JOURNAL OF COMPUTATIONAL PHYSICS 52, 163–186 (1983)

An Algorithm for Experimental Data Deconvolution Using Spline Functions

PAUL DIERCKX

Department of Computer Science,
Katholieke Universiteit Leuven, Celestijnenlaan 200A, B–3030 Heverlee, Belgium

Received October 6, 1982

This paper presents an efficient numerical method for resolution correction, requiring no a priori knowledge. A spline function is determined such that its convolution with a spline approximation for the response function of the measuring device fits closely enough to the measured data. The number of knots of the spline and their positions are determined automatically. The algorithm expects a parameter to control the trade-off between the closeness of fit and the smoothness of spline approximation. Confidence limits for this smoothing factor are available if the statistical errors on the data points can be estimated.

1. INTRODUCTION

In many laboratory experiments, the finite resolution of the measuring device causes a non-negligible distortion of the original signal. The restoration of the signal is an important but also very difficult problem and it is not surprising, therefore, that the literature on this subject is nearly inexhaustible (for some recent publications see [12, 14] and the references therein). Mathematically, the deconvolution problem can be formulated as a linear integral equation of the first kind, i.e.,

$$y(x) = \int_{-\infty}^{\infty} r(u) f(x-u)\, du, \tag{1.1}$$

where $r(x)$ is the resolution (or response) function of the measuring instrument (known either from measurements or through theoretical calculation), $f(x)$ is the original signal and $y(x)$ is the distorted signal. It is known to be an ill-posed problem. Slight perturbations of $y(x)$ might correspond to arbitrarily large perturbations on the solution $f(x)$ due to the smoothing effect of integration. This is particularly troublesome since normally $y(x)$ will be measured only as a finite, discrete set of data points (x_q, y_q), $q = 1, 2, \ldots, m$, with inevitably random noise on the y_q-values. As a consequence, the deconvolution method will have to apply some regularization, i.e., some restriction on the class of approximating functions for $f(x)$, otherwise it will be doomed to failure.

The method presented in this paper will make use of spline functions. Suppose that

0021-9991/83 $3.00

we have a finite interval $[a, b]$ and a good spline approximation $R(x)$ for $r(x)$ on it such that replacing r by R and the infinite integration limits by a and b has little effect on (1.1). Then for an interval $[c, d]$ with

$$c \leqslant x_q - a, \qquad d \geqslant x_q - b \tag{1.2}$$

we will approximate $f(x)$ by a second spline function $s(x)$ such that

$$g(x_q) \cong y_q, \qquad q = 1, 2, \ldots, m, \tag{1.3}$$

with

$$g(x) = \int_{\alpha(x)}^{\beta(x)} R(u)\, s(x-u)\, du \tag{1.4}$$

and

$$\begin{aligned} \alpha(x) &= \max(a, x-d), \\ \beta(x) &= \min(b, x-c). \end{aligned} \tag{1.5}$$

For the determination of this spline we will adapt to our specific deconvolution problem the smoothing criterion for curve fitting described in [5] and [6]. The basic idea will be to define a measure for the smoothness of $s(x)$ as well as for the closeness of fit (1.3), and then to find a way to control, by a single parameter, the extent to which these two (very often contradictory) properties will be satisfied. The knots of $s(x)$ will be automatically determined.

2. Spline Functions and Their B-Spline Representation

Consider the strictly increasing sequence of real numbers

$$c = t_0 < t_1 < \cdots < t_n < t_{n+1} = d. \tag{2.1}$$

The function $s(x)$ is called a spline of degree l on $[c, d]$, with knots $t_j, j = 1, 2, \ldots, n$, if the following conditions are satisfied:

(i) In each interval $[t_j, t_{j+1}], j = 0, 1, \ldots, n$, $s(x)$ is given by some polynomial of degree l or less.

(ii) $s(x)$ and its derivatives of orders $1, 2, \ldots, l-1$ are continuous everywhere in $[c, d]$.

If we introduce the additional knots

$$\begin{aligned} t_{-l} = t_{-l+1} = \cdots = t_{-1} &= c, \\ d = t_{n+2} = \cdots = t_{n+l} &= t_{n+l+1}, \end{aligned} \tag{2.2}$$

then every such spline has a unique representation of the form

$$s(x) = \sum_{j=-l}^{n} c_j N_{j,l+1}(x) \tag{2.3}$$

in which $N_{j,l+1}(x)$ denotes the normalized B-spline, defined as

$$N_{j,l+1}(x) = (t_{j+l+1} - t_j)[t_j, t_{j+1},..., t_{j+l+1}](\cdot - x)_+^l \tag{2.4}$$

with

$$x_+^l = (\max\{0, x\})^l \tag{2.5}$$

and where $[z_j, z_{j+1},..., z_{j+k}]\, H$ stands for the kth divided difference for the function H in the points $z_j,..., z_{j+k}$, i.e.,

$$\begin{aligned} [z_j]\, H &= H(z_j), \\ [z_j, z_{j+1},..., z_{j+k}]\, H &= \frac{[z_{j+1},..., z_{j+k}]\, H - [z_j,..., z_{j+k-1}]\, H}{z_{j+k} - z_j}. \end{aligned} \tag{2.6}$$

Normalized B-splines enjoy the interesting property that

$$N_{j,l+1}(x) = 0 \qquad \text{if} \quad x \leqslant t_j \qquad \text{or} \qquad x \geqslant t_{j+l+1}, \tag{2.7}$$

and they can be evaluated in a very stable way using the recurrence scheme of de Boor [2] and Cox [4]. Finally, we remark that $s(x)$ becomes a single polynomial on $[c, d]$ if the discontinuities of its lth derivative at the interior knots t_q all vanish, i.e., if

$$\sum_{j=-l}^{n} b_{q,j} c_j = 0, \qquad q = 1, 2,..., n, \tag{2.8}$$

with

$$b_{q,j} = N_{j,l+1}^{(l)}(t_q + 0) - N_{j,l+1}^{(l)}(t_q - 0),$$

or [5],

$$\begin{aligned} b_{q,j} &= 0 && \text{if} \quad j < q - l - 1 \ \text{ or } \ j > q, \\ &= \frac{(-1)^{l+1}\, l!\,(t_{j+l+1} - t_j)}{\Pi'_{j,l}(t_q)} && \text{if} \quad q - l - 1 \leqslant j \leqslant q, \end{aligned} \tag{2.9}$$

where

$$\Pi_{j,l}(t) = (t - t_j)(t - t_{j+1}) \cdots (t - t_{j+l+1}) \tag{2.10}$$

and the prime denotes derivative with respect to t.

3. The Deconvolution Method

3.1. *The Deconvolution Criterion*

Given a spline approximation $R(x)$ for $r(x)$ and the set of data values (x_q, y_q) with weights w_q, $q = 1, 2, \ldots, m$ $(x_q < x_{q+1})$, we will determine a spline approximation $s(x)$ for $f(x)$, trying to find a compromise between the following objectives:

(i) The convolution (1.4) of $R(x)$ and $s(x)$ should fit closely to the prescribed values y_q.

