A Perfect Smoother

Paul H. C. Eilers*

Department of Medical Statistics, Leiden University Medical Centre, P.O. Box 9604, 2300 RC Leiden, The Netherlands

The well-known and popular Savitzky−Golay filter has several disadvantages. A very attractive alternative is a smoother based on penalized least squares, extending ideas presented by Whittaker 80 years ago. This smoother is extremely fast, gives continuous control over smoothness, interpolates automatically, and allows fast leave-one-out cross-validation. It can be programmed in a few lines of Matlab code. Theory, implementation, and applications are presented.

In 1964, in this journal, Savitzky and Golay published an algorithm for smoothing and interpolation of noisy data.[1] This article has been cited many times in the chemical literature and elsewhere, and the Savitzy−Golay smoother (SGS) is still very popular. Savitzky and Golay were pioneers of the local least-squares (local likelihood) approach to smoothing that has become a cornerstone of modern regression methodology.[2−4] Advantages of the SGS, compared to traditional filters, are that it has no delay—so peaks in signals do not shift—and that it can handle (short stretches of) missing data quite well. The principle of the SGS is easy: fit a low-order polynomial through a data interval and keep the fitted value in the middle of the interval; shift the interval (one sample distance) to the right and repeat this procedure until all desired smoothed values have been computed. In practice, one has to take special care with (stretches of) missing values and with the computations at the boundary of the data domain. This makes programs for the SGS relatively complicated.[5] The SGS can also be slow: the amount of computation increases proportionally to the amount of smoothing, as longer and longer intervals are needed for polynomial fitting. There is little opportunity to exploit the speed of matrix and vector operations in languages such as Matlab.

In 1923, Whittaker published an algorithm for smoothing—he called it graduation—of life tables.[6] It is little known outside actuarial circles, although elements of it can be found in scattered places in the statistical literature. A very similar procedure is known as the Hodrick−Prescott filter in econometrics.[7] The aim of this article is to challenge the SGS with the Whittaker smoother, which has many attractive properties:

- It can be programmed in less than 10 lines of Matlab.
- It adapts to boundaries automatically.
- It handles missing values, even in large stretches, automatically by introducing a vector of 0-1 weights.
- When sparse matrices are being used, it smoothes even a series of a 100 000 observations in a few seconds on an average PC.
- It gives continuous control over smoothness with one parameter.
- It allows extremely fast cross-validation.

The next section presents the theory in an elementary way, pickled with small Matlab code fragments and application to simulated experimental data. Interpolation of missing data, extrapolation, and the effects of strong smoothing will also be presented there. Section 3 discusses (fast) cross-validation for automatic choice of the smoothing parameter. The discussion compares this smoother to smoothing splines, Fourier filters, and wavelets.

Throughout this paper, it will be assumed that the data have been sampled at equal intervals. The smoothing algorithm can easily be extended to nonuniform sampling. This is documented in the Supporting Information.

DISCRETE PENALIZED LEAST SQUARES

The Basic Algorithm. Suppose a (noisy) series y, of length m is given. The observations have been sampled at equal distances. For most applications, this will hold, because many instruments sample signals at a regular grid along time, wavelength, or frequency. The case of arbitrarily spaced observations is discussed in the Supporting Information.

We want to fit a smooth series z to y. We then have to balance two conflicting goals: (1) fidelity to the data and (2) roughness of z. The smoother z is, the more it will deviate from y. Roughness of z can be expressed with differences. For example, first differences are given by $\Delta z_i = z_i - z_{i-1}$. Squaring and summing the differences gives us a simple and effective measure of the roughness of z: $R = \sum_i (\Delta z_i)^2$. The lack of fit to the data can be measured as the usual sum of squares of differences: $S = \sum_i (y_i - z_i)^2$. A balanced combination of the two goals is the sum $Q = S + \lambda R$, where λ is a number chosen by the user. The idea of penalized least squares is to find the series z that minimizes Q. The larger λ is, the stronger the influence of R on the goal Q and the smoother z will be (at the cost of the fit to the data getting worse).

[*] To whom correspondence should be addressed. E-mail: p.eilers@lumc.nl.

[1] Savitzky, A.; Golay, M. J. E. *Anal. Chem*. **1964** *36*, 1627.

