General Least-Squares Smoothing and Differentiation of Nonuniformly Spaced Data by the Convolution Method

Peter A. Gorry

Department of Chemistry, University of Manchester, Manchester, England, M13 9PL

INTRODUCTION

The least-squares smoothing and differentiation of equally spaced data points using a convolution formalism has been extensively used since its introduction by Savitzky and Golay (*1*) in 1964. The original method suffered from truncation problems at each end of the data ($2m$ points are lost for a $2m + 1$ point filter). In a recent paper Gorry (*2*) presented a simple method, based on the properties of orthogonal polynomials, for calculating the least-squares convolution weights required for a general order polynomial fit, and all its derivatives, at all positions. This extended the Savitzky–Golay technique to include all data points in a spectrum.

In contrast to uniform data the problem of rapid smoothing/differentiation of *unequally* spaced data has received very little attention. While most experiments generate data with uniform spacing, this is not always the case. Furthermore, nonuniform spacing can arise from a transformation of (originally uniform) data; e.g. transforming ion time-of-flight spectra to velocity or energy distributions. In other cases nonuniformity arises from missing data, as occurs in environmental monitoring where instrument malfunction produces gaps in the record, or when data from different sources are combined.

In principle, nonequally spaced data can be treated in the same manner as equally spaced data. To find the smoothed value of a data point we take m points on each side of it and fit a polynomial, of order n, to the $2m + 1$ points. We then evaluate the polynomial at this central point to obtain the smoothed value. Moving to the next point we repeat the whole process—continuing until the entire spectrum is covered. Since a full least-squares calculation yields the least-squares coefficients, we can evaluate the polynomial fit at any position; thus the first $2m + 1$ and last $2m + 1$ points allow us to evaluate smoothed data for the first $m + 1$ and last $m + 1$ points, respectively, and hence avoid data truncation. Unfortunately, the computation involved in least-squares fitting of polynomials is considerable and usually involves matrix inversion techniques for the solution of the normal equations. A thorough discussion of this technique can be found in Bevington (*3*). For large spectra the computation required makes it unsuitable for use as a general tool.

We present here an extension of the convolution technique to include nonuniform data. Although the computation involved is still much greater than that for uniform data, it offers a great reduction on that required by the traditional least-squares method.

METHOD

(a) Basic Theory. We assume a spectrum containing p unequally spaced data points, $\{x_i, y_i\}$, which we wish to smooth, or differentiate (to order s), with a $2m + 1$ point filter (window) and polynomial order n. This involves the least-squares fitting by a polynomial of order n to $N = 2m + 1$ consecutive points and evaluating the polynomial (or its sth derivative) at the desired x_i value—usually the middle point. In what follows we consider the local fitting of these points and use an index of $i = 1...N$ to represent the $2m + 1$ data points. The least-squares polynomial has the form

$$f_n(x) = \sum_{k=0}^{n} b_k x^k \tag{1}$$

Application of the least-squares criterion

$$\frac{\partial}{\partial b_k}\left[\sum_{i=1}^{N}(f_n(x_i) - y_i)^2\right] = 0 \tag{2}$$

leads to $n + 1$ simultaneous equations in the unknown coefficients b_k. The solution of these normal equations can be found in most textbooks on data analysis.

We can equally represent eq 1 by an expansion in discrete orthogonal polynomials, $P_k(x)$, on the x_i points (*4*, *5*). In this case we have, analogous to eq 1

$$f_n(x) = \sum_{k=0}^{n} d_k P_k(x) \tag{3}$$

The orthogonality of the polynomials is defined by the relationships

$$\sum_{i=1}^{N} P_k(x_i)P_j(x_i) = 0 \quad \text{if} \quad j \neq k \tag{4}$$

$$\sum_{i=1}^{N} (P_k(x_i))^2 = \gamma_k \tag{5}$$

In fact $P_k(x)$ is orthogonal to any *arbitrary* polynomial of order $k - 1$ and below. Substitution of eq 3 into eq 2 yields solutions for the polynomial coefficients (*4*, *5*)

$$d_k = \sum_{i=1}^{N} \frac{y_i P_k(x_i)}{\gamma_k} \tag{6}$$

Evaluating the least-squares polynomial, eq 3, at the point $x = x_t$ yields the smoothed value $f_n(x_t)$

$$f_n(x_t) = \sum_{k=0}^{n}\sum_{i=1}^{N} \frac{y_i P_k(x_i)}{\gamma_k} P_k(x_t) \tag{7}$$

$$= \sum_{i=1}^{N}\left(\sum_{k=0}^{n} \frac{P_k(x_i)P_k(x_t)}{\gamma_k}\right) y_i \tag{8}$$

