SMOOTH INTERPOLATION OF SCATTERED DATA BY LOCAL THIN PLATE SPLINE METHOD

M A R 8 1  R FRANKE
SMOOTH INTERPOLATION OF SCATTERED DATA BY
LOCAL THIN PLATE SPLINES

Richard Franke

Technical Report for Period Jan 79 - Mar 81

Approved for public release; distribution unlimited
Prepared for: Chief of Naval Research
Arlington, VA 22217
The work reported herein was supported in part by the Foundation Research Program of the Naval Postgraduate School with funds provided by the Chief of Naval Research.

Reproduction of all or part of this report is authorized.

This report was prepared by:

RICHARD FRANKE
Associate Professor
Department of Mathematics

Reviewed by:
CARROLL O. WILDE, Chairman
Department of Mathematics

Released by:
W. M. TOLLES
Dean of Research
**Title and Subtitle**: Smooth Interpolation of Scattered Data by Local Thin Plate Splines


**Performer's Name and Address**: Foundation Research Program
Naval Postgraduate School
Monterey, CA 93940

**Contract or Grant Number(s)**:

**Program Element, Project, Task Area & Work Unit Numbers**:

**Report Date**: 26 March 1981

**Number of Pages**: 30

**Security Classification (of Report)**: Unclassified

**Distribution Statement (of Report)**: Approved for public release; distribution unlimited

**DISTRIBUTION STATEMENT (of this abstract entered in Block 20, if different from report)**: 

**Supplementary Notes**: 

**Keywords**: Scattered data interpolation, Multivariate approximation, Interpolation, Smooth interpolation, Thin Plate Splines, Bivariate interpolation

**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 

\[ F(x_k, y_k) = f_k, \quad k = 1, \ldots, N \]

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.
1.0 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 [20] contains extensive references. Other survey papers are by Barnhill [3] and Sabin [19]. The present author has surveyed and tested a number of algorithms for solution of the problem [10], [11]. 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 [9], 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 [1], [2], a version of which is available in the IMSL* library under the name IQRSCV, (2) a program based on a similar idea, due to Lawson [14], (3) two programs based on weighted local approximations by quadratic functions, due to Franke and Nielson [12]. A program by Little [15], and another by Nielson [18], both based on finite

*International Mathematical and Statistical Libraries, 7500 Bellaire Blvd., Houston, TX 77036.
element functions defined over triangles, will probably be available by press time. All of these programs have been tested by the author [10], [11], 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 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.

2.0 Theoretical Background

The general idea encompassing this scheme and many others are given in [9]. Consider that the points \((x_k, y_k, f_k)\), \(k = 1, ..., N\) 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 $\mathbb{R}^2$. We pause to give the necessary notation and definitions.

A set of functions, $w_i(x,y)$, $i = 1, \ldots, m$ is said to form a partition of unity if each $w_i(x,y) \geq 0$ and $\sum_{i=1}^{m} w_i(x,y) = 1$. 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 $Q_i(x,y)$, $i = 1, \ldots, m$, are defined on $S_i$ and have the property that they interpolate the data whose $(x,y)$ coordinates are in $\text{Interior} (S_i)$, 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 $F(x,y) = \sum_{i=1}^{m} w_i(x,y)Q_i(x,y)$. 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;** $F(x_k, y_k) = f_k$, $k = 1, \ldots, N$.

2. **Smoothness;** $F(x,y)$ is at least as smooth as the $w_i$ and $Q_i$, e.g.,
   
   if all of the functions $w_i$, $Q_i$, $i = 1, \ldots, m$ 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 $k \in \bigcup_{i \in J_{x,y}} I_i$: some $j \in J_{x,y}$ 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 [9].
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 \( \mathbb{R}^2 \), 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.

