ROCKY MOUNTAIN
JOURNAL OF MATHEMATICS
Volume 14, Number 1, Winter 1984

A TRIANGLE-BASED C^1 INTERPOLATION METHOD

R. J. RENKA AND A. K. CLINE

ABSTRACT. This paper discusses methods and software for C^1 interpolation at arbitrarily distributed data points in the plane. The primary results presented here are derivative-estimation procedures which lead to interpolatory surfaces constituting very accurate approximations for a variety of test functions.

I. Introduction. This paper is addressed to the following problem: given a set of nodes (abscissae) (X_i, Y_i), arbitrarily distributed in the $X - Y$ plane, with corresponding ordinates Z_i, $i = 1, \ldots, N$, construct a C^1 bivariate function $F(X, Y)$ which interpolates the data values, i.e., $F(X_i, Y_i) = Z_i$, $i = 1, \ldots, N$.

This problem arises in a wide variety of scientific fields in which the data represents observed or computed values of some physical phenomenon such as temperature, rainfall, elevation, or stress obtained by finite element methods. It is often impractical or impossible to obtain data at all of the points at which values are desired, thus making it necessary to compute values by an approximation technique such as interpolation. A smooth interpolatory surface is especially desirable when a visual impression of the data is called for. Since most available software for contour and surface perspective plotting requires that data be specified on a uniform mesh, it may be necessary to interpolate from a nonuniform mesh to a set of rectangular grid points.

The method consists of the following three steps.

1) Partition the convex hull of the set of nodes (the smallest convex region containing the nodes) into triangles by connecting the nodes with line segments.

2) Estimate partial derivatives of F with respect to X and Y at each of the nodes using the data values on either a set of nearby nodes (a local method) or all of the nodes (a global method).

3) For an arbitrary point (X, Y) in the convex hull of the set of nodes, determine which triangle contains the point, and compute an interpolated

Funding for the first author was provided by the Mathematics development program in Technical Applications, Computer Sciences at ORNL. The second author was partially supported by NASA grant NAG1-79.

value $F(X, Y)$ using the data values and estimated partial derivatives at each of the three vertices of the triangle. Capability of extrapolation for a point outside of the convex hull is also provided.

This basic solution method has been employed by McLain [9], Lawson [8], and Akima [1] with the exception that McLain and Akima estimate different quantities at Step 2 for use in the interpolation phase at Step 3. We have employed Lawson's interpolation algorithm in Step 3 but differ in our approach to the first two steps. The triangulation phase is described in a separate report [2] and will not be discussed here.

Software implementing the method is listed in Renka [11] and machine-readable code may be obtained from the first author.

§II describes the interpolant, §III discusses local derivative-estimation procedures, §IV is addressed to global derivative estimates, and test results are presented in §V.

II. C^1 interpolant. Our triangle-based interpolation method computes a value at a point based only on data values and first partial derivatives at the three vertices of the triangle containing the point. Given a point P and the coordinates of the vertices of a triangle containing P along with data values and estimated partial derivatives at the vertices, we compute $F(P)$ where

1) F is a true cubic (not bicubic) in each of the three subtriangles (of equal area) defined by connecting the vertices to the barycenter,

2) F is once continuously differentiable over the triangle, but has second derivative discontinuities across subtriangle boundaries, and

3) along each triangle side F is the Hermite cubic interpolant of the data values and tangential (directional) derivatives at the endpoints, and the derivative of F in the direction normal to each triangle side varies linearly, interpolating the normal derivatives at the endpoints.

This piecewise cubic element F is due to Clough and Tocher [3]. C^1 continuity over a union of triangles in a triangulation follows from the fact that the tangential and normal derivatives (and hence the partial derivatives) at a point on a triangle side are completely determined by their values at the endpoints of the side. As we would expect, interpolation is exact for data taken from the six-parameter family of quadratics, but not for cubics.

$F(P)$ is evaluated by an efficient computational procedure due to Lawson [7]. Our implementation of this procedure also provides for computing derivatives of F at the user's option. Table 2.1 specifies the operation counts required to evaluate F, as well as the operation counts associated with evaluating F and its partial derivatives F_x and F_y. Subtractions are included with additions and the number of compares is an expected value.

TABLE 2.1. Operation Counts for Interpolation

	Additions	Multiplies	Divisions	Compares	Assignment Statements
F	62	54	8	5 2/3	50
F, F_x, F_y	170	142	14	5 2/3	106

We now consider a method for extrapolation to points which are exterior to the convex hull of the nodes. The method consists of extending F linearly beyond the mesh boundary, i.e., we extrapolate to P by passing a linear function of one variable through the value and directional derivative of F at Q where Q is the projection of P onto the boundary. Thus

$$F(P) = F(Q) + \langle \nabla F(Q), P - Q \rangle$$

where the angular brackets denote inner product.

