Algorithm 792: Accuracy Tests of ACM Algorithms for Interpolation of Scattered Data in the Plane

ROBERT J. RENKA
University of North Texas
and
RON BROWN
AISN Software

We present results of accuracy tests on scattered-data fitting methods that have been published as ACM algorithms. The algorithms include seven triangulation-based methods and three modified Shepard methods, two of which are new algorithms. Our purpose is twofold: to guide potential users in the selection of an appropriate algorithm and to provide a test suite for assessing the accuracy of new methods (or existing methods that are not included in this survey). Our test suite consists of five sets of nodes, with node counts ranging from 25 to 100, and 10 test functions. These are made available in the form of three Fortran subroutines: TESTDT returns one of the node sets; TSTFN1 returns a value and, optionally, a gradient value, of one of the test functions; and TSTFN2 returns a value, first partials, and second partial derivatives of one of the test functions.

Categories and Subject Descriptors: D.3.2 [**Programming Languages**]: Language Classifications—*Fortran 77*; G.1.1 [**Numerical Analysis**]: Interpolation; G.1.2 [**Numerical Analysis**]: Approximation; G.4 [**Mathematics of Computing**]: Mathematical Software

General Terms: Algorithms

Additional Key Words and Phrases: Delaunay triangulation, interpolation, scattered data, Shepard method, surface fitting

1. INTRODUCTION

The problem of interpolating scattered data in the plane is the following: given a set of N arbitrarily distributed points $\{(x_i, y_i)\}_{i=1}^{N}$ in $\mathbf{R}^2$, referred to

Authors' addresses: R. J. Renka, Department of Computer Sciences, University of North Texas, Denton, TX 76203-1366; email: renka@cs.unt.edu; R. Brown, AISN Software, P.O. Box 449, Mapleton, OR 97453.

as *nodes*, along with data values z_i at the nodes, we wish to construct a smooth function $F : \mathbf{R}^2 \rightarrow \mathbf{R}$ such that $F(x_i, y_i) = z_i$ for $i = 1, \ldots, N$. Since continuity of first partial derivatives is sufficient smoothness for most applications, and smoother surfaces are more costly to construct, most scattered data methods produce $F \in C^1(\mathbf{R}^2)$. However, we also consider three methods that produce C^2 functions.

The scattered-data fitting problem arises in numerous areas of scientific inquiry with the data typically representing measured or computed values of some physical quantity—an underlying function which we seek to approximate. The fitting function serves to provide values at points where it is impossible or expensive to obtain measurements, to provide approximations to derivatives or integrals of the underlying function, or to provide a visual representation of the data in the form of a surface plot or contour plot. In some applications the problem is to select parameters (usually by a least-squares fit) that enter into a model of the underlying function, and in other applications the problem is to smooth out noise in the data values. We are concerned here only with applications for which a good model is not available and for which the data values may be assumed to be accurate.

The criteria for assessing the effectiveness of a scattered-data interpolation method include accuracy in reproducing test functions, computational costs in both preprocessing and evaluation, storage requirements, flexibility in handling constraints, and appearance of the interpolatory surface. In addition to these criteria, a software package may be judged on its portability, reliability, robustness, ease of use, modifiability, and flexibility in terms of additional capabilities such as preprocessing and postprocessing of data sets. Thus, the results reported in this survey, which is primarily addressed toward testing accuracy, should not be used as the sole criteria for selecting an algorithm. Furthermore, the most accurate method for one data set (or collection of data sets) may perform poorly on another. All of the methods here, on the other hand, are designed for general purpose, and accurately reproduce smooth underlying test functions with sufficiently dense node sets.

Unlike previous surveys of scattered-data methods by Barnhill [1977], Franke [1982], and Schumaker [1976], we have limited the selection of methods to those for which good implementations are readily available. By limiting attention to ACM algorithms, we ensure some degree of portability and reliability of the codes. The fact that several methods are now available as ACM algorithms reflects the growing maturity of the field. However, we make no claim that the methods tested are the best ones available, and we hope that this survey will encourage other researchers to develop and publish algorithms based on competing methods.

All of the algorithms reported on here have been incorporated into the latest release of TableCurve 3D [Brown 1997], a commercial data fitting package for Microsoft Windows developed by the second author. By building the algorithms, along with automated test procedures, into this interactive environment, we were able to very rapidly generate test results,

including tests of variations in the methods. These test results led not only to this survey and the two new algorithms (Renka and Brown [1999a; 1999b]), but also to improvements and bug corrections in existing algorithms, (Algorithm 761 in particular [Renka and Brown 1998]).

The remainder of the article is organized as follows. Section 2 describes the methods; the test suite is described in Section 3; and test results are presented in Section 4.

2. THE ALGORITHMS

The algorithms are ordered as follows:

(1) Algorithm 526 [Akima 1978a; 1978b]

(2) Algorithm 761 [Akima 1996]

(3) Algorithm 684, C^1 [Preusser 1990a; 1990b].

(4) Algorithm 684, C^2

(5) Algorithm 752, `GRADL` [Renka 1996b]

(6) Algorithm 752, `GRADC`

(7) Algorithm 752, `GRADG`

(8) Algorithm 660 (`QSHEP2D`) [Renka 1988a; 1988b]

(9) `CSHEP2D` [Renka and Brown 1999a]

(10) `TSHEP2D` [Renka and Brown 1999b]

We describe the first seven algorithms and their underlying methods in the first subsection below, and the last three algorithms in the second subsection.

2.1 Triangulation-Based Algorithms

The first seven algorithms employ a Delaunay triangulation of the nodes—a partition of the convex hull Ω of the nodes (the smallest convex set that contains the nodes) into triangles such that the set of triangle vertices coincides with the set of nodes and no circumcircle associated with a triangle contains a node in its interior. All except Algorithm 526 use Algorithm 751 [Renka 1996a] to construct the triangulation. Triangulation-based methods are generally faster than alternatives but require a special procedure for extrapolation (evaluation at points outside Ω). The extrapolation procedure of Algorithm 752 was added to Algorithm 684 for this study so that all of the algorithms provide for C^1 extrapolation. The fourth algorithm produces an interpolant in $C^2(\Omega)$, while the others are C^1 methods.