(ii) The approximating spline $s(x)$ should be smooth, in the sense that the discontinuities in its lth derivative are as small as possible. This will be our regularization.

In order to formulate this criterion mathematically we introduce a measure of smoothness and a measure of closeness of fit. For the latter, the weighted sum of squared residuals

$$\delta(\bar{c}) = \sum_{q=1}^{m} w_q (y_q - g(x_q))^2 \tag{3.1}$$

can be taken. δ is indeed a function of the B-spline coefficients $\bar{c}$, as follows from (1.4) and (2.3), i.e.,

$$g(x) = \sum_{j=-l}^{n} c_j g_j(x) \tag{3.2}$$

with

$$g_j(x) = \int_{\alpha(x)}^{\beta(x)} R(u)\, N_{j,l+1}(x-u)\, du. \tag{3.3}$$

As a suitable smoothing norm, i.e., a measure of the lack of smoothness,

$$\eta(\bar{c}) = \sum_{q=1}^{n} \left(\sum_{j=-l}^{n} b_{q,j} c_j \right)^2 \tag{3.4}$$

is proposed.

The deconvolution criterion is formulated mathematically as follows:

$$\text{minimize } \eta(\bar{c}), \tag{3.5}$$

$$\text{subject to the constraint } \delta(\bar{c}) \leqslant S, \tag{3.6}$$

where S is a non-negative constant which must be supplied by the user to control the extent of smoothing and therefore is called the "smoothing factor."

3.2. *Solution of the Minimization Problem*

Problem (3.5)–(3.6) is closely related to and can be solved in the same way as the minimization problem posed in the curve fitting algorihm [6]. It is easily verified that, using the method of Lagrange multipliers, this one finally results in the computation of the coefficients c_j of a spline $s_p(x)$, defined for positive values of the Lagrange parameter p as the least-squares solution of the system

$$\begin{aligned} &\sqrt{w_q} \sum_{j=-l}^{n} c_j g_j(x_q) = \sqrt{w_q}\, y_q, \qquad q = 1, 2, \ldots, m, \\ &1/\sqrt{p} \sum_{j=-l}^{n} c_j b_{q,j} = 0, \qquad q = 1, 2, \ldots, n, \end{aligned} \tag{3.7}$$

when p is given the value of the positive root of $F(p) = S$ with

$$F(p) = \sum_{q=1}^{m} w_q \left[y_q - \int_{\alpha(x_q)}^{\beta(x_q)} R(u)\, s_p(x_q - u)\, du \right]^2. \tag{3.8}$$

The system of Eq. (3.7) can be written in the matrix form

$$Q\bar{c} = \bar{e} \tag{3.9}$$

with

$$Q = \begin{vmatrix} WG \\ \dfrac{1}{\sqrt{p}} B \end{vmatrix}, \qquad \bar{e} = \begin{vmatrix} W\bar{y} \\ \bar{0} \end{vmatrix},$$

$$W = \begin{vmatrix} \sqrt{w_1} & & & 0 \\ & \sqrt{w_2} & \ddots & \\ 0 & & & \sqrt{w_m} \end{vmatrix},$$

$$G = \begin{vmatrix} g_{-l}(x_1) & \cdots & g_n(x_1) \\ \cdots & \cdots & \cdots \\ g_{-l}(x_m) & \cdots & g_n(x_m) \end{vmatrix}, \qquad B = \begin{vmatrix} b_{1,-l} & \cdots & b_{1,n} \\ \cdots & \cdots & \cdots \\ b_{n,-l} & \cdots & b_{n,n} \end{vmatrix}.$$

In Section 3.3 we will give conditions for the convolution matrix G to have full rank. In that case there is an unique least-squares solution of (3.9) for every $p > 0$. It can be determined in a stable way using an orthogonalization method. We have implemented a method which uses Givens rotations without square roots [9]. This is very suitable for our problem because the equations are treated row by row. So if (3.7) has to be solved for different values of p, the transformations on the first m rows of Q will be carried out only once.

From (3.7) we can see that as p tends to infinity, $s_p(x)$ will tend to the least-squares spline $S_n(x)$, i.e., the spline with fixed knots t_j, $j = 1, 2, \ldots, n$, for which (3.1)

is minimal. Let $F_n(\infty)$ be the corresponding limit value of $F(p)$. Also from (2.8) and (3.7) it follows that as p tends to zero, $s_p(x)$ will tend to $P_l(x)$, the polynomial of degree l for which (3.1) is minimal. The corresponding limit value of F is denoted by $F(0)$.

Finally, we can prove (as in [5] or [6]) that F is a convex and strictly decreasing function of p. Therefore we know that, once we have found a set of knots such that

$$F_n(\infty) \leqslant S < F(0), \tag{3.10}$$

there exists an unique positive root $\bar{p}$ of $F(p) = S$. This $\bar{p}$ can quickly be found by means of an iterative scheme based on rational interpolation [6].

3.3. *On Calculating the Elements of the Convolution Matrix*

3.3.a. *Direct computation.* In this section we will show how the elements of the convolution matrix G, i.e.,

$$g_j(x_q) = \int_{\alpha(x_q)}^{\beta(x_q)} R(u)\, N_{j,l+1}(x_q - u)\, du \tag{3.11}$$

can be calculated in an accurate and efficient way.

First of all from (2.7) it follows that

$$g_j(x_q) = 0 \qquad \text{if} \quad x_q \leqslant a + t_j \ \text{ or } \ x_q \geqslant b + t_{j+l+1}, \tag{3.12}$$

and, consequently, that G will have full rank if and only if there is at least one subset of $(n + l + 1)$ strictly increasing x-values x_{v_j} such that

$$a + t_j < x_{v_j} < b + t_{j+l+1}, \qquad j = -l, -l+1, \ldots, n. \tag{3.13}$$

If $a + t_j < x_q < b + t_{j+l+1}$ the integration limits in (3.11) can be adjusted as follows:

$$\begin{aligned} \alpha(x_q) &= \max(a, x_q - t_{j+l+1}), \\ \beta(x_q) &= \min(b, x_q - t_j). \end{aligned} \tag{3.14}$$

Let $R(x)$ be a spline function of degree k, with knots $a = \tau_0 < \tau_1 < \cdots < \tau_{h+1} = b$. Then the function $R(u)\, N_{j,l+1}(x_q - u)$ is a piecewise polynomial of degree $k + l$ in u with knots at the points τ_i, $i = 0, 1, \ldots, h+1$, and $x_q - t_i$, $i = j, j+1, \ldots, j+l+1$. If we arrange these knots, say, θ_i, such that

$$\alpha(x_q) = \theta_1 < \theta_2 < \cdots < \theta_I = \beta(x_q) \tag{3.15}$$

then

$$g_j(x_q) = \sum_{i=2}^{I} \int_{\theta_{i-1}}^{\theta_i} R(u)\, N_{j,l+1}(x_q - u)\, du. \tag{3.16}$$

Each integral in this sum can be numerically integrated exactly using a Gauss–Legendre formula of order $(k+l+2)/2$.

