Computational Statistics & Data Analysis 15 (1993) 329–342
North-Holland

Getting better contour plots with S and GCVPACK

Douglas Bates, Fred Reames and Grace Wahba
University of Wisconsin-Madison, Madison, WI 53706, USA

Received January 1991
Revised October 1991

Abstract: We show how to obtain esthetically pleasing contour plots using New S and GCVPACK. With these codes, thin plate splines can easily be used to interpolate "exact" data, and to produce smoothly varying contour plots, with none of the jagged corners that plague many other interpolation methods. It is noted that GCVPACK can also be used to interpolate data on the sphere and in Euclidean three space. We observe that a larger class of global interpolation methods (including the thin plate spline) have a Bayesian interpretation, and GCVPACK can be used to compute them.

Keywords: Bivariate interpolation; Smooth surfaces; Contouring methods; Thin plate splines; S; GCVPACK; Bayesian interpolation.

1. Introduction

Recently, two of us (F.R. and G.W.) have been involved in a Monte Carlo study, the outputs of which include computed values of a rather complicated response, as a function of two variables. We wished to obtain nice contour plots of these outputs so that our audience could visualize the behavior of this response. Ordinarily, one would evaluate the response on a sufficiently fine grid, for example, 80×80, and then, assuming that the underlying "true" response function is reasonably well behaved, any one of a number of readily available contouring programs which use simple linear interpolation between data points can make an attractive contour plot. We mention a grid of 80×80, because that typically provides sufficient resolution for an esthetically pleasing $8\ \frac{1}{2} \times 11$ plot. We have found the New S routines `contour` and `interp` (Becker, Chambers and Wilks, 1988, to be discussed further) to be quite easy to use and adequate in this situation. However, in our Monte Carlo study, it is quite expensive to calculate the response values. We computed our responses on a 13×21 grid of

Correspondence to: Prof. G. Wahba, Department of Statistics, 1210 West Dayton Street, University of Wisconsin, Madison, WI 53706, USA.

(x, y) values, and attempted to use the contouring programs in S to plot them. We were not happy with the results. Contouring relatively sparse or irregular data is in fact a difficult problem and not all interpolation methods are equally good for producing a smooth appearance on the resulting contours. We were not, however, prepared to compute on a finer grid, due to the large computation costs involved. We felt, since our response functions were known a priori to be visually "smooth", that there should be a method of producing smooth and esthetically pleasing contours. The first author (D.B.) suggested that we use the code GCVPACK (Bates et al., 1987) to interpolate the data to a finer grid via the use of a thin plate spline, and then use S to generate contour plots over this expanded data set.

GCVPACK contains code designed for computing thin plate smoothing splines, with generalized cross validation for choosing the smoothing parameter. It was not designed with interpolation of "exact" data in mind. However, it was designed with certain defaults which serendipitously allow the program to work as a "near interpolator", under conditions typically encountered when plotting data from "exact" computer experiments. That is, the interpolation can be controlled to be good to say, 5 or 6 figures, which is quite satisfactory for graphical work. We note that it is not necessarily desirable for a global interpolation scheme to be "exact" past the number of figures available, as the results can be sensitive to roundoff error, even if the computed data is good to 5 or 6 figures. That may well be one of the reasons for the lack of general use of good global interpolation schemes, although they have the potential for generating much nicer pictures in some cases than any local method. In our case, the design of GCVPACK allows the use of thin plate splines as a 5 or 6 figure interpolator of data sets of the size of a few hundred, while preventing this sensitivity to roundoff error. GCVPACK is available through `netlib` as a stand alone program, and there is also an S interface to GCVPACK. `netlib` may be accessed via `netlib@research.att.com`. Thus, it was not hard to carry out the proposed program. We were extremely pleased with the results. In this paper, we will describe those results, tell the reader some things about GCVPACK that are relevant to using it for contouring "exact" data, and remark on what might be expected on other examples. Considering the fact that jagged looking plots of smooth functions computed "exactly" appear in the literature all the time, we believe that this method will turn out to be extremely useful.

In Section 2 we describe our experience with GCVPACK and S, on our example. In Section 3 we discuss some of the features of GCVPACK which are relevant to smooth interpolation, so that the reader can better understand what might be expected in other examples. In Section 4 we describe the results of a major simulation study by Franke (1979) which compares a number of local and global interpolation methods, including thin plate splines. In Section 5 we note that global interpolation methods in a broad class, including thin plate splines, are Bayes estimates, and we indicate how S and GCVPACK may be used to obtain an interpolation scheme on the sphere, and in three dimensions. In the Appendix we briefly describe the Monte Carlo study that gave rise to our data.

2. Getting the contour plots

In this section we will guide the reader through the steps we went through to get our "beautiful" plots. We will describe several pitfalls that we had on the way, as these are just the traps that the unsuspecting user may fall into if trying to repeat our results.

