k + a + bx + cy\Big|_{(x,y)=(x_i,y_i)} = f_i, \qquad i \in I$$

$$\sum_{k \in I} A_k = 0$$

$$\sum_{k \in I} A_k x_k = 0$$

$$\sum_{k \in I} A_k y_k = 0.$$

The geometric effect of the last three equations is to suppress all terms in the approximation which grow faster than linear as distance from the interpolation points is increased. A linear system of order equal to the number of interpolation points plus three must be solved. To be nonsingular there must be at least three noncollinear points among the (x_k, y_k), $k \in I$. The system is symmetric, but not positive definite. While an equation solver designed for such systems could be used, we have found a general purpose solver, the DECOMP/SOLVE subroutines of Forsythe, Malcolm and Moler[20] has given more reliable results.

It is easily observed that the local interpolant has continuous derivatives of all orders except at the data points, (x_k, y_k), where a logarithmic singularity occurs in the second derivatives. The interpolant has linear precision, that is, if the (x_k, y_k, f_k) points all lie on a linear function, the interpolant will reproduce it.

While the thin plate spline is invariant with respect to scaling, translation, and rotation (not all of this is obvious), the condition number of the coefficient matrix for the system of equations is dependent on scaling. To minimize difficulties with that, and to remove the effect of scaling the variables by different amounts, we transform each rectangle $r_{i,j}$ onto the unit square $[0, 1]^2$, before the local approximation $Q_{i,j}$ is computed.

It remains to specify the process for determining the points (x_k, y_k, f_k) to be interpolated by the thin plate spline local approximation. Experience has shown that it is advantageous to include more points than is necessary, i.e. (x_k, y_k) which are outside of $R_{i,j}$. This tends to yield a better transition between local approximations than when only necessary points are included. Therefore, the set of (x_k, y_k) points transformed into the rectangle $R = [-0.1125, 1.1125]$ by the transformation taking $r_{i,j}$ onto $[0, 1]^2$ are included. This results in using interpolation points from a rectangle with about 150% of the area of r_{ij}. Let

$$I_{i,j} = \left\{k: \left(\frac{x_k - \bar{x}_{i-1}}{\bar{x}_{i+1} - \bar{x}_{i-1}}, \frac{y_k - \bar{y}_{j-1}}{\bar{y}_{j+1} - \bar{y}_{j-1}}\right) \in R\right\}.$$

This gives the basic set of interpolation points for the local approximation $Q_{i,j}$ associated with the rectangle $R_{i,j}$. Under certain conditions there may be fewer than the necessary three indices in $I_{i,j}$. When this happens, the set $I_{i,j}$ is augmented by including as interpolation points the necessary number of closest points to the rectangle $R_{i,j}$ (in the l_∞ norm), after the points (x_k, y_k) have undergone the transformation to the unit square. The minimum number of points per rectangle is a variable, MINPTS. This has been set to 3, but may be increased if it seems desirable.

After the interpolation points for each local approximation have been determined, the local thin plate splines, $Q_{i,j}$, can be determined by calculating the coefficients. This yields

$$Q_{i,j}(x, y) = \sum_{k \in I_{i,j}} A_{i,j,k} d_k'^2 \log d_k' + a_{i,j} + b_{i,j}x' + c_{i,j}y'$$

where the primes denote coordinates and distance after the transformation of $r_{i,j}$ to the unit square.

3.3 *Properties of the interpolant*

The overall interpolant is of the form

$$F(x, y) = \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} W_{i,j}(x, y) Q_{i,j}(x, y),$$

and as noted previously, there are at most four nonzero terms in the sum. Which terms are nonzero is easily determined. If $x \geq \bar{x}_{n_x}$, set $i' = n_x$; Otherwise let i' be the smallest index so that $\bar{x}_{i'+1} > x$. Determine j' in dual fashion from y and the $\bar{y}_j$'s. Then, the four nonzero terms have indices (i', j'), $(i' + 1, j')$, $(i', j' + 1)$, and $(i' + 1, j' + 1)$. If $i' = 0$ or $i' = n_x$, the terms involving i' or $i' + 1$, respectively, do not appear, and similarly if $j' = 0$ or $j' = n_y$, the terms involving j' or $j' + 1$, respectively, do not appear.

In addition to the properties outlined in Section 2, certain other properties hold. The approximation is invariant under translation and stretching (independently in each variable). It has symmetry with respect to planes parallel to coordinate planes whenever the data has that symmetry. The approximation is, in general, not invariant under rotations, however, since the rectangles depend on the individual coordinates of the data points. The exception is when only one grid line in each direction occurs, resulting in the approximation reducing to the global thin plate spline.

The approximation has continuous first derivatives, and jump discontinuities in the second derivatives across grid lines, as well as logarithmic singularities in the second derivatives at the data points. Plots of the surfaces generally appear to be quite smooth, however, Since the local approximations have linear precision, the overall approximation also has linear precision, i.e. if the data lies on a linear function, the interpolant is a linear function.

