A Two-Stage Filter for Smoothing Multivariate Noisy Data on Unstructured Grids

F. SALES-MAYOR AND R. E. WYATT
Institute of Theoretical Chemistry, Department of Chemistry and Biochemistry
University of Texas at Austin
Austin, TX 78712, U.S.A.

(Received January 2002; and accepted July 2003)

Abstract—Experimental data as well as numerical simulations are very often affected by "noise", random fluctuations that distort the final output or the intermediate products of a numerical process. In this paper, a new method for smoothing data given on nonstructured grids is proposed. It takes advantage of the smoothing properties of kernel-weighted averaging and least-squares techniques. The weighted averaging is performed following a Shepard-like procedure, where Gaussian kernels are employed, while least-squares fitting is reduced to the use of very few basis functions so as to improve smoothness, though at the price of interpolation accuracy. Once we have defined an n-point grid and want to make a smoothed fit at a given grid point, this method reduces to consideration of all m-point stencils (for a given value of m where m < n) that include that point, and to make a least-squares fitting, for that particular point, in each of those stencils; finally, the various results, thus obtained, are weight-averaged, the weights being inversely proportional to the distance of the point to the middle of the stencil. Then this process is repeated for all the grid points so as to obtain the smoothing of the input function. Though this method is generalized for the multivariate case, one-, two-, and five-dimensional test cases are shown as examples of the performance of this method. © 2004 Elsevier Ltd. All rights reserved.

Keywords—Smoothing, Noisy data, Unstructured grids, Multivariate data, Weighted-averaged least squares.

1. INTRODUCTION

Smoothing of noisy data has always been a topic of interest in many areas where computer simulations have been performed, including natural sciences as well as economics and social sciences. Many different techniques have been proposed for general cases and also for specific problems that display some particular tough feature. Many of them use some of the different types of splines as $D^m$-splines by Torrens [1], applying them to Lagrange problems, and for general parametric surfaces with tangent conditions, by Pasadas et al. [2]. Membrane splines, thin-plate splines, or a combination of them, where Lai and Vemuri [3] imbed the surface to smooth that is defined on an irregular domain, in a regular region, preserving prespecified discontinuities in the data. Bivariate monotone Powell-Sabin splines, which are overall $C^1$-continuous and piecewise...
quadratic on each grid cell have been used by Willemans and Dierckx [4] for the smoothing of surfaces defined on unstructured grids. Other references where splines of several variables are used may be found in [5], where hypersurfaces defined on scattered grid points are smoothed, while Dierckx et al. [6] provide a smoothing spline surface fitting method for cases in which planar sections of a surface are provided; in this case, surfaces are represented by tensor product splines in a radial and an angular coordinate. Ignatov and Pevnyi [7] use B-splines finite elements to smooth out surfaces in complex grid geometries, while Lu and Mathis use cubic splines [8]. Whiten [9] uses multidimensional cubic spline functions for least-squares fitting of scattered data. A similar work was done by Guan [10] with polynomial natural splines. For the particular case of L-shaped domains, Hu and Mao [11] propose another spline-based technique for unstructured grids. Other interesting references illustrating the use of splines for surface smoothing may be found in the work by Feng [12], Avdeev and Dikoussar [13] or Hrebicek et al. [14].

Some other methods have been proposed to attain the same goal. Roach and Martin [15] have worked out Fourier transform smoothing techniques, similar to those used in smoothing signal noise, for surface smoothing of objects whose hypersurface is crudely defined with polyhedra, this being divided in patches to facilitate the handling of a large number of grid points. This method also permits alteration of the shape of the interior of the patches. Finite elements have been applied by Potier and Vercken [16] for surface smoothing when sampling data are distributed irregularly. Demirbaş [17] proposes an algorithm for nonlinear multidimensional systems based upon quantization, multiple hypothesis testing, and the Viterbi decoding algorithm, where the estimation of state vectors is carried out sequentially, component-by-component, and in parallel. Cleveland and Devlin [18] and Grosse [19] apply the “locally weighted regression” method or loess, which estimates a regression surface from scattered data through a multivariate smoothing procedure by local fitting of linear or quadratic models with a neighborhood weighting that is moved across the domain, in an analogous way to how a moving average is computed for a time series.

