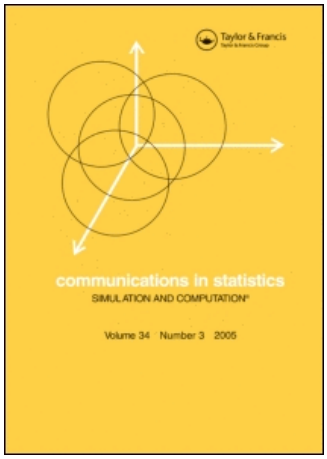

Communications in Statistics - Simulation and Computation
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713597237

Fitting noisy data using cross-validated cubic smoothing splines

S. B. Pope [a]; R. Gadh [b]
[a] Sibley School of Mechanical & Aerospace Engineering, Cornell University, Ithaca, NY
[b] Mechanical Engineering Department, Carnegie Mellon University, Pittsburgh, PA

Online Publication Date: 01 January 1988
To cite this Article: Pope, S. B. and Gadh, R. (1988) 'Fitting noisy data using cross-validated cubic smoothing splines', Communications in Statistics - Simulation and Computation, 17:2, 349 - 376
To link to this article: DOI: 10.1080/03610918808812668
URL: http://dx.doi.org/10.1080/03610918808812668

COMMUN. STATIST. -SIMULA., 17(2), 349-376 (1988)

FITTING NOISY DATA USING CROSS-VALIDATED CUBIC SMOOTHING SPLINES

S.B. Pope

Sibley School of Mechanical
& Aerospace Engineering
Cornell University
Ithaca, NY 14853

R. Gadh

Mechanical Engineering Department
Carnegie Mellon University
Pittsburgh, PA 15213

ABSTRACT

An algorithm is described for approximating an unknown function f(x), given many function values containing random noise. The approximation constructed is a cubic spline g(x) with sufficient basis functions to represent f(x) accurately. The basis-function coefficients are determined by minimizing a combination of the infidelity E (the mean-square error between g(x) and the data), and the roughness T (which is a measure of the tortuosity of g(x)). The quantity minimized is E+pT, where p is a smoothing parameter. A suitable value of p is determined by cross validation.

Results of numerical tests are reported which show that this algorithm is superior to least-squares cubic splines: in general the statistical errors are substantially less, and they are insensitive to the number of basis functions used.

INTRODUCTION

We address the familiar problem of approximating an unknown function f(x) in an interval [0,L] given samples of the function values containing random noise. The N sample values f_i are at the locations x_i (i = 1,2,...., N), and can be written

$$f_i = f(x_i) + \eta_i \,. \tag{1}$$

Here η_i are the (unknown) random errors that are assumed to be independent and to have mean zero

$$\langle \eta_i \rangle = 0 \,. \tag{2}$$

The variances

$$\langle \eta_i \eta_i \rangle = \sigma^2(x_i) \,, \tag{3}$$

may or may not be known.

The general problem described can take on different appearances depending on the number of samples N, the nature of the function f, and the magnitude of the random error σ. Here we are mainly concerned with a simple smooth function f, and with *dense data*. By this we mean that the number of samples N is very large (10,000 say), and that the number density of sample points is nowhere small. But at the same time the error associated with each sample is large, perhaps of the same magnitude as f itself. Problems of this type arise in Monte Carlo methods for simulating turbulent flows (Nguyen & Pope 1984, Pope 1985, Haworth & Pope 1987a,b, Haworth 1987, Anand, Pope & Mongia 1988).

In the simplest case x the single spatial variable, and $f_i \equiv F(x_i)$ are samples of a property (e.g. velocity, temperature, composition) of a fluid particle located at x_i. (F(x) is a random function.) The Monte Carlo method provides N sample pairs (x_i, f_i) from which the expectation $f(x) \equiv \langle F(x) \rangle$ is to be deduced.

The present work, and much previous work on this problem, is based on cubic splines (see de Boor 1978, Lancaster & Salkauskas 1986). We approximate f(x) by a cubic spline g(x):

$$g(x) \equiv \sum_{k=1}^{M} a_k b_k(x) \,, \tag{4}$$

where $b_k(x)$ are fixed cubic spline basis functions, and a_k are coefficients to be determined from the data. If the number of basis functions M is chosen to equal the number of samples N, then a spline g(x) can be formed that passes through all the data points. But since the data contain random errors, this is not a good approximation to f(x).

Since the function f(x) is, by assumption, simple and smooth, it can be "accurately" represented by far fewer basis functions. Although it is imprecisely defined, it is useful to denote by M_r the minimum number of basis functions needed to represent f(x) accurately. In most cases — and certainly for dense data — M_r is much less than N. Rather than choosing M = N, we could choose $M \approx M_r$ and determine the coefficients by least squares. That is, we choose the coefficients a_k to minimize the *infidelity* (or mean square error)

$$E \equiv \sum_{i=1}^{N} [f_i - g(x_i)]^2 w_i \,, \tag{5}$$

where w_i are numerical weights ascribed to each sample. Although simple and robust, the least-squares method has the disadvantage that the optimum choice of M is not known *a priori*. If M is too small, g cannot represent f accurately; if M is too large, g tends to follow the random errors. Test results to show this are presented in the third section.

We follow Reinsch (1967) in using smoothing splines (see also, de Boor 1978; and Schoenberg 1964). In this approach, the quantity minimized is a combination of the mean-square error and the tortuosity — or lack of smoothness. The *roughness* (or tortuosity) T of the approximant g is defined by

$$T \equiv \int_0^L [g''(x)]^2 dx \,, \tag{6}$$

where primes denote differentiation with respect to x. In Reinsch's algorithm the number of basis functions M is equal to the number of samples N with the spline knots being at the data points x_i: and for a given value of a (non-negative) smoothing parameter p, the basis function coefficients a_k are chosen to minimize

$$\chi \equiv E + pT \,. \tag{7}$$

The choice p = 0 yields the spline that passes through the data (or the least-squares spline if M is chosen to be less than N). At the opposite extreme, as p tends to infinity, g(x) tends to the straight line with the minimum least-squares error. Reinsch suggests a means of selecting a value of p based on the variance $\sigma^2(x_i)$ of the error in the data.

In the present context there are two difficulties with Reinsch's algorithm. First, the result g(x) is found to be extremely sensitive to the choice of p; and a simple strategy of specifying p *a priori* is unsatisfactory. Second, the choice M = N can lead to a very large (10,000, say) system of linear equations.

The algorithm that we present and demonstrate here overcomes both of these problems. First a near-optimum value of the smoothing parameter p is obtained by employing the statistical technique of cross validation. Second, the number of basis functions is chosen to be much less than N, but larger than M_r. In contrast to the least squares method, here the choice of M is not crucial, since smoothing is effected by minimizing the roughness, not by restricting the number of degrees of freedom (M) of the approximant g(x).

In the last ten years there has been considerable work on cross-validated cubic splines. Wahba and Wold (1975) used a "leaving-out-one" cross-validation techniques to determine the smoothing parameters p for the Reinsch spline; while Craven and Wahba (1979) did the same using "generalized cross validation" (GCV). Improved algorithms to determine p using GCV have been developed by Elden (1984), Hutchinson (1985) and Woltring (1986); while extensive theoretical results have been obtained by Wahba (1985) and Li (1986). The cross validation technique used here is somewhat different, being better suited to the case of dense data. It could be termed the "leaving-out-half" method, as explained in the next section.

The use of the Reinsch spline — in which the number of basis functions is equal to the number of data points — is clearly inappropriate to dense data. Using a convenient set of basis functions — independent of the data — was suggested by Wahba (1980) and has been used subsequently by Nychka et al. (1984) and by O'Sullivan & Wahba (1985).

In the next section the algorithm is described. In the third section, for a simple test problem, the method is comprehensively tested and compared to the least-squares method. The general performance of the method, variants and extensions are discussed in the fourth section. Finally the main conclusions are drawn.

ALGORITHM

To give an overview of the algorithm: the data are first divided into two independent sets of samples. Given p, a smoothing cubic spline is determined from

each data set. The cross error Z(p) is defined to be the mean-square error between each spline and the data set that was not used in its determination. The value of the smoothing parameter used is p^* — the value that minimizes Z(p). The value of p^* is determined iteratively.