[2] Hastie, T.; Tibshirani, R. *Generalized Additive Models*; Chapman and Hall: London, 1990.

[3] Loader, C. *Local Regression and Likelihood*; Springer: New York, 1999.

[4] Fan, J.; Gijbels, I. *Local Polynomial Modeling and Applications*; Chapman and Hall: London, 1996.

[5] Press: W. H.; Flannery, B. P.; Teukolsky, S. A.; Vetterling W. T. *Numerical Recipes in C: the Art of Scientific Computing*, 2nd ed.; Cambridge University Press: Cambridge, 1992.

[6] Whittaker, E. T. *Proc. Edinburgh Math. Soc.* **1923**, *41*, 63.

[7] Hodrick, R. J.; Prescott, E. C. *J. Money, Credit Banking* **1997** *29*, 1.

10.1021/ac034173t CCC: $25.00 © 2003 American Chemical Society
Published on Web 05/30/2003

To avoid a lot of tedious algebra, it is advantageous to introduce matrices and vectors

$$Q = |\mathbf{y} - \mathbf{z}|^2 + \lambda|\mathbf{Dz}|^2 \qquad (1)$$

where $|\mathbf{a}|^2 = \sum_i a_i^2$ indicates the quadratic norm of any vector **a**, and **D** is a matrix such that $\mathbf{Dz} = \Delta\mathbf{z}$. For example,when $m = 5$, **D** would be

$$\mathbf{D} = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \qquad (2)$$

Each row contains the pattern −1 1, shifted such that $d_{ii} = -1$ and $d_{i,i+1} = 1$. **D** has $m - 1$ rows and m columns.

Using results from matrix calculus, we find for the vector of partial derivatives

$$\partial Q/\partial \mathbf{z}' = -2(\mathbf{y} - \mathbf{z}) + 2\lambda\mathbf{D}'\mathbf{Dz} \qquad (3)$$

and equating this to 0, we get the linear system of equations

$$(\mathbf{I} + \lambda\mathbf{D}'\mathbf{D})\mathbf{z} = \mathbf{y} \qquad (4)$$

where **I** is the identity matrix.

Implementation in Matlab is very straightforward, because we can use the built-in function diff() to compute **D**:

```
m = length(y);
E = eye(m);
D = diff(E);
z = (E + lambda * D′ * D)\y;
```

This works fine for shorter data series, say $m \leq 1000$. Beyond that size, the time to solve the equations becomes relatively large. Storage requirements may also become nontrivial. These problems can be eliminated by the use of sparse matrices:

```
m = length(y);
E = speye(m);
D = diff(E);
C = chol(E + lambda * D′ * D);
z = C\(C′\y);
```

The function call speye(m) builds a sparse identity matrix, which is stored as three vectors, containing row, column, and value of only the nonzero elements of the identity matrix. This way, only 3m (floating point) memory locations are needed, instead of m^2. The diff() function and matrix multiplication are intelligent enough to maintain sparseness, so the storage of **D** takes only 6m locations and that of $I + \lambda\mathbf{D}'\mathbf{D}$, only 9$m$ locations. We could solve the system in the same way as in the last line of the nonsparse algorithm, but we use the Cholesky decomposition instead. The reason for this is that Matlab's default behavior is to try to order sparse equation systems for minimal bandwidth. This can take quite some time for large m, but is unnecessary in this application, because the bandwidth of $\mathbf{I} + \lambda\mathbf{D}'\mathbf{D}$ is already optimal (only the main diagonal and the first subdiagonals on each side are nonzero).

The computational bottom line is that on a 1000-MHz Pentium III PC, a series with length $m = 10^5$ is smoothed in 2 s; on the same machine, the nonsparse algorithm takes 6 s when $m = 1000$. For $m = 1000$, a Matlab implementation of the Savitzky−Golay smoother (see Supporting Information) takes ~2 s, largely independent of the window size. Computation time increases linearly in m. Of course, this 100-fold gain in speed relies heavily on the sparse matrix features of Matlab. If one is not in the lucky position to have such a tool at hand, one can use the sparse C implementation.[8]