In our computer experiment, described in the Appendix, we chose to evaluate the functions of interest on a 13×21 array of (x, y) coordinates. This grid is denser in the y direction because evaluation at different y's for the same x is relatively cheap. But every time the x coordinate is changed, we had to recompute the singular value decomposition of a 900×600 matrix and that takes a long time, even on the Cray Y-MP at the San Diego Supercomputer Center we were using.

We were doing the contouring locally on a DECstation 3100. Even though the evaluations on the Cray were accurate to about 7 or 8 significant figures, we initially transferred only three significant figures after the decimal point as shown in Table 1.

Figure 1 gives the result of entering this set of three figure data into the S `contour` routine. Contour uses linear interpolation between data points on a rectangular grid. The `contour` routine got confused in finding the contours for this data.

The next obvious step was to present `contour` with the 6 figure data; the result appears in Figure 2. Certainly useful, but still not very pretty. S has a routine `interp` which uses Akima's (1978) method to interpolate regular or

Table 1
Input data, rounded to three figures

		x												
		−6			−3			0			3			6
y	10	0.546	0.508	0.470	0.438	0.413	0.392	0.377	0.363	0.356	0.352	0.351	0.351	0.354
		0.518	0.478	0.444	0.417	0.395	0.378	0.365	0.355	0.349	0.347	0.347	0.349	0.352
		0.489	0.453	0.424	0.400	0.381	0.367	0.356	0.348	0.345	0.343	0.344	0.347	0.351
		0.465	0.433	0.407	0.387	0.370	0.358	0.350	0.343	0.341	0.341	0.343	0.346	0.351
		0.446	0.418	0.395	0.376	0.362	0.352	0.345	0.340	0.339	0.339	0.342	0.346	0.351
	5	0.431	0.405	0.385	0.368	0.356	0.347	0.341	0.338	0.337	0.339	0.342	0.346	0.351
		0.419	0.396	0.377	0.362	0.351	0.344	0.339	0.336	0.336	0.338	0.342	0.346	0.352
		0.410	0.389	0.371	0.358	0.348	0.341	0.337	0.336	0.336	0.338	0.342	0.347	0.353
		0.403	0.383	0.367	0.355	0.346	0.340	0.336	0.335	0.336	0.339	0.342	0.347	0.353
		0.398	0.379	0.364	0.353	0.345	0.339	0.336	0.335	0.336	0.339	0.343	0.348	0.354
	0	0.395	0.377	0.363	0.352	0.344	0.339	0.336	0.336	0.337	0.340	0.344	0.349	0.355
		0.392	0.375	0.361	0.351	0.344	0.339	0.336	0.336	0.338	0.340	0.345	0.350	0.356
		0.391	0.374	0.361	0.351	0.344	0.339	0.337	0.337	0.338	0.341	0.345	0.351	0.356
		0.390	0.374	0.361	0.351	0.344	0.340	0.338	0.338	0.339	0.342	0.346	0.351	0.357
		0.390	0.374	0.361	0.352	0.345	0.340	0.338	0.338	0.340	0.343	0.347	0.352	0.358
	−5	0.390	0.374	0.362	0.352	0.345	0.341	0.339	0.339	0.341	0.344	0.348	0.353	0.359
		0.390	0.375	0.362	0.353	0.346	0.342	0.340	0.340	0.341	0.344	0.348	0.353	0.359
		0.391	0.375	0.363	0.354	0.347	0.343	0.341	0.341	0.342	0.345	0.359	0.354	0.360
		0.392	0.376	0.364	0.354	0.348	0.344	0.341	0.341	0.343	0.346	0.350	0.355	0.360
		0.393	0.377	0.365	0.355	0.349	0.344	0.342	0.342	0.344	0.346	0.350	0.355	0.361
	−10	0.393	0.378	0.366	0.356	0.349	0.345	0.343	0.343	0.344	0.347	0.351	0.356	0.361

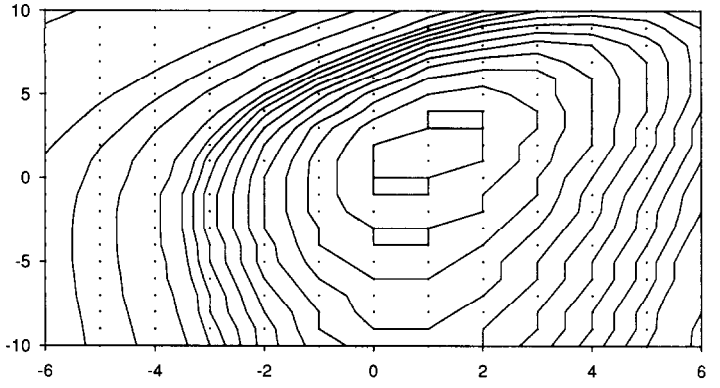

Fig. 1. Plot from `contour` for three figure data.