Since the term in parentheses is a pure number for each i, eq 8 represents a convolution calculation. If we require the smoothed sth derivative, we obtain this by differentiating eq 8 s times to yield

$$f_n^s(x_t) = \sum_{i=1}^{N}\left(\sum_{k=0}^{n} \frac{P_k(x_i)P_k^s(x_t)}{\gamma_k}\right) y_i = \sum_{i=1}^{N} h_{t,i}^{n,s} y_i \tag{9}$$

where

$$h_{t,i}^{n,s} = \sum_{k=0}^{n} \frac{P_k(x_i)P_k^s(x_t)}{\gamma_k} \quad \text{and} \quad P_k^s(x_t) = \left(\frac{\mathrm{d}^s}{\mathrm{d}x^s} P_k(x)\right)_{x=x_t} \tag{10}$$

The convolution weights $h_{t,i}^{n,s}$ are calculated from eq 10. In order to do this we require expressions for the orthogonal polynomials themselves. These can be generated by the following recursion formula (*4*, *5*)

$$P_k(x) = (x - B_{k-1})P_{k-1}(x) - C_{k-1}P_{k-2}(x) \tag{11}$$

where

$$B_k = \sum_{i=1}^{N} \frac{x_i(P_k(x_i))^2}{\gamma_k} \quad \text{and} \quad C_k = \frac{\gamma_k}{\gamma_{k-1}} \tag{12}$$

0003-2700/91/0363-0534$02.50/0 © 1991 American Chemical Society

Table I. Expressions for $P_k(x)$, γ_k, B_k, and the Smoothing Weights, $h^k_{t,i}$, for k = 0–2

k	$P_k(x)$	γ_k	B_k	$h^k_{t,i}$
0	1	N	$\frac{1}{N}\sum_{i=1}^{N} x_i$	$\frac{1}{N}$
1	$(x - B_0)$	$\sum_{i=1}^{N} x_i^2 - B_0\sum_{i=1}^{N} x_i$	$\frac{1}{\gamma_1}(\sum_{i=1}^{N} x_i^3 - B_0\sum_{i=1}^{N} x_i^2) - B_0$	$h^0_{t,i} + \frac{(x_i - B_0)(x_t - B_0)}{\gamma_1}$
2	$x^2 + A^1_2 x + A^0_2$	$\sum_{i=1}^{N} x_i^4 + A^1_2\sum_{i=1}^{N} x_i^3 + A^0_2\sum_{i=1}^{N} x_i^2$	$\frac{1}{\gamma_2}(\sum_{i=1}^{N} x_i^5 + A^1_2\sum_{i=1}^{N} x_i^4 + A^0_2\sum_{i=1}^{N} x_i^3) + A^1_2$	$h^1_{t,i} + \frac{P_2(x_i)P_2(x_t)}{\gamma_2}$

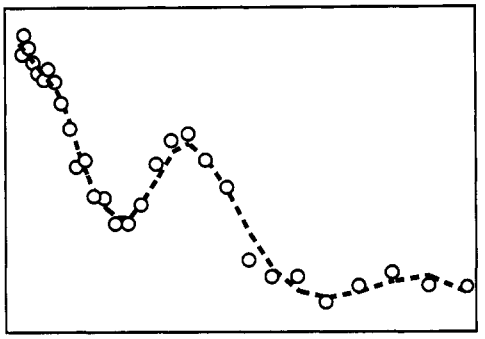

Figure 1. Least-squares smoothing of nonuniformly spaced data. The dashed line joins values from an 11-point cubic smooth.

and $P_0(x) = 1$, $P_{-1}(x) = 0$. The first three polynomials are $P_0(x) = 1$, $P_1(x) = (x - B_0)$, and $P_2(x) = (x - B_1)(x - B_0) - C_1$.

Equations 9–12 are all that are required to perform the least-squares calculation for point x_t. As an example consider a cubic smoothing calculation. We step through k from 0 to 3. At each k we calculate $P_k(x_i)$, for each point x_i from eq 11, we then calculate γ_k from eq 5, B_k and C_k from eq 12 and the weight factors $h^k_{t,i}$ from eq 10. Finally we use the weight factors for $k = 3$ to calculate the least-squares value from eq 9.