An exterior point lies in either a semi-infinite rectangle or a semi-infinite wedge defined by the lines which pass through the boundary nodes perpendicular to the boundary edges. See Figure 2.1. Q is easily determined by a traversal of the boundary and when Q lies on the interior of a boundary edge, $F(P)$ is readily computed from the properties of F on a triangle side.

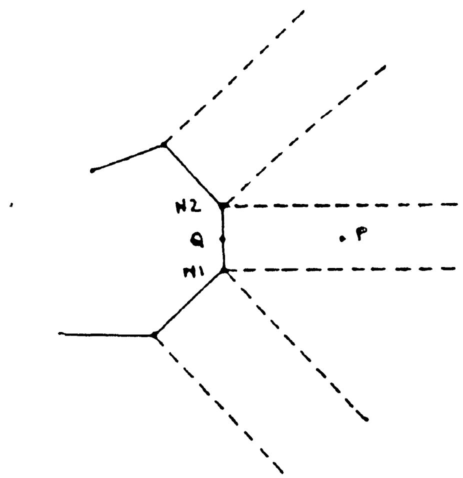

FIGURE 2.1. Partition of the Exterior Region into Semi-infinite Rectangles and Wedges.

With this extension to points exterior to the triangulation, the interpolatory surface F is defined and continuous over the entire $X - Y$ plane but has discontinuous first derivatives across boundaries between the semi-infinite regions. However, this lack of smoothness is negligible for points close to the triangulation boundary, and extrapolation is not appropriate for points far from the boundary. Thus, we feel a method for C^1 extrapolation is not worth the computational effort.

Details of the computational procedures for both interpolation and extrapolation are contained in [11].

III. Local derivative estimation. The interpolant F described in the previous section requires estimated partial derivatives at the nodes, and these must be determined by a local method in order that the interpolation method be local. Such a method is discussed in the first subsection below. The computational procedure associated with the method is described in the second subsection.

METHOD. For $k = 1, \ldots, N$, choose the partial derivatives at node k to be the partials of a quadratic function of X and Y which interpolates the data value Z_k at node k and which fits the data values at a set S_k of nearby nodes in a weighted least squares sense. Thus, derivative estimates, and hence interpolated values, are exact for quadratic data unless $N < 6$. (For $3 \leqq N \leqq 5$, the data is fitted with a linear rather than a quadratic function.)

This basic approach to derivative estimation is the method employed by Lawson [8] and is similar to interpolation methods proposed by McLain [9] and Franke and Nielson [4]. The difficulty arises in the choice of nearby nodes and weights. We use the following weighting function proposed by Franke and Little and recommended by Franke and Nielson [4]. For each node k, let D_i denote the distance between nodes i and k for $i = 1, \ldots, N$ and define the weight associated with node i, $i \neq k$, by

$$W_i \equiv (R_k - D_i)_+/(E_k * D_i) \text{ where } (R_k - D_i)_+ \equiv \begin{cases} R_k - D_i & \text{if } D_i \leqq R_k. \\ 0 & \text{if } D_i \geqq R_k \end{cases}$$

R_k is termed the radius of influence about node k. The weights are well defined when the nodes are distinct ($D_i > 0$ for $i \neq k$). Once R_k is chosen, S_k is taken to be the set of nodes within the radius of influence. Then for node j in S_k we have $W_j = (R_k - D_j)/(R_k * D_j) = 1/D_j - 1/R_k > 0$. Thus, as in the case of weighting by inverse distance ($W_j = 1/D_j$), this weighting method has the property that more distant data have less influence on derivative estimates.

We have obtained good results with the following choice of R_k. For $N \geqq 9$ let D be the distance from node k to the eighth closest node (other than k) to k. D is defined to be the distance from k to the furthest node from k if $N < 9$. Then R_k is taken to be the distance between k and the closest node to k (if it exists) whose distance from k is greater than D. If no such node exists, we arbitrarily choose $R_k = 2D$. Thus, for N sufficiently large, S_k consists of the eight closest nodes to k along with all nodes which lie at the same distance as the eighth closest.

In arriving at the derivative-estimation scheme outlined above, we tested a number of alternatives. These include weighting by inverse distance or various powers of inverse distance, taking S_k to be all nodes within some fixed radius (independent of k), and choosing S_k on the basis of adjacency information in the triangulation. Our testing consisted of computing both derivative and interpolation errors for sets of randomly chosen interpolation points. We used two sets of pseudo-randomly generated nodes with sizes 100 and 1000, and data values were taken from four simple bivariate test functions: cubic, quartic, exponential, and trigonometric.