Of special interest, for certain kinds of stochastic processes may be considered the smoothing algorithms associated with the Kalman filter. The use of this filter reduces computational difficulties in the maximum likelihood estimation of a smoothing parameter. These algorithms are derived by means of the so-called conditional mean technique or by the Bayesian approach. A very good introduction to this subject can be found in [20], where three particular algorithms are considered, namely, the fixed-point smoother, the fixed-interval smoother, and the intermittent smoother; the factorization methods [21] can improve the numerical properties of these smoothing algorithms (Bierman). A state-space model to perform discrete thin plate smoothing for data on a two-dimensional rectangular lattice with the use of the Kalman filter has also been proposed by Kashiwagi et al. [22]. Another numerical filter in one dimension has been proposed by Huang and Zhou by investigating the properties of the Whittaker-Vondrak method of data smoothing [23]. Time-invariant filters have been also developed as, for example, for the design of fixed interval linear least-squares smoothing filters, by Levy et al. [24]. Bubb developed a 1D filter for discrete time series, consisting of a finite number of weight factors that are obtained from a finite set of linear equations subjected to the mean square error criterion [25]. A detailed study of the application of averaging to multidimensional IIR (infinite impulse response) and FIR (finite impulse response) filters for uniform and nonuniform rectangular grids may also be found in the work of Radecki et al. [26]. Hoang et al. [27,28] design a stable adaptive filter to apply it to systems whose state space is very high dimensional and defined on three-dimensional rectangular grids.

Other methods are mainly focused on interpolation, though they also perform smoothing of the sampling data. A summary of the various different cases of scattered data fitting and references for dealing with them, may be seen in [29]. Surface fitting where a set of scattered data is interpolated into a set of intercalated points that form a regular grid, may be found in [30,31], where a least-squares fitting of the data with orthogonal polynomials generated by the Gram-Schmidt
A Two-Stage Filter

A two-stage filter is developed, along with other less accurate but simpler techniques. Lancaster and Salkauskas [32] propose some moving least-squares methods, both noninterpolating and interpolating, for unstructured grids; though they are explained for 2D cases, the results apply for any number of variables. Barnhill and Stead [33] interpolate arbitrarily placed data to a rectilinear grid by means of Boolean sums, then define local tensor products, and finally evaluate them on the rectilinear-grid points; a 3D example illustrates the performance of this method. A similar three-staged interpolant was developed by Foley [34] by means of operators that first make an interpolation to the scattered data to build a rectangular grid of function values which are further fitted to a hypersurface by other two operators. A fitting of a surface is provided as an example. A good survey of techniques for the interpolation of scattered data in 3D and beyond, was done by Alfeld [35]. Other algorithms for multivariate interpolation on unstructured grids may be found in [36], and by use of delta iterations, in [37]. Schumaker [38] exposes a survey of a variety of numerical methods for bivariate scattered-point interpolations, while Akima [39] also worked out the particular case of estimating partial derivatives for that same case.

In this study, we will take advantage of the smoothing capabilities of least-squares (mainly, linear and n-linear) techniques and weighted averaging; as our goal is intended to smooth out noisy functions given on unstructured grids. Functions of this kind have been very used recently, for example, in numerical models to solve the quantum hydrodynamic equations in the Lagrange picture [40-44] (where grids flow and deform as the wave packet evolves). We have refined the averaging stage of our filter by using a Shepard-like algorithm, that was initially intended to be used in nonstructured grids [45-48], and where the kernels depend inversely on the distance of each of the points that participate in the averaging, to the point where the smoothing is performed. Section 2 of this paper explains the details of the method (local least squares + weighted average), Section 3 discusses the results obtained in 1D, 2D, and 5D test examples, and Section 4 provides conclusions.

2. METHODOLOGY

The one-dimensional case will be explained first so as to facilitate understanding of the extension to the two-dimensional case. A nonstructured grid is assumed to be used in both the one- and two-dimensional cases. Let n denote the total number of grid points in 1D. The idea of the method that is going to be detailed in this section is, for a given target point, to average, at that point, the various values obtained by least-square fitting the prescribed function in each of the possible stencils of, say, k consecutive grid points (where k < n) that include the point where we are performing the calculation. This would be a weighted average, in the sense that the value obtained from a given least-squares fitting will be weighted according to the distance from the point to the middle of the stencil (defined as the average between the coordinate of the first and the last point of that stencil), since it is reasonable to expect that a fitting where the test point is in the middle of the stencil is better (therefore, it should carry a larger weight) than another fitting where that same point were near the boundaries of it. Therefore, what we have is a two-stage filter, which consists of a least-squares fitting and a weighted averaging in the way of Shepard interpolation [45–48], as will be shown later.