All seven methods use piecewise Hermite polynomial interpolants of nodal values, gradients, and, in the case of the first four methods, second partial derivatives. The fourth method also uses third- and fourth-order

nodal derivatives. The first three methods use quintic (degree-5) polynomials in which the 21 coefficients are chosen to satisfy the 18 interpolatory conditions (6 at each vetex) and to force continuity of the normal gradient component across each of the three triangle sides. The fourth method uses a nonic (degree-9) polynomial with 55 free parameters to interpolate 15 values at each vertex and force continuity of second partials across triangulation edges. The last three methods use the piecewise cubic (degree-3) Clough-Tocher element [Clough and Tocher 1965] on each triangle (cubic on each of the three subtriangles obtained by connecting the vertices to the barycenter). The evaluation cost is proportional to the polynomial degree, thus highest for the nonic and lowest for the cubic.

The algorithms are otherwise distinguished by the method used to estimate partial derivatives at the nodes. The quality of the interpolatory surface depends far more on the choice of nodal derivatives than on the polynomial degree. Local derivative-estimation methods, employed by the first, second, fifth, and sixth algorithms, consist of taking the derivatives at node k to be those of a nodal function—a polynomial that interpolates the data value z_k at node (x_k, y_k) and fits the data values at a set of nearby nodes in some sense. Algorithm 526 uses a quadratic interpolant of the values at node k and its five nearest neighbors. Algorithm 761 uses a rather costly but effective procedure involving empirically determined weights, a least-squares fit of a linear function and a cubic interpolant of 10 data values—the values at node k and its nine nearest neighbors. Algorithm 752 uses a weighted least-squares fit with inverse distance related weights (similar to the nodal functions described below for Algorithm 660). The original method, `GRADL`, uses a quadratic fit with the eight nearest neighbors. A new method, `GRADC`, uses a cubic polynomial with the 12 nearest neighbors. We added `GRADC` to Algorithm 761 to prevent the method from falling back to a lower-degree polynomial fit when the linear system defining the cubic interpolant was found to be ill conditioned [Renka and Brown 1998].

The global method of Algorithm 752 (`GRADG`) consists of minimizing an approximation to the linearized curvature of the Hermite interpolant over the set of nodal gradients, where the approximation is taken to be the sum over the triangulation edges of the linearized curvature (integral of squared second derivative) of the restriction of the interpolant to the edge (the univariate Hermite cubic interpolant of the endpoint values and directional derivatives). This minimization corresponds to a sparse positive definite linear system that can be solved cheaply by a small number (at most 10) of Gauss-Seidel iterations. Algorithm 684 uses `GRADG` with repeated application to obtain higher-order partial derivatives, e.g., second partials are obtained by applying the method with first partials in place of data values.

2.2 Modified Shepard Algorithms

Shepard's method was introduced in 1968 [Shepard 1968] and modified to a local method by Franke and Nielson in 1980 [Franke and Nielson 1980].

Algorithm 660 is a further modification of Franke and Nielson's method in which accuracy was improved and a cell-based search algorithm due to Bentley and Friedman [1979] was added for improved efficiency. The interpolant is defined by

$$F(x, y) = \sum_{k=1}^{N} W_k(x, y) Q_k(x, y) \Big/ \sum_{i=1}^{N} W_i(x, y),$$

where the nodal function Q_k is a bivariate quadratic polynomial that interpolates the data value z_k at node k and fits the data values on a set of nearby nodes in a weighted least-squares sense.

The unnormalized weights are inverse distance functions:

$$W_k(x, y) = \left[\frac{(R_w - d_k)_+}{R_w d_k} \right]^2$$

for

$$(R_w - d_k) + = \begin{cases} R_w - d_k & \text{if } d_k < R_w \\ 0 & \text{if } d_k \geq R_w \end{cases},$$

where $d_k(x, y)$ is the Euclidean distance between (x, y) and (x_k, y_k), and R_w is a radius of influence about (x_k, y_k). Note that an interpolated value at a point (x, y) depends only on the data at nodes whose radii include (x, y).

It follows from the above definition that F interpolates the data, maintains the local shape properties of the nodal functions (has first partial derivatives at (x_k, y_k) that agree with those of Q_k), has quadratic precision, and lies in the space $C^1(\mathbf{R}^2)$.

The weight associated with node i in the least-squares fit for nodal function Q_k is

$$\omega_{ik} = \left[\frac{(R_q - d_{ik})_+}{R_q d_{ik}} \right]^2,$$

where d_{ik} is the distance between nodes i and k, and R_q is another radius of influence about node k—Q_k depends only on the data values at nodes within distance R_q of (x_k, y_k).

The radii R_q and R_w vary with k and are taken to be just large enough to include N_q and N_w nodes, respectively, for fixed values of N_q and N_w. The optimal values of these parameters depend on the data set, but accuracy varies smoothly and gradually with variations in the values. The default recommendations, which we used for our test results, are $N_q = 13$ and $N_w = 19$.

