Moreover, we cannot derive the bias level δ from theory. In such cases we can estimate σ and especially δ empirically by the analysis of reference materials with certified concentrations of an analyte x_c. Then

$$\delta\ (\%) = \frac{|\bar{x}_e - x_c|}{x_c} \times 100 \qquad (6)$$

where $\bar{x}_e$ is an average result of the analysis of an RM (see details in ref 7).

We can expect that information theory can be successfully used in the optimization and comparison of various analytical methods and procedures, in the evaluation of the results obtained in collaborative tests for the certification of new RMs, and in many other applications.

LITERATURE CITED

(1) Eckschlager, K.; Štěpánek, V. *Anal. Chem.* **1982**, *54*, 1115A-1127A.
(2) Eckschlager, K.; Štěpánek, V. *Information Theory as Applied to Chemical Analysis*; Wiley: New York, 1979.
(3) Eckschlager, K.; Štěpánek, V. *Analytical Measurement and Information*; Research Studies Press: Letchworth, Great Britain, 1985.
(4) Eckschlager, K.; Štěpánek, V. *Chemom. Intell. Lab. Syst.* **1987**, *1*, 273-284.
(5) Obrusník, I.; Eckschlager, K. *J. Radioanal. Nucl. Chem., Articles* **1987**, *112*, 233-242.
(6) Eckschlager, K.; Štěpánek, V. *Collect. Czech. Chem. Commun.* **1985**, *50*, 1359-1367.
(7) Eckschlager, K.; Fusek, J. *Collect. Czech. Chem. Commun.* **1988**, *53*, 3021-3028.
(8) Lindstrom, R. M. *Nuclear Analytical Methods in Standards Certification*; IAEA-TECDOC-435; IAEA: Vienna, 1987.
(9) Becker, D. Analytical Design in Activation Analysis: The Role of Accuracy and Precision. In *Accuracy in Trace Analysis: Sampling, Sample Handling, and Analysis*; NBS Special Publication 422; NBS: Washington, DC, 1976; pp 1143-1155.
(10) De Bruin, M. Gamma-Ray Spectroscopy in Neutron Activation Analysis. In *Quality Assurance in Biomedical Neutron Activation Analysis*; IAEA-TECDOC-323; IAEA: Vienna, 1984; pp 163-177.
(11) Greenberg, R. R. *J. Radioanal. Nucl. Chem., Articles* **1987**, *113*, 233-247.
(12) Heft, R. E. *Control of Sample Configuration as an Aid to Accuracy in INAA*; NBS Special Publication 422; NBS: Gaithersburg, MD, 1976; pp 1275-1281.
(13) Ozmutlu, C.; Ortaovali, A. Z. *Nucl. Instrum. Methods* **1976**, *133*, 149-155.
(14) Eckschlager, K. *Anal. Chem.* **1977**, *49*, 1265-1267.
(15) Fleming, R. F.; Lindstrom, R. M. *J. Radioanal. Nucl. Chem., Articles* **1987**, *113*, 35-42.

RECEIVED for review August 7, 1989. Accepted November 27, 1989.

General Least-Squares Smoothing and Differentiation by the Convolution (Savitzky–Golay) Method

Peter A. Gorry
Department of Chemistry, University of Manchester, Manchester, England M13 9PL

Smoothing and differentiation of large data sets by piecewise least-squares polynomial fitting are now widely used techniques. The calculation speed is very greatly enhanced if a convolution formalism is used to perform the calculations. Previously tables of convolution weights for the center-point least-squares evaluation of $2m + 1$ points have been presented. A major drawback of the technique is that the end points of the data sets are lost ($2m$ points for a $2m + 1$ point filter). Convolution weights have also been presented in the special case of initial-point values. In this paper a simple general procedure for calculating the convolution weights at all positions, for all polynomial orders, all filter lengths, and any derivative is presented. The method, based on the recursive properties of Gram polynomials, enables the convolution technique to be extended to cover all points in the spectrum.

INTRODUCTION

In 1964 Savitzky and Golay (*1*) provided a simplified method for calculating smoothing and differentiation of data by a least-squares technique. Since then the Savitzky–Golay approach has been widely used because it produces a significant improvement in computational speed—replacing the traditional lengthy least-squares calculation by a simple, but equivalent, convolution. In this approach the least-squares value of a given point is calculated as a weighted combination of itself and m points on either side of it. This corresponds to performing a moving ($2m + 1$ point) least-squares fit across the data. The purpose of the original work was to provide a methodology for calculating the required weights and to provide tables of commonly used values. The original tables of convolution weights contained several errors however and were subsequently corrected by Steinier et al. (*2*) who recast the calculation in a matrix form.