The i^{th} of the N samples has value f_i, location x_i and is ascribed a numerical weight w_i. (This may be unity, or $\sigma(x_i)^{-1}$ if $\sigma^2(x)$ is known.) These N samples are divided into two independent sets:

$$f_i^{(s)},\ x_i^{(s)},\ w_i^{(s)};\quad i = 1,2,....,N^{(s)};\quad s = 1,2;\quad N^{(1)} + N^{(2)} = N. \tag{8}$$

It is important that the two sets be statistically independent; and it is desirable that the number density of samples along the x-axis be approximately the same. If the samples are ordered in x, or if their ordering is random, the division can simply be achieved (for even N) by:

$$N^{(1)} = N^{(2)} = N/2, \tag{9}$$

$$f_i^{(s)} = f_{2(i-1)+s},\quad i = 1,2,....,N^{(s)} \tag{10}$$

The number of basis functions M is selected, and the cubic spline basis functions $b_k(x)$ are determined. (We choose equally spaced knots.) Based on the two data sets and the value of p, two splines are formed:

$$g^{(s)}(x,p) \equiv \sum_{k=1}^{M} a_k^{(s)}(p) b_k(x);\quad s = 1,2. \tag{11}$$

The basis-function coefficients $a_k^{(s)}$ are determined by minimizing

$$\chi^{(s)} \equiv E^{(s)} + pT^{(s)}, \tag{12}$$

where $E^{(s)}$ and $T^{(s)}$ are defined in an obvious way by analogy to Eqs. (5) and (6).

By standard techniques (Dahlquist et al. 1974; de Boor 1978), the solution to the minimization problem can be written in matrix form as

$$\underline{a}_{(s)}(p) = (\underline{\underline{B}}_{(s)} + p\underline{\underline{C}})^{-1}\, \underline{Y}_{(s)}, \tag{13}$$

where

$$\underline{a}_{(s)} \equiv [a_1^{(s)} a_2^{(s)} \ldots a_M^{(s)}]^T , \tag{14}$$

and the components of $\underline{\underline{B}}_{(s)}$, $\underline{\underline{C}}$ and $\underline{Y}_{(s)}$ are

$$B_{(s)jk} \equiv \sum_{i=1}^{N^{(s)}} w_i^{(s)} b_j(x_i^{(s)}) b_k(x_i^{(s)}) , \tag{15}$$

$$C_{jk} \equiv \int_0^L b_j''(x) b_k''(x) dx , \tag{16}$$

and

$$Y_{(s)k} \equiv \sum_{i=1}^{N^{(s)}} w_i^{(s)} b_k(x_i^{(s)}) f_i^{(s)} . \tag{17}$$

In Eq. (13), the matrix $\underline{\underline{B}}_{(s)} + p\underline{\underline{C}}$ is M x M, banded (with bandwidth 7), symmetric, positive-definite. Hence the equation can be solved very efficiently using the Cholesky square-root method. Note that $\underline{a}_{(s)}$ depends on p, but that $\underline{\underline{B}}_{(s)}$, $\underline{\underline{C}}$ and $\underline{Y}_{(s)}$ do not.

For a given value of p we obtain two approximants $g^{(1)}(x,p)$ and $g^{(2)}(x,p)$ to the underlying function f(x). In order to determine a near-optimum value of p we use cross validation. The cross error is defined by

$$Z(p) \equiv \sum_{i=1}^{N^{(1)}} w_i^{(1)} (f_i^{(1)} - g^{(2)}(x_i^{(1)}))^2 + \sum_{i=1}^{N^{(2)}} w_i^{(2)} (f_i^{(2)} - g^{(1)}(x_i^{(2)}))^2 . \tag{18}$$

This is the mean-square error between the sets of data (1 and 2) and the approximants based on the other data sets (2 and 1). The minimizer p^* of Z(p) is chosen as the smoothing parameter.

The determination of p^* is accomplished by an iterative algorithm based on Newton's method. In order to implement Newton's method we need to determine

the first two derivatives of Z with respect to p; this in turn requires the determination of the first two derivatives of $\underline{a}_{(s)}$.

Let $\underline{a}'_{(s)}(p)$ and $\underline{a}''_{(s)}(p)$ be the first and second derivative of $\underline{a}_{(s)}(p)$. From Eq. (13) we obtain

$$\underline{a}'_{(s)} = - (\underline{\underline{B}}_{(s)} + p\underline{\underline{C}})^{-1}\underline{\underline{C}}\underline{a}_{(s)} , \tag{19}$$

and

$$\underline{a}''_{(s)} = -2(\underline{\underline{B}}_{(s)} + p\underline{\underline{C}})^{-1}\underline{\underline{C}}\underline{a}'_{(s)} . \tag{20}$$

To simplify the subsequent equations we replace $\underline{a}_{(s)}$, $\underline{a}'_{(s)}$ and $\underline{a}''_{(s)}$ by $\underline{\alpha}_{(s)}$, $\underline{\alpha}'_{(s)}$ and $\underline{\alpha}''_{(s)}$ defined by

$$\underline{\alpha}_{(1)} = \underline{a}_{(2)}, \ \underline{\alpha}_{(2)} = \underline{a}_{(1)}, \text{ etc.} \tag{21}$$

Now Eq. (18) can be rewritten in matrix form as

$$Z(p) = \sum_{s=1}^{2} \{ \sum_{i=1}^{N^{(s)}} w_i^{(s)}[f_i^{(s)}]^2 - 2\underline{\alpha}_{(s)}^T \underline{Y}_{(s)} + \underline{\alpha}_{(s)}^T \underline{\underline{B}}_{(s)}\underline{\alpha}_{(s)} \} . \tag{22}$$

Differentiating with respect to p we obtain

$$R(p) \equiv \frac{1}{2}\frac{dZ(p)}{dp}$$

$$= \sum_{s=1}^{2} (\underline{\alpha}_{(s)}^T \underline{\underline{B}}_{(s)} - \underline{Y}_{(s)}^T)\underline{\alpha}'_{(s)} , \tag{23}$$

and differentiating again

$$\frac{dR(p)}{dp} = \sum_{s=1}^{2} (\underline{\alpha}'_{(s)})^T \underline{\underline{B}}_{(s)}\alpha'_{(s)} + (\underline{\alpha}_{(s)}^T \underline{\underline{B}}_{(s)} - \underline{Y}_{(s)}^T)\underline{\alpha}''_{(s)} . \tag{24}$$

At $p = p^*$, Z(p) is a minimum and R(p) is zero. Starting from an initial guess $p^{(0)} = 0$, Newton's method to solve the equation $R(p^*) = 0$ results in the

iteration

$$p^{(n+1)} = p^{(n)} - R(p^{(n)}) / \frac{dR(p^{(n)})}{dp} . \tag{25}$$

In summary, one Newton iteration consists of the following steps. Given $p^{(n)}$, Eqs. (13), (19) and (20) are solved for $\underline{a}_{(s)}$, $\underline{a}'_{(s)}$ and $\underline{a}''_{(s)}$. (This requires two Cholesky decompositions and six back substitutions. Note that the coefficients $\underline{\underline{B}}_{(s)}$, $\underline{\underline{C}}$ and $\underline{Y}_{(s)}$ do not need to be re-evaluated on each iteration.) Then $R(p^{(n)})$ and its derivative are evaluated from Eqs. (23) and (24). The next Newton iterate $p^{(n+1)}$ is then obtained from Eq. (25).

Several comments about the iteration are called for. First, since Newton's method is not globally convergent, it is combined with a bisection method: if the next Newton iterate $p^{(n+1)}$ lies outside the currently known range $[p_{min}, p_{max}]$ of p^*, then $p^{(n+1)}$ is replaced by $(1/2)(p_{min} + p_{max})$. Initially p_{min} is zero and p_{max} is set to (machine) infinity. As the iteration proceeds p_{min} and p_{max} are updated. Second, it is recognized that the solution of $R(p) = 0$ guarantees only a local minimum of $Z(p)$. Tests indicate that usually $Z(p)$ is a simple convex function and hence the global minimum is obtained. But in general we accept the first local minimum obtained by the iteration.

Once the smoothing parameter p^* has been obtained, the coefficients $\underline{a}_{(1)}(p^*)$ and $\underline{a}_{(2)}(p^*)$ are determined. The final result is the spline approximant Eq. (4) with the coefficients being

$$\underline{a} = \frac{1}{2}[\underline{a}_{(1)}(p^*) + \underline{a}_{(2)}(p^*)] . \tag{26}$$

NUMERICAL TESTS

We present numerical results that determine the performance of the cross-validated cubic smoothing splines for a simple test problem. The performance of the new algorithm is compared to that of least-squares cubic splines.