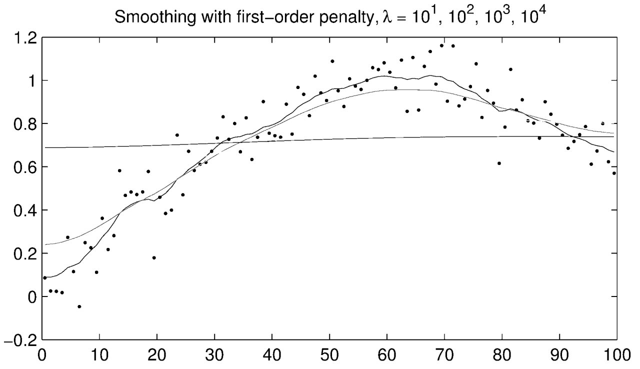

Figure 1. Smoothing of simulated data with a first-order penalty (d = 1) for several values of the smoothing parameter, λ. The larger λ is, the smoother the curve.

To simplify the presentation, the algorithm was introduced with first-order differences in the penalty. This the most simple case. Whittaker used third-order differences, and in many cases, second-order differences work fine. The formulas are

$$\Delta^2 z_i = \Delta(\Delta z_i) = (z_i - z_{i-1}) - (z_{i-1} - z_{i-2}) = z_i - 2z_{i-1} + z_{i-2} \qquad (5)$$

$$\Delta^3 z_i = \Delta(\Delta^2 z_i) = z_i - 3z_{i-1} + 3z_{i-2} - z_{i-3} \qquad (6)$$

but we do not need to know these details, as the Matlab diff() function allows an extra parameter, say d, for the order of the differences: D = speye(m, d).

Figure 1 shows smoothing with $d = 1$ for several values of λ, and Figures 2 and 3 do the same for $d = 3$. Note that with increasing smoothness, **z** approaches a horizontal line when $d = 1$, a tilted straight line when $d = 2$, and a parabola when $d = 3$. The reason for this will be discussed in the following. Also note that λ varies over a rather wide range, especially for $d = 3$.

Missing Data. Quite often data can be missing, such as in environmental monitoring time series, because of instrument malfunction or maintenance periods. The smoother can easily be modified to handle this situation. The missing elements of y are set to an arbitrary value, say 0, and a vector **w** of weights is introduced, with $w_i = 0$ for missing observations and $w_i = 1$ otherwise. The measure of fit is changed to

$$\mathrm{S} = \sum_i w_i\,(y_i - z_i)^2 = (\mathbf{y} - \mathbf{z})'\mathbf{W}(\mathbf{y} - \mathbf{z}) \qquad (7)$$

[8] Eilers, P. H. C. In *Graphic Gems IV*; Heckbert, P., Ed.; Academic Press: New York, 1994; pp 241−250.

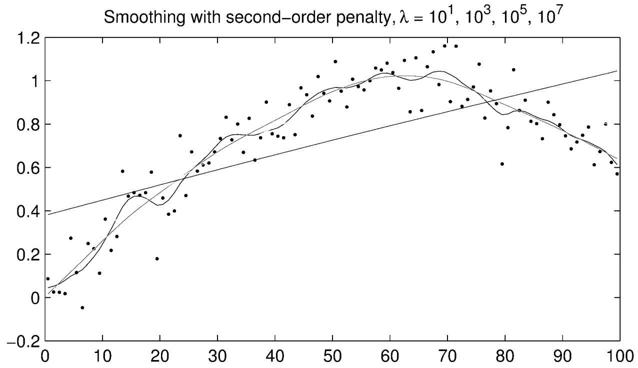

Figure 2. Smoothing of simulated data with a second-order penalty ($d = 2$) for several values of the smoothing parameter, λ. The larger λ is, the smoother the curve.

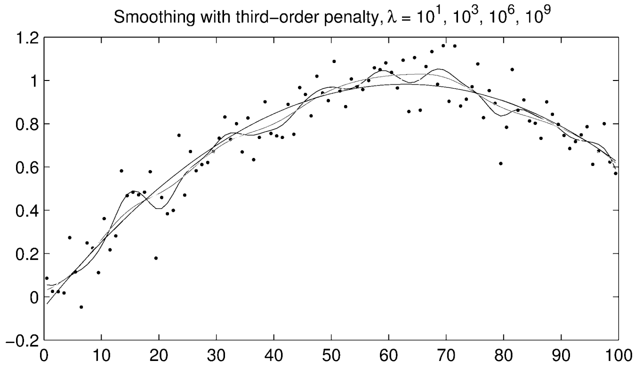

Figure 3. Smoothing of simulated data with a third-order penalty ($d = 3$) for several values of the smoothing parameter, λ. The larger λ is, the smoother the curve.