3.3.b. *Recursive computation.* In spite of this simple and apparently economical procedure, the computation of G is the most time-consuming step of all the deconvolution algorithm. However, G can also be computed recursively. If we add a knot to the set t_j, $j=1, 2,..., n$, and denote the new set and all related parameters by an asterisk, such that

$$\begin{aligned} &t_j^* = t_j, \qquad j=-l, -l+1,..., i, \\ &t_{i+1}^* \in (t_i, t_{i+1}), \\ &t_{j+1}^* = t_j, \qquad j=i+1, i+2,..., n+l+1, \end{aligned} \tag{3.17}$$

then obviously

$$\begin{aligned} N_{j,l+1}(x) &= N_{j,l+1}^*(x), \qquad j=-l, -l+1,..., i-l-1, \\ &= N_{j+1,l+1}^*(x), \qquad j=i+1, i+2,..., n. \end{aligned} \tag{3.18}$$

Also, one can easily prove [1] that

$$N_{j,l+1}(x) = \frac{(t_{i+1}^* - t_j^*)}{(t_{j+l+1}^* - t_j^*)} N_{j,l+1}^*(x) + \frac{(t_{j+l+2}^* - t_{i+1}^*)}{(t_{j+l+2}^* - t_{j+1}^*)} N_{j+1,l+1}^*(x), \qquad j=i-l,..., i. \tag{3.19}$$

Indeed, from (2.4) and (2.6) it follows that

$$\begin{aligned} \frac{(t_{i+1}^* - t_j^*)}{(t_{j+l+1}^* - t_j^*)} N_{j,l+1}^*(x) &= (t_{i+1}^* - t_j^*)[t_j^*,..., t_i^*, t_{i+1}^*, t_{i+2}^*,..., t_{j+l+1}^*](\cdot - x)_+^l \\ &= [t_{j+1}^*,..., t_i^*, t_{i+1}^*, t_{i+2}^*,..., t_{j+l+1}^*](\cdot - x)_+^l \\ &\quad - [t_j^*,..., t_i^*, t_{i+2}^*,..., t_{j+l+1}^*](\cdot - x)_+^l . \end{aligned} \tag{3.20}$$

Likewise,

$$\begin{aligned} \frac{(t_{j+l+2}^* - t_{i+1}^*)}{(t_{j+l+2}^* - t_{j+1}^*)} N_{j+1,l+1}^*(x) &= [t_{j+1}^*,..., t_i^*, t_{i+2}^*,..., t_{j+l+2}^*](\cdot - x)_+^l \\ &\quad - [t_{j+1}^*,..., t_i^*, t_{i+1}^*, t_{i+2}^*,..., t_{j+l+1}^*](\cdot - x)_+^l . \end{aligned} \tag{3.21}$$

Adding (3.20) and (3.21), and using (2.6), (3.17) and (2.4), the right-hand side of (3.19) becomes

$$
\begin{aligned}
&[t^*_{j+1},\ldots,t^*_i,t^*_{i+2},\ldots,t^*_{j+l+2}](\cdot-x)^l_+ - [t^*_j,\ldots,t^*_i,t^*_{i+2},\ldots,t^*_{j+l+1}](\cdot-x)^l_+\\
&\quad=(t^*_{j+l+2}-t^*_j)[t^*_j,t^*_{j+1},\ldots,t^*_i,t^*_{i+2},\ldots,t^*_{j+l+1},t^*_{j+l+2}](\cdot-x)^l_+\\
&\quad=(t_{j+l+1}-t_j)[t_j,\ldots,t_i,t_{i+1},\ldots,t_{j+l+1}](\cdot-x)^l_+\\
&\quad=N_{j,l+1}(x). \qquad (3.22)
\end{aligned}
$$

Q.E.D.

It immediately follows from (3.11), (3.18) and (3.19) that

$$
\begin{aligned}
g_j(x_q)&=g^*_j(x_q), && j=-l,\ldots,i-l-1,\\
&=\frac{(t^*_{i+1}-t^*_j)}{(t^*_{j+l+1}-t^*_j)}g^*_j(x_q)+\frac{(t^*_{j+l+2}-t^*_{i+1})}{(t^*_{j+l+2}-t^*_{j+1})}g^*_{j+1}(x_q), && j=i-l,\ldots,i,\\
&=g^*_{j+1}(x_q), && j=i+1,\ldots,n. \qquad (3.23)
\end{aligned}
$$

This means that if we have at our disposal the convolution matrix G and want to compute G^*, only one new column must be calculated through integration (3.16), e.g., $g^*_{i-(l-1)/2}(x_q)$, $q=1,2,\ldots,m$; all the other columns are then easily obtained using (3.23). This saves a lot of time if the convolution matrix must be calculated for different sets of knots.

3.4. *The Strategy for Choosing the Knots*

Our strategy for choosing the knots will try to take account of the specific behaviour of the function $y(x)$ underlying the data. However, it will not necessarily find the minimum number of knots nor their optimal positions.

The chosen set must satisfy condition (3.10). It is easy to check that there exists at least one such set. If $n=m-l-1$ and if, in locating the knots, condition (3.13) is satisfied, then the least-squares spline $S_n(x)$ with these knots has $F_n(\infty)=0$. Considering computation time and memory requirements, we are interested in a spline approximation with fewer knots. Therefore we determine the least-squares polynomial $P_l(x)$ which is simply the least-squares spline $S_0(x)$. If $F_0(\infty)\leqslant S$ this polynomial is a solution of our problem. However, usually $F_0(\infty)>S$. In that case we determine a number of least-squares splines $S_{n_j}(x)$ with increasing number of knots, until $F_{n_j}(\infty)\leqslant S$. To determine the number of knots to be added at each iteration we use the same formulae as in the curve fitting algorithm [6]. We take into account the number of knots added the last time and the result of these extra knots (the reduction in the sum of squared residuals $F_{n_{j-1}}(\infty)-F_{n_j}(\infty)$) as compared to what we hope to get with the new addition (a reduction $F_{n_j}(\infty)-S$).

To locate the knots, we first spread in a more or less meaningful way $F_{n_j}(\infty)$ over the different knot intervals, hoping hereby to find those regions where the fit $S_{n_j}(x)$ is poor. We compute the quantities

$$\Delta_q = w_q \left(y_q - \int_{\alpha(x_q)}^{\beta(x_q)} R(u)\, S_{n_j}(x_q - u)\, du \right)^2, \qquad q = 1, 2, \ldots, m, \tag{3.24}$$

$$\xi_i = \sum_{q=1}^{m} \Delta_q \frac{|g_i(x_q)|}{\sum_{k=-l}^{n_j} |g_k(x_q)|}, \qquad i = -l, \ldots, n_j, \tag{3.25}$$

and

$$\delta_k = \sum_{i=-l}^{n_j} \xi_i \frac{\int_{t_k}^{t_{k+1}} N_{i,l+1}(x)\, dx}{\int_c^d N_{i,l+1}(x)\, dx}, \qquad k = 0, 1, \ldots, n_j, \tag{3.26}$$