The Savitzky–Golay approach suffers from one major drawback: it truncates the data by m points at each end. This occurs because the convolution requires m points to the left and right of a point in order to calculate the required least-squares value. The problem of data truncation is compounded if repeated smoothing/differentiation is applied since each application removes a further $2m$ points. For large spectra with zero values at the ends this is not important, and indeed zero values can easily be reinserted. However for more limited data sets without base-line values at both ends, the Savitzky–Golay algorithm is not applicable.

In principle the above limitation is easy to overcome. The first $2m + 1$ point least-squares fit can simply be used to evaluate least-squares values for the first m points as well as the $m + 1$. Similarly the last $2m + 1$ point fit can be used to calculate the last m least-squares values. Indeed if a normal least-squares calculation has been performed, yielding the least-squares coefficients, evaluating the fit of any of the m values, rather than the middle, is a simple task. This approach has been adopted by Khan (*3*) in a matrix formulation of the full least-squares method. Unfortunately we now lose the main advantage of the convolution method—its greatly enhanced speed.

Extending the convolution approach to accommodate "end points" requires calculating a different set of weights for each position. The special case of initial-point smoothing and slope evaluation has been considered by Leach, Carter, and Harris (*4*) where the Savitzky–Golay approach was modified to provide quadratic convolution weights for smoothing and the first derivative for the initial point. The tables were presented as real numbers and were later calculated to higher accuracy by Baedecker (*5*).

It would be very desirable to have a means of extending the convolution approach to performing least-squares calculations for any general position. We present here a simple method of calculating the least-squares convolution weights required for a general order polynomial fit, and all its derivatives, at all positions.

METHOD

(a) The General Problem. First we must define the problem more fully. We assume a spectrum containing p evenly spaced data points, $\{y_i\}$, which we wish to smooth, or differentiate (to order s), with a $2m + 1$ point filter and polynomial of order n. Since this requires repetitively fitting a polynomial of order n to $2m + 1$ consecutive points, we convert each group of points to a temporary coordinate system (*1, 2*) in which the ordinate values range from $i = -m$ to $i = +m$, i.e. the midpoint is $i = 0$. The least-squares polynomial then has the form

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

Application of the least-squares criterion

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

leads to $n + 1$ simultaneous equations in the unknown coefficients b_k. The Savitzky–Golay approach evaluates eq 1 at $i = 0$ and hence only requires an expression for b_0. Equally, differentiation of eq 1 reveals that the sth derivative (evaluated at $i = 0$) requires an expression for b_s only. This allows eq 1 to be reduced to an expression of the form (*1, 2*)

$$f_n^s(0) = \sum_{i=-m}^{m} h_i^s y_i \tag{3}$$

where the n and s denote the polynomial and derivative order, h_i^s is the convolution weight of the ith point, and y_i its value.

In the Savitzky–Golay algorithm a $2m + 1$ point smooth/differentiation requires one set (of $2m + 1$) weights to calculate the midpoint values. If we wish to calculate least-squares values at *any* of the $2m + 1$ positions, then we require $2m + 1$ *sets* of weights. Thus if we are to calculate the convolution weights required for all possible cases of orders, derivatives, and positions, it becomes impractical to produce general numerical tables of the weights and an alternative strategy must be pursued. The calculation techniques of Savitzky and Golay (*1*), Steinier et al. (*2*), and Harris et al. (*4*), based on the direct solution of the simultaneous equations, are complex and unsuited for such a generalization. Instead we turn to an entirely different approach to the problem.