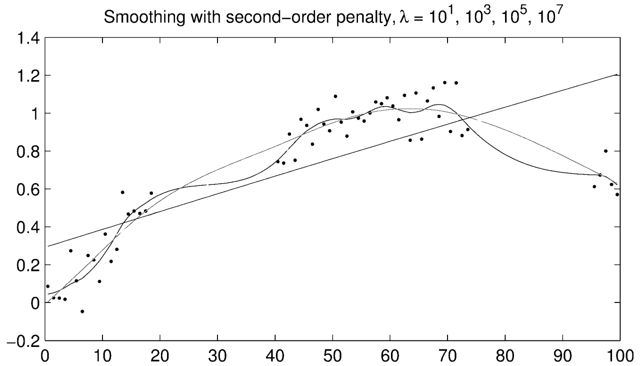

Figure 4. Smoothing and interpolation with missing data using a second-order penalty. The larger λ is, the smoother the curve.

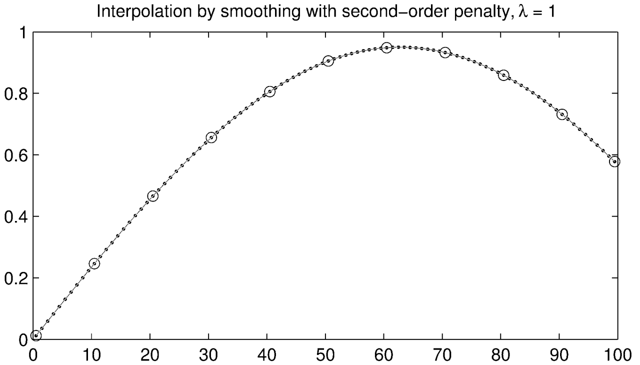

Figure 5. Interpolation by light smoothing of low-noise data. The circles indicate the data points; the small dots and the line, the smooth interpolated series.

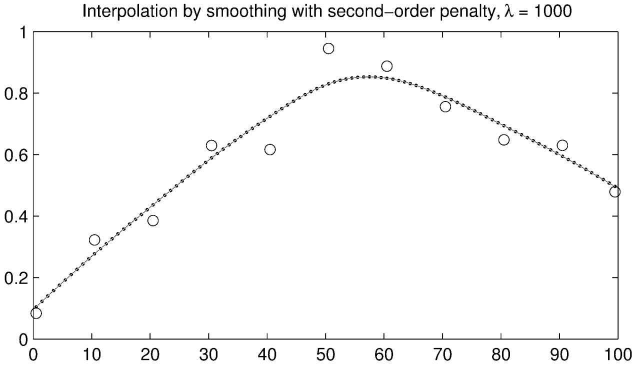

Figure 6. Interpolation by strong smoothing of noisy data. The circles indicate the data points; the small dots and the line, the smooth interpolated series.

where $\mathbf{W} = \mathrm{diag}(\mathbf{w})$, a diagonal matrix with w on its diagonal. The system of equations changes to

$$(\mathbf{W} + \lambda \mathbf{D}'_d \boldsymbol{D}_d)\, z = \mathbf{W} y \tag{8}$$

where the order of the differences, d, is indicated by the subscript in $\mathbf{D}_d$. The Matlab code becomes

```
m = length(y);
E = speye(m);
D = diff(E, d);
W = spdiags(w, 0, m, m);
C = chol(W + lambda * D′ * D);
z = C\(C′\(w.* y));
```

At the positions where y is missing, z is automatically and smoothly interpolated. This is illustrated in Figure 4 where some stretches of missing data were introduced deliberately.

We can use missing data as an easy device for smoothing and detailed interpolation. Suppose we want to derive a five times oversampled series from a given series $\mathbf{y}$. Then we construct a series u consisting of zeros, except that $u_1 = y_1$, $u_6 = y_2$, and so on, with generally $u_{5i-4} = y_i$. We also construct a series of weights $\mathbf{w}$ that is zero everywhere, except at the positions $5i - 4$, where it is one. Figure 5 illustrates this for a short low-noise series of simulated data with light smoothing, and Figure 6, for noisy data with stronger smoothing.

