Algorithm 760: Rectangular-Grid-Data Surface Fitting that Has the Accuracy of a Bicubic Polynomial

HIROSHI AKIMA
U.S. Department of Commerce, NTIA/ITS

A local algorithm for smooth surface fitting for rectangular-grid data has been presented. It has the accuracy of a bicubic polynomial.

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

General Terms: Algorithm

Additional Key Words and Phrases: Bivariate interpolation, cubic polynomial, interpolation, local interpolation

1. DESCRIPTION

This algorithm performs bivariate interpolation and surface fitting for rectangular-grid data and is presented as an improvement of Algorithm 474 by Akima [1974].

This algorithm has the accuracy of a bicubic (bivariate third-degree) polynomial, i.e., it correctly interpolates data taken from a surface that represents a bicubic polynomial. The accuracy of a cubic (third-degree) polynomial is considered desirable for an interpolation algorithm, as demonstrated in univariate interpolation by Akima [1991].

This algorithm is based on local procedures, as is Algorithm 474 [Akima 1974]. The method used by this algorithm consists of the following steps:

(1) estimation of three partial derivatives, z_x, z_y, z_{xy}, at each input grid point;
(2) locating, in which rectangle of the input grid, the point at which interpolation is to be performed;

Author's address: U.S. Department of Commerce, NTIA/ITS.N4, 325 Broadway, Boulder, CO 80303-3328; email: akima@ntia.its.bldrdoc.gov.

(3) determining a bicubic polynomial in x and y for the rectangle; and
(4) calculation of the z value by evaluating the bicubic polynomial.

Note that steps (1) and (2) are mutually independent, and either step can be performed first.

Step (1) of the method used in this algorithm is new. All other steps are essentially the same as in Algorithm 474 [Akima 1974]. Step (1) of the method consists of three substeps, i.e., substeps (1.1), (1.2), and (1.3) for estimation of partial derivative z_x, z_y, z_{xy}, respectively.

In substep (1.1), a set of four primary estimates of partial derivative z_x is calculated at each input grid point, each primary estimate calculated from a cubic polynomial in x fitted to a set of four contiguous input grid points in the x direction that includes the input grid point in question, and next the final estimate of the partial derivative is calculated at the input grid point as a weighted mean of the four primary estimates. The weight is the reciprocal of the product of the volatility factor and the distance factor: the volatility factor is the sum of the squares of the deviations in the z value at the four grid points from the least-squares-fit line, and the distance factor is the sum of the squares of the distances of the three input grid points from the input grid point in question.

In this substep, all four primary estimates of z_x are not available for an input grid point if its x coordinate is equal to one of the three minimum or maximum x values of the input grid. For such a grid point, the method uses only available primary estimate or estimates.

In substep (1.2), the final estimates of partial derivative z_y are calculated at each input grid point in a similar manner to substep (1.1).

In substep (1.3), a set of 16 primary estimates of partial derivative z_{xy} is calculated at each input grid point, each primary estimate calculated from a bicubic polynomial in x and y fitted to a set of 4-by-4 contiguous input grid points in the x and y directions that includes the input grid point in question, and next the final estimate of the partial derivative is calculated at each input grid point as a weighted mean of the 16 primary estimates. The weight is the reciprocal of the product of the volatility factor and the distance factor: the volatility factor is the sum of the squares of the deviations in the z value at the 16 grid points from the least-squares-fit plane, and the distance factor is the product of two distance factors in the x and y directions.

In this substep, all 16 primary estimates of z_{xy} are not available for an input grid point if its x coordinate is equal to one of the three minimum or maximum x values of the input grid or if its y coordinate is equal to one of the three minimum or maximum y values of the input grid. For such a grid point, the method uses only the available primary estimate or estimates.

It is clear from the procedures in step (1) that all the estimates of partial derivatives have the accuracy of a bicubic polynomial, and so does the whole procedure of bivariate interpolation and surface fitting.

Although the main function of this algorithm is interpolation inside the boundary of the input grid, extrapolation outside the boundary of the input

grid is included, just for the convenience of the user, as in Algorithm 474 [Akima 1974]. Since the extrapolated z value is linear along a line perpendicular to the boundary of the input grid in this algorithm (as in Algorithm 474), accuracy of the bicubic polynomial is not preserved outside the boundary of the input grid. It is recommended that this algorithm be used sparingly for extrapolation.