Test Problem

The function $f(x)$ selected for the test is

$$f(x) = \sin x , \tag{27}$$

in the interval $[0,4\pi]$. The N samples are divided into two sets of $N_{1/2} = N/2$ samples each. For the first set, the sample locations $x_i^{(1)}(i = 1,2,...,N_{1/2})$ are randomly distributed uniformly in the interval $[0,4\pi]$. For convenience, the sample locations of the second set are chosen to coincide with the first:

$$x_i^{(2)} = x_i^{(1)} . \tag{28}$$

(This choice simplifies the algorithm since then $\underline{\underline{B}}_{(1)}$ and $\underline{\underline{B}}_{(2)}$ are equal, see Eq. (15).) The samples $f_i^{(s)}$ have a uniform Gaussian error of standard derivation σ. That is

$$f_i^{(s)} = f(x_i^{(s)}) + \sigma\xi_i^{(s)} \tag{29}$$

where $\xi_i^{(s)}$ are N independent standardized Gaussian random numbers. The numerical weights $w_i^{(s)}$ are set to unity.

Mean-Square Errors

Numerical tests are performed by comparing the spline approximant g(x) to the known test function f(x). It is found that the error in the approximant is significantly greater near the boundaries (x=0 and $x=4\pi$) than it is in the central portion of the interval. The reason for this (and a means of reducing these boundary errors) is discussed in the next section. In order that the numerical tests are not unduly influenced by the boundaries, we base our measures of error on the interval $[\pi,3\pi]$.

The root-mean-square (rms) error ε_0 is defined by

$$\varepsilon_0^2 \equiv \frac{1}{2\pi} \int_{\pi}^{3\pi} \langle [f(x)-g(x)]^2 \rangle \, dx , \tag{30}$$

where angled brackets denote expected values. It should be realized that g(x) is a random function. For one realization (i.e. one set of random numbers $\xi_i^{(s)}$) we can

define the error $\tilde{\varepsilon}_0$ by

$$\tilde{\varepsilon}_0^2 \equiv \frac{1}{2\pi} \int_{\pi}^{3\pi} [f(x)-g(x)]^2 dx \ . \tag{31}$$

Then we have

$$\varepsilon_0^2 = \langle \tilde{\varepsilon}_0^2 \rangle \ . \tag{32}$$

In the results reported below, ε_0 is estimated by averaging $\tilde{\varepsilon}_0^2$ over 100 independent realizations: the resulting statistical error in ε_0 is found to be less than 10%.

In some applications — including that which motivated this study — in addition to an approximation to f(x), we require approximations to the derivatives f'(x) and f''(x). If g(x) is a good approximation to f(x), it by no means follows that g'(x) is a good approximation to f'(x). (Consider, for example, $g(x) \equiv f(x) + \psi \sin(x/\psi^2)$ for small ψ.) Hence we also examine ε_1 and ε_2, the rms errors in g'(x) and g''(x), defined by

$$\varepsilon_m^2 \equiv \frac{1}{2\pi} \int_{\pi}^{3\pi} \left\langle \left[\frac{d^m f(x)}{dx^m} - \frac{d^m g(x)}{dx^m} \right]^2 \right\rangle dx, \quad m = 1,2 \ . \tag{33}$$

As with ε_0, ε_1 and ε_2 are estimated from 100 independent realizations.

Independent Parameters

The errors ε_0, ε_1 and ε_2 depend on the values of the following parameters: M, the number of basis functions; σ, the standard deviation of each sample; and, $N_{1/2}$, the number of samples in the range $[\pi,3\pi]$. In the numerical tests, the values used were:

M: 10, 15, 20, 25, 30, 35, 40, 45, 50, 55.

σ: 1/256, 1/64, 1/16, 1/4, 1.

$N_{1/2}$: 162, 322, 642, 1282, 2562.

For the case of dense data ($N_{1/2} >> M_r$), we expect, and indeed find, that the errors ε_m depend on σ and $N_{1/2}$ only through the parameter

$$\gamma \equiv \sigma/\sqrt{N_{1/2}} \ . \tag{34}$$

This important quantity we term the *uncertainty* in the data, and its inverse γ^{-1} is the *certainty*.

To illustrate the significance of γ, we consider the quantity

$$\bar{g} \equiv \frac{1}{N_{1/2}} \sum_{i=1}^{N_{1/2}} f_i^{(1)} \ , \tag{35}$$

as an approximation to

$$\bar{g} \equiv \frac{1}{4\pi} \int_0^{4\pi} f(x)\, dx \ . \tag{36}$$

Elementary statistical calculations show that the rms statistical error in this approximation is γ. This result has two significances. First, the error depends on $N_{1/2}$ and σ only as they appear in γ: doubling $N_{1/2}$ has the same effect as decreasing σ by a factor of $\sqrt{2}$. Second, since estimating $\bar{f}$ is easier than estimating $f(x)$, the best that can be expected of the smoothing algorithm is that ε_0 is not much greater than γ.

In light of these considerations we define normalized rms errors by

$$\varepsilon_m^* \equiv \varepsilon_m/\gamma \ , \quad m = 0,1,2 \ . \tag{37}$$

Preliminary tests showed that indeed ε_m^* depend on $N_{1/2}$ and σ solely through γ. Hence the objective of the tests is to determine ε_m^* as functions of M and γ.

Results

Figure 1 shows the normalized error ε_0^* as a function of the number of basis functions M, with the uncertainty γ as a parameter. For moderate and large M

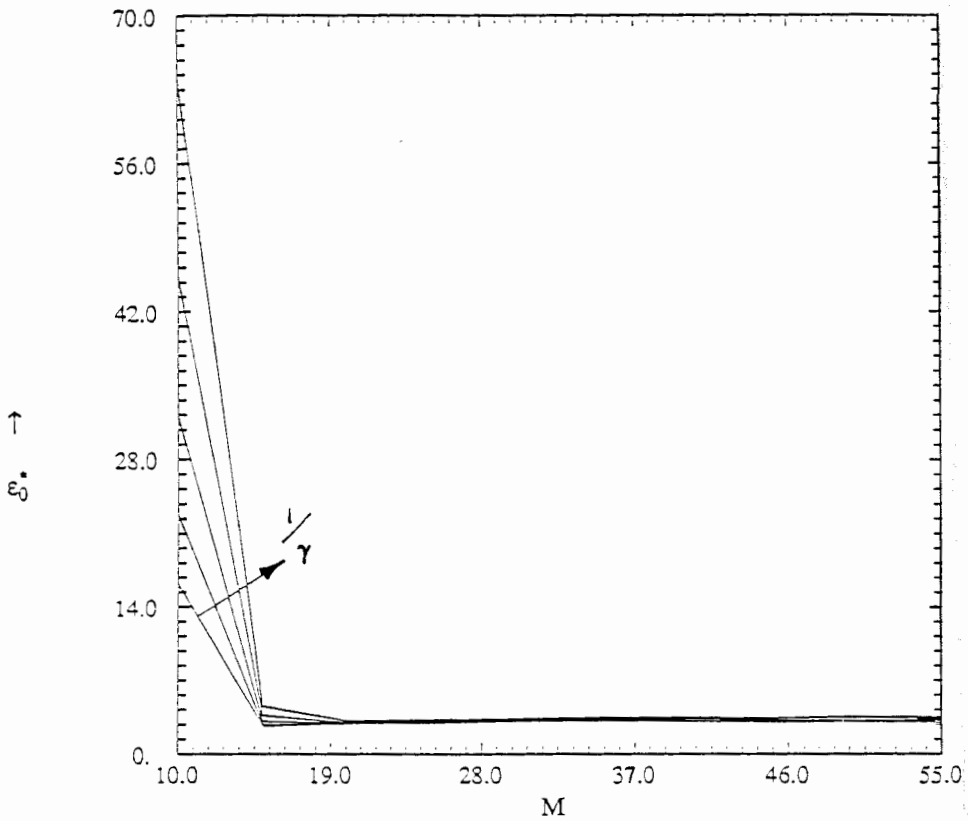

FIG. 1: Rms error in g, ε_0^*, against number of basis functions, M, for different uncertainties: γ = 3.1 x 10^4, 4.4 x 10^{-4}, 6.2 x 10^{-4}, 8.7 x 10^{-4}, 1.2 x 10^{-3}. (σ = 1/64). Cross-validated smoothing splines.

(M≥20), the error appears small ($\varepsilon_m^* \approx 3$) and independent of M and γ. But for small M (M=10 and M=15) the error is large and increases with γ^{-1}.

For small M (M=10, say) there are too few basis functions to represent the function f(x). Consequently, even if there is no random error (i.e. $\sigma = \gamma = 0$), there is a significant deterministic error. Because of this, for small γ, ε_0 is independent of γ and hence ε_0^* varies as γ^{-1}.