irregular data to a regular 40×40 grid. (`Contour` needs data on a regular grid.) We gave `interp` our 13×21 array (of 6 figure) data and put the resulting 40×40 array of data points from `interp` through `contour`. The resulting curves had visible corners similar to those in Figure 2, only more of them. We then changed the `interp` from its 40×40 default grid to 97×81, making the grid 8 times finer in x and 4 times finer in y than the original 13×21 array (keeping track of end points). The result appears in Figure 3. This picture is slightly more pleasing than picture we got from the 40×40 grid (not shown), and more pleasing than Figure 2, but not by much. We also read the 21 columns of data into `interp` in the opposite order than they had originally been entered to produce Figure 3; simultaneously the y values were reversed. In theory one would expect the same picture, but in practice Akima's method is based on a triangulation of the data points: entering data in a different order in `interp` can produce a different triangulation and hence a different interpolant. This apparently happened here, because the resulting 97×81 interpolated data points actually satisfied $\max |(m_1 - m_2)/m_2| = 0.0050$ where m_1 is a data point from the 97×81 array from the first try (Figure 3), and m_2 is the same data

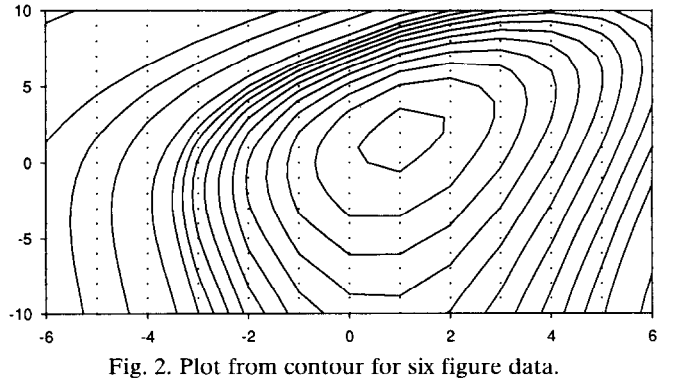

Fig. 2. Plot from contour for six figure data.

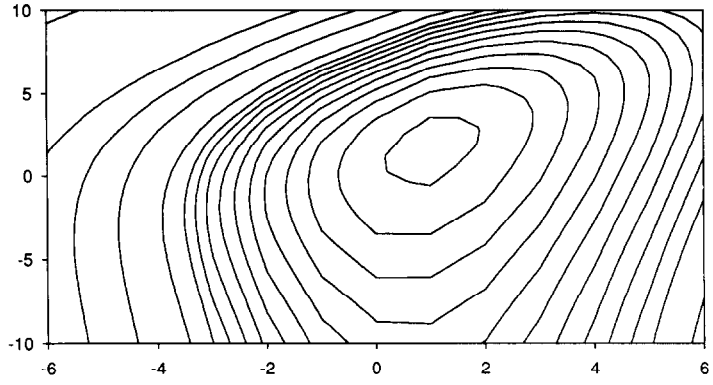

Fig. 3. Plot using `contour` on the `interp` output with 97×81 `interp` grid.

point from this second try with the reversed data order. The result appears in Figure 4.

Next, we gave the three figure data to GCVPACK to fit a thin plate spline. The thin plate spline we are discussing here (more details in Section 3) has a representation in terms of $n + 3 = (13 \times 21) + 3 = 276$ basis functions. The code computes the coefficients of these basis functions; then the thin plate spline can be evaluated to a high degree of accuracy at any (x, y) values. All the GCVPACK calculations were done on a Sun workstation in double precision: nothing less than this precision should be used. GCVPACK actually smooths the data and the degree of smoothing is determined by a smoothing parameter. In the examples here the smoothing parameter was set to its minimum GCVPACK default value, which is a computed function of the data design. (See Section 3 for more details.) GCVPACK returns the residual sum of squares of the difference between the thin plate spline at the 13×21 data points and the data. The result in this case was a root mean square deviation of 1.43×10^{-6}, so that the first three figures in the values of the spline at the data points are apparently matching the three figure input exactly; in fact, the spline is appar-

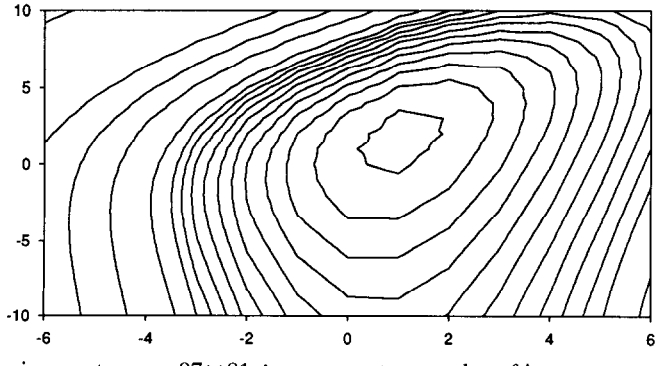

Fig. 4. Plot using contour on 97×81 `interp` output, order of input to `interp` reversed.