The steps to take are as follows. First, we start with the first set of k points of the n-point grid. We perform a least-squares fitting of the prescribed test function, to a given basis set of b functions (say $\phi_1, \phi_2, \ldots, \phi_b$) for those k points. As a result, we obtain a fitted value $g_k = g(x_k)$ for each of the first k grid points, according to the usual least-squares expression [49]

$$g_k = \Phi^T(x_k) (B^T VB)^{-1} B^T Vf,$$

where

$$\Phi^T = [\phi_1, \phi_2, \ldots, \phi_b].$$
are the values of the test, or input, function, at the \( x_1 \ldots x_k \) grid points. \( V \) is a "weight" matrix that depends on the standard deviation of the error of the function value at each point. In our test cases in Section 3, we have assumed that the error is randomly distributed between all grid points, which would correspond to taking \( V \) as the identity matrix. Another criterion might be to "assign" an error, to each point, that would be inversely proportional to the distance of each of them to the point where the fitting is being done, by means, for example, of a Lorentzian or Gaussian distribution that were peaked on the fitting point.

Obviously, there is no other combination of \( k \) consecutive points that include the first grid point; therefore, we will take as the final fitted value of the function at that point (which we will call \( F_1 \)) the value we have obtained from this operation \( (g_1) \).

Using the first least-squares fit, for the second point we have obtained the value \( g_2 \). But we could have obtained a different result for \( g_2 \) if rather than using the first \( k \) points as a stencil, we would have used the next set of \( k \) points: \( x_2, x_3, \ldots, x_{k+1} \). Therefore, in order to perform additional smoothing, we will average both results, giving them either the same or a different weight. In particular, in this study we have used a Gaussian kernel that depends on the distance from the point where we are performing the calculations, to the middle of the current stencil (the average between the coordinate of the first and the last point of that stencil). The criterion we have used to set up the value of the standard deviation of the Gaussian is, after placing the center of it in the middle of the stencil, to make the value of the Gaussian equal to \( 1/k \) at the edges of the stencil. The value \( 1/2k \) may be a bit better in some cases, but the results never differ much. Therefore, defining the kernel function

\[
w(x; x_l, x_{r=1+k-1}) = e^{-\left(\frac{(x-(x_l+x_r)/2)^2}{2\sigma(x;l,k)^2}\right)}
\]

where \( x_l \) and \( x_r \) are the coordinates of the first and last grid point of the stencil, the criterion we use would be \( w(x; x_l, x_{r=1+k-1}) = 1/k \), which yields, for the smoothing length

\[
\sigma(x; l, k) = \frac{|x_l - x_{l+k-1}|}{2\sqrt{2\log k}}.
\]

Therefore, the final result for the second point would be the kernel-weighted average

\[
F_2 = \frac{w(x_2; x_1, x_k)g_2^{(1)} + w(x_2; x_2, x_{k+1})g_2^{(2)}}{w(x_2; x_1, x_k) + w(x_2; x_2, x_{k+1})},
\]

where \( g_2^{(1)} \) is the value obtained for fitting the second point of the grid, according to expression (1), taking the first \( k \) points as a stencil, while for \( g_2^{(2)} \) the stencil considered includes the points \( x_2, x_3, \ldots, x_{k+1} \), and the denominator is just the normalization factor of the two weights. (If the grid is not very distorted, the term that is given in (5) as the center of the Gaussian, \((x_1 + x_r)/2\), may be a good choice. Otherwise, it might be more convenient to minimize the functional \( \sum (x_i - \bar{x})^2 \) where the sum is performed through all the points \( x_i \) of the stencil; \( \bar{x} \) would be the center of the Gaussian and its value would be the average of all the \( x_i \).) This resembles the weighted averaging used in the Shepard interpolation technique [45–48], where Lorentzian or other similar centered-decaying weight functions are used. Sometimes, a cut-off is also included so as to avoid contributions from far distant regions; this may result in some discontinuities in the
fitted function \[46\] and for that reason, no cut-off will be employed here. This is the motivation that led us to use Gaussians as weight functions since they decay more rapidly than, for example, Lorentzian functions, when distances to the fitting point are large.