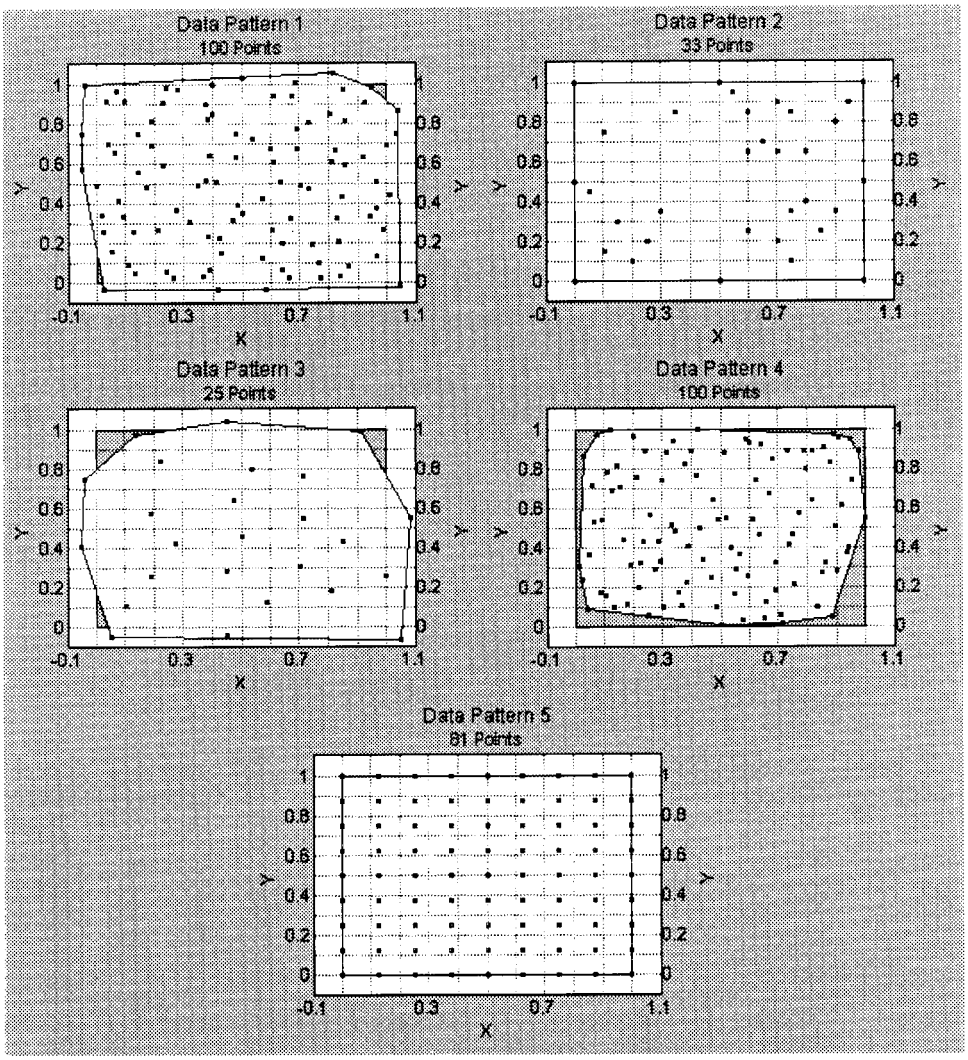

Fig. 1. Node sets.

With the cell-based search method described in Renka [1988b], assuming a uniform distribution of nodes, the expected time complexity is $O(N)$ for the preprocessing phase and constant for each evaluation. Worst-case operation counts are $O(N^2)$ for preprocessing and $O(N)$ for evaluation.

The two new algorithms, CSHEP2D and TSHEP2D, are nearly identical to Algorithm 660 (QSHEP2D), but use cubic and 10-parameter cosine series nodal functions, respectively, in place of the quadratics. Also, the relative weights are replaced by

$$W_k(x, y) = \left[\frac{(R_w - d_k)_+}{R_w d_k} \right]^3,$$

resulting in $F \in C^2(\mathbf{R}^2)$. CSHEP2D has cubic precision, and TSHEP2D exactly reproduces a bivariate cubic function of $cos(x)$ and $cos(y)$ for $x, y \in [0, \pi]$. The recommended parameter values for these methods are $N_q = 17$, $N_w = 30$ for CSHEP2D (except in the case of a nearly rectangular grid, in which case $N_q = 9$) and $N_q = 18$, $N_w = 32$ for TSHEP2D. For node set 3, N_w was limited to $N - 1 = 24$.

3. TEST SUITE

The five node sets are depicted in Figure 1, and the 10 test functions are displayed in Figures 2 and 3. The first three node sets and first six test functions are those originated by Franke [1979; 1982] and used by several other researchers. The error norms reported in the next section use a 33 $\times$ 33 rectangular grid of evaluation points in the unit square, and the shaded areas in Figure 1 thus display the regions in which extrapolation occurs. The fourth node set was chosen to emphasize the effects of extrapolation. Of the 1089 evaluation points, the number of extrapolation points is 13 for node set 1, 54 for node set 3, 190 for node set 4, and 0 for sets 2 and 5. The last node set is a 9 $\times$ 9 uniform rectangular grid. It was included because, in the case of missing data, it is necessary to use a scattered-data method for values on a nearly rectangular grid. It also enabled us to compare the accuracy of the scattered-data methods with a bicubic spline interpolant.

The last four test functions were chosen for their multiple features and abrupt transitions. Thus, while all of the test functions are smooth (except for a first derivative discontinuity at the point (0.5, 0.5) in the case of function 10), the last four are quite challenging. In fact, we omitted from our test results the eight combinations of the last four test functions with the two sparse node sets (sets 2 and 3).

The test functions are as follows:

$$\begin{aligned} F1(x, y) = {} & 0.75\exp(-((9x - 2)^2 + (9y - 2)^2)/4) \\ & + 0.75\exp(-(9x + 1)^2/49 - (9y + 1)/10) \\ & + 0.50\exp(-((9x - 7)^2 + (9y - 3)^2)/4) \\ & - 0.20\exp(-(9x - 4)^2 - (9y - 7)^2) \end{aligned}$$