3.0 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 \( x_0 < x_1 < x_2 < \ldots < x_{n_x} < x_{n_x+1} \) and \( y_0 < y_1 < y_2 < \ldots < y_{n_y} < y_{n_y+1} \) be given. For each \( i = 1, 2, \ldots, n_x \) and \( j = 1, 2, \ldots, n_y \) let \( r_{i,j} = [x_{i-1}, x_{i+1}] \times [y_{j-1}, y_{j+1}] \).

Let \( H_3(s) = 1 - 3s^2 + 2s^3 \), the Hermite cubic satisfying \( H_3(0) = 1 \), \( H_3(1) = H'_3(0) = H'_3(1) = 0 \). 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

\[ v_1(x) = \begin{cases} 
1, & x < x_1 \\
H_3 \left( \frac{x - x_1}{x_2 - x_1} \right), & x_1 \leq x < x_2 \\
0, & x \geq x_2
\end{cases} \]

for \( i = 2, \ldots, n_x - 1 \)

\[ v_{n_x}(x) = \begin{cases} 
0, & x < x_{n_x - 1} \\
1 - v_{n_x - 1}(x), & x_{n_x - 1} \leq x < x_n \\
1, & x \geq x_n
\end{cases} \]

The \( u_j(y) \) are dual. Then if we define

\[ W_{i,j}(x,y) = v_i(x)u_j(y), \quad i = 1, \ldots, n_x, \quad j = 1, \ldots, n_y, \]
it is easily observed that \( W_{i,j} \) has support \( R_{i,j} \), except for when \( i = 1 \) or \( n \), or \( j = 1 \) or \( n \), when the support extends to \( \infty \) 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 \( \mathbb{R}^2 \).

Since the \( x_i \) and \( y_j \) values give rise to a grid on the plane, we call them grid values. The choice of grid values \( x_i \) and \( y_j \) 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 \( \frac{1}{4} NPPR \) points. Since \( n_x = n_y \), we want \((n_x + 1)\frac{1}{4} NPPR = N\), leading us to take \( n_x \) to be the nearest integer to \((4N/NPPR)^{1/2} - 1\).

The grid values \( x_i \), are now chosen so that there are approximately equal numbers of points from the values \( x_k \), \( k = 1, \ldots, N \) in each interval \((x_i, x_{i+1})\). Specifically, let \( \hat{x}_k \) denote the \( x_k \) values arranged in non-decreasing 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 \( \Delta = \frac{N-1}{n_x+1} \). The values of \( \hat{x}_i \) are determined by taking them to be the values of \( g \) at the endpoints of the subintervals, i.e., \( \hat{x}_i = g(i) \), \( i = 0, 1, 2, \ldots, n_x + 1 \). The \( y_j \) 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 then 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 \( x_i \) and \( v_j \)) 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 [4-7] and Meinguet [16], [17]. 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 \( I = \{k: Q \text{ is to take on the value } f_k \text{ at } (x_k, y_k)\} \), 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
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 [31] 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 \((x_k, y_k)\) 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. The set of \((x_k, y_k)\) points transformed into the rectangle \(R = [-1.125, 1.125]^2\) by the transformation taking \(r_{i,j}\) onto \([0,1]^2\) are included. Let

\[
I_{i,j} = \{k: \left( \frac{X_k - X_{i-1}}{X_{i+1} - X_{i-1}}, \frac{Y_k - Y_{j-1}}{Y_{j+1} - Y_{j-1}} \right) \in R \}.
\]

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 \(l_\infty\) 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

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

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

\[
P(x, y) = \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} W_{i,j}(x, y) Q_{i,j}(x, y),
\]
and as noted previously, there are at most four nonzero terms in the sum. Which terms are nonzero is easily determined. If \( x \geq x_n \), set \( i' = n_x \); otherwise let \( i' \) be the smallest index so that \( x_{i'+1} > x \). Determine \( j' \) in dual fashion from \( y \) and the \( y_j \)'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 not invariant under rotations, however, since the rectangles depend on the individual coordinates of the data points.

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.