Planned "missing values" can also be used for extrapolation (at both ends) by extending both $\mathbf{y}$ and $\mathbf{w}$ with zeros up to the desired length. In the next subsection, we will see that interpolation sections are polynomial in i of order $2d$, while extrapolated sections are polynomials of degree d.

Some Polynomial Properties. Suppose we make λ very large. Then $(\mathbf{W} + \lambda\mathbf{D}'_d\mathbf{D}_d)\mathbf{z} = \mathbf{W}\mathbf{y}$ reduces to $\mathbf{D}'_d\mathbf{D}_d\mathbf{z} \approx \mathbf{W}\mathbf{y}/\lambda$, or $\mathbf{D}'_d\mathbf{D}_d\mathbf{z} \approx 0$. This will hold if $\mathbf{D}_d z \approx 0$. It is easily proved that $\Delta^d i^k = 0$ everywhere if k is an integer and $0 \le k < d$. From that follows a result for a polynomial of degree $d - 1$: $\Delta^d \sum_{k=0}^{d-1} \alpha_k i^k = 0$. So when $d = 1$, $\mathbf{z}$ approaches a constant, and when $d = 2$, it approaches a straight line when λ gets very large. The polynomial of degree $d - 1$ is not an arbitrary one, but the one that minimizes $\sum_i w_i(y_i - z_i)^2$, that is, the (weighted) least squares regression line through $\mathbf{y}$, taking $i = 1 \ldots m$ as the independent variable.

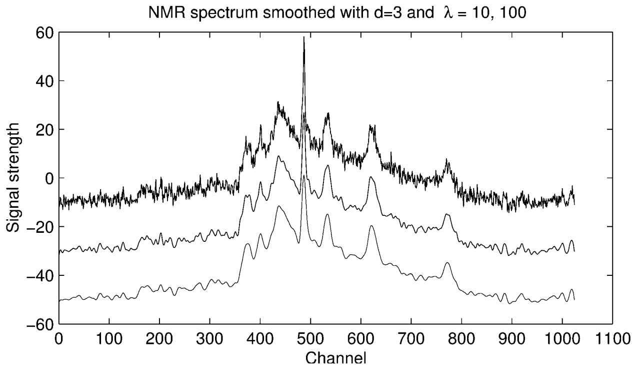

Figure 7. Smoothing of an NMR spectrum. For clarity, the smooth curves have been shifted vertically.

A similar result can be proved for extrapolation. It is easiest when we consider extrapolation at the left boundary of the data. Then for an initial segment, say $i = 1 \dots n$, we have that $w_{\text{i}} = 0$. For that segment, we have that $\tilde{\mathbf{D}}_d{}'\tilde{\mathbf{D}}_d\tilde{\mathbf{z}} = 0$, for any value of λ, where the symbol $\sim$ indicates that we restrict our attention to the segment $i < k$. This means that $\Delta^d\tilde{\mathbf{z}} = 0$ and, hence, that $\tilde{\mathbf{z}}$ is a polynomial in i of degree $d - 1$. The coefficients of the polynomial are determined by the smooth fit to the real data to the right.

Interpolation gives a different result. Let the symbol $\sim$ now indicate an interior stretch of missing data. Except for the d by d submatrices in the upper-left and lower-right corners, the rows of $\tilde{\mathbf{D}}'_d\tilde{\mathbf{D}}_d$ are equal to those of $\mathbf{D}_{2d}$, but centered on the diagonal. So we have that for that part of z, $\tilde{\mathbf{D}}_{2d}\tilde{\mathbf{z}} = 0$. This means that a polynomial of degree $2d - 1$ is used for interpolation. The coefficients are determined by the smooth fit to the available data at both sides.

Notice that polynomial interpolation and extrapolation all takes place automatically. In some cases the interpolated curve might be too flexible. One can combine first- and second-difference penalties to get tighter results.[8] In that case, exponential functions are used (again automatically) for interpolation and extrapolation.