Choosing S_k to contain nine nodes rather than eight led to comparable results, while other choices in the range 6 to 11 were slightly inferior. The inclusion of all neighbors of node k in the least squares fit has the apparent advantage that nodes are chosen on the basis of location relative to k as well as distance from k. However, this method did not prove to be effective in practice. Choosing S_k to be all nodes within a fixed distance R from node k also produced consistently inferior results for all choices of R. In fact, results were comparable or inferior when the constant R was optimized for each test function. Additional test results are presented in §V.

COMPUTATIONAL PROCEDURE. The quadratic to be fitted to Z_k and the data values at the nodes in S_k may be written $G(X, Y) = Z_k + a(X - X_k)^2 + b(X - X_k)(Y - Y_k) + c(Y - Y_k)^2 + ZX_k(X - X_k) + ZY_k(Y - Y_k)$. Then $G(X_k, Y_k) = Z_k$ and the partial derivatives of G at (X_k, Y_k) are ZX_k and ZY_k. The vector of unknown coefficients

$$\mathbf{u} = [a\ b\ c\ ZX_k\ ZY_k]^T$$

is determined by minimizing

$$\sum_{j\in S_k} [W_j*(G(X_j, Y_j) - Z_j)]^2 = \|\mathbf{Au} - \mathbf{v}\|^2$$

where $\mathbf{A}$ is an $m \times 5$ matrix, m being the number of elements in S_k. The row of $\mathbf{A}$ corresponding to node j is

$$W_j[(X_j - X_k)^2\,(X_j - X_k)(Y_j - Y_k)\,(Y_j - Y_k)^2\,(X_j - X_k)(Y_j - Y_k)]$$

and the corresponding component of $\mathbf{v}$ is $W_j(Z_j - Z_k)$.

The least squares minimization problem has a unique solution if and only if the columns of $\mathbf{A}$ are linearly independent, i.e., if and only if $G(X_j, Y_j) - Z_k = 0$ for all (X_j, Y_j) in $S_k \Rightarrow \mathbf{u} = 0$, or equivalently, $Q(X_j, Y_j) = 0$ for all (X_j, Y_j) in $S_k \cup (X_k, Y_k) \Rightarrow Q(X, Y) = 0$ for all (X, Y) for a general quadratic Q. Thus, a solution to the least squares problem is unique if and only if there is no conic section (zero contour of a linear or quadratic) which contains all the elements of $S_k \cup (X_k, Y_k)$.

Once all elements of S_k have been added to the least squares system, the system is tested for ill-conditioning and, if necessary, an additional node is added and the test is repeated. If it is not possible to add another node, a Marquardt stabilization factor is used to damp out the coefficients of the quadratic terms. Thus, the method is guaranteed to produce a unique solution unless all nodes are collinear.

The value of R_k and the indices of the elements of S_k are determined by the triangulation software [**2**, Algorithm 7]. A 6×6 array is used to store the transpose of the augmented regression matrix which is reduced to an upper triangular system by Givens rotations. The transpose of the matrix is stored so that the routines which set up an equation and apply a rotation operate on matrix elements stored in contiguous locations. Note that the unknown coefficients are ordered so that we need only solve a 2×2 triangular system for ZX_k and ZY_k.

IV. **Global derivative estimation.** We have investigated two global methods for computing derivative estimates, retaining the triangle-based interpolant previously described. Both methods involve the iterative solution of a linear system and require no storage other than the $2N$ locations for the derivatives. Method I was found to be computationally infeasible but is presented as motivation for the second method. Method II resulted in accuracy comparable to that of the local method described in the previous section, and required substantially less computation time. Thus, while Method II produces a global interpolant, it does not suffer the usual limitations of global methods in terms of storage requirements and computation time. The local method, however, allows derivative estimates to be computed as needed for each interpolation point, thus eliminating the necessity of storing derivatives. We have therefore chosen to include both methods in the software package.

METHOD I. Let $\mathbf{u}$ denote the vector of length $2N$ containing the vector of X-partials at the nodes followed by the vector of Y-partials, and let H denote the convex hull of the nodes. In a procedure somewhat analogous to the development of cubic splines, we determine the value of $\mathbf{u}$ which minimizes the $L2$ norm of the linearized curvature of the interpolant $F(X, Y)$ over H.

We take the curvature of F to be the vector 2-norm of the curvatures,

κ_1 and κ_2, in the directions of the X and Y axes. These are given by the solution of the generalized eigenvalue problem

$$\mathbf{Ax} = \kappa^{-1}\,\mathbf{Bx} \text{ for } \mathbf{A} = \begin{bmatrix} 1 + F_x^2 & F_xF_y \\ F_xF_y & 1 + F_y^2 \end{bmatrix}, \quad \mathbf{B} = a^{-1}\begin{bmatrix} F_{xx} & F_{xy} \\ F_{xy} & F_{yy} \end{bmatrix}$$