Table 2
First column of fitted three figure data demonstrating the interpolation of zero in the fourth place after the decimal point

0.546046	0.431021	0.398010	0.390005	0.391005
0.518040	0.419019	0.395006	0.390004	0.392007
0.489038	0.410016	0.392008	0.390004	0.393007
0.465032	0.403012	0.391003	0.390007	0.393011
0.446027				

ently matching at least one 0 after the given three figure input data. To see this graphically, Table 2 gives the values of the thin plate spline computed at the data points corresponding to the first column of the data in Table 1. The three figure data of Table 1 are being "interpolated" to 4 or 5 figures by fitting zeroes in the fourth and sometimes fifth place. We computed the values of this thin plate spline to over 6 figures on the same 97×81 grid as before, and gave these data to `contour`. The result appears in Figure 5. What is happening here is that rounding the input data to three figures has introduced a non-negligible amount of "noise" in the otherwise smooth input data, and GCVPACK, which is designed to run in double precision, is interpolating one or two figures of this roundoff noise.

Finally, we gave the 6 figure 97×81 GCVPACK output to `contour`, and the result is given in Figure 6, which gives the final contour plot of our data. This is the result we were looking for. Table 3 gives an overview of the six contour plots. We remark that no "cheating" is going on. Although a smoothing thin plate spline has been used, the rms difference between the 6 figure input data to GCVPACK, and the 6 figure output of GCVPACK at the data points was 0.797×10^{-6}. If we pretend that numbers rounded off to k places after the decimal have errors which are uniformly distributed between plus and minus $\frac{1}{2} \times 10^{-k}$ then their standard deviation is about 0.3×10^{-k}. One would then like the root mean square fit of the spline to the data to be a bit better than this

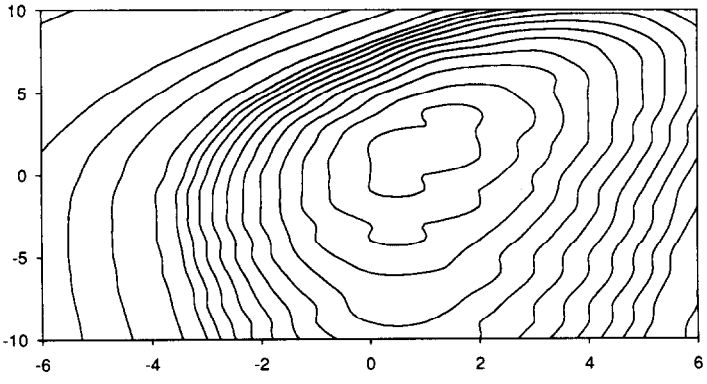

Fig. 5. Plot using contour on 97×81 GCVPACK output, using three figure input data.

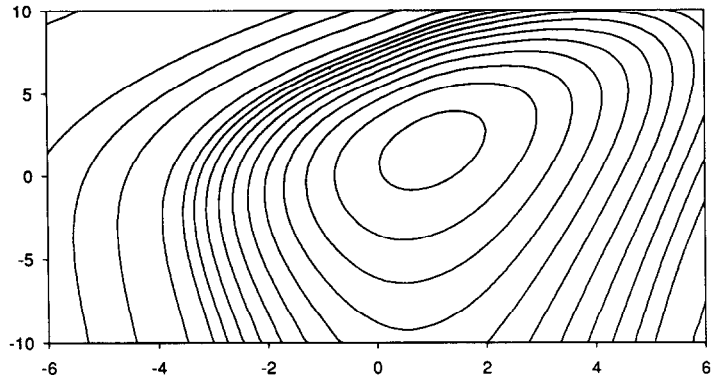

Fig. 6. Plot using contour on 97×81 GCVPACK output, using six figure input data and six figure GCVPACK output data.

(Wahba, 1975). Thus the desired interpolation is not "exact"; instead it is matching the input data to between 5 and 7 figures, resembling the accuracy with which the input data has been provided. In the next section we will describe how this comes about.

3. The thin plate smoothing spline

Given data (x_k, y_k, z_k) on a not necessarily regular grid, the thin plate spline with derivative parameter m is the solution to the following variational problem: Find f in a appropriate function space to minimize

$$\frac{1}{n}\sum_{k=1}^{n}\left(z_k - f(x_k, y_k)\right)^2 + \lambda J_m(f), \tag{1}$$

where $J_m(f)$ is the thin plate penalty functional

$$J_m(f) = \sum_{\nu=0}^{m}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\binom{m}{\nu}\left(\frac{\partial^m f}{\partial x^\nu\,\partial y^{m-\nu}}\right)^2 \mathrm{d}x\,\mathrm{d}y. \tag{2}$$

Table 3
Overview of alternative procedures for contour plots from 13×21 data set

Figure	Decimal figures used	Preprocessing procedure	Size of temporary data matrix [b]	Evaluation of contour plot
1	3	–	13×21	Unsatisfactory
2	6	–	13×21	Better than Figure 1
3	6	S-interp	97×81	Better than Figure 2
4 [a]	6	S-interp	97×81	Artifact of S-interp
5	3	GCVPACK	97×81	Responding to roundoff noise
6	6	GCVPACK	97×81	Desired result

[a] order of axes reversed.
[b] output from preprocessing, input for S-contour.