To illustrate the smoother in action, Figure 7 presents an NMR spectrum that is part of the Wavelab toolbox (www-stat.stanford-.edu/∼wavelab). Performance is quite good, except for the very sharp peak in the middle, which gets rounded even for rather small λ. To satisfactorily handle such pronounced local features one needs an adaptive approach, with a locally varying penalty. However, this falls outside the scope of this paper.

OPTIMAL SMOOTHING

One way of choosing a value for λ, the smoothing parameter, is tuning it until a visually pleasing result is obtained. A more objective choice can be made with cross-validation. The idea is to leave out each of the (nonmissing) elements of y in turn, smooth the remaining data and get a "prediction" $\hat{y}_{-\text{i}}$ for the left out y_{i}. By repeating this for all y_{i} we can compute the cross-validation standard error.

$$s_{\text{cv}} = \sqrt{\sum_i (y_i - \hat{y}_{-\text{i}})^2/m} \qquad (9)$$

To find the optimal value of λ, it is varied on a grid (taking log λ approximately linearly spaced) to search for a minimum of s_{cv}.

If followed literally, this would be a rather expensive recipe, as one has to smooth m series of length m, an amount of work proportional to m^2. Luckily, we can speed up the computations to make them proportional to m, as will now be shown. From eq 8 follows

$$\mathbf{z} = \hat{\mathbf{y}} = (\mathbf{W} + \lambda\mathbf{D}'_d\mathbf{D}_d)^{-1}\mathbf{W}\mathbf{y} = \mathbf{H}\mathbf{y} \qquad (10)$$

The matrix $\mathbf{H}$ is called the hat matrix in the regression literature, because it puts the "hat" (indicating a fitted value) on y. Hastie and Tibshirani[2] call it the smoother matrix. They also prove the following relationship, which is well-known in the regression literature.

$$y_i - \hat{y}_{-i} = \frac{y_i - \hat{y}_i}{1 - h_{ii}} \qquad (11)$$

So if the diagonal of $\mathbf{H}$ is available, it is trivial to compute the cross-validation residuals $y_i - \hat{y}_{-i}$ from the standard residuals $y_i - \hat{y}_i$. Alternatively, one can use generalized cross-validation,

$$s_{\text{cv}} = \sqrt{\sum_i \left(\frac{y_i - \hat{y}_{-i}}{1 - \bar{h}}\right)^2 / m} \qquad (12)$$

where $\bar{h} = \sum_i h_{ii}/m$, the average of the diagonal elements of $\mathbf{H}$. This means that in eq 11, $\bar{h}$ is substituted for h_{ii}.

For small m, it is no problem to compute $\mathbf{H}$ in full and take its diagonal. But for long data series, we lose the advantages of sparse matrices, because $\mathbf{H}$ is a full matrix. But there is a way out. We can write, in the case of no missing data,

$$(\mathbf{I} + \lambda\mathbf{D}'_d\,\mathbf{D}_d)\,\mathbf{H} = \mathbf{I} \qquad (13)$$

So the columns of $\mathbf{H}$ can be found one by one by smoothing the corresponding columns of the identity matrix. This gives an opportunity to avoid the storage problem if we were to compute the columns in turn and keep only the element on the diagonal of $\mathbf{H}$. But it does not solve the time problem.

The key to an efficient solution is the following observation: the shape of the curve of h_{ii} vs i is nearly the same for a problems of size m_1 and m_2 if we normalize the corresponding λ_1 and λ_2 such that $\lambda_2 = (m_2/m_1)^{2d}\lambda_1$. This is illustrated by Figure 8. The normalization condition can be understood if we interpret a column of $\mathbf{H}$, when $m = m_1$, as samples of an impulse response, and $\mathbf{D}_d\mathbf{H}$ as an estimate of the dth derivative, implicitly taking the distance between the samples being 1. If we decrease the sample distance by a factor $k = m_2/m_1$ but keep the continuous impulse response the same, we have to multiply by k^d to get the right estimates of the dth derivative. In the penalty occurs a square, hence k^{2d}.

We use this property to compute the ratio of $\bar{h}$, the average of the diagonal of $\mathbf{H}$ to h_{nn} with $n = \lfloor m/2 \rfloor$. To find h_{nn} we smooth an impulse signal of size m, but we compute the full $\mathbf{H}$ for a much smaller size, say 100.

Figure 9 shows cross-validation in action for the NMR spectrum.