$$F2(x, y) = \frac{\tanh(9y - 9x) + 1}{9}$$

$$F3(x, y) = \frac{1.25 + \cos(5.4y)}{6 + 6(3x - 1)^2}$$

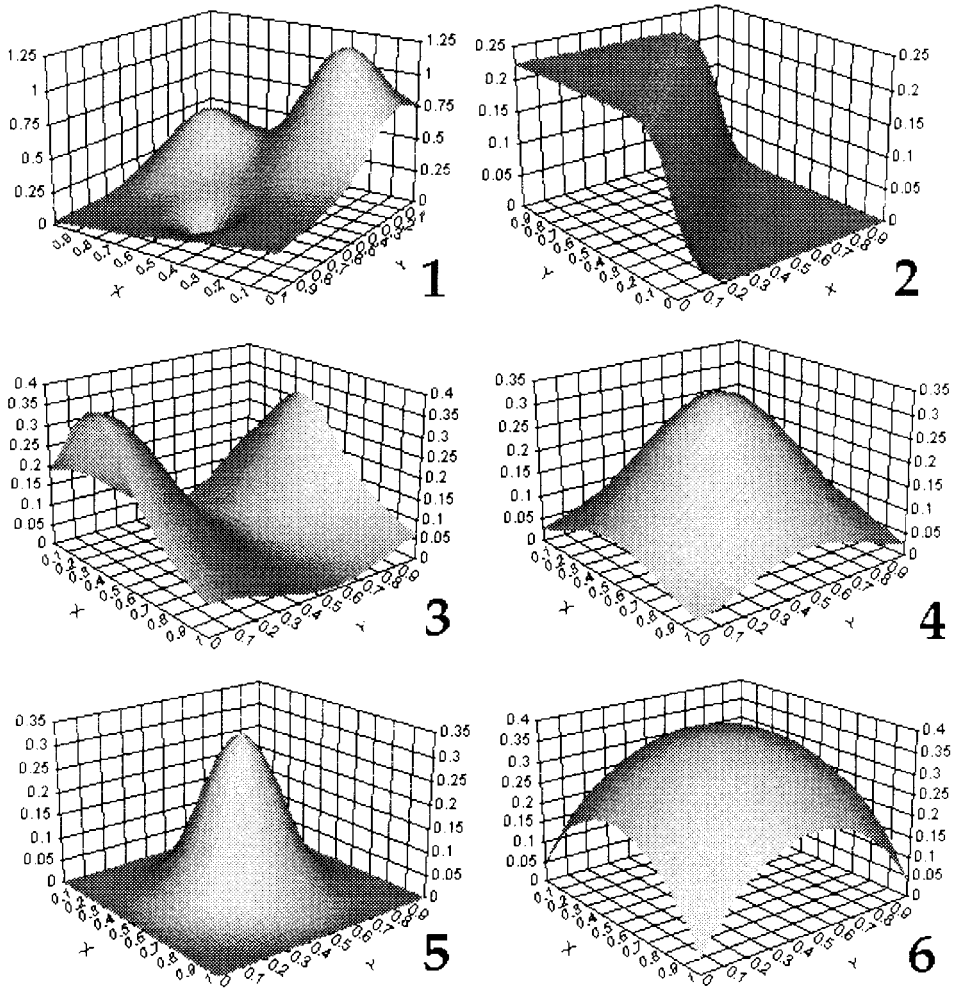

Fig. 2. Franke's test functions.

$$F4(x, y) = \frac{1}{3}\exp(-5.0625((x - 0.5)^2 + (y - 0.5)^2))$$

$$F5(x, y) = \frac{1}{3}\exp(-20.25((x - 0.5)^2 + (y - 0.5)^2))$$

$$F6(x, y) = \frac{1}{9}\sqrt{64 - 81((x - 0.5)^2 + (y - 0.5)^2)} - 0.5$$

$$F7(x, y) = 2\cos(10x)\sin(10y) + \sin(10xy)$$

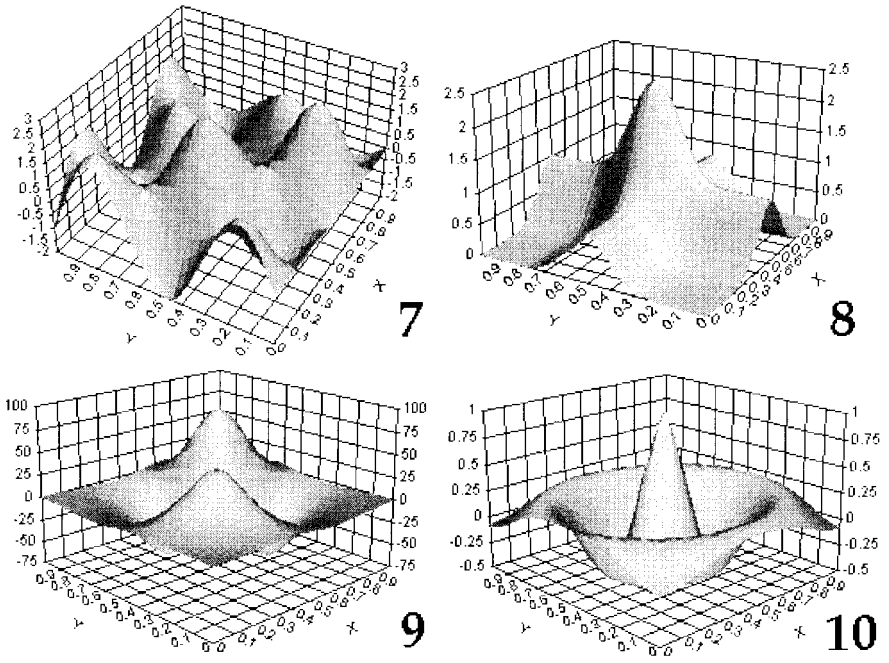

Fig. 3. Additional test functions.

$$F8(x, y) = \exp\left(-\frac{(5 - 10x)^2}{2}\right) + 0.75\exp\left(-\frac{(5 - 10y)^2}{2}\right)$$

$$+ 0.75\exp\left(-\frac{(5 - 10x)^2}{2}\right)\exp\left(-\frac{(5 - 10y)^2}{2}\right)$$

$$F9(x, y) = \left(\left(\frac{20}{3}\right)^3 \exp\left(\frac{10 - 20x}{3}\right)\exp\left(\frac{10 - 20y}{3}\right)\right)^2$$

$$\times \left(\left(\frac{1}{1 + \exp\left(\frac{10 - 20x}{3}\right)}\right)\left(\frac{1}{1 + \exp\left(\frac{10 - 20y}{3}\right)}\right)\right)^5$$

$$\times \left(\exp\left(\frac{10 - 20x}{3}\right) - \frac{2}{1 + \exp\left(\frac{10 - 20x}{3}\right)}\right)$$

$$\times \left(\exp\left(\frac{10 - 20x}{3}\right) - \frac{2}{1 + \exp\left(\frac{10 - 20y}{3}\right)}\right)$$