For the third grid point, the procedure is similar to that followed for the second, but in this case, we have three different stencils that include the third point, at our disposal: the first two are those that were used for the second point, and the third one would be formed by the points \(x_3, x_4, \ldots, x_{k+2}\). In general, for the \(i^{th}\) point, the value \(F_i\) of the fitting would be

\[
F_i = \frac{\sum_{j=\max(i-k+1,1)}^{\min(i, n-k+1)} w(x_i; x_j, x_j+k-1) g^{(j)}_i}{\sum_{j=\max(i-k+1,1)}^{\min(i, n-k+1)} w(x_i; x_j, x_j+k-1)},
\]

where \(g^{(j)}_i\) would be calculated according to expressions (1)–(4): calling \(H\) the product of matrices whose inverse is calculated in expression (1) and whose elements are

\[
H(\alpha, \beta) = \sum_{\gamma, \delta=1}^k B(j)\gamma_\alpha V(j)\gamma_\beta B(j)_{\delta\beta} \quad (\alpha, \beta = 1, \ldots, b),
\]

where \(B(j)\) is the generalization of (1) for any stencil whose points are \(x_j, \ldots, x_{j+k-1}\),

\[
B(j) = \begin{bmatrix}
\phi_1(x_j) & \phi_2(x_j) & \cdots & \phi_b(x_j) \\
\vdots & & & \\
\phi_1(x_{j+k-1}) & \phi_2(x_{j+k-1}) & \cdots & \phi_b(x_{j+k-1})
\end{bmatrix},
\]

and \(V(j)\) the standard deviation error matrix for that same stencil, \(g^{(j)}_i\) would become

\[
g^{(j)}_i = \sum_{\xi, \eta=1}^b \sum_{\gamma, \delta=1}^k \phi_\xi(x_i) (H(j)^{-1})_{\xi\gamma} B(j)_{\gamma\beta} V(j)_{\beta\delta} f(x_{\beta+\delta-1}).
\]

For the two-dimensional case, the extension is as follows: we call \(k_1\) the number of points of the stencil along the first dimension, and \(k_2\) along the second one. (At the end of this section, we will discuss the case when this tensorially decomposed indexing is not possible.) The whole grid will be comprised of \(n_1 \times n_2\) points. We call the \(x\) coordinate \(x_1\), and the \(y\) coordinate \(x_2\), for a straightforward extension to the multidimensional case. Now, the task is to find all possible \(k_1 \times k_2\)-point stencils that include the target point where we perform the fitting. If the number of stencils is too large, we may also restrict ourselves, if we want, to those where the point is not too far from the middle. However, here the general method that includes all possible stencils will be shown. For each stencil, we apply the same least-square method that was done for 1D, but using a basis set in \(x_1\) and \(x_2\) coordinates. Matrix \(B\) in expression (3) will include all \(k_1 \times k_2\) points of the stencil, as well as \(f^T\) in (4). The results obtained this way, for each stencil, will be weight-averaged according to an analogous expression to (7); for the \((i_1, i_2)\)th point of the 2D grid (where the labels \(i_1 = 1, \ldots, n_1\) and \(i_2 = 1, \ldots, n_2\),

\[
F_{i_1, i_2} = \frac{\sum_{j_1=\max(i_1-k_1+1,1)}^{\min(i_1, n_1-k_1+1)} \sum_{j_2=\max(i_2-k_2+1,1)}^{\min(i_2, n_2-k_2+1)} w(x_{i_1j_1}, x_{i_1j_1+k_1-1}) \sum_{j_2=\max(i_2-k_2+1,1)}^{\min(i_2, n_2-k_2+1)} w(x_{i_2j_2}, x_{i_2j_2+k_2-1}) g^{(j_1j_2)}_{i_1i_2}}{\sum_{j_1=\max(i_1-k_1+1,1)}^{\min(i_1, n_1-k_1+1)} \sum_{j_2=\max(i_2-k_2+1,1)}^{\min(i_2, n_2-k_2+1)} w(x_{i_1j_1}, x_{i_1j_1+k_1-1}) \sum_{j_2=\max(i_2-k_2+1,1)}^{\min(i_2, n_2-k_2+1)} w(x_{i_2j_2}, x_{i_2j_2+k_2-1}) g^{(j_1j_2)}_{i_1i_2}},
\]
\[ g_{(j_1j_2)}(x_{1_{j_1j_2}}, x_{2_{j_1j_2}}) = \sum_{i=1}^{b} \phi_{\xi}(x_{1_{i_{j_1j_2}}}, x_{2_{i_{j_1j_2}}}) \begin{pmatrix} H(j_1j_2)^{-1} \end{pmatrix}_{\xi_{\eta}} \sum_{\gamma_1=1}^{k_1} \sum_{\gamma_2=1}^{k_2} B(j_1j_2)_{\gamma_1\gamma_2} \]