(b) Gram Polynomials. It is well-known that the least-squares approximation to a function can equally well be cast in terms of a weighted expansion of discrete orthogonal polynomials, rather than a simple powers series (*6, 7*). Particularly suitable for this application are the Gram polynomials, and we have, analogous to eq 1

$$f_n(t) = \sum_{k=0}^{n} b_k P_k^m(t) \tag{4}$$

where $P_k^m(t)$ is the Gram polynomial of order k, over $2m + 1$ points, evaluated at point t. The Gram polynomials (*6*) are defined by

$$P_k^m(t) = \sum_{j=0}^{k} \frac{(-1)^{j+k}(j + k)^{(2j)}(m + t)^{(j)}}{(j!)^2(2m)^{(j)}} \tag{5}$$

where $(a)^{(b)}$ is a generalized factorial function: $(a)(a - 1) \ldots (a - b + 1)$, and $(a)^{(0)} = 1$. Substitution of eq 5 into eq 4 and application of the least-squares criterion eq 2 yields an expression of the form

$$f_n(t) = \sum_{i=-m}^{m}\sum_{k=0}^{n} \frac{(2k + 1)(2m)^{(k)}}{(2m + k + 1)^{(k+1)}} P_k^m(i) P_k^m(t) y_i \tag{6}$$

We can generalize eq 6 for the sth derivative very simply and thus produce an expression in exactly the form required for a convolution calculation

$$f_n^s(t) = \sum_{i=-m}^{m}\sum_{k=0}^{n} \frac{(2k + 1)(2m)^{(k)}}{(2m + k + 1)^{(k+1)}} P_k^m(i) P_k^{m,s}(t) y_i \tag{7}$$

where

$$P_k^{m,s}(t) = \left(\frac{d^s}{dx^s} P_k^m(x)\right)_{x=t}$$

Comparison of eq 3 and eq 7 provides the required expression for the convolution weight for data point i ($-m \leq i \leq m$), with polynomial order n, and sth derivative, evaluated at position t

$$h_i^{t,s} = \sum_{k=0}^{n} \frac{(2k + 1)(2m)^{(k)}}{(2m + k + 1)^{(k+1)}} P_k^m(i) P_k^{m,s}(t) \tag{8}$$

The tables of Savitzky and Golay are obtained from eq 8 by setting $t = 0$. Indeed this approach had been used previously by Ernst (*8*) to derive analytic expressions for the weights in the special case of smoothing ($s = 0$, $t = 0$) and small n ($n = 2$ and 4), corresponding to Tables I and II of Savitzky and Golay. The tables of Harris (*4*) and Baedecker (*5*) correspond to $t = -m$ and $s = 0$, $s = 1$.

In the case of differentials ($s > 0$) where the original data points are separated by Δx, rather than unity, we must modify eq 7 to take this into account (*1*)

$$f_n^s(t) = \frac{\sum_{i=-m}^{m} h_i^{t,s} y_i}{\Delta x^s} \tag{9}$$

The division by Δx^s is easily carried out after the convolution calculation.

Although eq 8 is a powerful way of expressing the weight values, the ordinary definition of the Gram polynomials, eq 5, is not very practical for calculation purposes, especially in the case of the derivatives. We thus need an alternative approach for actual calculation. Fortunately it is possible to derive a recursive relationship between the Gram polynomials

$$P_k^m(i) = \frac{2(2k - 1)}{k(2m - k + 1)} i P_{k-1}^m(i) - \frac{(k - 1)(2m + k)}{k(2m - k + 1)} P_{k-2}^m(i) \tag{10}$$

with $P_0^m(i) = 1$ and $P_{-1}^m(i) = 0$.

This provides an easy method of calculating the Gram polynomial values, but more importantly, it also provides a straightforward method of evaluating the general sth derivative, simply by differentiating eq 10

$$P_k^{m,s}(i) = \frac{2(2k - 1)}{k(2m - k + 1)}[iP_{k-1}^m(i) + sP_{k-1}^{m,s-1}(i)] - \frac{(k - 1)(2m + k)}{k(2m - k + 1)} P_{k-2}^{m,s}(i) \tag{11}$$

with $P_0^{m,s}(i) = 0$ and $P_{-1}^{m,s}(i) = 0$.