4.0 The Computer Program

The overall hierarchy of the subroutines is given in Diagram 1, which shows the communication links between them. No COMMON is used as array sizes are problem dependent and thus specified by the user. We briefly discuss them, mentioning important parameters.
Calling Program

This program is supplied by the user and must supply the \((x_k, y_k, f_k)\) points, plus two arrays of points \(x_{0j}\) and \(y_{0j}\) for the grid of points \((x_{0i}, y_{0j})\) 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., \(I_{i,j}, x_i, y_j\), 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 1 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. 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.

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 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 \( \bar{x}_0 = \min_k x_k, \bar{x}_{n_k + 1} = \max_k x_k \), and the dual in y. This is not necessary, although all points to be interpolated should lie in \([\bar{x}_0, \bar{x}_{n_k + 1}] \times [\bar{y}_0, \bar{y}_{n_y + 1}]\). To prevent different scaling (internally) in the two variables, a square grid covering the \((x_k, y_k)\) points could be specified.

**Subroutine LOTPS**

This subroutine 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 the local approximations, (4) evaluate the interpolant at a grid of points.

**Subroutine GRID**

This subroutine selects values of \( \tilde{x}_i \) and \( \tilde{y}_j \) in accordance with the discussion in Section 3.

**Subroutine LOCLIP**

This subroutine determines the interpolation points for each local interpolant, in accordance with Section 3.

**Subroutine CLOTS**

This subroutine generates the system of equations for the coefficients of the local approximations and calls an equation solver to obtain them.
Internally the routine has a maximum of 30 local interpolation points. This can easily be altered by changing two statements, as noted in the program listing.

Subroutine EVLIPS

This subroutine evaluates the interpolant on the grid of points specified by the user. Use of a grid, when that is required, facilitates the process of locating the rectangle in which a particular evaluation point is located. The $x_0$ and $y_0$ values should be in increasing order for maximum efficiency. Evaluation time for a grid of points should be nearly independent of $N$ for large $N$, which is borne out by the timing information in Table 2.

5.0 Examples and Observations

Example 1. This example shows a typical 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 Figure 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 $NPR = 10$, has rms error of about .3%, and is virtually indistinguishable from a plot of the parent surface. It is shown in Figure 2.

Examples 3, 4, 5. These examples use the same set of 60 points lying in the square $(-\frac{1}{18}, 19, 18)^2$, and chosen by a pseudorandom number generator. The function is explicitly given by $f(x,y) = .1 + \frac{\sin 3(x+y)}{12(x+y)}$, and is shown in Figure 3. Figures 4, 5, and 6 show the interpolant over the square $[0,1]^2$ for $NPR$ values of 6, 10, and 15.

From these examples we see that $NPR = 10$ works well, not too much difference is observed when $NPR$ is increased, but $NPR = 6$ gives a less
smooth appearing surface. The smaller NPPR is, the more localized the surface becomes (although NPPR = 1 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 of NPPR. The data points were chosen by a pseudorandom number generator in the square \([-\frac{1}{18}, \frac{19}{18}\]^2\), and the approximation was evaluated on a 33x33 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.