where $a = (1 + F_x^2 + F_y^2)^{1/2}$. To obtain the linearized curvatures, we assume the partials of F are small and take F_x^2, F_y^2, and F_xF_y to be zero. Then $\mathbf{A}$ is the identity matrix $\mathbf{I}$, and $a = 1$. Hence $|\mathbf{A} - \kappa^{-1}\mathbf{B}| = |\mathbf{I} - \kappa^{-1}\mathbf{B}| = 0 \Rightarrow |\kappa\,\mathbf{I} - \mathbf{B}| = 0$, i.e., κ_1 and κ_2 are the eigenvalues of $\mathbf{B}$. Since $\mathbf{B}$ is symmetric, its Frobenius norm is equal to that of the diagonal matrix of its eigenvalues. It follows that $\kappa_1^2 + \kappa_2^2 = F_{xx}^2 + 2F_{xy}^2 + F_{yy}^2$. Thus, the problem is to find $\mathbf{u}$ which minimizes the quadratic functional

$$Q(\mathbf{u}) = \int_H (F_{xx}^2 + 2F_{xy}^2 + F_{yy}^2)dH.$$

The interpolant can be written

$$F(X, Y) = \sum_{i=1}^{2N} u_i{*}f_i(X, Y) + \sum_{i=1}^{N} Z_i{*}g_i(X, Y)$$

where f_i and g_i are the appropriate patch functions (cardinal functions with local support). We may define a symmetric bilinear form on the space of piecewise cubic interpolants by

$$a(F, G) \equiv \int_H (F_{xx}G_{xx} + 2F_{xy}G_{xy} + F_{yy}G_{yy})dH.$$

Then $Q(\mathbf{u}) = a(F, F) = \mathbf{u}^T\mathbf{A}\mathbf{u} + \mathbf{b}^T\mathbf{u} + c$
where

$$A_{ij} = a(f_i, f_j), \qquad b_i = 2\sum_{j=1}^{N} a(f_i, g_j)Z_j,$$

and

$$c = a\left(\sum_{i=1}^{N} Z_i{*}g_i,\ \sum_{j=1}^{N} Z_j{*}g_j\right).$$

To show that the minimization problem has a unique solution, it suffices to show that $\mathbf{A}$ is positive definite. Suppose $a(F, F) = 0$. Then, since the terms F_{xx}^2, F_{xy}^2, and F_{yy}^2 are nonnegative, they must integrate to zero over any portion of H—in particular, over each of the subtriangles on which F is a true cubic. It follows that F is linear on H. Since $\mathbf{A}$ is independent of the data values, we may assume without loss of generality that $Z_i = 0$, $i = 1, \ldots, N$. Then F has at least three zeros in H and is therefore identically zero. Thus, we have shown that $F \neq 0$ implies $a(F, F) > 0$.

Note that the minimization problem is the variational equivalent of the biharmonic equation $\nabla^2(\nabla^2 F) = 0$. This is verified by substituting the integrand I of Q into the Euler-Lagrange equation

$$\frac{\partial^2}{\partial x^2}\left(\frac{\partial I}{\partial F_{xx}}\right) + \frac{\partial^2}{\partial x \partial y}\left(\frac{\partial I}{\partial F_{xy}}\right) + \frac{\partial^2}{\partial y^2}\left(\frac{\partial I}{\partial F_{yy}}\right) = 0.$$

The biharmonic equation governs the deflection due to bending in a thin plate, and Q may be thought of as the strain energy associated with this bending.

Setting the partials of Q with respect to u_j to zero, we obtain the order $2N$ linear system $2\mathbf{Au} + \mathbf{b} = \mathbf{0}$ where $\mathbf{A}$ is symmetric, positive definite, and sparse. Since f_i and f_{N+i}, $i = 1, \ldots, N$, have support on the set of triangles containing node i, and since the average number of neighbors per node is less than six, the expected number of nonzero elements in a row of $\mathbf{A}$ is $2(6 + 1) = 14$ and $\mathbf{A}$ has at most $28N$ nonzeros. Since the zero structure depends on the ordering of the nodes, a reordering algorithm might result in increased storage and computational efficiency for a direct solution method. In order to minimize the storage requirement, however, we chose to employ an iterative method.

Note that the integrands involved in the residual $2\mathbf{Au} + \mathbf{b}$ are true quadratic functions of X and Y on each subtriangle for which F is cubic. Hence, in order to evaluate the integrals over a triangle, we employed a nine-point quadrature rule composed of three-point rules, exact for quadratics, on each subtriangle.

For the 100-node triangulation of Fig. 5.1, $\mathbf{A}$ was found to have an estimated condition number of 10^{10}. Thus, while guaranteed to converge, the Gauss-Seidel method applied to this system was so slow as to be computationally infeasible. In an attempt to increase the rate of convergence, we applied the Successive Overrelaxation (SOR) method with an algorithm for estimating the optimal relaxation parameter borrowed from ITPACK [6].There is no theory which guarantees that this method would result in an improvement over Gauss-Seidel and no improvement was observed. No other methods were tried since Method II was found to be generally more accurate as well as more efficient. We conclude that Method I cannot be recommended.