In fact eq 11 alone can be used to generate both the Gram polynomials ($s = 0$) and their derivatives ($s > 0$) providing account is taken of the different starting values in these two cases. Thus eq 8 and eq 11 together provide a simple and

Table I. Convolution Weights for Quadratic Smoothing and Differentiation

	5 pt, quadratic, smooth					7 pt, quadratic, smooth						
i	−2	−1	0	1	2	−3	−2	−1	0	1	2	3
−3						32	5	1	−2	−2	−1	5
−2	31	9	−3	−5	3	15	4	3	3	1	0	−3
−1	9	13	12	6	−5	3	3	4	6	3	1	−6
0	−3	12	17	12	−3	−4	2	4	7	4	2	−4
1	−5	6	12	13	9	−6	1	3	6	4	3	3
2	3	−5	−3	9	31	−3	0	1	3	3	4	15
3						5	−1	−2	−2	1	5	32
norm	35	35	35	35	35	42	14	14	21	14	14	42

	5 pt, quadratic, 1st deriv					7 pt, quadratic, 1st deriv						
i	−2	−1	0	1	2	−3	−2	−1	0	1	2	3
−3						−13	−29	−19	−3	1	11	7
−2	−54	−34	−2	6	26	−2	−6	−6	−2	−6	−6	−2
−1	13	3	−1	−17	−27	5	9	3	−1	−9	−15	−7
0	40	20	0	−20	−40	8	16	8	0	−8	−16	−8
1	27	17	1	−3	−13	7	15	9	1	−3	−9	−5
2	−26	−6	2	34	54	2	6	6	2	6	6	2
3						−7	−11	−1	3	19	29	13
norm	70	70	10	70	70	28	84	84	28	84	84	28

practical method for calculating the required weights.

(c) Implementation. We are now in a position to calculate the convolution weights by using the recursive definition of the Gram polynomials and their derivatives eq 11. A function to calculate the general convolution weight eq 8, as a real number, can be achieved in just a few lines of PASCAL code as shown in Figure 1. PASCAL was chosen since recursion is a standard feature of the language. However many modern implementations of more traditional languages (e.g. FORTRAN, BASIC) also now provide this facility and conversion is straightforward. The code in Figure 1 has been made to resemble the expressions in eq 8 and eq 11 as closely as possible with no attempt to optimize speed. If only low-order smoothing is required it is easy to avoid recursion simply by providing explicit expressions for the Gram polynomials (e.g. $P_0{}^m(i) = 1$, $P_1{}^m(i) = i/m$, $P_2{}^m(i) = (3i^2 - m(m+1))/(m(2m - 1))$ etc.).

```
function GramPoly(i,m,k,s : integer) : real;
{ Calculates the Gram Polynomial (s=0), or its s'th    }
{ derivative evaluated at i, order k, over 2m+1 points  }
 begin
  if k>0 then
   GramPoly := (4*k-2)/(k*(2*m-k+1))*(i *GramPoly(i,m,k-1,s) +
   s*GramPoly(i,m,k-1,s-1)) - ((k-1)*(2*m+k))/(k*(2*m-k+1))*GramPoly(i,m,k-2,s)
  else
   if (k=0) and (s=0) then GramPoly:=1 else GramPoly:=0;
 end;

 function GenFact(a,b: integer) : real;
{ Calculates the generalised factorial (a)(a-1)...(a-b+1) }
  var
   j : integer;
   gf : real;
 begin
  gf:=1; for j:=(a-b+1) to a do gf:=gf * j;  GenFact:=gf;
 end;

 function Weight(i,t,m,n,s : integer) : real;
{ Calculates the weight of the i'th data point for the t'th Least-Square  }
{ point of the s'th derviative, over 2m+1 points, order n .               }
  var
   k : integer;
   sum : real;
 begin
  sum:=0;
  for k:=0 to n do sum := sum + (2*k+1)*(GenFact(2*m,k)/GenFact(2*m+k+1,k+1))
            *GramPoly(i,m,k,0)*GramPoly(t,m,k,s);
  Weight:=sum;
 end;
```

Figure 1. Pascal code to perform the calculation of a general weight value $h_i^{t,s}$. Function *Weight* corresponds to eq 8, function *GramPoly* to eq 11, and function *GenFact* to the generalized factorial defined after eq 5.

A program to produce integer weights over a common normalizing factor requires the code to be modified to accumulate the numerator and denominator separately and to provide for common factor cancellations. Both versions of the program are available from the author.