As M increases, the deterministic error decreases rapidly. At what stage it becomes negligible depends on the magnitude of the random error. Figure 1

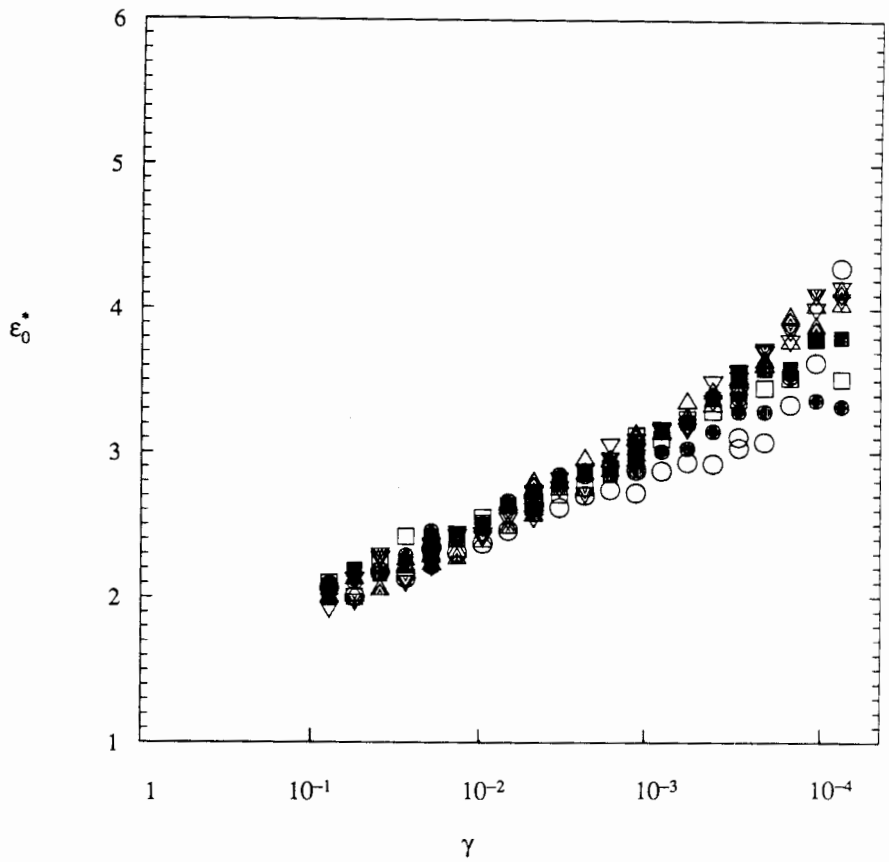

FIG. 2: Rms error ε_0^* as a function of the uncertainty γ for different numbers of basis functions. Symbols: ○, M = 20; ●, M = 25; □, M = 30; ■, M = 35; △, M = 40; ▲, M = 45; ▽, M = 50; ▼, M = 55. Cross-validated smoothing splines.

suggests that with 20 basis functions the deterministic error is negligible. The results to follow confirm this conclusion provided γ is greater than 10^{-3}.

To study more closely the error ε_0^* in cases where the deterministic error is small, in Fig. 2 we show ε_0^* against γ for different values of $M \geq 20$. For not too small values of γ ($\gamma > 10^{-3}$, say), ε_0^* is in the range 2-3 and increases weakly with the certainty γ^{-1}. Most importantly, the error depends little on M: thus, the choice of the number of basis functions is not crucial (provided only that $M \geq M_r \approx 20$).

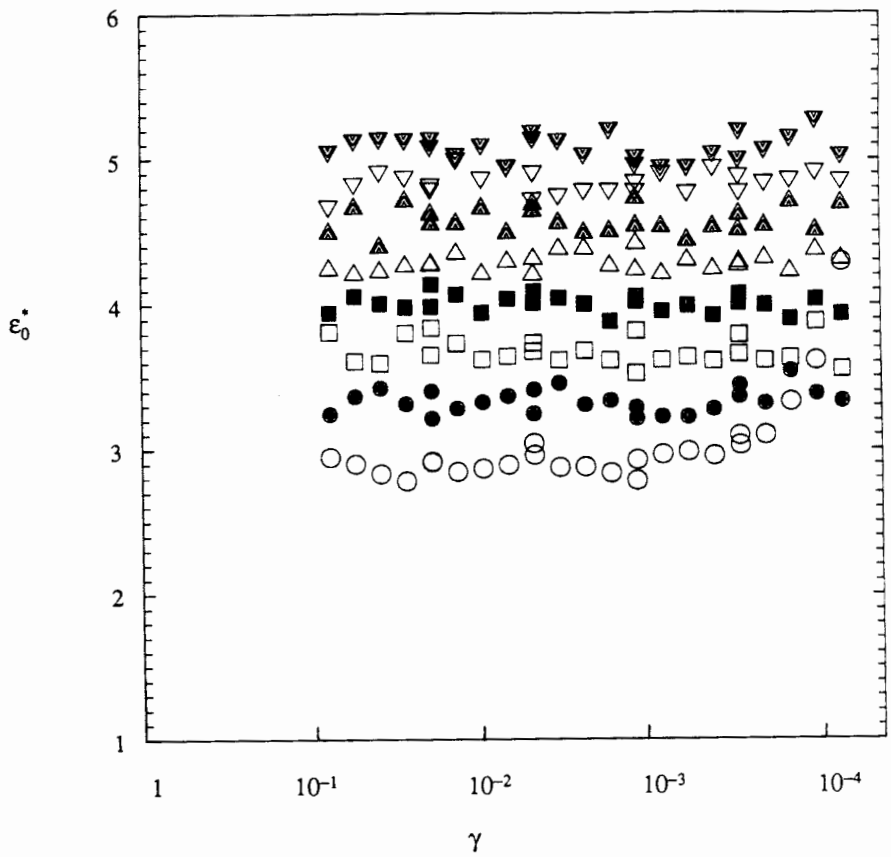

FIG. 3: Same as Fig. 2. Least-squares cubic splines.

For very small uncertainties ($\gamma < 10^{-3}$) there is more variation of ε_0^* with M. Note that with the smallest number of basis functions (M=20) the error begins to increase rapidly, indicating that the deterministic error is no longer negligible.

Figure 3 is the same plot as Fig. 2, but for least-squares splines. The conclusions to be drawn are quite different. The errors ε_0^* are generally larger (in the range 3-5); they are essentially independent of γ, and, most importantly, they increase with M. For least squares, then, the choice of M is crucial.

Figures 4 and 5 show similar plots for ε_1^*, and Figs. 6 and 7 show ε_2^*.

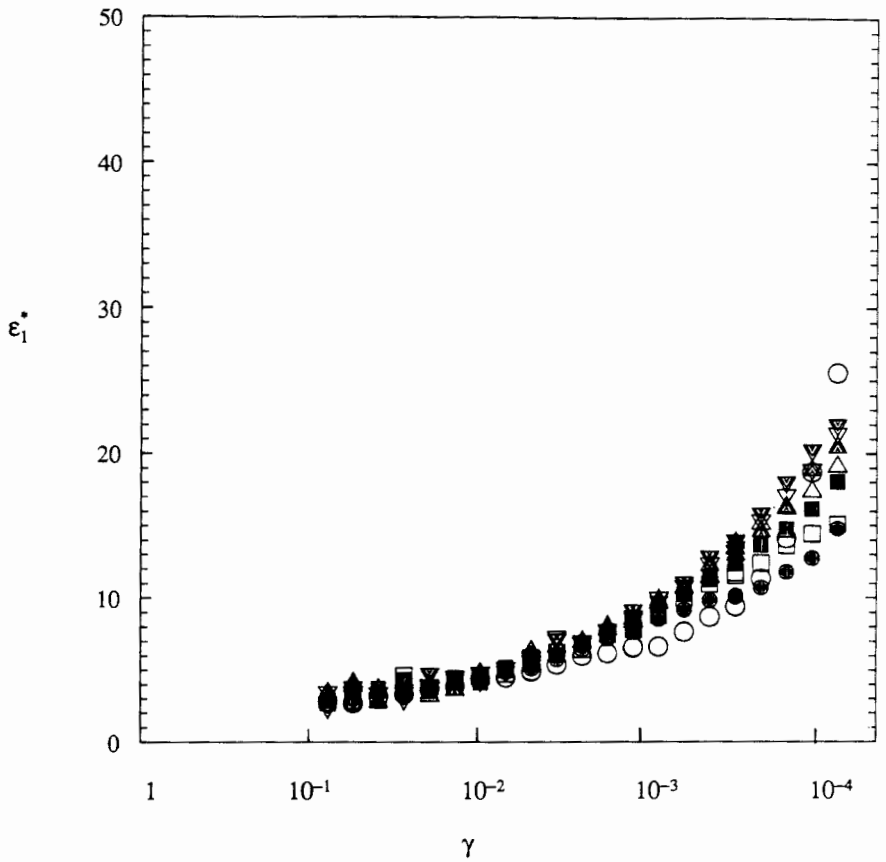

FIG. 4: Rms error in first derivative g', ε_1^*, as a function of the uncertainty γ. Symbols same as Fig. 2. Cross-validated smoothing splines.