For $m = 2$ this becomes

$$J_2(f) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \left(f_{xx}^2 + 2f_{xy}^2 + f_{yy}^2\right) \mathrm{d}x\ \mathrm{d}y. \tag{3}$$

In the discussion that follows, we describe only the $m = 2$ case. Provided the design points $(x_k, y_k) = P_k$, $k = 1, \ldots, n$ do not all fall on a straight line, there is a unique minimizer f_λ to (1) for every $\lambda > 0$. Furthermore, f_λ has a representation

$$f_\lambda(P) = \sum_{\nu=1}^{3} \beta_i \phi_i(P) + \sum_{k=1}^{n} \delta_k E(P, P_k), \tag{4}$$

where

$$P = (x, y), \quad P_k = (x_k, y_k), \qquad \phi_1(P) = 1, \quad \phi_2(P) = x, \quad \phi_3(P) = y \tag{5}$$

and

$$E(P, Q) = \| P - Q \|^2 \ln(\| P - Q \|). \tag{6}$$

Here $\| P - Q \|$ is the Euclidean distance between P and Q. See Duchon (1977), Wahba and Wendelberger (1980). We will reproduce the equations for $\beta = (\beta_1, \beta_2, \beta_3)$, and $\delta = (\delta_1, \ldots, \delta_n)$. For details on their calculation, see Bates et al., (1987). Let T be the $n \times 3$ matrix with (k, ν) entry $\phi_\nu(P_k)$ and let

$$T = (F_1 : F_2)\begin{pmatrix} G \\ 0 \end{pmatrix} \tag{7}$$

be the Q–R decomposition of T. The three columns of F_1 span the column space of T, and the $n - 3$ columns of F_2 are perpendicular to the columns of T. G is 3×3. Let K be the $n \times n$ matrix with (k, l) entry $E(P_k, P_l)$, and let $F_2' K F_2 = UDU'$. Then (see Bates et al. (1987), Wahba (1990), and references cited therein)

$$\delta = F_2 U (D^2 + n\lambda I)^{-1} U' F_2' z \tag{8}$$

$$G\beta = F_1'(z - K\delta). \tag{9}$$

Given data (x_k, y_k, z_k), $k = 1, \ldots, n$, GCVPACK chooses λ, and returns β and δ; the user evaluates (4) on as fine a grid as desired. If the data come (exactly) from a plane, then z is in the column span of T, hence $F_2' z = 0$, and it can be checked that f_λ is (in theory) this plane, whatever the value of λ. As $\lambda \to 0$ then $f_0 = \lim_{\lambda \to 0} f_\lambda$ is (in theory, in an appropriate space) the minimizer of $J_2(f)$ subject to the interpolation conditions $f_\lambda(P_i) = z_i$, $i = 1, \ldots, n$. If you try data taken from a plane with known β's, or if you try $\lambda = 0$, on your computer, you can see the numerical accuracy of your computations by comparing known and computed β's or known and computed data. We also remark that $f_\beta = \lim_{\lambda \to \infty} f_\lambda$ is the plane best fitting the data in a least squares sense.

In the examples of size $n = 13 \times 21$ presented here, the $270 = n - 3$ eigenvalues of this problem, that is, the entries of D^2, ranged from $d_1^2 = 133.6741$ to $d_{270}^2 = 0.011756$ for a condition number of 11.37×10^3, probably not big enough

to cause problems in double precision, even for $n\lambda = 0$. For regularly spaced data the theoretical decay rate of these eigenvalues is $O(n^{-2})$; see Wahba (1979), Cox (1984). If the data are irregular and/or n is larger, the condition number will be bigger. Sibson and Stone (1991) have looked at preconditioning methods useful in the very ill conditioned case.

In GCVPACK the (default) method of choosing the smoothing parameter is by minimizing the GCV function $V(\lambda)$ given by

$$V(\lambda) = \frac{\Sigma\left[n\lambda/\left(d_k^2 + n\lambda\right)\right]^2 w_k^2}{\left[\Sigma n\lambda/\left(d_k^2 + n\lambda\right)\right]^2}, \tag{10}$$

where $(w_1, \ldots, w_{n-3})' = w = F_2' z$. The minimizer of $V(\lambda)$ has certain well known properties for the model

$$z_k = f(P_k) + e_k, \quad k = 1, \ldots, n, \tag{11}$$

when the e_k are independent, identically distributed N(0, σ^2), see e.g. Craven and Wahba (1979), Li (1986). GCVPACK was designed for the case in which σ^2 was assumed to correspond to independent "engineering sized" errors as opposed to possibly correlated, small roundoff errors. GCVPACK does a search over λ between default limits of $\lambda_{\min} \approx 0.01 d_{n-3}^2/n$ and $\lambda_{\max} \approx 100 d_1^2/n$, where d_{n-3}^2 and d_1^2 are respectively the smallest and largest eigenvalues in (8). Alternatively, the user may specify the upper and lower search limits, including the possibility of choosing λ by setting them equal. These choices in the design of GCVPACK were made for the purpose of maintaining numerical stability within the context of a completely automatic search procedure. The experiments carried out during the development of GCVPACK suggested that $f_{\lambda_{\min}}$ and $f_{\lambda_{\max}}$ were, for most practical purposes, indistinguishable from f_0 and f_∞.