Table I. Error Norms for Triangulation-Based Algorithms on Node Set 1

Fcn	526	761	684, C^1	684, C^2	GRADL	GRADC	GRADG
1	0.002093	0.000852	0.000671	0.000895	0.001246	0.000752	0.000714
2	0.004036	0.001363	0.001386	0.001540	0.002651	0.002853	0.001622
3	0.001514	0.000049	0.000432	0.000499	0.000270	0.000201	0.000410
4	0.000687	0.000025	0.000077	0.000128	0.000036	0.000011	0.000041
5	0.004211	0.000374	0.000322	0.000593	0.001445	0.000418	0.000385
6	0.001243	0.000013	0.000341	0.000404	0.000081	0.000023	0.000328
Avg	0.002297	0.000446	0.000538	0.000677	0.000955	0.000710	0.000583
7	0.045132	0.012406	0.017065	0.016671	0.024248	0.017414	0.022873
8	0.020119	0.005007	0.005946	0.005682	0.015092	0.013438	0.009209
9	0.023018	0.006877	0.004076	0.005029	0.009631	0.005131	0.004996
10	0.078449	0.022701	0.018081	0.019472	0.052491	0.026646	0.025809
Avg	0.041680	0.011748	0.011292	0.011714	0.025366	0.015657	0.015722

$$F10(x, y) = \exp\left(-0.04\sqrt{(80x - 40)^2 + (90y - 45)^2}\right) \times \cos\left(0.15\sqrt{(80x - 40)^2 + (90y - 45)^2}\right)$$

4. TEST RESULTS

For each of the 10 algorithms and 42 of the 50 data sets (node set/test function pairs), an error norm was computed using a 33×33 uniform rectangular grid of evaluation points in the unit square $[0,1] \times [0, 1]$. The measure of error was taken to be SSE/SSM, where SSE is the sum of squared errors (deviations from the test function values), and SSM is the sum of squares of deviations of the 1089 test function values from their mean. The normalization of the error measure to account for the variance in the true function values enables us to compute meaningful averages of error norms over the set of test functions. However, as noted in the previous section, the last four functions are more difficult to approximate than the first six. We therefore separate the set of test functions into two categories and compute an average error for each—except with the two sparse node sets, in which case the difficult function category is omitted. Tables I to V correspond to the five node sets and the triangulation-based methods. Tables VI to X correspond to the Shepard methods. The rows labeled Avg are averages of the values for functions 1–6 or 7–10.

Figure 4 displays values of $r^2 = 1 - \text{SSE/SSM}$ averaged over functions 1 to 6 for node sets 1, 4, and 5, Figure 5 displays averages over the same functions for sets 2 and 3, and Figure 6 displays averages over functions 7 to 10 for sets 1, 4, and 5. These values are obtained from the averages in Tables I–X. In statistics, r^2 is referred to as a coefficient of determination, and its value may be interpreted as follows: $r^2 = 0$ can be obtained by a least-squares fit of a constant function to the data and implies no accuracy

Table II. Error Norms for Triangulation-Based Algorithms on Node Set 2

Fcn	526	761	684, C^1	684, C^2	GRADL	GRADC	GRADG
1	0.055885	0.037804	0.024353	0.029570	0.024802	0.033427	0.023359
2	0.017793	0.030015	0.020797	0.024181	0.020437	0.095503	0.016877
3	0.028114	0.008382	0.015836	0.015708	0.023656	0.011459	0.021365
4	0.007452	0.001485	0.001270	0.001920	0.004186	0.002118	0.002024
5	0.090912	0.052050	0.044145	0.036343	0.063380	0.069237	0.074102
6	0.022039	0.001018	0.004463	0.005078	0.001396	0.000728	0.003933
Avg	0.037033	0.021792	0.018477	0.018800	0.022976	0.035412	0.023610

Table III. Error Norms for Triangulation-Based Algorithms on Node Set 3

Fcn	526	761	684, C^1	684, C^2	GRADL	GRADC	GRADG
1	0.015184	0.026354	0.013685	0.014297	0.028412	0.076085	0.012716
2	0.071839	0.019954	0.055678	0.055739	0.087156	0.057899	0.058622
3	0.047664	0.013908	0.034031	0.034027	0.045388	0.012305	0.034783
4	0.008533	0.002343	0.002744	0.002792	0.006930	0.002592	0.002824
5	0.024673	0.056551	0.006893	0.007008	0.021608	0.088805	0.007867
6	0.028923	0.001009	0.009594	0.010251	0.002591	0.001765	0.009178
Avg	0.032803	0.020020	0.020438	0.020686	0.032014	0.039909	0.020998

Table IV. Error Norms for Triangulation-Based Algorithms on Node Set 4

Fcn	526	761	684, C^1	684, C^2	GRADL	GRADC	GRADG
1	0.003187	0.002720	0.001012	0.001082	0.004125	0.001762	0.001112
2	0.007085	0.001254	0.003964	0.004006	0.003497	0.008705	0.004000
3	0.002461	0.000861	0.001808	0.001862	0.001467	0.001773	0.001842
4	0.000962	0.000124	0.000157	0.000189	0.000331	0.000074	0.000143
5	0.001536	0.000417	0.000327	0.000510	0.001203	0.000372	0.000225
6	0.002942	0.000272	0.001281	0.001342	0.000699	0.000572	0.001242
Avg	0.003029	0.000941	0.001425	0.001499	0.001887	0.002210	0.001427
7	0.098711	0.023801	0.049575	0.049306	0.073205	0.038078	0.054774
8	0.031143	0.011066	0.036545	0.036155	0.037952	0.047086	0.038984
9	0.025148	0.002690	0.006805	0.006368	0.019777	0.018589	0.009734
10	0.030876	0.018643	0.013654	0.016793	0.041680	0.020586	0.012728
Avg	0.046470	0.014050	0.026645	0.027156	0.043154	0.031085	0.029055

in the fit; $r^2 = 0.9$ is considered a fair fit; $r^2 = 0.95$ indicates a good fit; $r^2 = 0.99$ is very good; $r^2 = 0.999$ is excellent; and $r^2 = 0.9999$ implies essentially no error in empirical data, for which there are seldom more than four significant digits.