6.0 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 document-ation 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.
<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.35</td>
<td>0.35</td>
<td>0.5</td>
</tr>
<tr>
<td>-0.05</td>
<td>0.25</td>
<td>0.0</td>
</tr>
<tr>
<td>0.10</td>
<td>-0.05</td>
<td>0.0</td>
</tr>
<tr>
<td>0.50</td>
<td>0.05</td>
<td>0.0</td>
</tr>
<tr>
<td>0.00</td>
<td>0.90</td>
<td>0.0</td>
</tr>
<tr>
<td>0.30</td>
<td>0.70</td>
<td>0.0</td>
</tr>
<tr>
<td>0.60</td>
<td>0.50</td>
<td>0.0</td>
</tr>
<tr>
<td>0.90</td>
<td>0.00</td>
<td>0.0</td>
</tr>
<tr>
<td>0.40</td>
<td>1.05</td>
<td>0.0</td>
</tr>
<tr>
<td>0.85</td>
<td>0.80</td>
<td>0.0</td>
</tr>
<tr>
<td>1.05</td>
<td>0.20</td>
<td>0.0</td>
</tr>
<tr>
<td>1.10</td>
<td>1.10</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Table 1: Data for "Cardinal" function.
<table>
<thead>
<tr>
<th>NPPR</th>
<th>N</th>
<th>n_x = n_y</th>
<th>NWKU</th>
<th>NWWKU</th>
<th>Time (Preprocessing)</th>
<th>Time (Evaluation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>60</td>
<td>3</td>
<td>334</td>
<td>273</td>
<td>4.2</td>
<td>14.3</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>7</td>
<td>619</td>
<td>506</td>
<td>7.5</td>
<td>17.6</td>
</tr>
<tr>
<td>6</td>
<td>500</td>
<td>18</td>
<td>3596</td>
<td>2893</td>
<td>70.5</td>
<td>21.2</td>
</tr>
<tr>
<td>6</td>
<td>1000</td>
<td>25</td>
<td>7441</td>
<td>6140</td>
<td>201.5</td>
<td>21.3</td>
</tr>
<tr>
<td>10</td>
<td>60</td>
<td>4</td>
<td>284</td>
<td>243</td>
<td>5.3</td>
<td>18.0</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>5</td>
<td>488</td>
<td>427</td>
<td>9.9</td>
<td>26.4</td>
</tr>
<tr>
<td>10</td>
<td>500</td>
<td>13</td>
<td>3014</td>
<td>2649</td>
<td>73.3</td>
<td>29.8</td>
</tr>
<tr>
<td>10</td>
<td>1000</td>
<td>19</td>
<td>6565</td>
<td>5804</td>
<td>202.6</td>
<td>31.7</td>
</tr>
<tr>
<td>15</td>
<td>60</td>
<td>3</td>
<td>224</td>
<td>199</td>
<td>6.8</td>
<td>23.0</td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>4</td>
<td>414</td>
<td>373</td>
<td>12.4</td>
<td>32.8</td>
</tr>
<tr>
<td>15</td>
<td>500</td>
<td>11</td>
<td>2856</td>
<td>2591</td>
<td>91.6</td>
<td>39.3</td>
</tr>
<tr>
<td>15</td>
<td>1000</td>
<td>15</td>
<td>5920</td>
<td>5439</td>
<td>258.5</td>
<td>40.6</td>
</tr>
</tbody>
</table>

Table 2: Storage/Times for Various Size Problems.
Figure 1: Cardinal Function

Figure 2: Saddle Function
Figure 3: Parent Function

Figure 4: NPPR = 6

Figure 5: NPPR = 10

Figure 6: NPPR = 15
References


[18] NIELSON, Gregory M. - "A Method for Interpolating Scattered Data Based Upon a Minimum Norm Network", a manuscript.


Appendix

SUBROUTINE LOTPS (MODE, NPPR, NPI, XI, YI, FI, NXO, XO, NYO, YO, IWK, NIWK, WK, NWK, NWKU, FO, KER)  
LOT 1
LOT 2
LOT 3
LOT 4
LOT 5
LOT 6
LOT 7
LOT 8
LOT 9
LOT 10
LOT 11
LOT 12
LOT 13
LOT 14
LOT 15
LOT 16
LOT 17
LOT 18
LOT 19
LOT 20
LOT 21
LOT 22
LOT 23
LOT 24
LOT 25
LOT 26
LOT 27
LOT 28
LOT 29
LOT 30
LOT 31
LOT 32
LOT 33
LOT 34
LOT 35
LOT 36