The RGBI3P/RGSF3P subroutine package that implements the method consists of five subroutines: two master subroutines, which interface the user, and three supporting subroutines. The subroutines are as follows:

RGBI3P:	a master subroutine that performs bivariate interpolation, i.e., estimates the z values at the specified points in the x-y plane;
RGSF3P:	a master subroutine that performs surface fitting, i.e., estimates the z values at the specified rectangular output grid points in the x-y plane and generates a two-dimensional array containing these estimated z values;
RGPD3P:	a supporting subroutine that estimates partial derivatives at the input grid points;
RGLCTN:	a supporting subroutine that locates points in an input grid in the x-y plane, i.e., determines to which rectangle each of the points to be located belongs; and
RGPLNL:	a supporting subroutine that determines a polynomial in x and y for each rectangle in the x-y plane and calculates the z value by evaluating the polynomial for each of the desired points.

All the subroutines are written in the 1977 ANSI Standard Fortran.

Mutual dependencies of the subroutines of the RGBI3P/RGSF3P subroutine package are described in the following:

The user	calls RGBI3P or RGSF3P.
RGBI3P	calls RGPD3P, RGLCTN, and RGPLNL.
RGSF3P	calls RGPD3P, RGLCTN, and RGPLNL.
RGPD3P	calls none.
RGLCTN	calls none.
RGPLNL	calls none.

2. TESTS

Accuracy of the algorithm has been tested with the following test functions used by Franke [1979] in his study of scattered-data surface fitting:

(1) a combination of four Gaussian functions;
(2) a hyperbolic tangent function (almost a cliff);
(3) a saddle-shaped function;
(4) a Gaussian function (gentle slope);
(5) a Gaussian function (steep slope); and

Table I. Comparison of RGSF3P with RGSFOA and RGSFOS

Input			Function					
Grid	Error	Subroutine	F1	F2	F3	F4	F5	F6
6*6	Mean	RGSF3P	0.03168	0.00659	0.00128	0.00036	0.00943	0.00032
		Better of OA and OS	0.02238	0.00683	0.00301	0.00257	0.00439	0.00064
		Ratio	1.42	0.96	0.42	0.14	2.15	0.49
	Max	RGSF3P	0.19941	0.02000	0.00811	0.00161	0.03083	0.00194
		Better of OA and OS	0.08578	0.02773	0.01571	0.00650	0.05755	0.00342
		Ratio	2.32	0.72	0.52	0.25	0.54	0.57
	—	Is RGSF3P Better?	X	O	O	O	X	O
7*7	Mean	RGSF3P	0.01357	0.00365	0.00110	0.00026	0.00411	0.00015
		Better of OA and OS	0.01490	0.00431	0.00179	0.00124	0.00358	0.00038
		Ratio	0.91	0.84	0.61	0.21	1.15	0.40
	Max	RGSF3P	0.09428	0.01455	0.00629	0.00082	0.01489	0.00116
		Better of OA and OS	0.09404	0.02099	0.00783	0.00306	0.01659	0.00220
		Ratio	1.00	0.69	0.80	0.27	0.90	0.53
	—	Is RGSF3P Better?	O	O	O	O	X	O
8*8	Mean	RGSF3P	0.00555	0.00203	0.00034	0.00021	0.00084	0.00008
		Better of OA and OS	0.00891	0.00280	0.00102	0.00068	0.00211	0.00023
		Ratio	0.62	0.72	0.34	0.32	0.40	0.35
	Max	RGSF3P	0.04050	0.01178	0.00228	0.00065	0.00599	0.00079
		Better of OA and OS	0.05844	0.01532	0.00758	0.00184	0.00717	0.00163
		Ratio	0.69	0.77	0.30	0.35	0.84	0.49
	—	Is RGSF3P Better?	O	O	O	O	O	O
9*9	Mean	RGSF3P	0.00307	0.00121	0.00036	0.00016	0.00108	0.00005
		Better of OA and OS	0.00605	0.00186	0.00061	0.00039	0.00098	0.00015
		Ratio	0.51	0.65	0.59	0.41	1.10	0.31
	Max	RGSF3P	0.02079	0.00888	0.00207	0.00045	0.00441	0.00056
		Better of OA and OS	0.05380	0.01175	0.00509	0.00108	0.00619	0.00124
		Ratio	0.39	0.76	0.41	0.41	0.71	0.45
	—	Is RGSF3P Better?	O	O	O	O	O	O
10*10	Mean	RGSF3P	0.00217	0.00077	0.00014	0.00012	0.00065	0.00003
		Better of OA and OS	0.00348	0.00128	0.00040	0.00026	0.00089	0.00011
		Ratio	0.63	0.60	0.34	0.46	0.73	0.27
	Max	RGSF3P	0.02422	0.00669	0.00138	0.00033	0.00607	0.00040
		Better of OA and OS	0.01844	0.00981	0.00299	0.00070	0.00952	0.00095
		Ratio	1.31	0.68	0.46	0.47	0.64	0.42
	—	Is RGSF3P Better?	O	O	O	O	O	O
11*11	Mean	RGSF3P	0.00132	0.00051	0.00018	0.00007	0.00017	0.00002
		Better of OA and OS	0.00270	0.00089	0.00027	0.00017	0.00044	0.00008
		Ratio	0.49	0.57	0.66	0.41	0.38	0.23
	Max	RGSF3P	0.01746	0.00544	0.00118	0.00022	0.00064	0.00029
		Better of OA and OS	0.02832	0.00775	0.00250	0.00048	0.00206	0.00073
		Ratio	0.62	0.70	0.47	0.46	0.31	0.40
	—	Is RGSF3P Better?	O	O	O	O	O	O