4. THE COMPUTER PROGRAM

The calling program must supply the (x_k, y_k, f_k) points, plus two arrays of points $x0_i$, and $y0_j$ for the grid of points $(x0_i, y0_j)$ at which the interpolant is to be evaluated. In addition, the user must supply two workspace arrays, IWK and WK in which information calculated during preprocessing (e.g. $I_{i,j}$, $\bar{x}_i$, $\bar{y}_j$, and coefficients for the local approximations), is stored, and an array FO for the returned interpolant values.

The amount of storage required for the arrays IWK and WK is not known *a priori*. The estimated space required is about 6N for IWK and about 7N for WK. Table 2 gives exact results for several different sized problems based on random (x, y) points. Oddly distributed point sets may result in somewhat more storage being required, although it is unlikely that storage will exceed 10N for either array. In any case, the precise number of locations required is returned to the calling program from Subroutine LOTPS, the only routine referenced by the user. If an insufficient number are allowed, an error return occurs. Subroutine LOTPS provides the interface between the user's program and the set of routines implementing the method. Generally, LOTPS sets up storage areas in the arrays IWK and WK, determines parameters required by other subroutines, and calls other subroutines to (1) generate the grid, if necessary, (2) determine the interpolation points for the local approximations, (3) compute coefficients for local approximations, (4) evaluate the interpolant at a grid of points.

Under the usual option, the user specifies NPPR > 0, the suggested value being 10. If the user wishes to specify the grid lines, he may do so by setting a parameter, NPPR $= 0$, and then giving grid line information in the arrays IWK and WK, as explained in the argument description for Subroutine LOTPS. Typically one should take $\bar{x}_0 = \min_k x_k$, $\bar{x}_{n_x+1} = \max_k x_k$, and the dual in y. This is not necessary, although all points to be interpolated should lie in $[\bar{x}_0, \bar{x}_{n_x+1}][\bar{y}_0, \bar{y}_{n_y+1}]$. To prevent different scaling (internally) in the two variables, a square grid covering the (x_k, y_k) points could be specified.

5. EXAMPLES AND OBSERVATIONS

Example 1. This example shows a local approximation function and is for the data in Table 1. This function is a "cardinal" function for the first point, and as such shows the effect of a nonzero value at a single point on the interpolant. The plotted surface, shown in Fig. 1, is over $[0, 1]^2$.

Example 2. This example is given to show a surface generated from 100 randomly generated points with the function value being obtained from an explicitly given function. The surface was generated using NPPR $= 10$, has rms error of about 0.3%, and is virtually indistinguishable from a plot of the parent surface. It is shown in Fig. 2.

Examples 3–5. These examples use the same set of 60 points lying in the square $(-1/18, 19/18)^2$, and chosen by a pseudorandom number generator. The function is explicitly given by

$$f(x, y) = 0.1 + \frac{\sin 3(x + y)}{12(x + y)},$$

and is shown in Fig. 3. Figures 4–6 show the interpolant over the square $[0, 1]^2$ for NPPR values of 6, 10 and 15.

From these examples we see that NPPR = 10 works well, not too much difference is observed when NPPR is increased, but NPPR = 6 gives a less smooth appearing surface. The smaller NPPR is, the more localized the surface becomes (although NPPR = 4 will probably be the least value practicable, and some local interpolants will likely become planes due to the minimum of three interpolation points being reached). In line with this comment, very smooth surfaces with small gradients will probably be amenable to larger NPPR, while surfaces with large gradients may be best approximated by taking NPPR smaller, thus localizing the behavior.

Table 2 gives the results of a series of problems with various numbers of points and values

Table 1. Data for "Cardinal" function

x	y	f
0.35	0.35	0.5
-0.05	0.25	0.0
0.10	-0.05	0.0
0.50	0.05	0.0
0.00	0.90	0.0
0.30	0.70	0.0
0.60	0.50	0.0
0.90	0.00	0.0
0.40	1.05	0.0
0.85	0.80	0.0
1.05	0.20	0.0
1.10	1.10	0.0

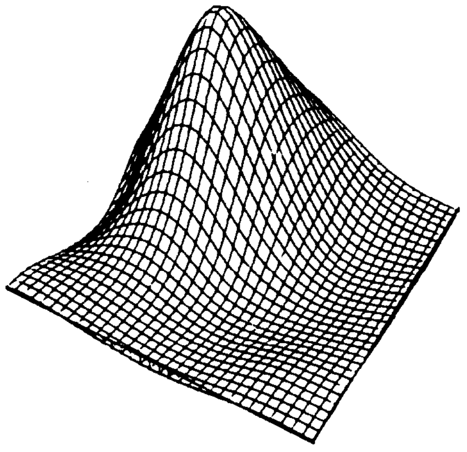


Fig. 1. Cardinal function.