THIS SUBROUTINE SERVES AS A USER INTERFACE TO THE SET OF  
SUBROUTINES THAT IMPLEMENT FRANKE’S METHOD OF SURFACE INTERPOLATION. RECTANGULAR REGIONS ARE USED WITH PRODUCT CUBIC HERMITE WEIGHT FUNCTIONS. THE RECTANGLES ARE CHOSEN IN AN ATTEMPT TO OBTAIN ABOUT NPPR POINTS IN EACH REGION. THE SAME NUMBER OF GRID LINES IS USED IN EACH DIRECTION. LOCAL INTERPOLATION FUNCTIONS ARE THE THIN PLATE SPLINES DESCRIBED BY DUCHON AND OTHERS. A DESCRIPTION OF THE METHOD AND REFERENCES APPEAR IN SMOOTH INTERPOLATION OF SCATTERED DATA BY LOCAL THIN PLATE SPLINES - 
NAVAL POSTGRADUATE SCHOOL REPORT NPS-53-81-002 - 
RICHARD FRANKE - 
DEPARTMENT OF MATHEMATICS - 
NAVAL POSTGRADUATE SCHOOL - 
MONTEREY, CALIFORNIA 93940 - 
(408) 646-2758 / 2206 - 
DIFFICULTIES AND SUCCESSES WITH THIS PROGRAM SHOULD BE COMMUNICATED TO THE AUTHOR. - 
THE ARGUMENTS ARE AS FOLLOWS. - 
MODE - INPUT. INDICATES THE STATUS OF THE CALCULATION. - 
1, SET UP THE PROBLEM. COMPUTE THE COEFFICIENTS FOR THE LOCAL APPROXIMATIONS BY THIN PLATE SPLINES, AND RETURN THE GRID OF INTERPOLATED FUNCTION VALUES INDICATED BY NXO, XO, NYO, YO IN FO. - 
2, THE PROBLEM HAS BEEN SET UP PREVIOUSLY. CALCULATE THE GRID OF INTERPOLATED POINTS INDICATED BY NXO, XO, NYO, YO IN FO. THE PROGRAM ASSUMES THAT THE ARRAYS XI, YI, IWK, AND WK ARE UNCHANGED FROM THE PREVIOUS CALL.
C NPPR - INPUT. DESIRED AVERAGE NUMBER OF POINTS PER REGION. LOT 37
C THE SUGGESTED VALUE IS TEN. SHOULD BE AT LEAST LOT 38
C FOUR. VALUES LARGER THAN FIFTEEN COULD REQUIRE LOT 39
C MINOR PROGRAM MODIFICATIONS TO ALLOW MORE LOT 40
C STORAGE FOR THE EQUATION SOLVER. THIS DEPENDS LOT 41
C ON THE DISPOSITION OF THE POINTS. LOT 42
C IF THE USER WISHES TO SPECIFY HIS OWN GRID LINES LOT 43
C X TILDA AND Y TILDA, HE MAY DO SO BY SETTING LOT 44
C NPPR = 0 AND SETTING NECESSARY VALUES IN THE LOT 45
C ARRAYS IWK AND WK, AS NOTED BELOW. DATA WHICH LOT 46
C HAS A POOR DISTRIBUTION OVER THE REGION OF LOT 47
C INTEREST SHOULD PROBABLY HAVE THE GRID SPECIFIED LOT 48
C THIS IS ALSO ADVISABLE IF THE X-Y POINTS OCCUR LOT 49
C ON LINES. LOT 50
C
C NPI - INPUT. NUMBER OF INPUT DATA POINTS. LOT 51
C XI - \ LOT 52
C YI - INPUT. THE DATA POINTS (XI,YI,FI), I=1,...,NPI. LOT 53
C FI - / LOT 54
C NXO - INPUT. THE NUMBER OF XO VALUES AT WHICH THE INTERP- LOT 55
C OLABRATION FUNCTION IS TO BE CALCULATED. LOT 56
C XO - INPUT. THE VALUES OF X AT WHICH THE INTERPOLATION LOT 57
C FUNCTION IS TO BE CALCULATED. THESE SHOULD LOT 58
C BE IN INCREASING ORDER FOR MOST EFFICIENT LOT 59
C EVALUATION. LOT 60
C NYO - INPUT. THE NUMBER OF YO VALUES AT WHICH THE INTERP- LOT 61
C OLABRATION FUNCTION IS TO BE CALCULATED. LOT 62
C YO - INPUT. THE VALUES OF Y AT WHICH THE INTERPOLATION LOT 63
C FUNCTION IS TO BE CALCULATED. THESE SHOULD LOT 64
C BE IN INCREASING ORDER FOR MOST EFFICIENT LOT 65
C EVALUATION. LOT 66
C
C IWK - INPUT AND OUTPUT. THIS ARRAY IS OUTPUT WHEN MODE = 1 LOT 67
C AND IS INPUT WHEN MODE = 2. THIS MUST BE LOT 68
C AN ARRAY DIMENSIONED APPROXIMATELY 6*NPI. LOT 69
C WHEN NPPR IS INPUT AS ZERO THE USER MUST LOT 70
C SPECIFY THE NUMBER OF VERTICAL GRID LINES (THE LOT 71
C NUMBER OF X TILDA VALUES) IN IWK(1) AND THE LOT 72
C NUMBER OF HORIZONTAL GRID LINES (THE NUMBER OF LOT 73
C Y TILDA VALUES) IN IWK(2). LOT 74
NIWK - INPUT.  ON ENTRY WITH MODE = 1 THIS MUST BE SET TO THE LOT 75
DIMENSION OF THE ARRAY IWK.