METHOD II. The following derivative-estimation procedure is essentially the same as the method used by Nielson (in conjunction with a different interpolant) in his minimum norm network [10]. Let P_k denote the patch of triangles containing node k and define

$$Q_k(\mathbf{ZX}, \mathbf{ZY}) \equiv \int_{P_k} (F_{xx}^2 + 2F_{xy}^2 + F_{yy}^2)dx\,dy$$

where $\mathbf{ZX}$ and $\mathbf{ZY}$ are the N-vectors of unknown partials at the nodes.

Then the linear system described in the previous subsection may be written

$$\frac{\partial Q}{\partial ZX_k} = \frac{\partial Q_k}{\partial ZX_k} = 0, \frac{\partial Q}{\partial ZY_k} = \frac{\partial Q_k}{\partial ZY_k} = 0$$

for $k = 1, \ldots, N$.

In order to reduce the computational effort associated with Method I, we introduced the following approximation to Q_k. Let $\hat{Q}_k$ be the quadratic functional defined by the sum of the squares of the $L2$ norms of the linearized curvatures of F along the line segments from node k to neighbors of k.

Thus, for m arcs incident on node k,

$$\hat{Q}_k(\mathbf{ZX}, \mathbf{ZY}) = \sum_{i=1}^{m} \int_0^{L_i} [W_i''(t)]^2 \, dt$$

where t varies along the arcs, L_i is the length of arc i, and $W_i(t)$ is the restriction of the interpolant F to arc i. Expressions for $\hat{Q}_k$ can be derived by using the fact that W_i is the Hermite cubic interpolant of the values and directional derivatives of F at the endpoints of the arc.

If the unknowns are ordered so that **ZY** follows **ZX**, the matrix in the linear system obtained by replacing Q_k by $\hat{Q}_k$ has the same zero structure as the matrix associated with Method I. To solve the system, however, we employed a block Gauss-Seidel method with 2 × 2 blocks. An iteration is as follows. For $k = 1, \ldots, N$ set ZX_k and ZY_k to the solution of the 2 × 2 linear system

$$\frac{\partial \hat{Q}_k}{\partial ZX_k} = 0, \frac{\partial \hat{Q}_k}{\partial ZY_k} = 0.$$

(The k-th components of **ZX** and **ZY** are updated at each step.) The proof that the system is positive definite and hence that the Gauss-Seidel method converges is completely analogous to the proof given for Method I.

Test results for two sets of nodes and six test functions are presented in the following section. Starting with the derivatives initialized to zero, effective convergence was achieved with only three iterations in all cases. Four iterations resulted in slightly smaller derivative errors but no significant difference in interpolation errors. Using the optimizing compiler on the IBM 3033, the time requirement for a pair of derivative estimates was found to be approximately .1467 milliseconds for each iteration —.44 milliseconds for three iterations. (The operation count for an iteration is proportional to N and the observed rate of convergence is independent of N.) The local method was found to require 2.09 milliseconds per node and is thus slower by a factor of 4.75.

V. **Test results.** Franke [5] has published a comparison of 29 methods for smooth interpolation of scattered data in the plane. Among the criteria used to judge the methods were accuracy, efficiency, storage requirements, and appearance of the resulting surface. We have duplicated some of Franke's tests and replicated his results so that our method could be included in the comparison. Our computations were performed on the IBM 3033 system at ORNL.

Data values were taken from the following test functions:

$$\begin{aligned}
\text{EXPONENTIAL}\quad F1 &= .75\exp[-((9X-2)^2+(9Y-2)^2)/4] \\
&\quad + .75\exp[-(9X+1)^2/49-(9Y+1)/10] \\
&\quad + .5\exp[-((9X-7)^2+(9Y-3)^2)/4] \\
&\quad - .2\exp[-(9X-4)^2-(9Y-7)^2] \\
\text{CLIFF}\quad F2 &= [\tanh(9Y-9X)+1]/9 \\
\text{SADDLE}\quad F3 &= [1.25+\cos(5.4Y)]/[6+6(3X-1)^2] \\
\text{GENTLE}\quad F4 &= \exp[-(81/16)((X-.5)^2+(Y-.5)^2)]/3 \\
\text{STEEP}\quad F5 &= \exp[-(81/4)((X-.5)^2+(Y-.5)^2)]/3 \\
\text{SPHERE}\quad F6 &= \sqrt{64-81((X-.5)^2+(Y-.5)^2)}/9-.5.
\end{aligned}$$

Data values were computed on a set of 100 nodes determined in a manner which led to a somewhat uniform distribution, and whose convex hull nearly covers the unit square. A Thiessen triangulation [2] of this set is depicted in Figure 5.1.