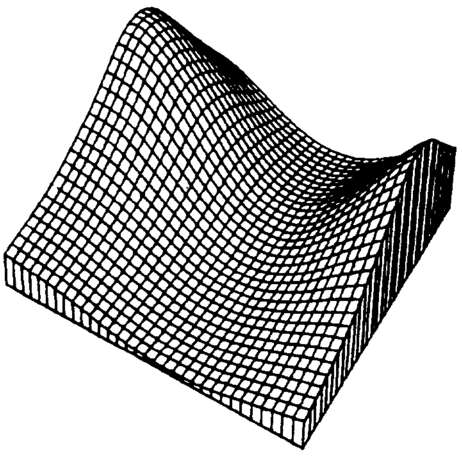


Fig. 2. Saddle function.

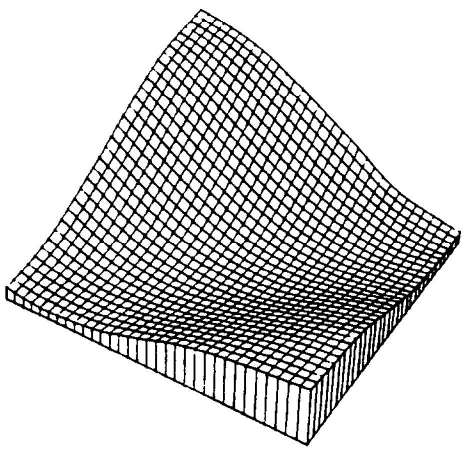


Fig. 3. Parent function.

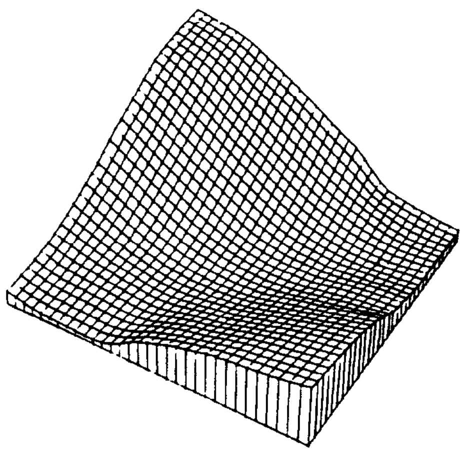


Fig. 4. NPPR = 6.

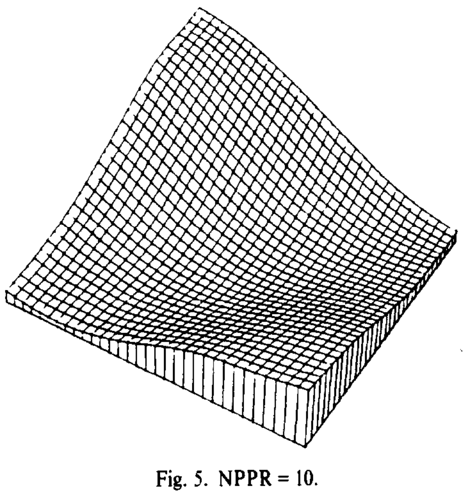


Fig. 5. NPPR = 10.

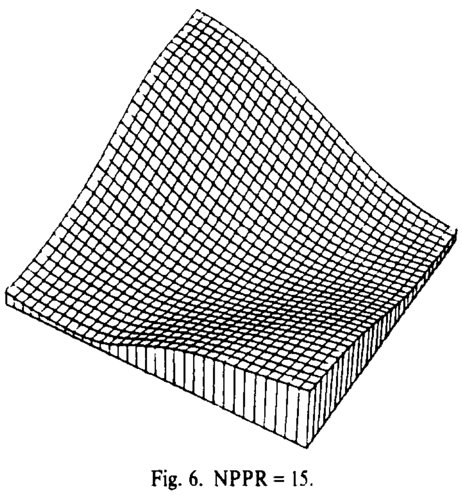


Fig. 6. NPPR = 15.

Table 2. Storage/times for various size problems

No. of points/rect. (NPPR)	No. of pts. (N)	No. of grid lines ($n_x = n_y$)	Size of WK array	Size of IWK array	Time (sec.,P400) (Preprocessing)	Time (sec.,P400) (Evaluation)
6	60	5	334	273	4.2	14.3
6	100	7	619	506	7.5	17.6
6	500	18	3596	2893	70.5	21.2
6	1000	25	7441	6140	201.5	21.3
10	60	4	284	243	5.3	18.0
10	100	5	488	427	9.9	26.4
10	500	13	3014	2649	73.3	29.8
10	1000	19	6565	5804	202.6	31.7
15	60	3	224	199	6.8	23.0
15	100	4	414	373	12.4	32.8
15	500	11	2856	2591	91.6	39.3
15	1000	15	5920	5439	258.5	40.6

of NPPR. The data points were chosen by a pseudorandom number generator in the square $[-1/18, 19/18]^2$, and the approximation was evaluated on a 33×33 grid (of 1089 points total) on $[0, 1]^2$. We observe that increasing NPPR increases execution time while decreasing the amount of storage needed. Preprocessing time should be about proportional to N^2, but apparently there is a strong linear component for small N. Preprocessing time should also be about proportional to NPPR, although this is not readily apparent. Evaluation time should be nearly independent of N for large N, and proportional to NPPR, which is approximately shown.