NIWKU - OUTPUT.  THE ACTUAL NUMBER OF LOCATIONS USED IN THE LOT 76
ARRAY IWK.

WK - INPUT AND OUTPUT.  THIS ARRAY IS OUTPUT WHEN MODE = 1 LOT 77
AND IS INPUT WHEN MODE = 2.  THIS MUST BE AN LOT 78
ARRAY DIMENSIONED APPROXIMATELY 7*NPI LOCATIONS.LOT 79
LOCATIONS.

WHEN NPPR IS INPUT AS ZERO THE USER MUST SPECIFY LOT 80
THE VALUES OF X TILDA AND Y TILDA AS FOLLOWS.  LOT 81
WK(2), ..., WK(NXG+1) ARE THE NXG (= IWK(1)) LOT 82
X GRID VALUES, X(I) TILDA, IN INCREASING ORDER.  LOT 83
TYPICALLY WK(1) = MIN X(I), ALTHOUGH IT NEED LOT 84
NOT BE.  WK(1) MUST BE LESS THAN OR EQUAL TO LOT 85
WK(2), AND SHOULD BE LESS THAN OR EQUAL TO LOT 86
MIN X(I).  WK(NXG+2) IS USUALLY MAX X(I), ALLOT 87
THOUGH IT NEED NOT BE.  WK(NXG+2) MUST BE LOT 88
GREATER THAN WK(NXG+1), AND SHOULD BE GREATER LOT 89
THAN OR EQUAL TO MAX X(I).

NIWK - INPUT.  ON ENTRY WITH MODE = 1 THIS MUST BE SET TO THE LOT 90
DIMENSION OF THE ARRAY WK.

NIWKU - OUTPUT.  THE ACTUAL NUMBER OF LOCATIONS USED IN THE LOT 91
ARRAY WK.

FO - OUTPUT.  VALUES OF THE INTERPOLATION FUNCTION AT THE LOT 92
GRID OF POINTS INDICATED BY NXO, XO, NYO, YO.  LOT 93
FO IS ALWAYS DIMENSIONED (NXO,NYO) IN THE LOT 94
CALLING PROGRAM.
KER - OUTPUT. RETURN INDICATOR.

= 0, NORMAL RETURN.

= -1, PROBLEM HAS NOT BEEN PREVIOUSLY SET UP (LOTPS CALLED WITH MODE = 1).

= 1, ERROR RETURN FROM CLOTPS, SINGULAR MATRIX IN THE LOT.

= 2, ERROR RETURN FROM CLOTPS. SOME RECTANGLE (I, J) HAS MORE THAN THE ALLOWED NUMBER OF POINTS ASSOCIATED WITH IT. SEE CLOTPS FOR THE FIX.