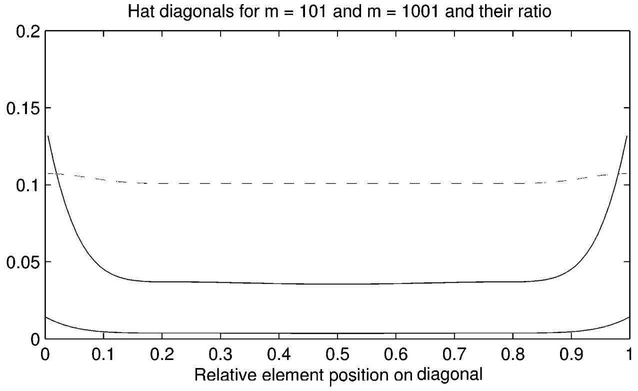

Figure 8. Diagonal elements h_{ii} of the hat matrix for $m = 101$ (upper full line) and $m = 1001$ (lower full line) plotted against i/m. Second order penalty and appropriately scaled smoothing parameters. The broken line shows the ratios of corresponding elements.

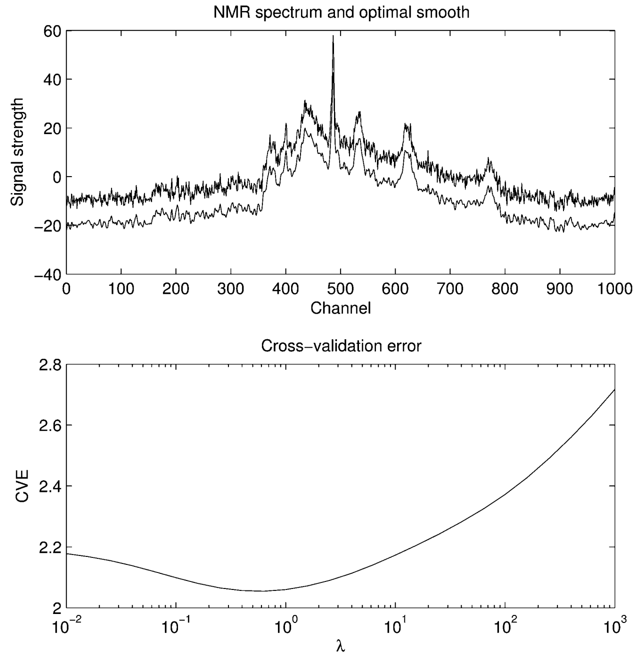

Figure 9. Automatic selection of the smoothing parameter for the NMR spectrum using cross-validation. Upper panel: data and smooth; for clarity, the smoothed curve has been shifted down by 10 units. Lower panel: cross-validation profile.

One should realize that cross-validation assumes a smooth trend with independent errors. Only in that case, automatically chosen λs will be in line with expectations obtained from visual inspection of the data. If the errors have serial correlation, one may find that cross-validation chooses a rather small amount of smoothing. Although at first this may seem surprising, it is, in fact, reasonable, because only with such light smoothing will the residuals be independent. A simple solution that appears to work quite well on longer series was suggested by Hans Boelens (University of Amsterdam): use only every second, fifth, or tenth sample. This breaks the serial correlation effectively. An effective way to implement this is to introduce 0/1 weights.

Figure 10 shows the surface profile of a sanded piece of wood.[9] Cross-validation with the original data chooses a λ that gives curve that hardly differs from the data itself. Using every tenth sample for cross-validation, we get a curve that much better represents the trend that the human eye sees in the data. Notice that the logarithm of λ was varied in steps of 0.5 on a linear grid to search for the minimum of s_{cv}. There is little point in trying to find a minimum with sophisticated search algorithms.