(6) a part of a sphere.

In the Fortran notation, these six functions are written as

```
      F1(X,Y) = 0.75*EXP(−((9.0*X − 2.0)**2 + (9.0*Y − 2.0)**2)/4.0)
     1        +0.75*EXP(−((9.0*X + 1.0)**2)/49.0 − (9.0*Y + 1.0)/10.0)
     2        +0.50*EXP(−((9.0*X − 7.0)**2 + (9.0*Y − 3.0)**2)/4.0)
     3        −0.20*EXP(−(9.0*X − 4.0)**2 − (9.0*Y − 7.0)**2)
      F2(X,Y) = (TANH(9.0*Y − 9.0*X) + 1.0)/9.0
      F3(X,Y) = (1.25 + COS(5.4*Y))/(6.0*(1.0 + (3.0*X − 1.0)**2))
      F4(X,Y) = EXP(−81.0*((X − 0.5)**2 + (Y − 0.5)**2)/16.0)/3.0
      F5(X,Y) = EXP(−81.0*((X − 0.5)**2 + (Y − 0.5)**2)/4.0)/3.0
      F6(X,Y) = SQRT(64.0 − 81.0*((X − 0.5)**2 + (Y − 0.5)**2))/9.0−0.5
```

Performance of the RGSF3P subroutine has been compared with those of the RGSFOA and RGSFOS subroutines. The RGSFOA is the same as Algorithm 474 [Akima 1974], and the RGSFOS subroutine is the surface-fitting subroutine based on the osculatory interpolation, also described in Akima [1974]. These three subroutines have been run with uniformly spaced 6-by-6, 7-by-7, 8-by-8, 9-by-9, 10-by-10, and 11-by-11 input grids in a unit square and for a uniformly spaced 33-by-33 output grid. For each run, the mean and maximum (absolute) errors (deviations from the expected values) have been calculated. Table I shows the errors resulting from the test runs of the RGSF3P subroutine in the "RGSF3P" rows, and the better of the RGSFOA and RGSFOS subroutines in the "Better of OA and OS" rows. Ratios of the errors resulting from RGSF3P to the better of RGSFOA and RGSFOS are calculated and listed in the "Ratio" rows. We consider that RGSF3P is better when the product of the two ratios for the mean and maximum is less than 1.0 for each combination of test function and input grid. In the "Is RGSF3P Better?" rows, the symbol "O" signifies the case and "X" otherwise. Table I indicates that RGSF3P outperforms the other two algorithms in 33 combinations out of a total of 36 combinations of six functions and six input grids.

REFERENCES

AKIMA, H. 1974. A method of bivariate interpolation and smooth surface fitting based on local procedures. *Commun. ACM 17*, 1 (Jan.), 18–20. See also the companion algorithm. Algorithm 474. *Commun. ACM 17*, 1 (Jan.), 26–31.

AKIMA, H. 1991. A method of univariate interpolation that has the accuracy of a third-degree polynomial. *ACM Trans. Math. Softw. 17*, 3 (Sept.), 341–366. See also the companion algorithm. Algorithm 697. *ACM Trans. Math. Softw. 17*, 3, 367.

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

Received August 1994; accepted August 1995