The following observations are consistent with what would be expected of the underlying methods. For the sparse node sets, GRADL performed better than GRADC, and QSHEP2D performed better than CSHEP2D (quadratic fits are more accurate than cubics), while, on the dense node sets, GRADC outperformed GRADL, and CSHEP2D outperformed QSHEP2D (cubics are more accurate than quadratics). With the exception of function 6, Algorithm 684 (C^1) is more accurate than GRADG. Since these algorithms use the same

Table V. Error Norms for Triangulation-Based Algorithms on Node Set 5

Fcn	526	761	684, C^1	684, C^2	GRADL	GRADC	GRADG
1	0.001253	0.000496	0.000411	0.000396	0.001213	0.000557	0.000492
2	0.002903	0.001026	0.001512	0.001381	0.002620	0.001427	0.001903
3	0.000590	0.000020	0.000111	0.000121	0.000124	0.000023	0.000108
4	0.000202	0.000004	0.000031	0.000034	0.000041	0.000005	0.000031
5	0.001427	0.000155	0.000039	0.000037	0.001077	0.000241	0.000116
6	0.000931	0.000002	0.000239	0.000259	0.000018	0.000002	0.000217
Avg	0.001218	0.000284	0.000391	0.000371	0.000849	0.000376	0.000478
7	0.021122	0.002369	0.004560	0.004483	0.011812	0.003905	0.006199
8	0.005735	0.001892	0.000390	0.000429	0.004448	0.002213	0.000721
9	0.010976	0.003174	0.002184	0.001971	0.011304	0.004433	0.003631
10	0.022273	0.007865	0.005454	0.006702	0.016552	0.008442	0.005909
Avg	0.015027	0.003825	0.003147	0.003396	0.011029	0.004748	0.004115

Table VI. Error Norms for Shepard Algorithms on Node Set 1

Fcn	QSHEP2D	CSHEP2D	TSHEP2D
1	0.001029	0.000560	0.000610
2	0.001599	0.002090	0.002290
3	0.000308	0.000182	0.000286
4	0.000062	0.000010	0.001124
5	0.000705	0.000345	0.000170
6	0.000041	0.000010	0.009820
Avg	0.000624	0.000533	0.002383
7	0.013383	0.011088	0.016936
8	0.006639	0.008277	0.004016
9	0.010449	0.003507	0.002821
10	0.025393	0.022579	0.021196
Avg	0.013966	0.011363	0.011242

Table VII. Error Norms for Shepard Algorithms on Node Set 2

Fcn	QSHEP2D	CSHEP2D	TSHEP2D
1	0.025933	0.027103	0.041226
2	0.035452	0.050291	0.070631
3	0.015924	0.009804	0.008986
4	0.008858	0.001336	0.015680
5	0.049137	0.063871	0.030250
6	0.001418	0.000445	0.100699
Avg	0.022787	0.025475	0.044579

gradient-estimation and extrapolation procedures, the results reflect the fact that the piecewise quintic Hermite interpolant is more accurate than the piecewise cubic. Finally, on the rectangular grid, none of the scattered-data methods is as accurate as the bicubic spline (0.99987 for functions 1–6, 0.99750 for functions 7–10), but Algorithm 761, Algorithm 684, GRADC, GRADG, and CSHEP2D are very close.

Table VIII. Error Norms for Shepard Algorithms on Node Set 3

Fcn	QSHEP2D	CSHEP2D	TSHEP2D
1	0.020568	0.040828	0.016455
2	0.074131	0.060934	0.080080
3	0.034142	0.018707	0.016894
4	0.006235	0.002377	0.009501
5	0.025420	0.054079	0.022628
6	0.001928	0.001309	0.103223
Avg	0.027071	0.029706	0.041464

Table IX. Error Norms for Shepard Algorithms on Node Set 4

Fcn	QSHEP2D	CSHEP2D	TSHEP2D
1	0.003450	0.002356	0.003156
2	0.002690	0.001795	0.003726
3	0.000397	0.000151	0.001709
4	0.000357	0.000059	0.004192
5	0.000976	0.000914	0.000193
6	0.000180	0.000034	0.027067
Avg	0.001342	0.000885	0.006674
7	0.030211	0.029959	0.036122
8	0.007238	0.006531	0.003057
9	0.010479	0.009313	0.005077
10	0.031790	0.026761	0.019050
Avg	0.019930	0.018141	0.015827

Table X. Error Norms for Shepard Algorithms on Node Set 5

Fcn	QSHEP2D	CSHEP2D	TSHEP2D
1	0.001452	0.000542	0.001988
2	0.002127	0.001040	0.001330
3	0.000165	0.000026	0.001673
4	0.000070	0.000008	0.003333
5	0.001358	0.000193	0.000165
6	0.000022	0.000003	0.017854
Avg	0.000866	0.000302	0.004391
7	0.013501	0.003174	0.014951
8	0.006147	0.001862	0.001046
9	0.013527	0.003977	0.004196
10	0.019582	0.010463	0.012458
Avg	0.013189	0.004869	0.008163

Comparing GRADC with GRADG, GRADC is more accurate for functions 3, 4, 6, and 7, while GRADG is superior for the other six functions. Apparently the broad sweeping features of functions 3, 4, 6, and 7 are better handled by the local gradient-estimation procedure, while the surfaces with sharp localized features favor the global method.