The automatic grid selection process works well when the data is fairly uniform in (x, y) and lies nearly in a square region. If the data is very irregular, or lies in an oblong rectangular area, it will probably be useful to explore results with a user specified grid to obtain better coverage by rectangles which are not too oblong. Limited experience has been accumulated in these situations.

6. HOW TO OBTAIN THIS PROGRAM

A copy of this program, written in Fortran, and including a sample driver program, can be obtained from the author. To do so, send a (short) 1/2 inch tape to the author indicating the format desired.

Acknowledgements—This work was supported, in part, by the Foundation Research Program at the Naval Postgraduate School. The final form of the program and documentation was accomplished while the author was on sabbatical leave at Drexel University. The author extends his thanks to the Center for Scientific Computation and Interactive Graphics at Drexel for use of their Prime 400 computer system, on which all examples were run.

REFERENCES

1. L. L. Schumaker, Fitting surfaces to scattered data, In *Approximation Theory II* (Edited by G. G. Lorentz, C. K. Chui and L. L. Schumaker), pp. 203–268. Academic Press, New York (1976).
2. R. E. Barnhill, Representation and approximation of surfaces, In *Mathematical Software III* (Edited by J. R. Rice), pp. 69–120. Academic Press, New York (1977).
3. M. A. Sabin, Contouring—A review of methods for scattered data, In *Mathematical Methods in Computer Graphics and Design*, (Edited by K. W. Brodlie), pp. 63–85. Academic Press, New York (1980).
4. R. Franke, *A Critical Comparison of Some Methods for Interpolation of Scattered Data.* Naval Postgraduate School, TR#NPS-53-79-003, Available from NTIS, #AD-A081 688/4 (1979).
5. R. Franke, Scattered data interpolation: tests of some methods. *Math Comp.* (To be published).
6. R. Franke, Locally determined smooth interpolation at irregularly spaced points in several variables. *JIMA* **19**, 471–482 (1977).
7. H. Akima, A method of bivariate interpolation and smooth surface fitting for irregularly distributed data points. *ACM TOMS* **4**, 148–159 (1978).
8. H. Akima, Algorithm 526: Bivariate Interpolation and smooth surface fitting for irregularly distributed data points. *ACM TOMS* **4**, 160–164 (1978).
9. C. L. Lawson, Software for C^1 surface interpolation, In *Software III* (Edited by J. R. Rice), pp. 159–192. Academic Press, New York (1977).
10. R. Franke and G. Nielson, Smooth Interpolation of large sets of scattered data. *Int. J. Num. Meth. Engng.* **15**, 1691–1704 (1980).
11. F. Little, University of Utah CAGD Report (forthcoming).
12. G. M. Nielson, *A Method for Interpolating Scattered Data Upon a Minimum Norm Network.* (To be published).
13. R. L. Harder and R. N. Desmarais, Interpolation using surface splines. *J. Aircraft* **9**, 189–191 (1972).
14. J. Duchon, *Fonctions—Spline du Type Plaque Mince en Dimencion* 2. Report 231, University of Grenoble (1975).
15. J. Duchon, *Fonctions—Spline a Energie Invariate par Rotation.* Report 27, University of Grenoble (1976).
16. J. Duchon, Interpolation des fonctions de deux variables suivant le principe de la flexion des plaques minces. *R.A.I.R.O. Analyse Numerique* **10**, 5–12 (1976).
17. J. Duchon, Splines minimizing rotation—invariant semi-norms in Sobolev spaces, In *Constructive Theory of Functions of Several Variables* (Edited by W. Schempp and K. Zeller), pp. 85–100. Lecture Notes in Math 571, Springer (1977).
18. J. Meinguet, Multivariate interpolation at arbitrary points made simple. *ZAMP* **30**, 292–304 (1979).
19. J. Meinguet, An intrinsic approach to multivariate spline interpolation at arbitrary points, In *Polynomial and Spline Approximation* (Edited by B. N. Sahney), pp. 163–190. D. Reidel, New York (1979).
20. G. E. Forsythe, M. A. Malcolm and C. B. Moler, *Computer Methods for Mathematical Computations.* Prentice-Hall, Englewood Cliffs, New Jersey (1977).