Tables I and II give examples of the complete tables required for smoothing and calculating the first derivative for some small values of m and n. The column heading is the position at which the least-squares value is to be evaluated (t) and the row heading is the data point index (i). The center column ($t = 0$) values are the weights provided in the original work of Savitzky and Golay and Steinier et al. So, for instance, if we wish to calculate the least-squares fit to the first point ($t = -2$) in a five-point quadratic smooth, we have from Table I

$$\bar{Y}_{-2} = \frac{31y_{-2} + 9y_{-1} - 3y_0 - 5y_1 + 3y_2}{35} \qquad (12)$$

Equation 12 is an example of the weights provided by Harris

Table II. Convolution Weights for Cubic Smoothing and Differentiation

	5 pt, cubic, smooth					7 pt, cubic, smooth						
i	−2	−1	0	1	2	−3	−2	−1	0	1	2	3
−3						39	8	−4	−2	1	4	−2
−2	69	2	−3	2	−1	8	19	16	3	−4	−7	4
−1	4	27	12	−8	4	−4	16	19	6	2	−4	1
0	−6	12	17	12	−6	−4	6	12	7	12	6	−4
1	4	−8	12	27	4	1	−4	2	6	19	16	−4
2	−1	2	−3	2	69	4	−7	−4	3	16	19	8
3						−2	4	1	−2	−4	8	39
norm	70	35	35	35	70	42	42	42	21	42	42	42

	5 pt, cubic, 1st deriv					7 pt, cubic, 1st deriv						
i	−2	−1	0	1	2	−3	−2	−1	0	1	2	3
−3						−257	−122	−29	22	31	−2	−77
−2	−125	−19	1	5	−29	122	17	−46	−67	−46	17	122
−1	136	−1	−8	−13	88	185	62	−19	−58	−55	−10	77
0	48	12	0	−12	−48	72	48	24	0	−24	−48	−72
1	−88	13	8	1	−136	−77	10	55	58	19	−62	−185
2	29	−5	−1	19	125	−122	−17	46	67	46	−17	−122
3						77	2	−31	−22	29	122	257
norm	84	42	12	42	84	252	252	252	252	252	252	252

Table III. Convolution Weights for Quadratic Initial-Point Smoothing: Polynomial Order = 2, Derivative = 0

	h (2*m* + 1)								
i	21	19	17	15	13	11	9	7	5
-10	631								
-9	513	257							
-8	405	204	409						
-7	307	156	315	158					
-6	219	113	231	117	47				
-5	141	75	157	81	33	83			
-4	73	42	93	50	21	54	109		
-3	15	14	39	24	11	30	63	32	
-2	-33	-9	-5	3	3	11	27	15	31
-1	-71	-27	-39	-13	-3	-3	1	3	9
0	-99	-40	-63	-24	-7	-12	-15	-4	-3
1	-117	-48	-77	-30	-9	-16	-21	-6	-5
2	-125	-51	-81	-31	-9	-15	-17	-3	3
3	-123	-49	-75	-27	-7	-9	-3	5	
4	-111	-42	-59	-18	-3	2	21		
5	-89	-30	-33	-4	3	18			
6	-57	-13	3	15	11				
7	-15	9	49	39					
8	37	36	105						
9	99	68							
10	171								
norm	1771	665	969	340	91	143	165	42	35

Table IV. Convolution Weights for Quadratic Initial-Point First Derivative: Polynomial Order = 2, Derivative = 1

	h (2*m* + 1)								
i	21	19	17	15	13	11	9	7	5
-10	-23 370								
-9	-17 233	-5661							
-8	-11 696	-4012	-792						
-7	-6 759	-2543	-533	-7917					
-6	-2 422	-1254	-306	-4966	-330				
-5	1 315	-145	-111	-2435	-187	-945			
-4	4 452	784	52	-324	-68	-456	-1428		
-3	6 989	1533	183	1367	27	-67	-511	-13	
-2	8 926	2102	282	2638	98	222	166	-2	-54
-1	10 263	2491	349	3489	145	411	603	5	13
0	11 000	2700	384	3920	168	500	800	8	40
1	11 137	2729	387	3931	167	489	757	7	27
2	10 674	2578	358	3522	142	378	474	2	-26
3	9 611	2247	297	2693	93	167	-49	-7	
4	7 948	1736	204	1444	20	-144	-812		
5	5 685	1045	79	-225	-77	-555			
6	2 822	174	-78	-2314	-198				
7	-641	-877	-267	-4823					
8	-4 704	-2108	-488						
9	-9 367	-3519							
10	-14 630								
norm	336 490	67830	7752	61880	2002	4290	4620	28	70

(*4*) and Baedecker (*5*)—but in integer format—so that they can be evaluated to the full accuracy of the computer. Since these initial-point values are so important for initial slope evaluations (*4*, *5*, *9*) we present their integer values in Tables III and IV.