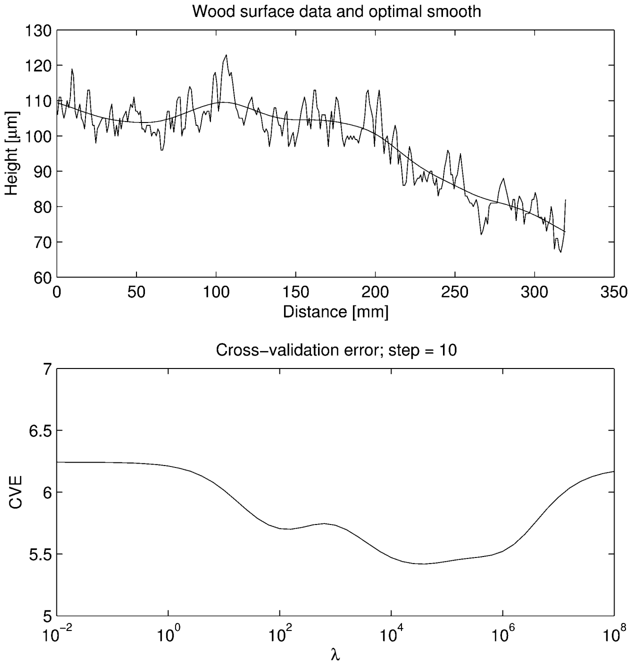

Figure 10. Upper panel: optimal smoothing (with second-order penalty) of surface height measurements of a piece of sanded wood. Lower panel: cross-validation profile; to break the strong serial correlation, every tenth sample was used.

Of course, this is no more than a simple ad-hoc solution. Questions remain about the optimal step size between observations to use for cross-validation. One can also think of combining results from stepped series that start at observations 1, 2, 3, and so on. A more fundamental solution would try to model the data explicitly as a trend plus correlated (autoregressive) noise. But this falls outside the scope of this paper. In any real application, it is advisable not to rely blindly on an automatic choice of the smoothing parameter.

DISCUSSION

Despite its popularity, the Savitzky−Golay smoother is rather inflexible and slow. Adapting the graduation smoother of Whittaker to modern software leads to dramatic improvements in speed, flexibility, and ease of cross-validation. The basic algorithm can be programmed in a few lines of Matlab.

The starting theme of this paper was to present an improved alternative to the Savitzky−Golay smoother (or more generally, local likelihood methods). It will finish with a discussion of contrasts to other popular smoothers.

The smoothing spline is similar in that it estimates a model that contains a smoothed value at each observation. It assumes an unknown piecewise polynomial smooth curve and puts a penalty on the integral of the squared second derivative. A banded system of linear equations results, which can be solved efficiently with software for sparse matrices (such as Matlab). With ap-

[9] Pandit S. M.; Wu, S. M. *Time Series and System Analysis with Applications*; Wiley: New York, 1983.

propriate weights, the smoothing spline can be used for interpolation and extrapolation. With heavy smoothing, the linear least squares line through the data will be approached. The construction of the equation is not an automatic procedure, and algorithms for penalties on derivatives of higher order than the second are hard to find.

Fourier filtering assumes that the signal to be smoothed is periodic, leading to serious trouble at the boundaries when the signal has a trend. Modern software computes Fourier transforms very quickly. Equally spaced samples are a must, and arbitrary interpolation or extrapolation are impossible, though it is possible to accomplish systematic subsampling with the Fourier transform.

Wavelets also assume periodicity of the signal. Fast algorithms demand the number of samples to be a power of two. Observations have to be equally spaced and interpolation or extrapolation is impossible. On the other hand, there are some very interesting developments in nonlinear approaches to wavelet smoothing that use thresholding of (high-frequency) wavelet coefficients.[10] They seem to deal better with sharp peaks, such as the one in the middle of the NMR spectrum in Figure 7.

The difference penalty smoother can be interpreted as a special case (with zero-degree B-splines, centered at the data points) of P-splines,[11] a combination of regression on a B-spline basis and the difference penalty (on the B-spline coefficients). In that paper, it is also explained how to extend this type of smoothing to nonnormal data such as counts or binary observations.

SUPPORTING INFORMATION AVAILABLE

The Supporting Information consists of the following: a document that presents Whittaker smoothing for unequally spaced data and a detailed comparison to the Savizky−Golay filter, a small toolbox of Matlab functions for Whittaker smoothing, and the data for the NMR spectrum and the wood profile. This material is available free of charge via the Internet at http://pubs.acs.org.

Received for review February 21, 2003. Accepted April 28, 2003.

AC034173T

[10] Percival, D. B.; Walden, A. T. *Wavelet Methods for Time Series Analysis*; Cambridge University Press: Cambridge, 2000.

[11] Eilers, P. H. C.; Marx, B. D. *Stat. Sci.* **1996** *11*, 89.