Evenly spaced data allow calculation of the weight factors to be performed independently of the actual data. It is this fact which produces the extraordinary speed increase of the Savitzky–Golay method. Unfortunately, when the data are unevenly spaced, the polynomials (and hence the weights) are different for *every* set of $2m + 1$ points. This means that we must perform the above calculation even when moving along the experimental spectrum by a single point. For small values of n the computational effort required by the polynomial approach is similar to the solution of the normal least-squares equations by conventional means. At higher n the calculations are faster, since they involve only simple summations and are not followed by matrix inversion or determinant expansion, furthermore the orthogonal polynomial approach is more stable since one can never encounter problems with ill-conditioned matrices.

If the above treatment was all we could achieve, the speed improvement would be rather modest except for high-order fits. However, we can hope to reduce the calculation time further if we can make use of earlier calculations as the filter slides along the data. If we consider the smoothing of two consecutive points in the spectrum with an N point filter, then the calculations for the second point involve $N - 1$ of the same points as the first—considerable duplication of calculation is involved—albeit in a complex way. We now turn our attention to making use of this coherence in the smoothing process.

The key to the problem is to find a way in which, as the filter slides along the spectrum, the weight factors of eq 10 can be calculated by using as much information from previous points as possible. The difficulty is that eqs 5 and 10–12, scramble the contributions of the various data points in a complex manner; hence the contribution of a single point to the γ, B, and C coefficients is difficult to unravel. We do better to recast the polynomials, $P_k(x)$, in their power series form

$$P_k(x) = \sum_{r=0}^{k} A^r_k x^r \tag{13}$$

The sth derivative is also now easy to calculate

$$P^s_k(x) = \sum_{r=s}^{k} \frac{r!}{(r-s)!} A^r_k x^{r-s} \tag{13a}$$

The coefficients of x can be generated by the simple recursion relationship

$$A^r_k = A^{r-1}_{k-1} - B_{k-1}A^r_{k-1} - C_{k-1}A^r_{k-2}$$
$$\text{with } A^0_0 = 1 \quad \text{and} \quad A^r_k = 0\{r \gtrless k\ 0 \tag{14}$$

which for the first three polynomials yields

$$A^0_0 = 1$$
$$A^1_1 = 1 \qquad A^0_1 = -B_0$$
$$A^2_2 = 1 \qquad A^1_2 = -B_0 - B_1 \qquad A^0_2 = B_1B_0 - C_1 \tag{15}$$

these can be seen to be identical with those obtained by expanding the polynomials given after eq 12. The major advantage of this form of the polynomials is that it allows us to evaluate γ_k, B_k, and C_k in a much more useful way. By expanding one of the $P_k(x)$ in eq 5 and noting that the remaining $P_k(x)$ is orthogonal to any polynomial of order $k - 1$, we have

$$\gamma_k = \sum_{i=1}^{N} x^k P_k(x_i) = \sum_{i=1}^{N} x^k \sum_{r=0}^{k} A^r_k x_i^r$$
$$\gamma_k = \sum_{r=0}^{k} A^r_k (\sum_{i=1}^{N} x_i^{r+k}) \tag{16}$$

Similarly, the expression for B_k becomes

$$B_k = \frac{1}{\gamma_k}\sum_{r=0}^{k} A^r_k (\sum_{i=1}^{N} x_i^{r+k+1}) + A^{k-1}_k \tag{17}$$

The important factor here is that now $P_k(x)$, γ_k, and B_k can be calculated simply from terms involving sums of the x_i to various powers. Such sums are ideal for simplifying the smoothing calculation of successive data points. For instance, if we have calculated $\sum x_i^3$ for a set of data points x_i, $i = 1...N$. When we move along the spectrum to smooth the next point we simply subtract x_1^3 and add x_{N+1}^3. We can easily maintain a sliding set of sums in this manner. Smoothing to polynomial order n requires that we calculate sums to x^{2n+1}.

Table I gives expressions for $P_k(x)$, γ_k, and B_k and the smoothing weights, $h^k_{t,i}$, for k = 0–2. These are not required for calculation since eqs 13, 14, 16, and 17 are completely general for all orders. However, for small k, using the explicit form can improve computational speed.