where

\[ B(j_1j_2)_{\gamma_1\gamma_2} = \phi_{\eta}(x_{1_{j_1+\gamma_1-1}}, x_{2_{j_2+\gamma_2-1}}) \]

\[ V(j_1j_2)_{\gamma_1\gamma_2} = \frac{\phi_{\eta}(x_{1_{j_1+\gamma_1-1}}, x_{2_{j_2+\gamma_2-1}})}{\sum_{\delta_1=1}^{k_1} \sum_{\delta_2=1}^{k_2}} \]

\[ H(j_1j_2)_{\alpha\beta} = \sum_{\lambda_1=1}^{k_1} \sum_{\lambda_2=1}^{k_2} B(j_1j_2)_{\lambda_1\lambda_2} \phi_{\eta}(x_{1_{j_1+\gamma_1-1}}, x_{2_{j_2+\gamma_2-1}}) \]

In an analogous way, the algorithm may be extended to \( m \) dimensions as follows:

\[ F_{(j_1 \ldots j_m)} = \frac{\min_{i_1 \ldots i_m-k+1} \sum_{j_1=\max(i_1-k+1,1)}^{\min(i_1 \ldots i_m-k_m+1)} w(x_{1_{i_1}}, x_{1_{j_1}}, x_{j_1+k-1}) \ldots \sum_{j_m=\max(i_m-k_m+1,1)}^{\min(i_1 \ldots i_m-k_m+1)} w(x_{i_m}, x_{j_m}, x_{j_m+k_m-1})}{\prod_{l=1}^{m} \sum_{j_l=\max(i_l-k_l+1,1)}^{\min(i_l \ldots i_m-k_l+1)} w(x_{l_{i_l}}, x_{l_{j_l}}, x_{l_{j_l+k_l-1}})} \]

where

\[ g_{(j_1 \ldots j_m)} = \sum_{i=1}^{b} \phi_{\xi}(x_{1_{i_1}}, \ldots, x_{i_{j_1}}, \ldots, x_{i_{j_m}-1}) \begin{pmatrix} H(j_1 \ldots j_m)^{-1} \end{pmatrix}_{\xi_{\eta}} \sum_{\gamma_1=1}^{k_1} \ldots \sum_{\gamma_m=1}^{k_m} B(j_1 \ldots j_m)_{\gamma_1 \ldots \gamma_m} \]

\[ B(j_1 \ldots j_m)_{\gamma_1 \ldots \gamma_m} = \phi_{\eta}(x_{1_{j_1}}, \ldots, x_{j_1+k_1-1}) \ldots \phi_{\eta}(x_{i_{j_m}}, \ldots, x_{j_m+k_m-1}) \]

\[ V(j_1 \ldots j_m)_{\gamma_1 \ldots \gamma_m} = \frac{\phi_{\eta}(x_{1_{j_1}}, \ldots, x_{j_1+k_1-1}) \ldots \phi_{\eta}(x_{i_{j_m}}, \ldots, x_{j_m+k_m-1})}{\sum_{\delta_1=1}^{k_1} \ldots \sum_{\delta_m=1}^{k_m}} \]

\[ H(j_1 \ldots j_m)_{\alpha\beta} = \sum_{\lambda_1=1}^{k_1} \ldots \sum_{\lambda_m=1}^{k_m} B(j_1 \ldots j_m)_{\lambda_1 \ldots \lambda_m} \phi_{\eta}(x_{1_{j_1}}, \ldots, x_{j_1+k_1-1}) \ldots \phi_{\eta}(x_{i_{j_m}}, \ldots, x_{j_m+k_m-1}) \]