= 3, PREVIOUS ERROR RETURN FROM CLOTPS HAS NOT BEEN CORRECTED.

= 4, IXK AND/OR WK ARRAYS HAVE NOT BEEN DIMENSIONED LARGE ENOUGH IN THE CALLING PROGRAM. REDIMENSION IXK AND WK TO AT LEAST THE SIZE INDICATED BY NIWKU AND NWKU, RESPECTIVELY.

= 5, MODE IS OUT OF RANGE.

SUBPROGRAMS CALLED BY THIS ROUTINE ARE GRID, LOCLIP, CLOTPS, AND EULIPS. ALSO REQUIRED ARE USRTA FROM THE IMSL LIBRARY AND THE ROUTINES DECOMP AND SOLVE FROM FORSYTHE, MALCOLM, AND MOLER.
<table>
<thead>
<tr>
<th>DISTRIBUTION LIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defense Technical Information Center</td>
</tr>
<tr>
<td>Cameron Station</td>
</tr>
<tr>
<td>Alexandria, VA 22314</td>
</tr>
<tr>
<td>Dudley Knox Library</td>
</tr>
<tr>
<td>Naval Postgraduate School</td>
</tr>
<tr>
<td>Monterey, CA 93940</td>
</tr>
<tr>
<td>Dean of Research</td>
</tr>
<tr>
<td>Naval Postgraduate School</td>
</tr>
<tr>
<td>Monterey, CA 93940</td>
</tr>
<tr>
<td>Department of Mathematics</td>
</tr>
<tr>
<td>C. O. Wilde, Chairman</td>
</tr>
<tr>
<td>Professor F. D. Faulkner</td>
</tr>
<tr>
<td>Professor A. Schoenstadt</td>
</tr>
<tr>
<td>Professor R. Franke</td>
</tr>
<tr>
<td>Naval Postgraduate School</td>
</tr>
<tr>
<td>Monterey, CA 93940</td>
</tr>
<tr>
<td>Dr. Richard Lau</td>
</tr>
<tr>
<td>Office of Naval Research</td>
</tr>
<tr>
<td>1030 East Green St.</td>
</tr>
<tr>
<td>Pasadena, CA 91106</td>
</tr>
<tr>
<td>Professor R. E. Barnhill</td>
</tr>
<tr>
<td>Department of Mathematics</td>
</tr>
<tr>
<td>University of Utah</td>
</tr>
<tr>
<td>Salt Lake City, UT 84112</td>
</tr>
<tr>
<td>Professor G. M. Nielson</td>
</tr>
<tr>
<td>Department of Mathematics</td>
</tr>
<tr>
<td>Arizona State University</td>
</tr>
<tr>
<td>Tempe, AZ 85281</td>
</tr>
<tr>
<td>Chief of Naval Research</td>
</tr>
<tr>
<td>ATT: Mathematics Program</td>
</tr>
<tr>
<td>Arlington, VA 22217</td>
</tr>
<tr>
<td>Rosemary E. Chang</td>
</tr>
<tr>
<td>Sandia Laboratories</td>
</tr>
<tr>
<td>Applied Mathematics, Division 2-8235</td>
</tr>
<tr>
<td>Livermore, CA 94550</td>
</tr>
<tr>
<td>L. H. Seitelman</td>
</tr>
<tr>
<td>Pratt and Whitney Aircraft</td>
</tr>
<tr>
<td>400 Main Street</td>
</tr>
<tr>
<td>East Hartford, CT 06108</td>
</tr>
<tr>
<td>William W. Bozek</td>
</tr>
<tr>
<td>Engineering Data Processing</td>
</tr>
<tr>
<td>Philips ECG, Inc.</td>
</tr>
<tr>
<td>Johnston Street</td>
</tr>
<tr>
<td>Seneca Falls, NY 13148</td>
</tr>
</tbody>
</table>