Qualitatively the behavior is the same to that of ε_0^*, although the magnitudes of the errors can be much larger.

To emphasize one virtue of the new algorithm, Figs. 8 and 9 show ε_2^* against M for cross-validated smoothing splines and for least-squares splines. For least-squares (Fig. 9) the error increases rapidly with M: with M = 55, the error is 16 times that with M = 20. With cross validation, on the other hand, the error grows slowly with M: with M = 55 it is less than twice that with M = 20.

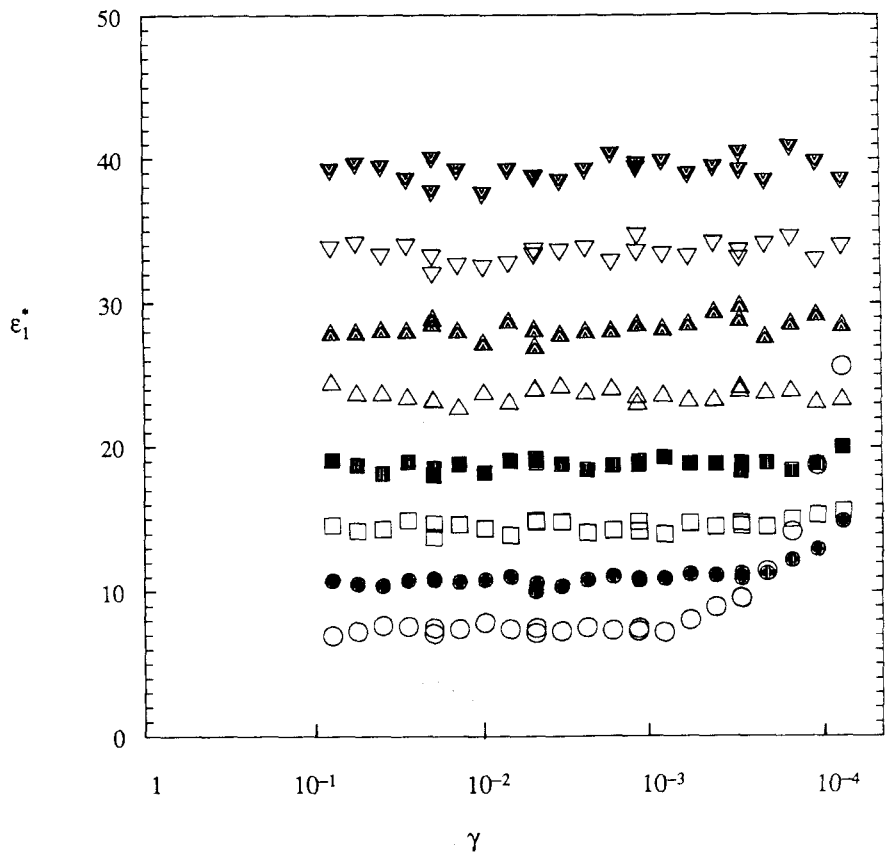

FIG 5: Same as Fig. 4. Least-squares cubic splines.

It is evident that with least squares it is crucial to use a near-optimum number of basis functions. In nearly all applications this number is not known a priori, if at all. Nevertheless, we ask the question: With the optimum number of basis functions, how do the errors ε_m^* compare for cross validation and least squares? We define $\tilde{\varepsilon}_m(\gamma)$ to be the minimum of $\varepsilon_m^*(M,\gamma)$ over all the values of M used. Figure 10 shows $\tilde{\varepsilon}_m$ against γ. It may be seen that, for γ greater than 10^{-3}, the error using cross validation is consistently and significantly less than the error using least squares. For very small uncertainties ($\gamma<10^{-3}$) the two methods yield the same errors (with the near-optimum number of basis functions).

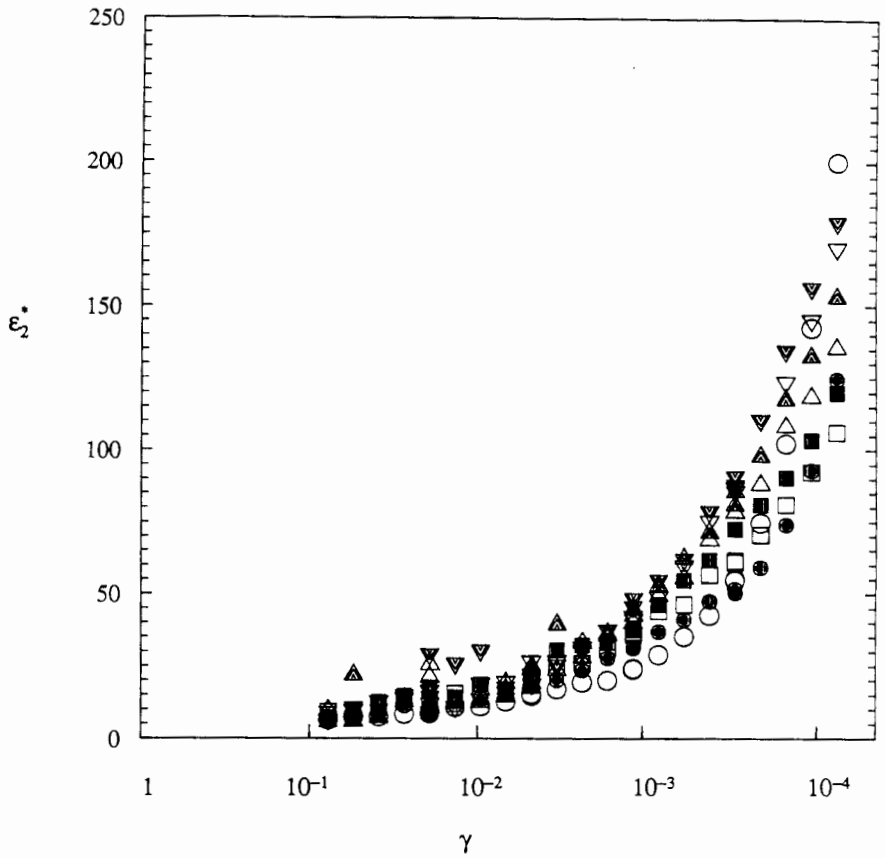

FIG. 6: Rms error in second derivative g", ε_2^*, as a function of the uncertainty γ. Symbols as Fig. 2. Cross-validated smoothing splines.

Bias

Above we have used the root mean-square errors ε_0 to characterize the difference between f(x) and its approximant g(x). We can be more precise and differentiate between three errors: g(x) can be decomposed as

$$g(x) = f(x) + d(x) + \beta(x) + r(x) . \qquad (38)$$

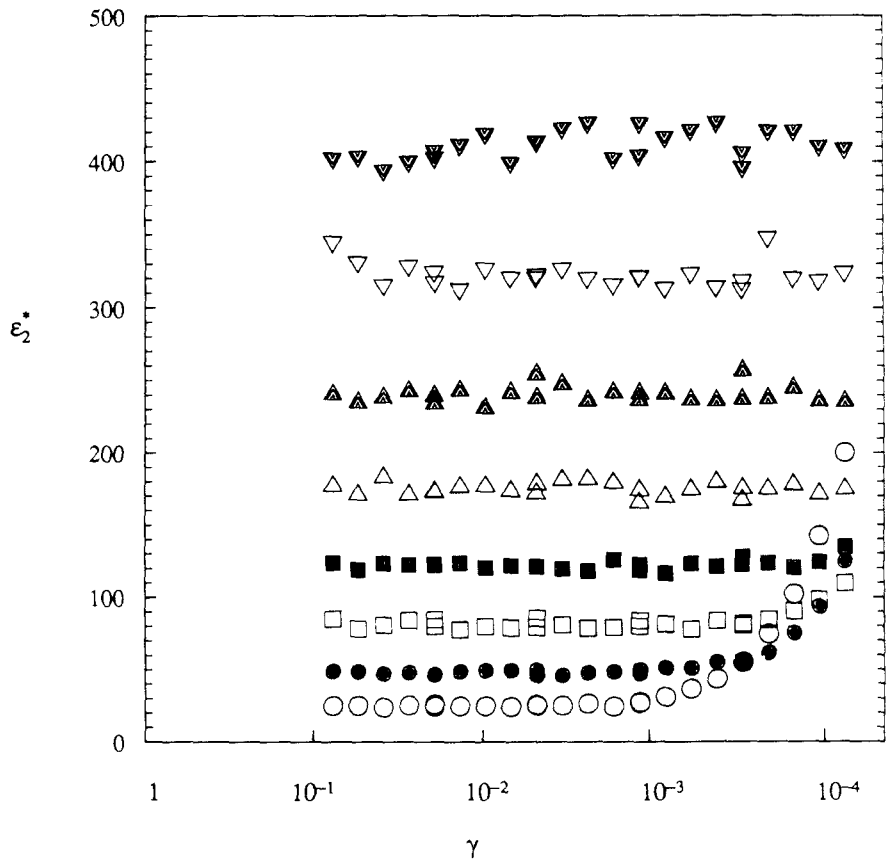

FIG. 7: Same as Fig. 6. Least-squares cubic splines.