i.e., we compute for each basis function $N_{i,l+1}(x)$ its contribution ξ_i to the total sum of squared residuals by using the elements of the convolution matrix as weighting factors. Then ξ_i is spread over those intervals where $N_{i,l+1}(x)$ is non-zero (positive), i.e., $[t_k, t_{k+1}]$, $k = i, i+1, \ldots, i+l$. As weighting factors here we use the integral value of the B-spline over the knot interval. Finally, note that the δ-values can easily be computed using the formulae of Gaffney [8] for integrating spline functions, i.e.,

$$\delta_k = \int_{t_k}^{t_{k+1}} \tilde{s}(x)\, dx \tag{3.27}$$

with

$$\tilde{s}(x) = \sum_{i=-l}^{n_j} d_i N_{i,l+1}(x) \tag{3.28}$$

and

$$d_i = \frac{(l+1)\,\xi_i}{(t_{i+l+1} - t_i)}. \tag{3.29}$$

The additional knots are located midway in the intervals $[t_k, t_{k+1}]$ with the largest δ_k-numbers. Before adding the knot, however, we check whether condition (3.13) is fulfilled. We also allow more than one knot to be located in the interval $[t_k, t_{k+1}]$. The new knot intervals also get δ-numbers, i.e., $\delta_k \cdot \delta_{k-1}/(\delta_{k-1} + \delta_{k+1})$ and $\delta_k \cdot \delta_{k+1}/(\delta_{k-1} + \delta_{k+1})$, respectively (or both $\delta_k/2$ if $k = 0$ or $k = n_j$). Finally, since the knots are only added and not relocated we know that the sum of squared residuals must decrease, i.e.,

$$F_{n_{j+1}}(\infty) \leqslant F_{n_j}(\infty), \qquad F_{m-l-1}(\infty) = 0. \tag{3.30}$$

Hence, it is also valid to use the recursive procedure of Section 3.3.b for calculating the elements of the convolution matrices.

4. Some Practical Considerations

The algorithm described in the previous sections has been implemented in a Fortran subroutine package called DECOSP [7].

All programs are written in Standard Fortran and have been checked with the PFORT verifier of Bell Labs. The package has been successfully implemented on an IBM 3033 and on a PDP 11/60 minicomputer. A magnetic tape copy of the package, together with an example program, can be obtained from the author.

Apart from $R(x)$, i.e., the spline approximation for the resolution function $r(x)$, the set of data points (x_q, y_q) with the corresponding weights w_q and the degree l of the requested spline $s(x)$, the user merely has to provide the smoothing factor S to control the trade-off between the "roughness" of the fit, as measured by the smoothing norm η (3.4), and the infidelity to the data, as measured by the weighted sum of squared residuals δ (3.1).

Recommended values of S depend on the relative weights w_q. If these are taken as $(\delta y_q)^{-2}$, with δy_q an estimate of the standard deviation of y_q, then a good S-value should be found in the range $m \pm \sqrt{2m}$ [13]. If nothing is known about the statistical error in y_q, each w_q can be set equal to one and S can be determined by trial and error. To decide whether an approximation corresponding to a certain S is satisfactory or not, the user should then examine the results graphically, i.e., by plotting $s(x)$ and by comparing $g(x)$ against the data values y_q. If S is too large the spline will be too smooth and signal will be lost (underfit); $g(x)$ will fit not closely enough to the y_q-values. If S is too small, too much noise will be picked up and $s(x)$ will reflect some unrealistic peaks and oscillations.

Note that an appropriate smoothing factor principally depends on the accuracy of the measured data values y_q and not on the signal itself (as contrasted, e.g., with an appropriate position for the knots of $s(x)$). Therefore, after having corrected a number of signals (possibly corresponding to an exactly known input) the user will certainly have a good idea in what range to choose the smoothing factor. The algorithm will then automatically adapt itself to other signals.

To economize the search for a good S-value DECOSP provides different modes of computation. At the first call of the routine or whenever he wants to restart with the initial set of knots (see Section 3.4), the user must set a parameter IOPT = 0. If IOPT = 1 the program will continue with the set of knots found at the last call to the routine. So, if the user wishes to call DECOSP repeatedly with decreasing S-values, he can save a lot of computation time by specifying IOPT = 1 from the second call on. Also, if he wants to calculate another spline corresponding to new data values y_q or weights w_q, he can start with the last found set of knots but then, in addition to IOPT = 1, he must reset a second parameter IFLAG = 0.

5. Numerical Results

The present method was applied to tests taken form [12, 14] and to variants of these. We report some of the results and illustrate how our deconvolution algorithm can be used efficiently.

In each example a known function $f(x)$ is convolved with a known resolution function $r(x)$. The resulting function $y(x)$ is sampled at different points x_q, $q = 1, 2,..., m$, and pseudo-random noise is added to simulate experimental data values y_q.

In the examples where $r(x)$ cannot be represented by a first-degree spline, we determine a cubic spline approximation $R(x)$ using a curve fitting algorithm (see, e.g., [3, 6]). Then the deconvolution algorithm can be applied. The computed spline $s(x)$ is plotted against the exact function $f(x)$. To indicate the quality of approximation we calculate the overall error [11, 14]:

$$\text{ERROR} = \left\{1/m \sum_{q=1}^{m} (s(x_q) - f(x_q))^2\right\}^{1/2} \Big/ \max\{|f(x_q)|, q = 1, 2,..., m\}. \tag{5.1}$$

5.1. *Example* 1

In the first example we check the influence of the smoothing factor S and illustrate how to iteratively find an appropriate value for this parameter.

We use the test data of Johnson [11, 14] which are listed in Table I. If $f(x)$ and $r(x)$ are both Gaussian, i.e.,

$$f(x) = \frac{1}{\sigma_f \sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma_f^2}\right)$$

and

$$r(x) = \frac{1}{\sigma_r \sqrt{2\pi}} \exp\left(\frac{-x^2}{2\sigma_r^2}\right),$$

then $y(x)$ is also Gaussian with a standard deviation $\sigma_y = \sqrt{\sigma_f^2 + \sigma_r^2}$. The data of Johnson were obtained for $\sigma_f = 1$ and $\sigma_r = 0.7$; $y(x)$ was sampled at the points $x_q = 0.2(q - 22)$, $q = 1, 2,..., 44$, and normally distributed noise with a standard deviation of 2% was added to the $y(x_q)$-values. In Fig. 1 the data points (x_q, y_q) are marked with crosses.