When this tensorially-decomposed indexing is not possible, for each grid point we have to locate its \( k - 1 \) closest points: this will define the first stencil. Then, we repeat that same operation with each of its \( n \) closest points (for any \( n < k - 1 \)): this will provide us the rest of the stencils (\( n \)), so we will have \( n + 1 \) stencils at our disposal, and therefore, \( n + 1 \) least-square fittings that will be weight-averaged to provide us the smoothed value for that target point. If some stencils do not include it, they should be discarded. Let us still call \( n + 1 \) the total number of stencils considered. The grid-point indexing will be denoted by one single label, \( i \), and the weights of the
A Two-Stage Filter

various \( n + 1 \) stencils, by \( w(x_i; \sum_{j \in \text{stencil}_j} x_i/k) \), where \( x_i \) denotes the \( l \)th coordinate of point \( x_i \) and \( \sum_{j \in \text{stencil}_j} x_i/k \) would be the estimated average of the \( l \)th coordinate along stencil \( j \) (the sum, being performed to all the \( k \) points of the stencil, for all \( j = 1, \ldots, n+1 \)) and it will be the center of the Gaussian weight of that coordinate, \( z_l \), while according to the criterion in expression (6), its width \( \sigma \) would be \( |x_f - z_l|/\sqrt{2\log k^{1/m}} \), where \( m \) is the dimension and \( x_f \), the furthest, or one of the furthest, \( x_i \) coordinates with respect to the \( z_l \) value of that stencil. Therefore, being \( g_i^{(j)} \) the least-square fitting at the target point \( i \) for stencil \( j \), the final smoothed value would be given by

\[
F_i = \frac{\sum_{j=1}^{k+1} \prod_{l=1}^{m} w \left( x_i; \sum_{\text{stencil}_j} x_l/n \right) g_i^{(j)}}{\prod_{l=1}^{m} w \left( x_i; \sum_{\text{stencil}_j} x_l/n \right)}.
\] (20)

3. DISCUSSION OF RESULTS

We first implemented this algorithm for the one-dimensional case, for a Gaussian distorted by noisy fluctuations. We have generated structured noise by a sine-like function, so that the final test function we used was

\[
f(x) = e^{-(x-x_0)^2/2\sigma^2} \left[ 1 + \alpha \cos(\beta x) \right] = f_{\text{smooth}}(x) + f_{\text{noise}}(x),
\] (21)

where \( x_0 \) and \( \sigma \) are the parameters for the Gaussian, \( \alpha \) is a gauge of the amplitude of the noise and \( \beta \) is its frequency. The values used in our example have been \( x_0 = 0, \sigma^2 = 1.2, \) and \( \beta = 33. \)

The grid employed has an irregular mesh. Though this noise has just one frequency, white noise (random frequencies for each grid point as well as random amplitudes, both following a Gaussian distribution error peaked around the values used in expression (21)) has also been tested, yielding very similar accuracies.

Figure 1a shows the input function (\( \alpha = 0.1 \)) along with the smoothed output function and its derivative. In order to calibrate our expectation about how the analytical derivative should be if

![](image)

(a) Input function (curve with ‘+’) and smoothed output function (curve with ‘x’), its derivative (curve with ‘*’) and analytical derivative (curve with ‘-’) of the input function without noise. Grid points below. Noise parameter \( \alpha = 0.1 \). Derivatives are shifted.

Figure 1.
the function were not affected by noise, we have plotted it so as to compare it with the derivative of the fitted (output) function.

In this example, we have used a grid of 99 nonequally spaced points and stencils of 15 points. As expected, the smoothing of the function is better than that of the derivative. The latter is not properly a smoothed fitting but the analytical derivative of the resulting fitted function itself. A much more noisy test function ($\alpha = 0.4$) has also been tried (Figure 1b); the fitted function was still very smooth though the quality of the derivative was worse. The linear basis set used included just 1 and $x$. It is possible to extend it, though for this particular case we have already seen a very good performance. However, this will always depend on the nature of the noise and the function to fit. In this case, adding $x^2$ as a basis function actually worsens the results in the sense of smoothness (but not of fitting, which gets better), which is recovered by using larger stencils (again, at the expense of the quality of the interpolation). In addition, a larger basis set requires more computational time. The optimal choice of the size of the stencil will depend on the nature of the function and the noise, but in general, large sizes provide, sometimes, too much smoothing (and a poorer interpolation) though for that same reason it may be better for evaluating derivatives. Smaller sizes yield better interpolations, but worse derivatives; they are also suitable for dealing with high frequency noise, since this consists in a random fluctuation on a short space interval. It is also possible to evaluate second derivatives, as we will show in a 2D example, but for that, rather than evaluating them directly from the fitted function, it is much better to evaluate the first derivative twice.