Table 5.1 contains mean and maximum absolute interpolation errors on the 1089 grid points of a 33 by 33 uniform mesh in the unit square. Thirteen of the grid points required extrapolation and were included in the computation of the error norms for all methods except Lawson's. The first four methods use the interpolant described in §II but with different derivatives. The table includes the four most accurate local methods tested by Franke, along with Nielson's method which uses essentially the same global derivative-estimation procedure as our method. All of them produce a C^1 surface, and all are triangle-based except the modified Shepard's method. We feel that the low storage and computational requirements of the methods using global derivative estimates justifies comparison of those methods with the local methods. The results associated with local derivative estimates have been verified by Franke.

The maximum errors associated with the first four methods demonstrate that errors in the interpolant are sometimes offset by derivative errors, and thus improved derivative estimates do not necessarily lead to a more accurate interpolant. With regard to mean interpolation errors for our local method, in no case was more than half a digit of precision lost due to

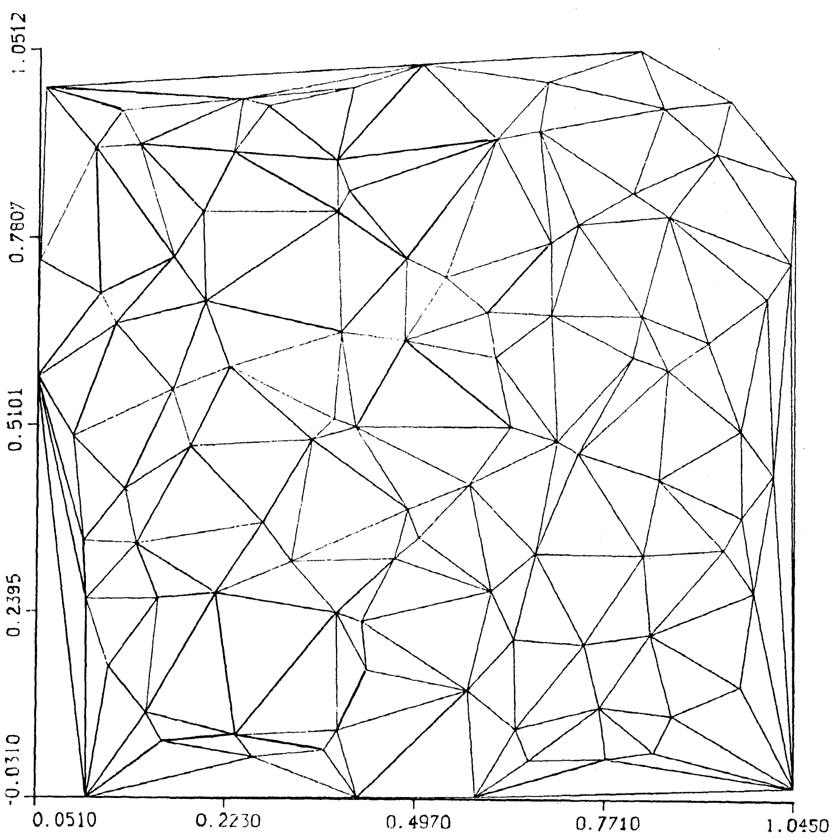

FIGURE 5.1. 100-Node Thiessen Triangulation

errors in the derivative estimates. As expected, the largest errors occurred near the boundary where there are long thin triangles. A comparison of root-mean-square interpolation errors resulted in much the same pattern as that of the mean errors tabulated here.

Figure 5.2 depicts a surface perspective plot of $F1$ (exponential) which was generated from the 1089 interpolated values associated with our local method. The only apparent flaw occurs near the corner at the bottom of the plot where the surface should be nearly flat. This is apparently due to the long thin triangles along the right side of the bloundary.

In order to test the effect of larger variations in the density of the nodal distribution, the tests described above were repeated with the 33-node set which Franke generated for this purpose [5]. In this case, extrapolation was required at 125 of the 1089 uniform grid points. Results are tabulated in Table 5.2.