Calculation of the weights, eq 10, for the special case of smoothing when the order, n, is large can be performed more

quickly by utilizing the *Christoffel–Darboux identity*

$$\sum_{k=0}^{n} \frac{P_k(x_i)P_k(x_t)}{\gamma_k} = \frac{P_{n+1}(x_i)P_n(x_t) - P_n(x_i)P_{n+1}(x_t)}{\gamma_n(x_i - x_t)} \quad i \neq t$$
$$= \frac{P'_{n+1}(x_i)P_n(x_t) - P'_n(x_i)P_{n+1}(x_t)}{\gamma_n} \quad i = t \tag{18}$$

This closed form for the summation requires $P_{n+1}(x)$; however the calculation of the polynomial coefficients, eq 14, does not need any quantity which has not already been calculated for order n and is thus quite efficient.

(b) Implementation. The implementation of the smoothing/differentiation algorithm is fairly straightforward. As the filter slides along the spectrum we utilize a local copy of the N $(2m + 1)$ points that are currently being fitted. Also calculated are the powers of x_i up to x^{2n+1} for each x_i. The sums of the powers of x_i are also maintained.

$$\begin{bmatrix} x_1 & x_2 & \cdots & x_N \\ x_1^2 & x_2^2 & \cdots & x_N^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{2n+1} & x_2^{2n+1} & \cdots & x_N^{2n+1} \end{bmatrix} \begin{matrix} \sum_{i=1}^{N} x_i \\ \sum_{i=1}^{N} x_i^2 \\ \vdots \\ \sum_{i=1}^{N} x_i^{2n+1} \end{matrix} \tag{19}$$

This local block is all that is required to perform the weight calculations. The order of the calculation for each order k is

- Calculate the A_k^r from eq 14.
- Calculate the γ_k and B_k from eqs 16 and 17 and C_k from eq 12.
- Calculate the $h_{t,i}^{k,s}$ from eqs 10 and 13/13a.
- When $k = n$, calculate the smoothed value from eq 9.

For the first N points of the spectrum the local copy is used to calculate weights and smoothed/differentiated values for the first $m + 1$ positions ($t = 1...m + 1$). For the last N points the local copy is used to calculate weights and smoothed/differentiated values for the last $m + 1$ positions ($t = m + 1...N$). At spectrum points in between it is used to calculate weights and the smoothed/differentiated value for $t = m + 1$ only (the middle point). When moving from one spectrum point to the next, the first column values of eq 19 are subtracted from their associated sums. The local data are then shuffled one place to the left and the last vacated column is filled with values for the new end point. The values from this column are then added to the accumulated sums. The local data are now ready for the next calculation.

This algorithm provides a simple way to slide the filter along the spectrum while carrying considerable calculation forward each time. A general subroutine to perform these calculations for any order, filter length, and derivative at all spectral positions is available from the author. Figure 1 shows the results of using this routine to smooth nonuniform data in which the x spacing increases by a factor of 5 across the spectrum. The dashed line connects the smoothed points produced by an 11-point cubic filter. If only special cases are required (e.g. quadratic, first derivative) then explicit expressions for the γ_k, B_k, C_k, and weights can be coded to improve speed still further.

The increase in speed over conventional least-squares techniques is difficult to quantify exactly, since it depends on the order of the equations, the method used to solve the least-squares normal equations, and the spectrum length. In various trial tests with realistic spectra the speed improvement was usually about 1 order of magnitude. This factor increases as the order increases—due to the increased time spent solving the least-squares normal equations by traditional methods.

LITERATURE CITED

(1) Savitzky, A.; Golay, M. J. E. *Anal. Chem.* **1964**, *36*, 1627–1639.
(2) Gorry, P. A. *Anal. Chem.* **1990**, *62*, 570–573.
(3) Bevington, P. R. *Data Reduction and Error Analysis for the Physical Sciences*; McGraw-Hill Book Co.: New York, 1969; Chapter 8.
(4) Ralston, A. *A First Course in Numerical Analysis*; McGraw-Hill Book Co.: New York, 1965; Chapter 6.
(5) Hildebrand, F. B. *Introduction to Numerical Analysis*; McGraw-Hill Book Co.: New York, 1974; Chapter 7.

RECEIVED for review October 1, 1990. Accepted November 19, 1990.