Such tables can be used to write programs to perform the smoothing/differentiation for particular cases. However, the ease of calculation of weights from eq 8 and eq 11 means that they can be generated "in situ" to provide a completely general smoothing/differentiation routine for all *m*, *n*, and *s*. Such a routine first calculates a weight table, $\mathbf{h}[t,i]$, for all positions t ($-m \leq t \leq m$) and all filter points i ($-m \leq i \leq m$) for a given *m*, *n*, and *s* (see Table I for examples). The table is then used to evaluate the least-squares smooth/differentiation for every point in the spectrum. The first *m* points and the last *m* points are calculated by using the appropriate columns in **h** ($t = -m$ to -1, and $t = 1$ to m, respectively). The rest of the spectrum uses the center point weighting $\mathbf{h}[0,i]$. The advantage of constructing a weight table is that it only needs calculating once providing the *m*, *n*, and *s* values are not changed.

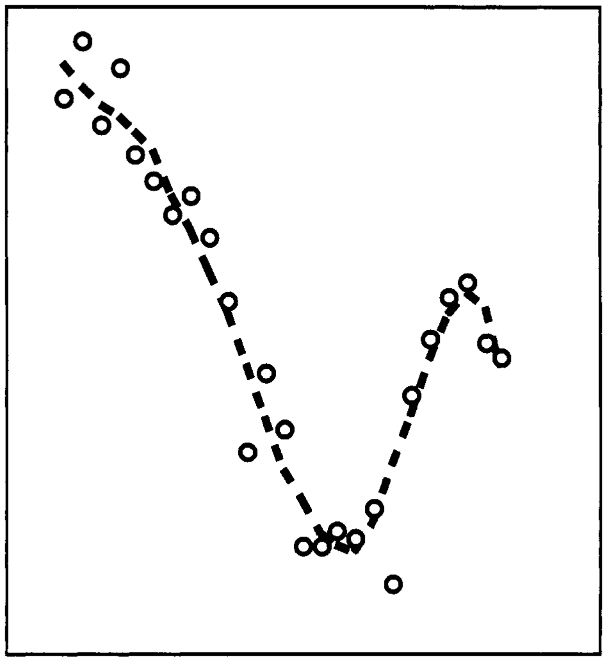
Figure 2. Least-squares smoothing using convolution weights for all points in the spectrum. The circles are the original data points. The dashed line joins the values from an 11 point cubic smooth. The correct smoothing of the end point values is clearly shown.

Figure 2 shows an example of smoothing (11 point cubic) using the algorithm described above. The data were chosen to provide an example of where the end-point information is particularly important. Whereas the traditional Savitzky–Golay smooth would have truncated the results by five points at each end, the current method correctly smooths the end points.

CONCLUSIONS

The convolution approach to least-squares smoothing and differentiation has been extended to remove the data truncation problem of the original Savitzky and Golay algorithm. A formalism based on the recursive properties of Gram polynomials provides a simple and direct means of calculating the convolution weights for all cases and thus enables a short but completely general routine to be written.

LITERATURE CITED

(1) Savitzky, A.; Golay, M. J. E. *Anal. Chem.* **1964**, *36*, 1627–1639.
(2) Steinier, J.; Termonia, Y.; Deltour, J. *Anal. Chem.* **1972**, *44*, 1906–1909.
(3) Khan, A. *Anal. Chem.* **1987**, *59*, 654–657.
(4) Leach, R. A.; Carter, C. A.; Harris, J. M. *Anal. Chem.* **1984**, *56*, 2304–2307.
(5) Baedecker, P. A. *Anal. Chem.* **1985**, *57*, 1477–1479.
(6) Ralston, A. *A first Course in Numerical Analysis*; McGraw-Hill Book Co.: New York, 1965; pp 235–254.
(7) Hamming, R. W. *Numerical Methods for Scientists and Engineers*; McGraw-Hill: New York, 1962; Chapter 24.
(8) Ernst, R. R. *Adv. Magn. Reson.* **1966**, *2*, 1.
(9) Khan, A. *Anal. Chem.* **1988**, *60*, 369–371.

RECEIVED for review September 22, 1989. Accepted November 28, 1989.