In order to quantify the error, we have used the following estimators:

\[
\text{absol. err} = \frac{1}{n} \sum_{j=1}^{n} |f_{\text{fit}}(x_j) - f_{\text{smooth}}(x_j)|, \tag{22}
\]

\[
\text{relat. err} = \frac{1}{n} \sum_{j=1}^{n} \left| \frac{f_{\text{fit}}(x_j) - f_{\text{smooth}}(x_j)}{f_{\text{fit}}(x_j)} \right|, \tag{23}
\]

where $f_{\text{smooth}}$ corresponds to the smooth test function without noise. The first estimator is the absolute error and the second, the relative one. For the latter, the straight line on top of the
summation symbol means that points where the value of the function is very close to zero have to be excluded, skipping over, thus, singularities and quasisingularities that would make the relative error be artificially large. The threshold value should be much smaller than the range of variation (within the grid used) of the test function. For the 2D test case, a value of 1% of the range of variation (which is around 0.175 for the function and its first and second derivative) has been chosen, while for this 1D test case (whose range of variation is around 0.4), we have simply omitted the left and right tail points, as well as the central point for the evaluation of the relative error of the derivative, since \( f_{\text{fit}} \) there is very close to zero.

For the 2D case, we have employed a 99 by 99-point unstructured grid, for the following test function:

\[
f(x_1, x_2) = e^{-\frac{1}{2} \sum_{i,j=1}^{2} (x_i-\bar{x}_i)(x_j-\bar{x}_j)} \left[ 1 + \alpha_1 \cos(\beta_1 x_1) + \alpha_2 \cos(\beta_2 x_2) \right],
\]

(24)

where \((x^{-1})_{ij}\) is the inverse of the covariance matrix of the 2D Gaussian and \(\alpha_1, \beta_1\) and \(\alpha_2, \beta_2\) are the noise parameters for the \(x_1\) and \(x_2\) coordinates (\(x_1 = x_2 = 0\), \(x_{11} = x_{22} = 1.2\), \(x_{12} = 0.5\), \(\alpha_1 = \alpha_2 = 0.1\), \(\beta_1 = 33\), and \(\beta_2 = 30\). The basis functions \(1, x, y, \text{ and } xy\) were used.

Figure 2 shows a contour plot of function (24) as well as the fitting, while Figure 3 compares the fitting and the function (24) without noise (\(\alpha_1 = \alpha_2 = 0\)); the stencils used comprised 15 \(\times\) 15 points. If we were to examine 1D slices of Figure 2, we would see that most of the fitted curves fall well below the 1D slices, rendering an apparently very poor fitting though the smoothing visually would look good; the reason for this is that the 2D smoothing process is not a direct product of two 1D problems, but a full algorithm (both because of the basis set used and the fact that the stencils employed are two dimensional). Note also, the cross term of the test function. As a result of that, a comparison has to be made for the full 2D function, since the contribution from the other dimension may lower the resulting fitted 1D slices. That full 2D comparison is made in Figure 2 and there it is seen that they match quite well. We also see that the quality is similar to the 1D test performed in Figure 1a.

Figure 4 shows the various fitted values obtained for a fixed target when the fitting is done at each of the 15 \(\times\) 15 stencils that contains it, along with the weights corresponding to each of them. We see that the values at the edges (those obtained from stencils where the target point is at some edge or nearby) are very different from those obtained elsewhere. However, we also see that their corresponding weights are very small, therefore, giving an almost negligible contribution.
Figure 3. Contour plot of the smoothed output function of Figure 2 (solid line) and the 2D Gaussian of that same figure without noise (dashed line).

Figure 4. Output fitted values obtained for a fixed target point, in each of the stencils (bottom surface), the final weighted average (plane), and weights corresponding to each stencil (dotted surface -shifted and stretched-).

to the final averaged fitted value. This fact may be used to just skip over calculations in those stencils, thus, speeding up the numerical process (something especially useful for the multidimensional case) and ignoring ill-posed values as it is seen in the plot.