In all of the examples presented here, $\hat{\lambda} = \lambda_{\min}$; however, in some similar examples $\hat{\lambda}$ turned out to be slightly larger. GCVPACK will return $\hat{\lambda}$ as well as the residual sum of squares, so that if the user's goal is "interpolation", one can check to see if the root mean square residual corresponds to matching the input data to as many significant figures as appropriate. Note that our example shows that GCVPACK may interpolate zeroes beyond the number of figures provided in the data. If this happens and causes problems, then λ should be made larger until the residual mean square corresponds to a roundoff error in the last significant figure in the input data.

The thin plate spline with $m = 3$ will be exact on quadratic functions and will have a slightly different "bendiness" to the contours. See Boult (1987). We tried $m = 3$ but the difference between the resulting contours was just barely visible and the plot is not reproduced here.

As a final question we wondered whether we had really needed as much data as we had computed. To answer this question we removed the data for every other x value, leaving a 7×21 grid of points. The contours for both the $m = 2$ and $m = 3$ thin plate splines were again just barely visibly different from Figure 6, and we do not reproduce them here. Thus, for the purposes of generating the

contour plot to essentially visual accuracy, we could, in this example, have done with about half as many data points. In the $m = 2$ case, the rms difference between the 7×21 six figure input data and the fitted spline was 0.110×10^{-5}, while the rms difference between the spline and the 6×21 data that had been set aside was 0.491×10^{-2}. The code can easily be used during the course of an experiment to see whether further data should be taken, since regular data are not necessary to fit the spline and the spline may be evaluated anywhere. The experimenter can do this by withholding one or a few special data points, or by obtaining new data points and comparing them to the values "forecasted" by the spline based on the previous data points, or comparing the two splines computed with and without the special or new data points. These comparisons may of course vary over the (x, y) plane. We remark that we had a particular interest in the minimizer of the function being studied and we conjecture that the minimizer of the fitted spline is a quite reasonable estimate of the minimizer of the unknown function in examples like this one.

4. Franke's comparisons

In 1979 shortly after the original thin plate spline theory became available Franke (1979, 1980) did an extensive numerical study comparing different methods of interpolating scattered, "exact" data on the plane, including inverse distance weighted methods, methods based on triangulation, finite element methods and a group of methods which consisted of thin plate splines and Hardy's multiquadrics. This latter group (which can all be shown to be Bayes methods, more on that later) were the only methods which in Franke's study received an "A" for visual appearance. Existence properties of Hardy's multiquadrics were not even known until the important work of Micchelli (1986). Our visual results agree with Franke's in the case of the thin plate spline. From Franke's pictures, there is little visually to distinguish between the thin plate spline and the multiquadrics, on the examples he tried. The thin plate splines will interpolate a plane by a plane ($m = 2$), and a quadratic by a quadratic ($m = 3$); however, the main advantage of using the thin plate spline for us at this time is the availability of GCVPACK and the ability to use it easily in conjunction with S.

5. Interpolation formulae as Bayes estimates, interpolation on the sphere and in Euclidean 3-space

For this discussion it is not necessary that P_k be in Euclidean 2-space. P_k may be in Euclidean 3-space, or, just in an abstract metric space $\mathcal{T}$. Let $\phi_1(P), \ldots, \phi_M(P)$ be $M < n$ linearly independent functions with the property that least squares regression on their span for the design $P_1, \ldots, P_n$ is unique, that is, the $n \times M$ matrix T with i, ν entry $\phi(P_{i\nu})$ is of rank M. Let $K(P, P')$ be

any positive definite function on $\mathscr{T} \times \mathscr{T}$, that is, $\Sigma_{j,k} a_j a_k K(P_j, P_k) > 0$ for any $a = (a_1, \ldots, a_n)$ not identically 0. These two ingredients will always lead to an interpolation formula which is exact on span $\phi_1, \ldots, \phi_M$ via the formula

$$f(P) = \sum_{\nu=1}^{M} \beta_\nu \phi_\nu(P) + \sum_{k=1}^{n} \delta_k K(P, P_k) \tag{12}$$

where, if we let T be the $n \times M$ matrix with $i\nu$ entry $\phi_\nu(P_i)$ and K be the $n \times n$ matrix with jk entry $K(P_j, P_k)$, then $\beta = (\beta_1, \ldots, \beta_M)$ and $\delta = (\delta_1, \ldots, \delta_n)$ are given by (8) and (3) with 3 replaced by M. See Kimeldorf and Wahba (1971), Micchelli (1986), Wahba (1990).