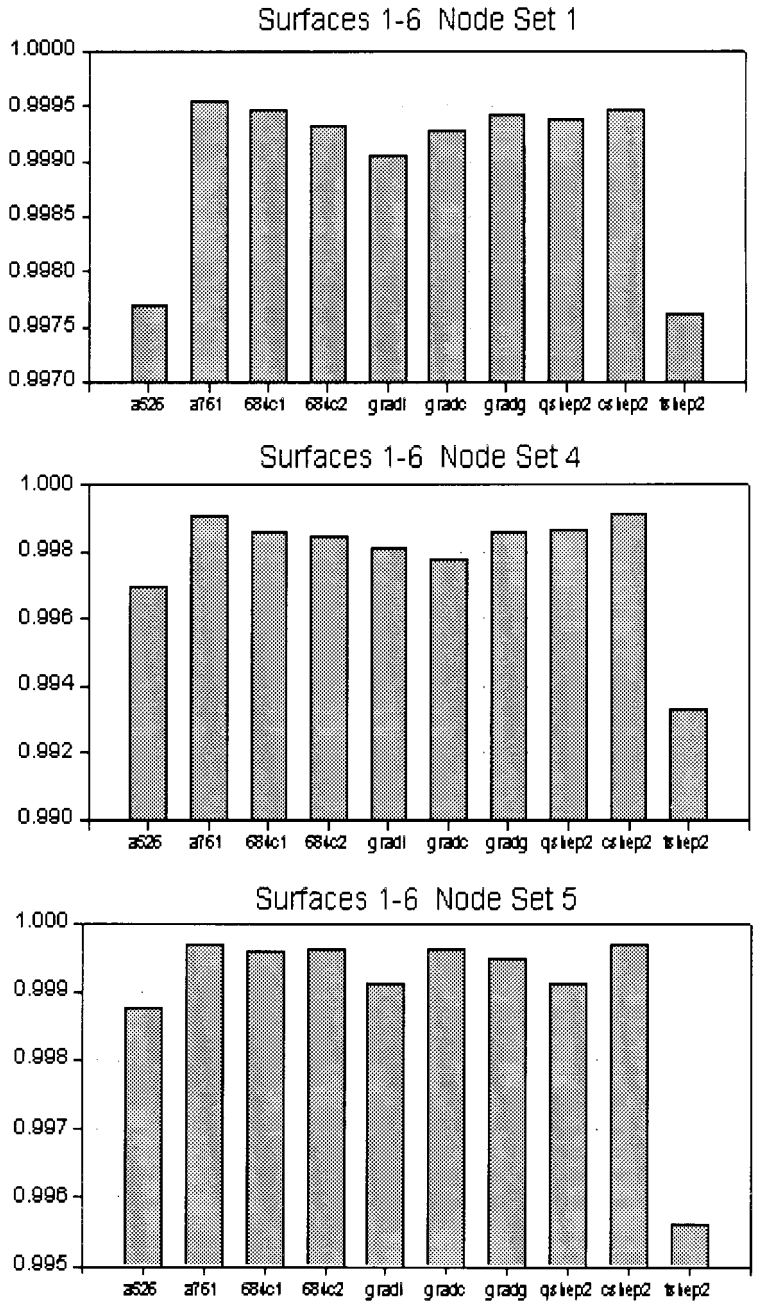

Fig. 4. Average r^2 values for functions 1–6, node sets 1,4,5.

With the exception of the uniform rectangular grid, the C^1 option of Algorithm 684 is more accurate than the C^2 option, which should therefore only be used if a C^2 fit is required. In seeking the cause of the lower

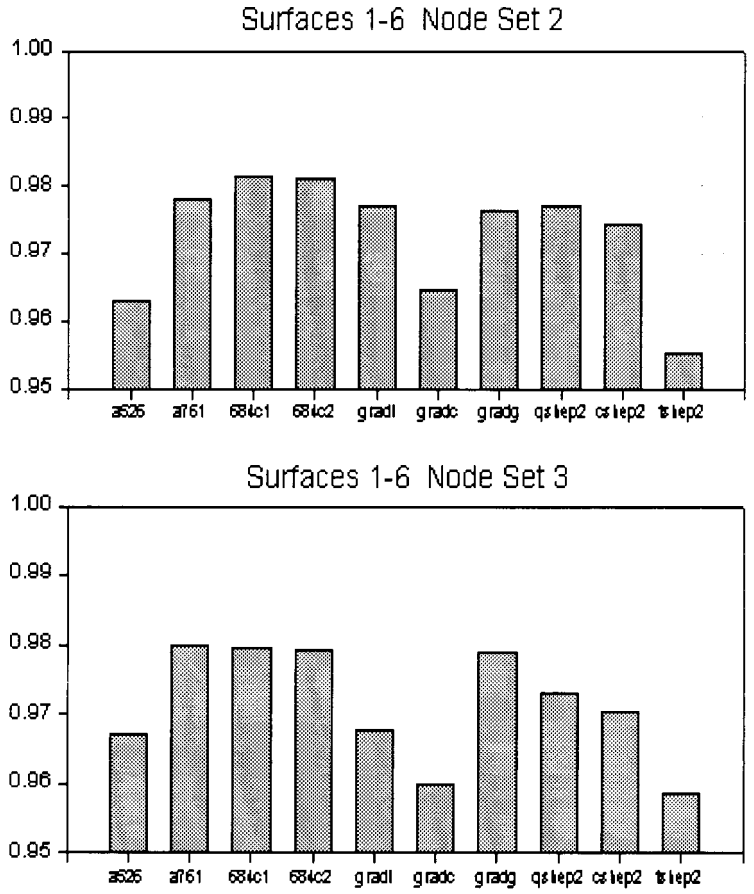

Fig. 5. Average r^2 values for functions 1–6, node sets 2,3.

accuracy, it was found that the choice of third-and-fourth order nodal derivatives had little effect on the accuracy of the C^2 method. The advantage of the additional smoothness is apparently not sufficient to offset the disadvantage of the high-degree polynomials.

Our conclusions are as follows. With the dense node sets (including the uniform grid), the most accurate algorithms are TSHEP2D, Algorithm 761, and Algorithm 684 (C^1) for the more difficult test functions, or CSHEP2D and Algorithm 761 for the easier functions. For the sparse node sets, the best performers are Algorithm 761 and Algorithm 684 with the C^1 option. Algorithm 526 did not perform well on any of the data sets and should be considered obsolete. TSHEP2D is also a poor choice unless the data set exhibits significant variation or periodicity. In any data fitting application, the fitting function should be assessed visually by plotting the surface.

If cheap evaluation is needed, Algorithm 752 is the best choice, and GRADG is usually more accurate than either of the local methods. Algorithm