Before applying our deconvolution algorithm we must find a spline approximation $R(x)$ for $r(x)$. In order to reduce the computation time in our algorithm (see Section 3.3a), we seek a spline of low degree with not too many knots. Actually we determine a cubic ($k = 3$) spline approximation for the interval $[a, b] = [-3, 3]$ using a least-squares spline algorithm (nearly every software library contains such a program). The knots and B-spline coefficients of $R(x)$ are given in Table II. Then we seek a good cubic ($l = 3$) spline approximation $s(x)$ for $f(x)$ on the interval $[c, d] = [-4.4,$

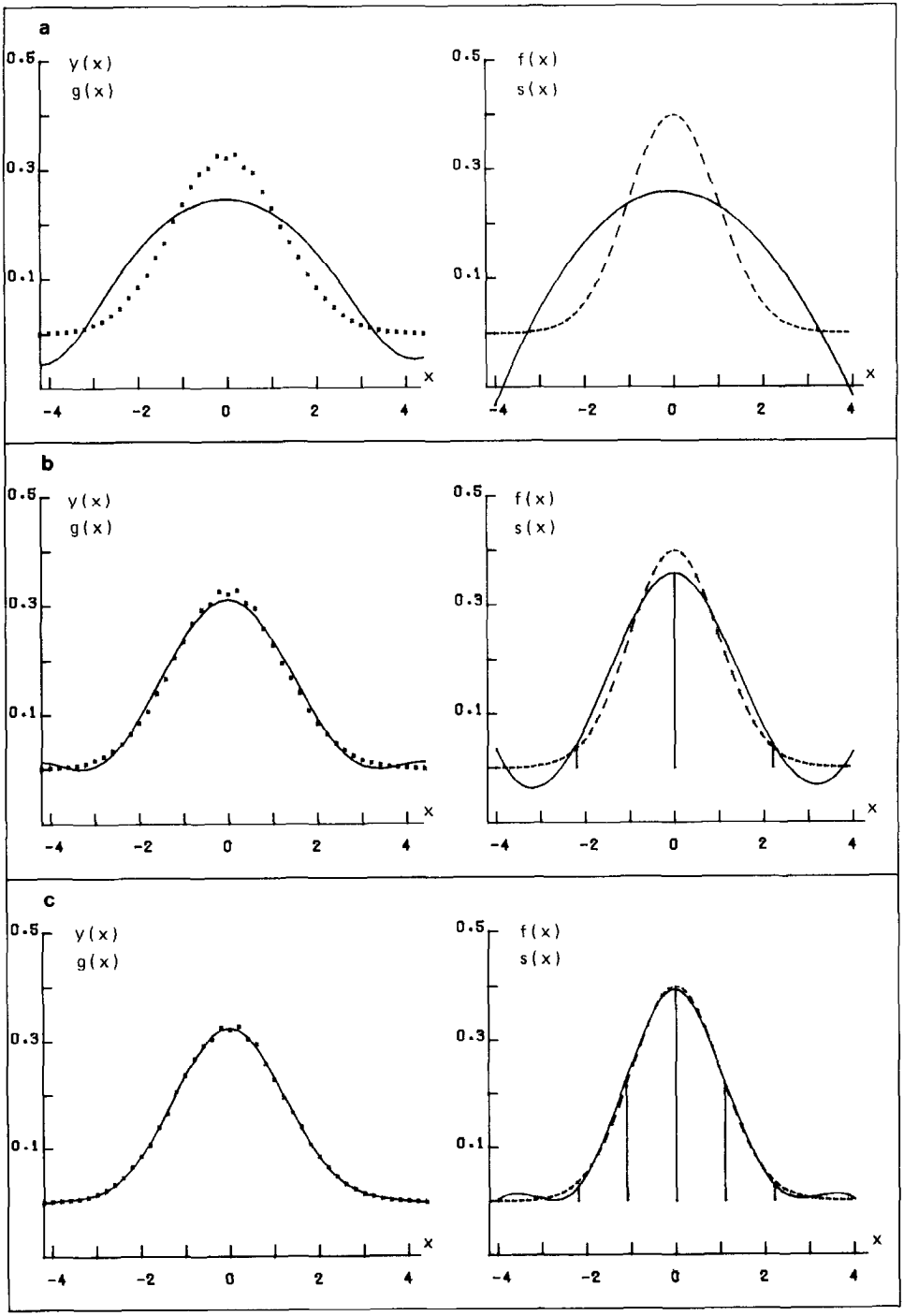

FIG. 1. Deconvolution of Johnson's data with weights $w_q = \delta^{-2}$, showing the influence of the smoothing factor: (a) $S \gg 0$, (b) $S = 500$, (c) $S = 44$, (d) $S = 38$, (e) $S = 34$, (f) $S = 38$.