Now consider the stochastic model

$$Z(P) = \sum_{\nu=1}^{M} \theta_\nu \phi_\nu(P) + Z_1(P), \quad P \in \mathscr{T}, \tag{13}$$

where Z_1 is a zero mean Gaussian stochastic process with $EZ_1(P)Z_1(P') = K(P, P')$ and $\theta \sim \mathrm{N}(0, \xi I)$. It is well known (see Wahba, 1978) that

$$\lim_{\xi \to \infty} E\left[Z(P) \mid Z(P_k) = z_k,\ k = 1, \ldots, n\right] = f_0(P), \tag{14}$$

so that this type of interpolation formula has a Bayesian interpetation. Note that if Z_1 is replaced by $Z_1 + \Sigma_{\nu=1}^{M} \rho_\nu \phi_\nu$ for any ρ_ν, then f_0 will be unchanged. It can thus be seen that it is not necessary to know K completely: if $\tilde{K}$ is any symmetric function with the property that $\Sigma_k \delta_k (K(P, P_k) - \tilde{K}(P, P_k)) = 0$ whenever $T'\delta = 0$ then $\tilde{K}$ may be substituted for K and the result will be the same. For the existence of an interpolation formula of the form (12) it is only necessary that K be conditionally positive definite with respect to $\phi_1, \ldots, \phi_M$. This means that $\Sigma_{j,k} \delta_j \delta_k K(P_j, P_k)$ is required to be positive only when $T'\delta = 0$, equivalently when δ is in the column span of F_2. The variogram of kriging estimates have this property, see Matheron (1973). For an example of a non-negative definite function which is equivalent to $E(P, P')$ in this sense, see Wahba and Wendelberger (1980). See also Sacks et al. (1989).

Wahba (1981,1982) gives some relatively simple positive definite functions on the sphere. GCVPACK may be used with them to provide an interpolation of scattered data on the sphere. The partial spline model in GCVPACK may be used to obtain interpolation formulae which are "exact" on specified functions, see Wahba (1990), Bates et al., (1987). Alfeld (1989) provides a review of three-dimensional interpolation formulae. GCVPACK can also be used to interpolate three (and higher) dimensional data with thin plate splines. It is necessary that $2m$ be greater than the number of dimensions. The resulting thin plate spline will provide an "exact" interpolation on polynomials of total degree less than m.

6. Summary and conclusions

It has been shown how to use S and GCVPACK to obtain esthetically pleasing contour plots from very accurate data on a function known to be smooth, as

might arise from a computer experiment. GCVPACK represents (according to one referee) "heavy machinery"; however, since the code is free from `netlib`, and, we believe, easy to use, the cost of using this machinery to go from Figure 2 or Figure 3 to Figure 6 is essentially a modest amount of data entry time. This form of interpolation is proposed for surfaces that are known a priori to be very smooth, as our example was, and for which the corners generated using S alone represent a visual distraction. We have provided a cheap and easy way to remove this visual distraction. The code may also be used easily as an aid in deciding whether more data should be taken.

Finally, we have discussed thin plate splines as Bayes estimates and described their relationship to a broader class of global interpolation procedures, and we have noted that GCVPACK an be used for interpolation on Euclidean d space and on the sphere.

Appendix. The source of the data

We were studying a new cross-validatory-risk estimate for simultaneously estimating a smoothing and a weighting parameter and wanted to get nice plots to demonstrate how well the method worked on a realistic example. The model was

$$y_1 = X_1 f + \epsilon_1, \tag{15}$$

$$y_2 = X_2 f + \epsilon_2, \tag{16}$$

where y_i is of dimension n_i, f is of dimension p and $\epsilon_i \sim \mathrm{N}(0, \sigma_i^2 I)$, where the σ_i's are unknown. We estimated f by penalized least squares, and we estimated both a smoothing parameter and a relative weight to be given to the two data sets, by a new method whose properties are being studied by Monte Carlo methods. Our simulation study was geared to a particular application and we were interested in obtaining realistic accuracy estimates to assess how well the method might work in practice. The application had to do with estimating the global 500 millibar height (the height at which the atmospheric pressure is 500 millibars), given direct observational data from radiosondes (data source 1), and from a forecast (data source 2). The vector f contains the Fourier coefficients of the 500 millibar height with respect to an expansion in spherical harmonics used as basis functions. y_1 and y_2 represent the two different sources of data. In order to mimic a realistic problem in numerical weather forecasting, the n's and p had to be very large; we used $n_1 \sim 900$ and $n_2 = p \sim 600$.

For a given r, α the estimate $f_{r,\alpha}$ of f is the minimizer of

$$\frac{1}{r} \| y_1 - X_1 f \|^2 + r \| y_2 - X_2 f \|^2 + \alpha f' J f, \tag{17}$$

where J is a suitable penalty matrix. Letting $A_{ij}(r, \alpha)$ be the matrices satisfying

$$X_1 f_{r,\alpha} = A_{11} y_1 + A_{12} y_2, \tag{18}$$

$$X_2 f_{r,\alpha} = A_{21} y_1 + A_{22} y_2, \tag{19}$$

our cross-validation-risk estimate is the minimizer over r and α of

$$C(r, \alpha) = \frac{1}{n_1 + n_2}\left[\frac{\| y_1 - X_1 f_{r,\alpha} \|^2}{((1/n_1)\mathrm{tr}(I - A_{11}))^2} + \frac{\| y_2 - X_2 f_{r,\alpha} \|^2}{((1/n_2)\mathrm{tr}(I - A_{22}))^2}\right]. \quad (20)$$

Each time C was computed with a new r, it was necessary to compute a 600×900 singular value decomposition.