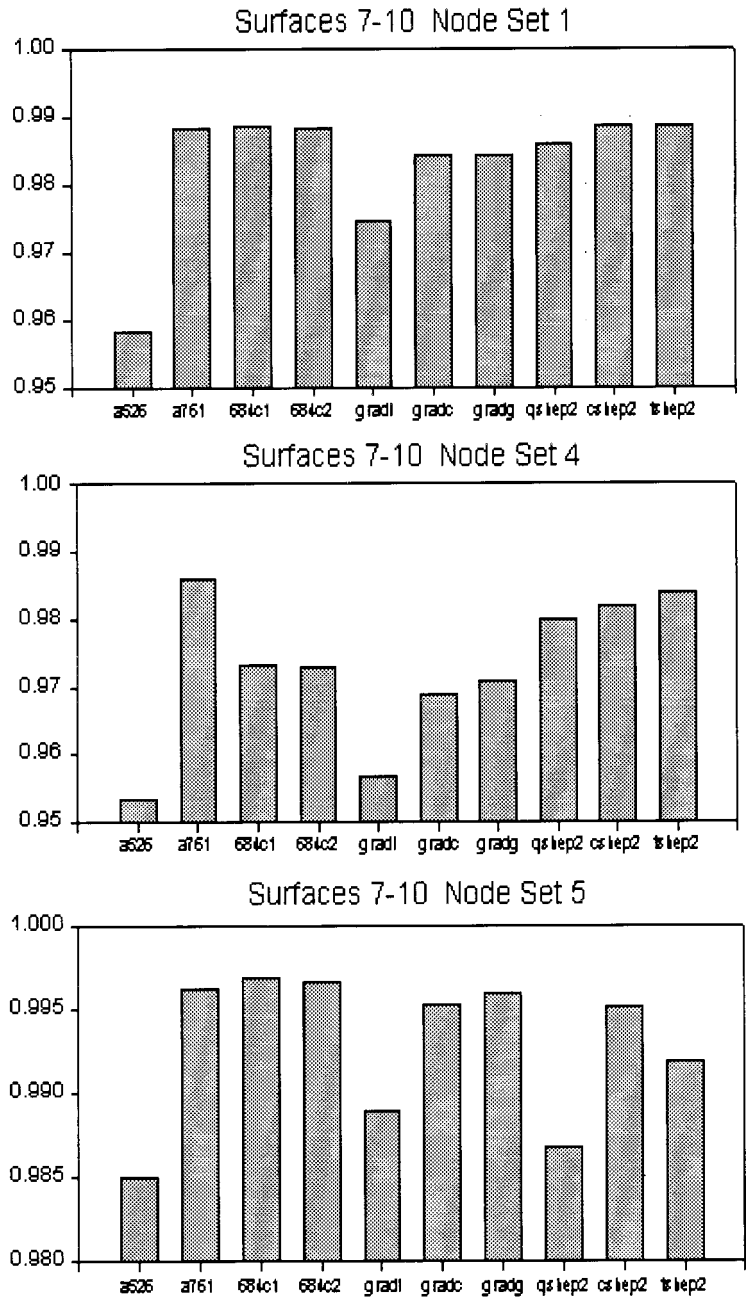

Fig. 6. Average r^2 values for functions 7–10.

752 and the Shepard algorithms provide procedures for evaluating first partial derivatives at arbitrary points. CSHEP2D and TSHEP2D also provide for computing second partials.

REFERENCES

AKIMA, H. 1978a. Algorithm 526: Bivariate interpolation and smooth surface fitting for irregularly distributed data points. *ACM Trans. Math. Softw. 4*, 2 (June 1978), 160–164.

AKIMA, H. 1978b. A method of bivariate interpolation and smooth surface fitting for irregularly distributed data points. *ACM Trans. Math. Softw. 4*, 2 (June 1978), 148–159.

AKIMA, H. 1996. Algorithm 761: Scattered data surface fitting that has the accuracy of a cubic polynomial. *ACM Trans. Math. Softw. 22*, 3 (Sept.), 362–371.

BARNHILL, R. E. 1977. Representation and approximation of surfaces. In *Mathematical Software III*, J. R. Rice, Ed. Academic Press, Inc., New York, NY, 69–120.

BENTLEY, J. L. AND FRIEDMAN, J. H. 1979. Data structures for range searching. *ACM Comput. Surv. 11*, 4 (Dec.), 397–409.

BROWN, R. 1997. *TableCurve 3D*. Version 3. SPSS, Inc., Chicago, IL.

CLOUGH, R. W. AND TOCHER, J. L. 1965. Finite elements stiffness matrices for analysis of plates in bending. In *Proceedings of the Conference on Matrix Methods in Structural Mechanics*.

FRANKE, R. 1979. A critical comparison of some methods for interpolation of scattered data. NPS-53-79-003. Dept. of Mathematics, Naval Postgraduate School, Monterey, CA.

FRANKE, R. 1982. Scattered data interpolation: Tests of some methods. *Math. Comput. 38*, 157 (Jan.), 181–200.

FRANKE, R. AND NIELSON, G. 1980. Smooth interpolation of large sets of scattered data. *Int. J. Numer. Method. Eng. 15*, 1691–1704.

PREUSSER, A. 1990a. Algorithm 684: C^1- and C^2-interplation on triangles with quintic and nonic bivariate polynomials. *ACM Trans. Math. Softw. 16*, 3 (Sept. 1990), 253–257.

PREUSSER, A. 1990b. Efficient formulation of a bivariate nonic C^2-hermite polynomial on triangles. *ACM Trans. Math. Softw. 16*, 3 (Sept. 1990), 246–252.

RENKA, R. J. 1988a. Algorithm 660: QSHEP2D: Quadratic Shepard method for bivariate interpolation of scattered data. *ACM Trans. Math. Softw. 14*, 2 (June 1988), 149–150.

RENKA, R. J. 1988b. Multivariate interpolation of large sets of scattered data. *ACM Trans. Math. Softw. 14*, 2 (June 1988), 139–148.

RENKA, R. J. 1996a. Algorithm 751: TRIPACK: Constrained two-dimensional Delaunay triangulation package. *ACM Trans. Math. Softw. 22*, 1 (Mar.), 1–8.

RENKA, R. J. 1996b. Algorithm 752: SRFPACK: Software for scattered data fitting with a constrained surface under tension. *ACM Trans. Math. Softw. 22*, 1 (Mar.), 9–17.

RENKA, R. J. AND BROWN, R. 1998. Remark on Algorithm 761. *ACM Trans. Math. Softw. 24*, 4 (Dec.), 383–385.

RENKA, R. J. AND BROWN, R. 1999a. Algorithm 790: CSHEP2D: Cubic Shepard method for bivariate interpolation of scattered data. *ACM Trans. Math. Softw. 25*, 1 (Mar.). This issue.

RENKA, R. J. AND BROWN, R. 1999b. Algorithm 791: TSHEP2D: Cosine series Shepard method for bivariate interpolation of scattered data. *ACM Trans. Math. Softw. 25*, 1 (Mar.). This issue.

SCHUMAKER, L. L. 1976. Fitting surfaces to scattered data. In *Approximation Theory*, G. G. Lorentz, C. K. Chui, and L. L. Schumaker, Eds. Academic Press, Inc., New York, NY.

SHEPARD, D. 1968. A two-dimensional interpolation function for irregularly spaced data. In *Proceedings of the 23rd ACM National Conference*. ACM, New York, NY, 517–524.

Received: March 1997; revised: October 1998; accepted: November 1998