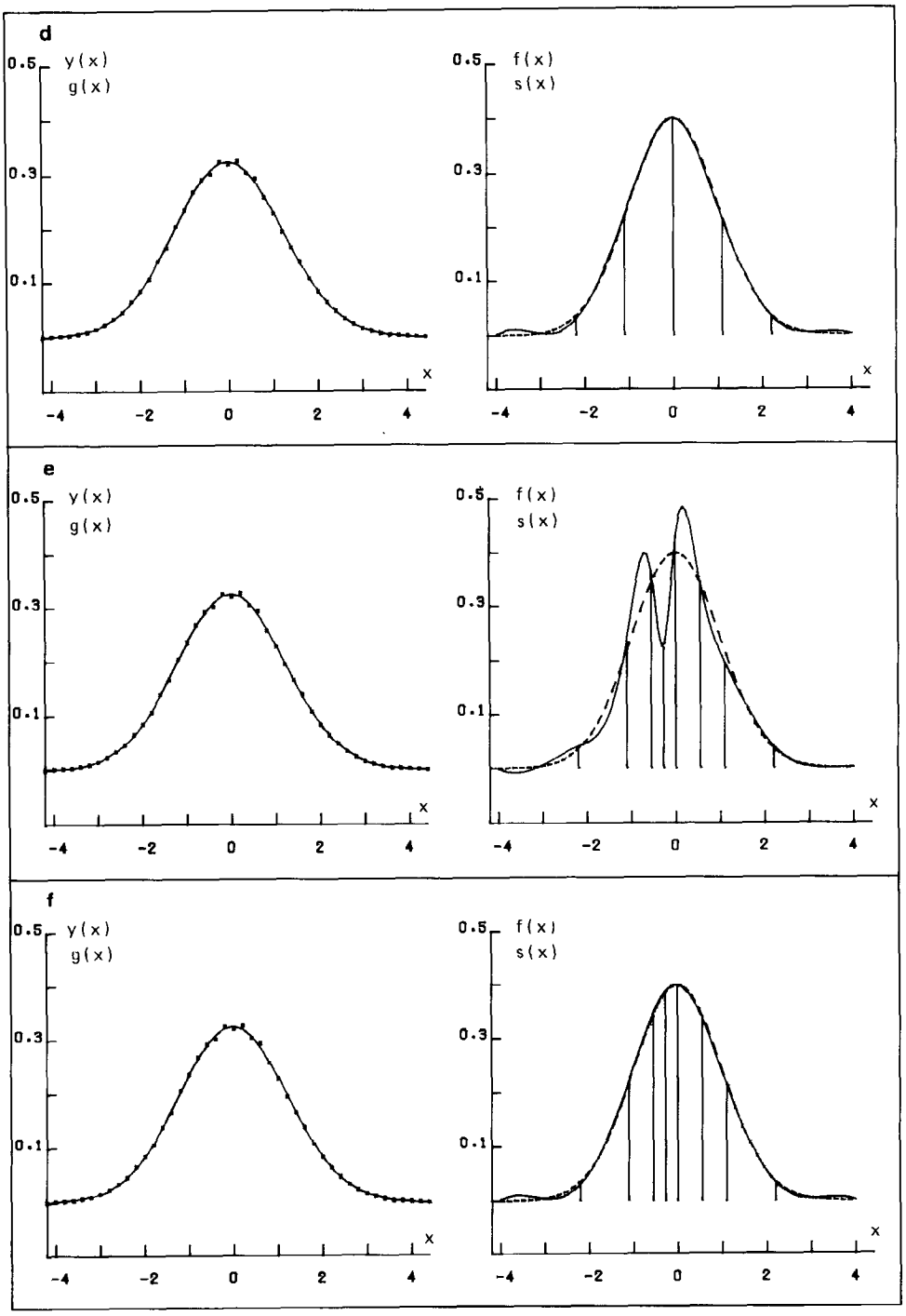

FIGURE 1 *(continued)*

TABLE I

Johnson's Test Data

q	x_q	y_q	q	x_q	y_q
1	−4.2	0.001	23	0.2	0.328
2	−4.0	0.002	24	0.4	0.305
3	−3.8	0.003	25	0.6	0.295
4	−3.6	0.004	26	0.8	0.259
5	−3.4	0.007	27	1.0	0.229
6	−3.2	0.010	28	1.2	0.196
7	−3.0	0.016	29	1.4	0.167
8	−2.8	0.023	30	1.6	0.140
9	−2.6	0.034	31	1.8	0.108
10	−2.4	0.046	32	2.0	0.084
11	−2.2	0.066	33	2.2	0.065
12	−2.0	0.085	34	2.4	0.048
13	−1.8	0.107	35	2.6	0.034
14	−1.6	0.140	36	2.8	0.024
15	−1.4	0.166	37	3.0	0.016
16	−1.2	0.206	38	3.2	0.011
17	−1.0	0.236	39	3.4	0.007
18	−0.8	0.269	40	3.6	0.004
19	−0.6	0.293	41	3.8	0.003
20	−0.4	0.303	42	4.0	0.002
21	−0.2	0.326	43	4.2	0.001
22	0.0	0.321	44	4.4	0.0

4.4] by using subroutine DECOSP. We shall call this routine repeatedly with decreasing S-values until a satisfactory result is obtained. But first, we determine the weights w_q. In order to check the confidence interval for S we calculate an estimate δ of the standard deviation of the y_q-values, i.e.,

$$e_q = y_q - y(x_q), \qquad q = 1, 2, \ldots, 44,$$

$$\bar{e} = \sum_{q=1}^{44} e_q/44,$$

$$\delta^2 = \sum_{q=1}^{44} (\bar{e} - e_q)^2/43,$$

and we set the weights $w_q = \delta^{-2}$. A good S-value should be found in the range $[m - \sqrt{2m},\ m + \sqrt{2m}] \cong [35, 53]$. However, we will not further use this information and make as if δ^{-2} is an arbitrary constant (this would then correspond to setting $w_q = 1$ and multiplying the succeeding smoothing factors S by δ^2).

If initially we call DECOSP with a very large smoothing factor S, the program returns the least-squares polynomial $P_3(x)$ (i.e., the least-squares spline $S_0(x)$ without

TABLE II

The Knots and B-Spline Coefficients of the Cubic Spline Approximation for $r(x) = \exp(-x^2/0.98)/(0.7\sqrt{2\pi})$, $-3 \leqslant x \leqslant 3$

i	Knots τ_i	B-spline coefficients d_i
−3	−3.0	0.000131
−2	−3.0	−0.000124
−1	−3.0	−0.000218
0	−3.0	0.022900
1	−2.2	0.191249
2	−1.4	0.382632
3	−1.0	0.533457
4	−0.6	0.588262
5	−0.3	0.533457
6	0.	0.382632
7	0.3	0.191249
8	0.6	0.022900
9	1.0	−0.000218
10	1.4	−0.000124
11	2.2	0.000131
12	3.0	
13	3.0	
14	3.0	
15	3.0	

interior knots) and the corresponding weighted sum of squared residuals $F(0)$. Figure 1a shows the result; on the right we see that $s(x) = P_3(x)$ (full line) is a poor approximation for $f(x)$ (dashed line). This is not surprising since $g(x)$, the convolution of $R(x)$ and $s(x)$, also poorly approximates $y(x)$.

So we then proceed by calling DECOSP with decreasing S-values less than $F(0)$ (e.g., $F(0)/10$, $F(0)/100$,...). Figure 1b shows the result according to $S = 500$. We still have oversmoothing; $s(x)$ now is a spline with $n = 3$ interior knots, the position of which is given by the vertical lines. According as $g(x)$ fits closer to the data values (x_q, y_q) we decrease the smoothing factor more carefully (e.g., $S = 250$, 100, 50,...). Figure 1c shows the result according to $S = m = 44$; $g(x)$ is a good approximation for $y(x)$ and the quality of $s(x)$ is acceptable. Decreasing the smoothing factor a bit more produces even better results. Figure 1d shows the approximation according to $S = 38$. This spline has the same knots ($n = 5$) as the spline of Fig. 1c but less smoothing is applied. The overall error (5.1) is 1.36% whereas Johnson [11] obtains 2.7% and Verkerk [14] 1.3%.

Now, if we still decrease S a little more we suddenly get "overfitting." The approximation of Fig. 1e, which corresponds to $S = 34$, shows some unrealistic wiggles and peaks, telling us that we have gone too far. From the second call of the routine we have used the mode of computation IOPT = 1 (see Section 4). So we always continue

with the set of knots found at the last call, saving a lot of computation time in this way. So if we finally calculate a new approximation $s(x)$ according to $S = 38$ we get a spline with the same knots ($n = 8$) as the one corresponding to $S = 34$. The result can be seen in Fig. 1f. The quality of fit is the same as for the spline in Fig. 1d. So, the fact that we can obtain a good approximation even if the number of knots is too high clearly demonstrates the regularization in our algorithm.

5.2. *Example* 2

In the preceding example all data points had the same weight. However, it would be better to give the points around the peak a relatively smaller weight, since statistical error is proportional to y_q. To check the influence of individual weighting, we use Johnson's data with $w_q = (y_q - y(x_q))^{-2}$. On the assumption that the spline $s(x)$ to be expected is such that $g(x_q) = y(x_q)$, $q = 1, 2, \ldots, m$, then from (3.8) we can compute the corresponding smoothing factor, i.e., $S = m = 44$. The results, for different degrees of spline approximation, are shown in Fig. 2. The overall error (5.1) of the cubic spline in Fig. 2a is only 0.7%.

5.3. *Example* 3

In their tests McKinnon *et al.* [12] simulate photoluminescence data by means of the resolution function $r(x) = 5.802x^2 \exp(-0.4x)$. Using a variant of the curve fitting algorithm in [6], we approximate this function on the interval $[0, 25]$ by a cubic spline $R(x)$ with boundary conditions $R(0) = R'(0) = 0$. The knots and B-spline coefficients of $R(x)$ are given in Table III.