Here d(x) is the deterministic error, mentioned above, due to the number of basis functions M being insufficient to represent f(x) accurately. With $g_0(x)$ denoting the value of g(x) obtained by least squares without random error ($p = \sigma = \gamma = 0$), the deterministic error is

$$d(x) = g_0(x) - f(x) . \tag{39}$$

Note that d(x) depends on M and is, of course, non-random.

The basis $\beta(x)$ is the systematic error due to the random error in the samples:

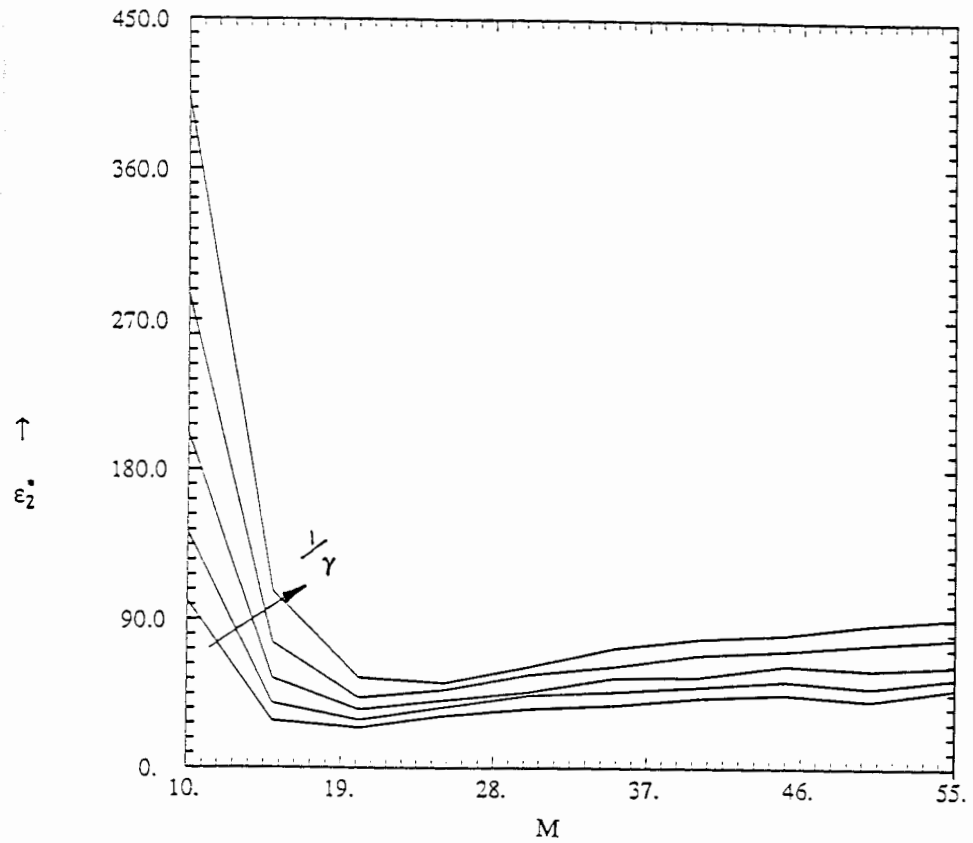

FIG. 8: Rms error in second derivative g", ε_2^*, against number of basis functions, M, for different uncertainties: γ and σ same as Fig. 1. Cross-validated smoothing splines.

$$\beta(x) \equiv \langle g(x) \rangle - f(x) - d(x)$$
$$= \langle g(x) \rangle - g_0(x) \,. \qquad (40)$$

The remainder, r(x), is the random error which, it follows, has zero mean.

In some applications (e.g. some Monte Carlo methods) a non-zero bias may be less desirable than a smaller (unbiased) random error. One reason is that unbiased random errors can be reduced at will by averaging results over many realizations. But in such a procedure the bias remains unchanged.

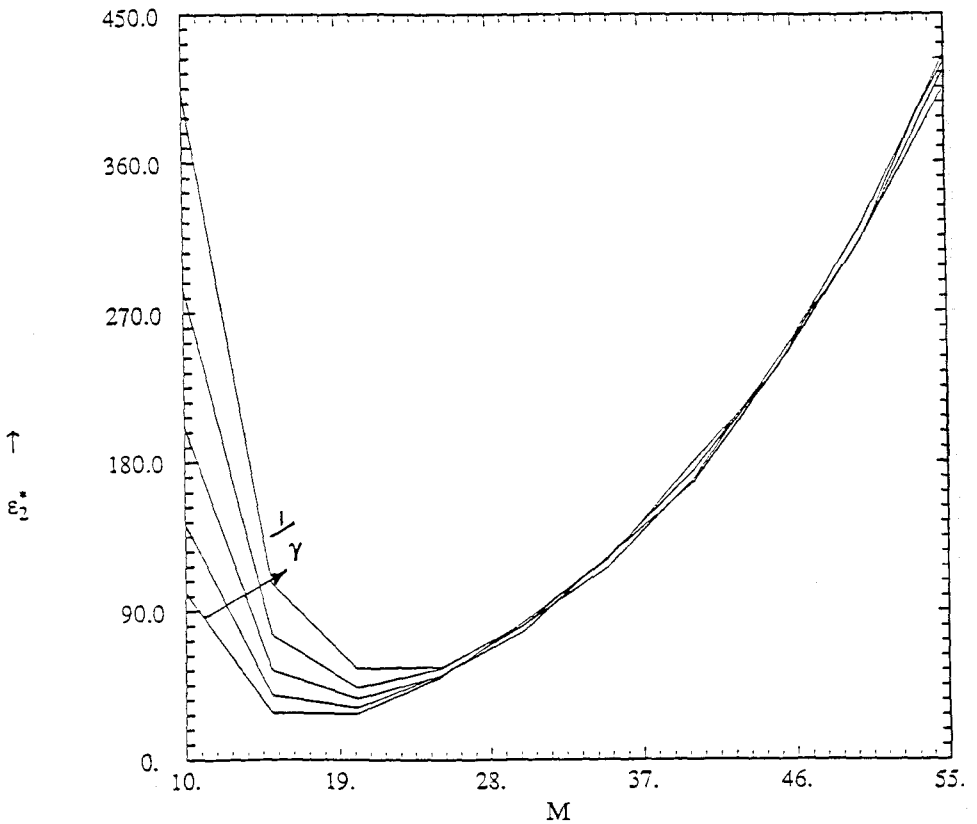

FIG. 9: Same as Fig. 8. Least-squares cubic splines.

In this respect the least-squares method has an advantage in that it is unbiased. With some mild assumptions, it follows from Eq. (13) that the spline coefficients $\underline{a}_\beta$ of the bias $\beta(x)$ are given by

$$\underline{a}_\beta = - p \langle \underline{\underline{B}} \rangle^{-1} \underline{\underline{C}} \langle \underline{a} \rangle . \tag{41}$$

Clearly, then, with least squares (p=0) the bias is zero. With cross-validated smoothing (p>0) the bias is non-zero except in particular circumstances. (For example, if $\langle \underline{a} \rangle$ corresponds to a straight line (i.e. a curve of zero roughness) then $\underline{\underline{C}} \langle \underline{a} \rangle$ is zero.)