TABLE 5.1. Interpolation Errors for 100-Node Set

Method	*F*1	*F*2	*F*3	*F*4	*F*5	*F*6
Mean Interpolation Errors						
True Derivatives	.00413	.00086	.00018	.00008	.00035	.00010
Global Derivative Estimates	.00540	.00191	.00094	.00046	.00100	.00079
Local Derivative Estimates	.00619	.00241	.00076	.00035	.00146	.00026
Lawson	.00783	.00221	.00149	.00061	.00154	.00038
Nielson-Franke Quadratic	.00741	.00265	.00110	.00058	.00176	.00022
Modified Quadratic Shepard	.00785	.00264	.00112	.00065	.00182	.00026
Akima Mod III	.00729	.00293	.00105	.00049	.00171	.00058
Nielson Minimum Norm	.00537	.00181	.00091	.00047	.00101	.00077
Maximum Interpolation Errors						
True Derivatives	.0985	.0341	.0069	.0019	.0084	.0056
Global Derivative Estimates	.0499	.0484	.0217	.0032	.0196	.0115
Local Derivative Estimates	.0505	.0320	.0108	.0020	.0190	.0066
Lawson	.0951	.0280	.0565	.0090	.0216	.0095
Nielson-Franke Quadratic	.0782	.0721	.0168	.0052	.0206	.0034
Modified Quadratic Shepard	.0573	.0468	.0125	.0039	.0218	.0036
Akima Mod III	.0520	.0958	.0142	.0033	.0212	.0080
Nielson Minimum Norm	.0492	.0424	.0195	.00303	.0195	.0117

TABLE 5.2. Interpolation Errors for 33-Node Set

Method	*F*1	*F*2	*F*3	*F*4	*F*5	*F*6
Mean Interpolation Errors						
True Derivatives	.01656	.00473	.00302	.00118	.00554	.00059
Global Derivative Estimates	.03020	.00812	.01047	.00364	.01049	.00284
Local Derivative Estimates	.03201	.00852	.00893	.00353	.00974	.00167
Lawson	.0462	.0126	.0133	.00552	.0129	.00210
Nielson-Franke Quadratic	.0326	.0137	.00939	.00422	.0104	.00585
Modified Quadratic Shepard	.0340	.0121	.00907	.00451	.0113	.00400
Akima Mod III	.0372	.0106	.0104	.00394	.0119	.00556
Nielson Minimum Norm	.0305	.00800	.0102	.00371	.0106	.00273
Maximum Interpolation Errors						
True Derivatives	.1400	.0310	.0317	.0119	.0778	.0038
Global Derivative Estimates	.1487	.0582	.0577	.0211	.1137	.0192
Local Derivative Estimates	.1609	.0604	.0513	.0189	.0953	.0128
Lawson	.287	.0956	.0685	.0269	.139	.0137
Nielson-Franke Quadratic	.150	.0878	.0679	.0312	.0835	.0983
Modified Quadratic Shepard	.184	.0876	.0724	.0272	.110	.101
Akima Mod III	.164	.0680	.0597	.0204	.115	.0819
Nielson Minimum Norm	.150	.0582	.0571	.0214	.115	.0186

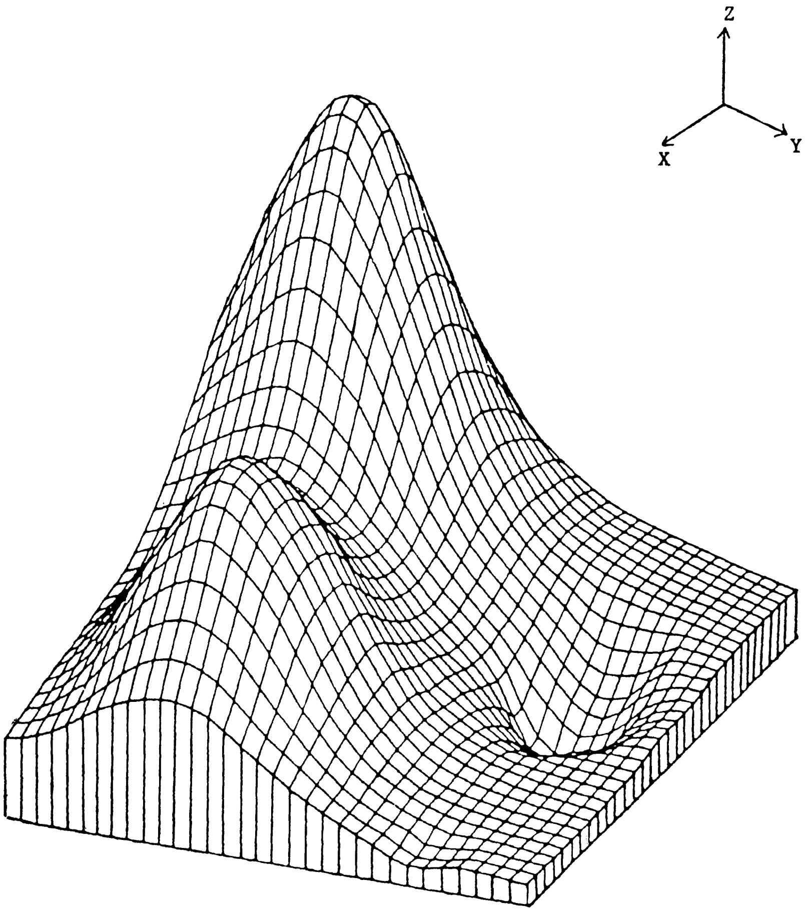

FIGURE 5.2. Surface Plot of $F1$.