Figures 5a and 5b show a front view of a 3D plot of the x-derivative of the fitting of the test function and the analytical derivative of the test function without noise, for two different stencils: the former with $10 \times 10$ points and the latter with $15 \times 15$. Here, we see that big sizes yield smoother results and worse interpolations also for the derivatives, while small sizes work the opposite way. Figure 6 shows a contour plot of the results in Figure 5b and Figure 7 shows the comparison between the second x-derivatives (the analytical and the smoothed one, the former being referred to the test function without noise). In this case, as mentioned before, we performed the first derivative of the smoothed fitting and then we used the result as the input of a new fitting whose derivative is the second derivative of the original function. Again, we see that the agreement is good, especially taking into account the ill-posed nature of the original function. And actually, though the interpolation is not very good (though the smoothness of the results is),
Figure 5. $x$-derivative of the smoothed output function of Figure 2 (dark lines) and analytical derivative of the 2D Gaussian of that same figure (grey lines). Front 3D view.

Figure 6. Contour plot of Figure 5b. Thin line is the smoothed output, and thick line, the analytical derivative.
this comparison cannot be considered as more than an estimation of what the true value should be, since when the test function is affected by noise, many functions might be possible candidates for being the original, noiseless function. We do not know whether the test function is actually a pure Gaussian or a sort of 'distorted' Gaussian-like function. Because of this, expressions (22) and (23)—and their extension to 2D—are just vague estimators that yield an approximate gauge concerning the accuracy of the smoothing.

The natural extension of the absolute error estimator (22) to the 2D case would be

\[
\text{absol. err} = \frac{1}{n_1 n_2} \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \left| f_{\text{fix}} \left( x_{1i_1}, x_{2i_2} \right) - f_{\text{smooth}} \left( x_{1i_1}, x_{2i_2} \right) \right| ,
\]

and similarly for the relative error estimator (23). Finally, we have included a contour plot of a 2D slice of a 5D Gaussian function (with cross terms) distorted by noise, and its smoothed fitting (Figure 8). The grid was comprised of 19 points along each direction and the stencils,
by 3. Due to the smallness of these numbers (larger would have been impractical due to storage requirements), accuracy is not as good as for the 2D test, yet we can see that the smoothing is still good. Also, the configuration of the grid points does not affect substantially the precision of the smoothness; actually, a slightly better accuracy is observed when considering a random scattered-point grid than a tensorial product of five (equal or different) 1D irregular grids (for Figure 8, the latter option was used).

Sometimes we may be aware of some discontinuity or highly oscillating structure of the test function near a localized region of the grid. In that case, we can use more basis functions only for the grid points of that region, or shorter stencils or a combination of both. This would yield a better interpolation in that region, avoiding too much smoothing that might ignore a real, not noisy structure. In other cases, especially when noise is small, it may be useful to average the result of the smoothing with a standard interpolating technique, giving different weights to each of the two outputs according to the magnitude of the noise. CPU times of computation for the 1D case has been of the order of 0.05 seconds, while for the 2D test it has been around a second, for a 99-point grid in each of the dimensions. Sometimes we may be aware of some discontinuity or highly oscillating structure of the test function near a localized region of the grid. In that case, we can use more basis functions only for the grid points of that region, or shorter stencils or a combination of both. This would yield a better interpolation in that region, avoiding too much smoothing that might ignore a real, not noisy structure. In other cases, especially when noise is small, it may be useful to average the result of the smoothing with a standard interpolating technique, giving different weights to each of the two outputs according to the magnitude of the noise. CPU times of computation for the 1D case has been of the order of 0.05 seconds, while for the 2D test it has been around a second, for a 99-point grid in each of the dimensions, and 2.7 seconds for the 2D slice of the 5D case, where 19 points were used for every of the five dimensions of the grid, and three for each of the dimensions of the stencils.

4. CONCLUSIONS

A new and numerically fast technique for smoothing noisy functions on nonstructured grids, taking advantage of the smoothing properties of least-squares fitting and weighted averaging in the Shepard sense, has been successfully developed. It may be applied to any number of dimensions and does not require direct product grids. The results obtained for three test problems, in 1, 2, and 5 dimensions certify the quality of this method when its results are compared with a test function where random noise was added. A direct first-derivative evaluation of the resulting fitted function has also proved to be successful, as well as the second derivative, obtained by the same procedure performed upon the first derivative of the smoothed function.

REFERENCES