Then we generate a set of normally distributed stochastic variates e_q (expected value 0, standard deviation 0.01) and consider the data

$$x_q = q/4, \qquad q = 1, 2, \ldots, 200,$$
$$y_q = \left(\int_0^{x_q} r(x) f(x_q - x)\, dx\right)(1 + e_q),$$

in order to find a cubic spline approximation for

$$f(x) = 0.1 \exp(-0.02x) + 0.01 \Big/ \left(\left(\frac{x - A}{12.5}\right)^2 + 0.05\right)$$

using subroutine DECOSP. Since the errors are proportional to the y_q we set $w_q = y_q^{-2}$. From (3.8) we find an estimate for the smoothing factor, i.e.,

$$S \cong \sum_{q=1}^{200} e_q^2/(1 + e_q)^2 \cong 0.02.$$

Figure 3 shows the results according to $S = 0.018$ and for $A = 12.5$ (Fig. 3a), resp. $A = 25$ (Fig. 3b). The overall error (5.1) is 0.77%, resp. 0.70%. This example illustrates that our algorithm is adaptive in placing the knots (see the vertical lines).

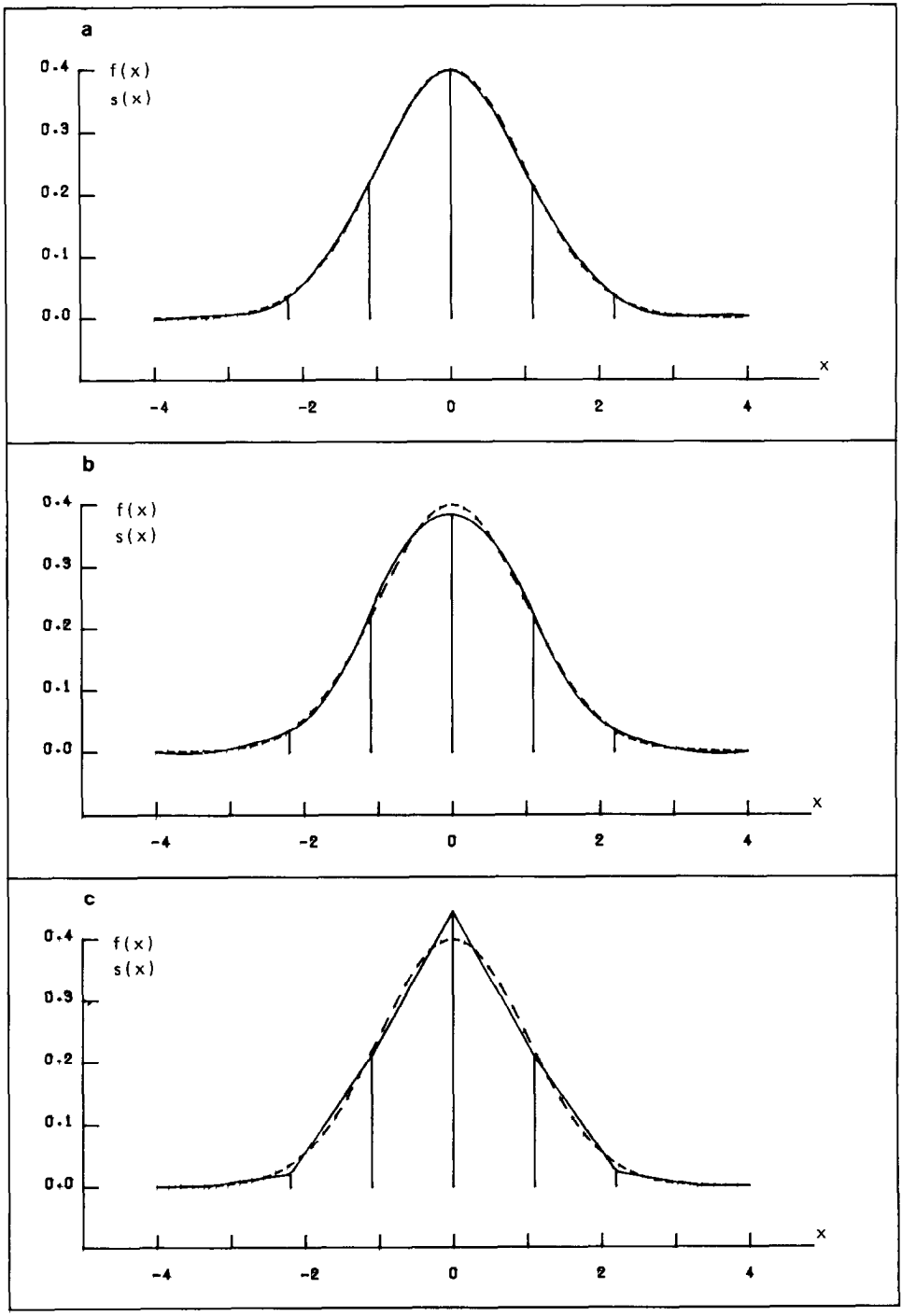

FIG. 2. Deconvolution of Johnson's data with weights $w_q = (y_q - y(x_q))^{-2}$ and using different degrees of spline approximation: (a) $l = 3$, (b) $l = 2$, (c) $l = 1$.

TABLE III

The Knots and B-Spline Coefficients of the Cubic Spline Approximation for $r(x) = 5.802x^2 \exp(-0.4x)$. $0 \leqslant x \leqslant 25$

i	Knots τ_i	B-spline coefficients d_i
−3	0.	0.
−2	0.	0.
−1	0.	3.01567
0	0.	10.6970
1	1.	17.6811
2	1.75	20.3386
3	3.25	18.7294
4	4.75	15.4066
5	6.25	10.4402
6	8.	3.61714
7	9.5	0.603071
8	12.5	0.289518
9	18.75	0.163039
10	25.	
11	25.	
12	25.	
13	25.	

In both cases there is a concentration near the peak. The concentration of knots and the oscillations of $s(x)$ near the origin can be somewhat explained by the excessively large weights for the data points in that region.

5.4. *Example* 4

In the next example we illustrate how DECOSP can be used efficiently in case several similar signals coming from the same device are to be deconvolved. We consider the response function $r(x) = \max\{0, (1 - |x|/\sigma_r)/\sigma_r)\}$ [14] with $\sigma_r = 0.1$. It can be represented as a first-degree spline with knots $\tau_{-1} = \tau_0 = -\sigma_r$, $\tau_1 = 0$, $\tau_2 = \tau_3 = \sigma_r$ and B-spline coefficients $d_{-1} = d_1 = 0$, $d_0 = 1/\sigma_r$. The different functions we want to restore on the interval $[-0.5, 0.5]$ are

$$f_i(x) = \sin(4\pi x + \alpha_i) + \cos(8\pi x)/2$$

with

$$\alpha_i = (i - 1)\,\pi/16, \qquad i = 1, 2, \ldots, 32.$$

Then from (1.1) it is easily verified that the distorted signals are

$$y_i(x) = \frac{2(1 - \cos 4\pi\sigma_r)}{(4\pi\sigma_r)^2} \sin(4\pi x + \alpha_i) + \frac{(1 - \cos 8\pi\sigma_r)}{(8\pi\sigma_r)^2} \cos(8\pi x).$$