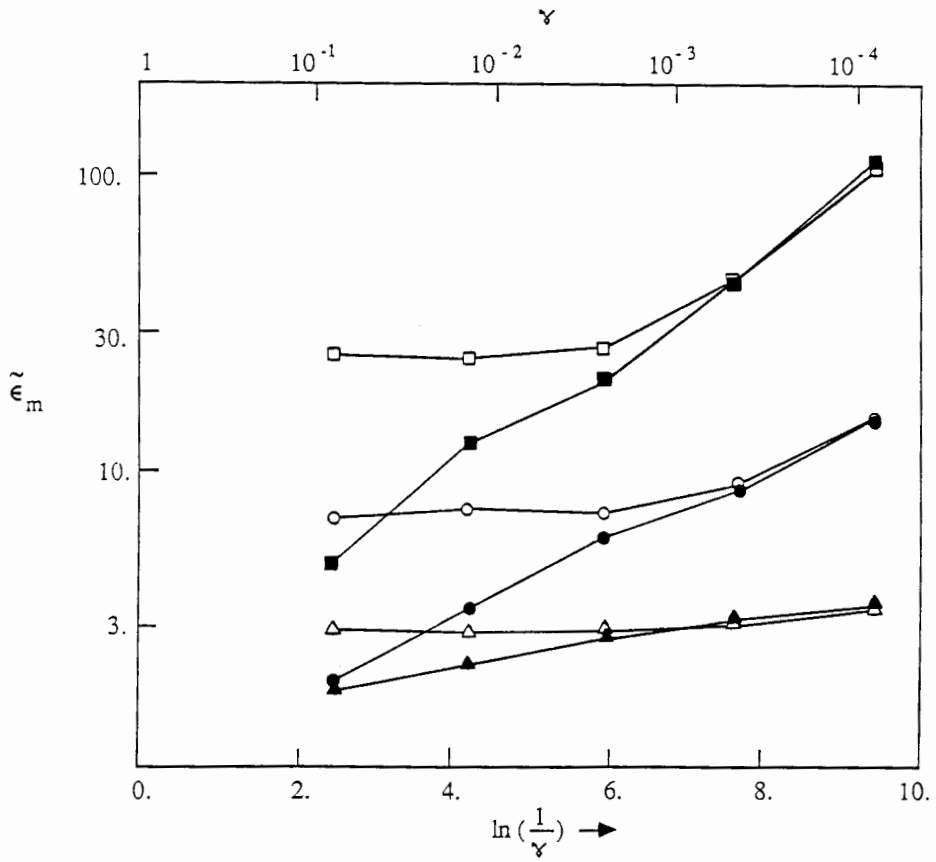

FIG. 10: Minimum (over basis functions used) rms errors $\tilde{\varepsilon}_0$, $\tilde{\varepsilon}_1$, $\tilde{\varepsilon}_2$ as functions of the uncertainty γ. Triangles, $\tilde{\varepsilon}_0$; circles, $\tilde{\varepsilon}_1$; squares, $\tilde{\varepsilon}_2$; solid symbols, cross-validated smoothing splines; open symbols, least-squares cubic splines.

We define the rms bias error ε_β by

$$\varepsilon_\beta^2 = \frac{1}{2\pi} \int_\pi^{3\pi} \langle \beta(x)^2 \rangle \, dx \quad . \tag{42}$$

As before, ε_β is estimated by averaging over many realizations. In this case, however, the estimation and control of the statistical error is less straightforward.

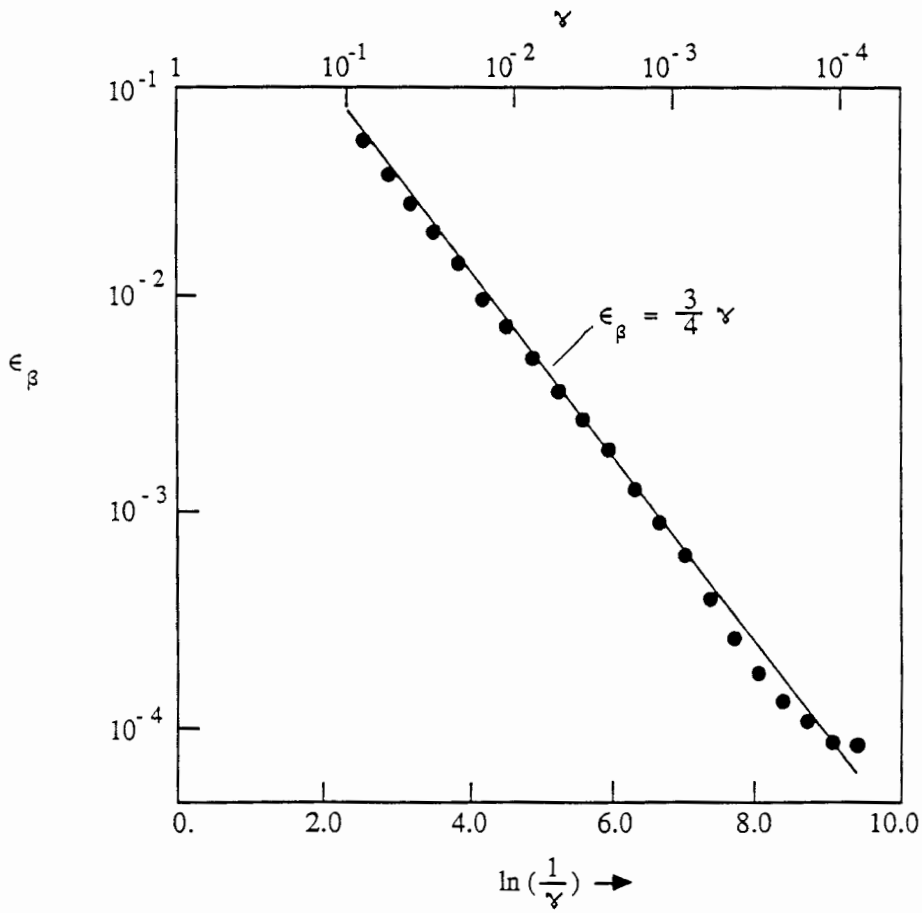

FIG. 11: Rms bias error ε_β as a function of the uncertainty γ. (M = 25).

The method used (see Gadh 1987, for details) results in a statistical error of less than 8%.

Figure 11 shows ε_β as a function γ for M = 25. It may be seen that the data are well represented by the empirical relation

$$\varepsilon_\beta/\gamma = 0.75 \, . \tag{43}$$

Although all three errors (d, β and r) contribute to the rms error ε_0, it should

be noted that bias makes a small contribution. If ε_0^* is 2, say, and Eq. (43) holds, then the bias makes less than a 10% contribution to ε_0^*.

In summary, unlike the least squares, the cross-validated smoothing algorithm is biased. However, the bias is quite small (10%) compared to the random error.

DISCUSSION

The algorithm described has been used extensively in Monte Carlo calculations (Nguyen and Pope 1984; Pope and Correa 1987; Haworth and Pope 1987a,b, for example). It has proved completely reliable when applied to widely differing functions f(x), with different types of random error. We now discuss some observations concerning the algorithm and extensions to it.

Rate of Convergence

The smoothing parameter p^* is determined iteratively. It is found that the convergence of this iteration is not as fast as desirable: typically 15 iterations are needed. It may be that an analytical investigation into the nature of the function R(p) (Eq. 23) could guide the development of a more rapidly converging algorithm.

Variants

Two variants of the algorithm were investigated. The first is to define the roughness T (Eq. 6) in terms of the third derivative g'''(in place of g''). The second variant pertains to the cross validation. The samples are divided in $K > 2$ independent data sets (rather than just two). A spline is determined from each data set; and the cross error Z(p) is defined as the sum of the mean square errors between the spline s ($s = 1,2,...,K$) and the other data sets t ($t = 1,2,...,K$; $t \neq s$).

Numerical tests were performed on both of those variants (with K up to 12). Although the tests were not as comprehensive as those described in the previous section, the general conclusion is that neither variant has a major effect on the magnitude of the statistical errors ε_m^*. Hence, on grounds of simplicity and computational efficiency, the basic algorithm is to be preferred.

Cross-Validation Technique

As mentioned in the Introduction, the "leaving-out-half" cross validation technique used here differs from the "leaving-out-one" (Wahba & Wold 1975) and GCV (Craven & Wahba 1979) techniques. It has been conjectured (Wahba 1988) that for the present case of dense data all three methods give similar values for the smoothing parameter p: but it is further conjectured (Wahba 1988) that the current method is inferior for sparse data.

Boundaries

It was mentioned above that the discrepancies between f(x) and the spline approximant g(x) tend to be greatest at the boundaries x = 0 and x = L. This is because the first few and last few basis functions lie partly outside the interval [0,L]. Consequently, they are determined by fewer samples and hence have greater statistical error.