Root-mean-square errors again displayed a pattern similar to that of the mean errors. The results show that the accuracy of our methods is not restricted to nodal distributions with uniform density. The similar accuracy of the two global methods (for both node sets) indicates that the choice of interpolant is much less important than the gradient estimates.

While triangle-based interpolation methods are more likely to suffer defects near the boundary than are other methods, they are generally more efficient computationally. In terms of the total time required to compute the 1089 interpolated values, Lawson's method was found to be about 3.5 times as fast as Nielson-Franke Quadratic and Akima Mod III, and about 7 times as fast as the Modified Quadratic Shepard method [5]. The efficiency of our method relative to Lawson's depends on the application: the triangulation phase is slower (see Reference 2); the local derivative-estimation procedure is approximately the same but the global procedure is faster (see §IV); and the interpolation phase is about the same. Central-processor time averaged .133 milliseconds per interpolated value with given derivative estimates.

Approximate storage requirements for arrays other than those containing nodal coordinates and data values are as follows:

Renka and Cline	$-7N$ or $9N$
Lawson	$-20N$
Nielson-Franke Quadratic	$-32N$
Modified Quadratic Shepard	$-5N$
Akima Mod III	$-33N$
Nielson Minimum Norm	$-32N$

Shepard's method has the advantage of not requiring a triangulation but pays a price in computational efficiency as noted above. The Nielson-Franke, Akima, and Nielson methods use Akima's triangulation software which requires approximately $32N$ storage locations. Our method gives the user the option of computing and storing the $2N$ estimated partial derivatives before entering the interpolation phase. In this case, the total storage requirement is $9N$. The partials need not be stored, however. If relatively few interpolated values at points scattered throughout a large region are needed, the user can set a flag indicating that, for each interpolation point, derivative estimates are to be computed at the vertices of the triangle containing that point.

Another criterion used to judge an interpolation package is ease of use. Our method suffers from the requirement that the user must call more than one subroutine — a node-presorting routine (optional), the triangulation routine, a derivative-estimation routine (optional), and an interpolation routine. We feel the gain in versatility and efficiency more than justifies this requirement. Also, our parameter lists are very short and, unlike most methods, we require no user-specified ad hoc parameters to define the interpolant. For example, the Nielson-Franke Quadratic, Modified Quadratic Shepard, and Akima Mod III methods all use weight functions involving a fixed radius of influence. This radius must be either user-specified or estimated from global information such as the number of nodes and the diameter of their convex hull. Franke points out that such an estimate results in a method which is, by a strict definition, global.

The variable radius of influence based on a fixed number of nodes within the radius rather than vice versa is apparently the key idea behind the success of our local method. It seems likely that this idea could be effectively employed in a number of interpolation methods.

The software discussed here, as well as a software package which extends our method to the surface of a sphere, is available from the first author.

References

1. H. Akima, *A method of bivariate interpolation and smooth surface fitting for irregularly distributed data points*, ACM TOMS **4**, (1978), 148–64.

2. A. K. Cline and R. J. Renka, *A storage-efficient method for construction of a Thiessen triangulation*, this Journal, this issue.

3. R. W. Clough and J. L. Tocher, *Finite element stiffness matrices for analysis of plates in bending*, Proc. Conf. Matrix Methods in Struct. Mech., Air Force Inst. of Tech., Wright-Patterson A.F.B., Ohio, 1965.

4. R. Franke and G. Nielson, *Smooth interpolation of large sets of scattered data*, International J. for Numerical Methods in Engineering **15** (1980), 1691–1704.

5. R. Franke, *A critical comparison of some methods for interpolation of scattered data*, Naval Postgraduate School Technical Report, NPS-53-79-003, 1979.

6. R. G. Grimes, D. R. Kincaid, W. I. MacGregor, and D. M. Young, *ITPACK report: adaptive iterative algorithms using symmetric sparse storage*, CNA-139, Center for Numerical Analysis, Univ. of Texas at Austin.

7. C. L. Lawson, *C^1-compatible interpolation over a triangle*, Jet Propulsion Laboratory, TM33-770, 1976.

8. C. L. Lawson, *Software for C^1 surface interpolation*, Mathematical Software III, J. R. Rice, Ed., Academic Press, New York, 1977, 161–194.

9. D. H. McLain, *Two Dimensional Interpolation from Random Data*, Computer J. **19** (1976), 178–181, and *Errata*, 384

10. G. Nielson, *A method for interpolating scattered data based upon a minimum norm network*, Math. Comp. **40** (1983), 253–271.

11. R. J. Renka, *Triangulation and Bivariate Interpolation for Irregularly Distributed Data Points*, Ph.D. Dissertation, The University of Texas at Austin, 1981.

Computer Sciences at Oak Ridge National Laboratory, Union Carbide Corporation, Nuclear Division, Oak Ridge, TN 37830

Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712