It can be shown using the arguments in Craven and Wahba (1979) that the minimizer $(\hat{r}, \hat{\alpha})$ of $C(\cdot, \cdot)$ is approximately a minimizer of

$$R(r, \alpha) = \frac{1}{n_1 + n_2}\left[\| X_1(f - f_{r,\alpha}) \|^2 + \| X_2(f - f_{r,\alpha}) \|^2\right]. \quad (21)$$

The data were values of $R(x, y)$ for a selected example, where $x = \text{const} \log r^2$ and $y = \text{const} \log(\alpha/\hat{\alpha})$. One of the goals of the original study was to compare the minimizers of R and C. Some corresponding plots of the "C" data eliminated here at the suggestion of a referee may be found in Bates et al. (1990).

Acknowledgements

Research supported by AFOSR under Grant AFOSR 90-0103, by NSF under Grant DMS-8801996 and by NASA under Contract NAG5-316. Some of the computing was done at the San Diego Supercomputer Center.

References

Akima, H., A method of bivariate interpolation and smooth surface fitting for irregularly distributed data points, *A.C.M. Trans. Math. Software*, **4** (1978) 148–164.

Alfeld, P., Scattered data interpolation in three or more variables, in: T. Lyche and L. Schumaker (Eds), *Mathematical Methods in Computer Aided Geometric Design* (Academic Press, New York, 1989) 1–34.

Bates, D., M. Lindstrom, G. Wahba and B. Yandell, GCVPACK-Routines for generalized cross validation, *Commun. Statist. Simul. Comput.*, **16** (1987) 263–297.

Bates, D., F. Reames and G. Wahba, Getting better contour plots with S and GCVPACK. Technical Report 865 (Statistics Dept., University of Wisconsin, Madison, 1990).

Becker, R., J. Chambers and A. Wilks, *The New S Language* (Wadsworth, Belmont, CA, 1988).

Boult, T., Optimal algorithms: tools for mathematical modeling. *J. Complexity*, **3** (1987) 183–200.

Cox, D., Multivariate smoothing spline functions, *SIAM J. Numer. Anal.*, **21** (1984) 789–813.

Craven, P. and G. Wahba, Smoothing noisy data with spline functions: estimating the correct degree of smoothing by the method of generalized cross-validation. *Numer. Math.*, **31** (1979) 377–403.

Duchon, J., Splines minimizing rotation-invariant semi-norms in Sobolev spaces, in: W. Schempp and K. Zeller, eds., *Constructive Theory of Functions of Several Variables* (Springer-Verlag, Berlin, 1977) 85–100.

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

Franke, R., Scattered data interpolation: test of some methods, *Math. Comp.*, **38** (1982) 181–200.

Kimeldorf, G. and G. Wahba, Some results on Tchebycheffian spline functions. *J. Math. Anal. Applic.*, **33** (1971) 82–95.

Li, K.C., Asymptotic optimality of C_L and generalized cross validation in ridge regression with application to spline smoothing, *Ann. Statist.*, **14** (1986) 1101–1112.

Matheron, G., The intrinsic random functions and their applications, *Adv. Appl. Prob.*, **5** (1973) 439–468.

Micchelli, C., Interpolation of scattered data: distance matrices and conditionally positive definite functions. *Constructive Approximation*, **2** (1986) 11–22.

Sacks, J., W. Welch, T. Mitchell and H. Wynn, Design and analysis of computer experiments, *Statistical Science*, **4** (1989) 409–435.

Sibson, R. and G. Stone, Computation of thin-plate splines, *SIAM J. Sci. Stat. Comput.*, **12** (1991) 1304–1313.

Wahba, G., Smoothing noisy data by spline functions. *Numer. Math.*, **24** (1975) 383–393.

Wahba, G., Improper priors, spline smoothing and the problem of guarding against model errors in regression. *J. Roy. Stat. Soc. Ser. B*, **40** (1978) 364–372.

Wahba, G., Convergence rates of "thin plate" smoothing splines when the data are noisy. in: T. Gasser and M. Rosenblatt. (Eds), *Smoothing Techniques for Curve Estimation, Lecture Notes in Mathematics, No. 757*, (1979) 232–246.

Wahba, G., Spline interpolation and smoothing on the sphere, *SIAM J. Sci. Stat. Comput.*, **2** (1981) 5–16.

Wahba, G., Erratum: Spline interpolation and smoothing on the sphere, *SIAM J. Sci. Stat. Comput.*, **3** (1982) 385–386.

Wahba, G., *Spline Models for Observational Data*, CBMS-NSF Regional Conference Series in Applied Mathematics, SIAM, Vol. 59, 1990.

Wahba, G. and J. Wendelberger, Some new mathematical methods for variational objective analysis using splines and cross-validation, *Monthly Weather Review*, **108** (1980) 1122–1145.