It may be — as it is in the Monte Carlo applications mentioned — that some information is known about the function at the boundaries. General linear boundary conditions are

$$\omega_1 f(0) + \omega_2 f'(0) + \omega_3 = 0 , \tag{44}$$

and

$$\omega_4 f(L) + \omega_5 f'(L) + \omega_6 = 0 , \tag{45}$$

where $\omega_1,...,\omega_6$ are prescribed constants. The splines g(x) can be constrained to satisfy these conditions, without affecting the structure of the algorithm. It is found that specifying boundary information in this way greatly improves the behavior of the splines near the boundaries.

Pre-Processing

For the case of dense data ($N > 100\, M_r$, say) the computational expense can be decreased by pre-processing the data $(f_i^{(s)}, x_i^{(s)}, w_i^{(s)};\ i = 1,N^{(s)};\ s = 1,2)$. First the interval [0,L] is divided into $\tilde{N}_{1/2}$ bins, generally of equal size. (The number $\tilde{N}_{1/2}$ is typically 4 times the number of basis functions.) The j^{th} bin is centered at $\tilde{x}_j$ and the samples falling in it are averaged to form $\tilde{f}_s^{(s)}$ and $\tilde{w}_j^{(s)}$. Then the spline algorithm is applied to the data $\tilde{f}_j^{(s)}$, $\tilde{x}_j^{(s)} = \tilde{x}_j$, $\tilde{w}_j^{(s)}$; $j = 1, \tilde{N}_{1/2}$; $s = 1,2$.

The principal advantage of this pre-processing is that the computational work required to form the coefficients $\underline{\underline{B}}_{(s)}$ and $\underline{Y}_{(s)}$ (Eqs. 15 and 17) is greatly reduced. A secondary advantage is that the sample locations $\tilde{f}_j^{(1)}$ and $\tilde{f}_j^{(2)}$ coincide. Provided the bin width $L/\tilde{N}_{1/2}$ is small compared to the scale of variation of the function L/M_r, a negligible deterministic error is incurred.

A second form of pre-processing is to estimate the variance $\sigma^2(x)$ from the samples, and to prescribe numerical weights $w_i^{(s)}$ based, in part, on this estimate. Since this process is common to many algorithms we do not elaborate on it here.

Extension to Several Dimensions

For the case of d dimensions (d > 1), there is an obvious extension of the algorithm, in which there are M d-dimensional basis functions, and the roughness is defined in terms of $\nabla^2 g(\underline{x})$. Such an algorithm, though feasible, would be considerably more expensive than the one-dimensional algorithm. This is because the number of basis functions required increases exponentially with d (all other things being equal).

More attractive, when possible, is the tensor-produced method (see de Boor 1978). In this method the d-dimensional spline $g(\underline{x})$ is formed by constructing one-dimensional splines in each of the d-directions in turn. This method has been implemented in two dimensions (Anand, Pope & Mongia 1988) and in three dimensions (Haworth 1987). Preliminary tests suggest that good results are obtained by using the cross-validated smoothing algorithm in the first direction, and the more economical least-squares method in the remaining directions.

CONCLUSION

We have developed an algorithm to approximate an unknown function f(x) given function values containing random noise. The approximant g(x) is a cubic spline with a sufficient number of basis functions to accurately represent f(x). The basis-function coefficients are determined by minimizing a combination of the infidelity E (i.e. the mean-square error between g(x) and the data) and the roughness T of g(x), Eq. (6). The quantity minimized $\chi \equiv E + pT$ depends on a smoothing parameter p. A "leaving-out-half" cross-validation technique is used to determine a suitable value of p.

For a simple sinusoidal test function, the performance of this method has been determined, and compared to that of least-squares cubic splines. The main conclusions from these tests are as follows.

i) The choice of the number of basis functions is not crucial, providing only that there are sufficient to represent the function f(x) accurately.

ii) For least-squares splines, on the other hand, the error between f(x) and g(x) increases rapidly with the number of basis functions, beyond some optimum number.

iii) Even if the optimum number of basis functions is used, the errors in the cross-validated smoothing method are substantially less than those in the least-squares method (except when there is little uncertainty in the data, in which case the performance of the two methods is similar).

iv) The least-squares method is unbiased, whereas the cross-validated smoothing method has a small bias. The magnitude of the bias is typically 10% of the unbiased random error.

The method has been used extensively in Monte Carlo calculations. Its performance is enhanced by pre-processing the data, and by incorporating any known boundary information. By using the tensor product representation, the method has been extended to two and three dimensions.

ACKNOWLEDGMENTS

This work was supported by grant number DAA929-84-K-0020 from the U.S. Army Research Office. The computations were performed at the Cornell National Supercomputer Facility which is supported in part by the National Science Foundation, New York State and the IBM Corporation. The authors are grateful to Professor Grace Wahba for valuable comments on the original version of this paper.

BIBLIOGRAPHY

Anand, M.S., Pope, S.B. and Mongia, H.C. (1988), "A pdf method for turbulent recirculating flows," in *Turbulent Reactive Flows, Volume II: Structure and Predictions*, Springer-Verlag.

de Boor, C. (1978), *A Practical Guide to Splines*, Berlin: Springer-Verlag.

Craven, P. and Wahba, G. (1979), "Smoothing noisy data with spline functions," *Numer. Math.*, **31**, 377-403.

Dahlquist, G. and Björck, A. (1974), *Numerical Methods*, New Jersey: Prentice-Hall.

Elden, L. (1984), "A note on the computation of the generalized cross-validation function for ill-conditioned least squares problems," *BIT*, **24**.

Gadh, R. (1987), "Numerical aspects of Monte Carlo methods as applied to turbulent premixed flames," M.S. Thesis, Cornell University.

Haworth, D.C. (1987), "Developments in applying a pdf/Monte Carlo approach to engine cylinder flows," General Motors Research Laboratories Report

Haworth, D.C. and Pope, S.B. (1987a), "A pdf modeling study of self-similar turbulent free shear flows," *Phys. Fluids*, **30**, 1026-1044.

Haworth, D.C. and Pope, S.B. (1987b), "Monte Carlo solutions of a joint pdf equation for turbulent flows in general orthogonal coordinates," *J. Comput. Phys.*, **72**, 311-346.

Hutchinson, M.F. (1985), "Alogorithm 642 A fast procedure for calculating minimum cross validation cubic smoothing splines," *ACM Trans. Math. Software*, **12**, 150-153.

Lancaster, P. and Salkauskas, K. (1986), *Curve and Surface Fitting*, London: Academic Press.

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

Nguyen, T.V. and Pope, S.B. (1984), "Monte Carlo calculations of turbulent diffusion flames," *Combust. Sci. Technol.*, **42**, 13-45.

Nychka, D., Wahba, G., Goldfarb, S. and Pugh, T. (1984), "Cross-validated spline methods for the estimation of three-dimensional tumor size from observations on two-dimensional cross sections," *J. Am. Stat. Assoc.*, **79**, 832-846.

O'Sullivan, F. and Wahba, G. (1985), "A cross validated Bayesian algorithm for nonlinear remote sensing," *J. Comput. Phys.*, **59**, 441-455.

Pope, S.B. (1985), "PDF methods for turbulent reactive flows," *Prog. Energy Combust. Sci.*, **11**, 119-192.

Pope, S.B. and Correa, S.M. (1988), "Joint pdf calculations of a non-equilibrium turbulent diffusion flame," *Twenty-First Symposium (International) on Combustion*, The Combustion Institute, 1341-48.

Reinsch, C.M. (1967), "Smoothing by spline functions," *Numer. Math.*, *10*, 177-183.

Schoenberg, I.J. (1964), "Spline functions and the problem of graduation," *Proc. Nat. Acad. Sci.*, **52**, 947-950.

Wahba, G. (1988), Personnal Communication.

Wahba, G. (1985), "A comparison of GCV and GML for choosing the smoothing parameter in the generalized spline smoothing problem," *Ann. Statist.*, **13**, 1378-1402.

Wahba, G. (1980), "Spline bases, regularization and generalized cross validation for solving approximation problems with large quantities of noisy data," in *Approximation Theory III*, (W. Cheney, ed.) London: Academic Press.

Wahba, G. and Wold, S. (1975), "A completely automatic French curve: fitting spline functions by cross validation," *Communications in Statistics*, **4**, 1-

Woltring, H.J. (1986), "A FORTRAN package for generalized, cross-validatory spline smoothing and differentiation," *Adv. Engng. Software*, **8**, 104-113.

Received by Editorial Board member February, 1987; Revised March, 1988.

Recommended by Grace G. Wahba, University of Wisconsin-Madison, Madison, Wisconsin.