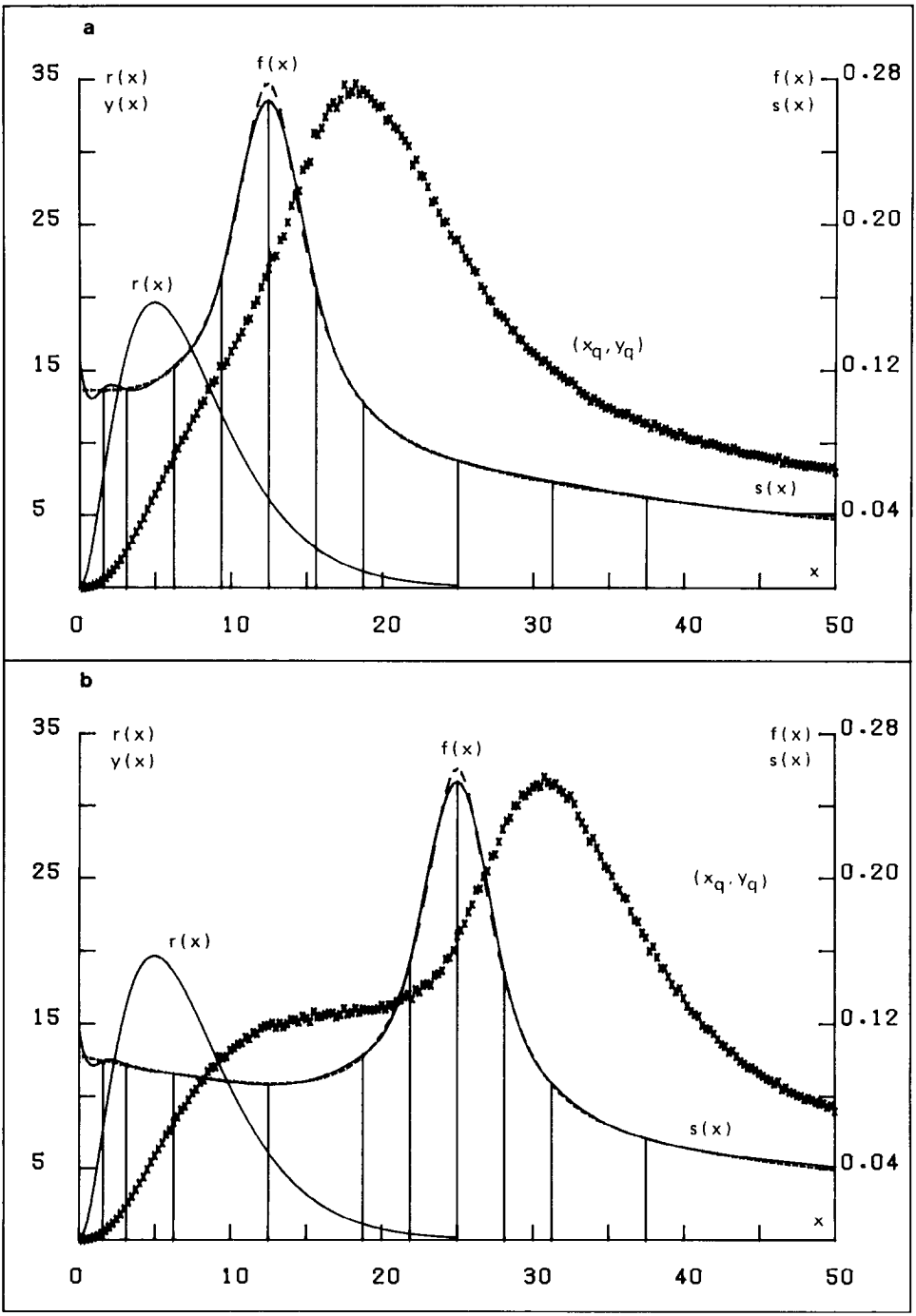

FIG. 3. Deconvolution of simulated data according to $f(x) = 0.1 \exp(-0.02x) + 0.01/(((x - A)/12.5)^2 + 0.05)$ and $r(x) = 5.802x^2 \exp(-0.4x)$: (a) $A = 12.5$, (b) $A = 25$.

The data points are obtained as follows. Using a random number generator we generate sets of normally stochastic variates $e_{q,i}$ with expected value 0 and standard deviation 0.01, and then for $i = 1, 2,..., 32$ we consider the data

$$x_q = (q - 51)/100.$$
$$y_{q,i} = y_i(x_q) + e_{q,i}.$$
$$w_q = (0.01)^{-2}, \qquad q = 1, 2,..., 101.$$

Although the spline approximations $s_i(x)$ will only be used in $[-0.5, 0.5]$ we set the approximation interval $[c, d] = [-0.5 - \sigma_r, 0.5 + \sigma_r]$. In this way no errors are introduced by breaking up the infinite integration interval in (1.3)–(1.4)–(1.5).

Further, we select seven values out of the confidence interval for the smoothing factor, i.e., $S_j = 120 - 5j$, $j = 1, 2,..., 7$. Then we determine successively spline approximations according to these decreasing smoothing factors and proceed with the next data set as soon as the corresponding overall error (5.1) drops below some preset value (actually we require an error of 1.35%). So, we simulate the action of a user who knows from experience in what range to choose the smoothing factor and then effectively tries some S-values until he can accept (through graphical inspection) the corresponding approximation as being satisfactory.

The results of our test are summarized in Table IV. For each data set we give the

TABLE IV

Deconvolution of Simulated Data According to $f(x) = \sin(4\pi x + \alpha_i) + 0.5\cos(8\pi x)$ and $r(x) = \max\{0, 10(1 - 10|x|)\}$ (ERROR and TIME results in brackets correspond to a smoothing factor $S = 115$)

i	α_i	S	ERROR (%)	Time (sec)	i	α_i	S	ERROR (%)	Time (sec)
1	0	90	1.18(1.70)	1.45(1.08)	17	π	90	1.30(1.63)	0.37(0.11)
2	$\pi/16$	100	1.30(1.48)	0.27(0.12)	18	$17\pi/16$	95	1.26(1.50)	0.32(0.11)
3	$\pi/8$	90	1.23(1.71)	0.35(0.12)	19	$9\pi/8$	100	1.25(1.50)	0.27(0.11)
4	$3\pi/16$	115	1.18(1.18)	0.10(0.10)	20	$19\pi/16$	115	1.29(1.29)	0.12(0.12)
5	$\pi/4$	100	1.15(1.57)	0.24(0.11)	21	$5\pi/4$	85	1.29(1.54)	0.49(0.13)
6	$5\pi/16$	85	1.28(1.79)	0.42(0.11)	22	$21\pi/16$	100	1.31(1.50)	0.27(0.11)
7	$3\pi/8$	85	1.32(1.71)	0.43(0.11)	23	$11\pi/8$	95	1.25(1.62)	0.33(0.13)
8	$7\pi/16$	110	1.29(1.38)	0.14(0.10)	24	$23\pi/16$	115	1.21(1.21)	0.11(0.11)
9	$\pi/2$	90	1.33(1.87)	0.35(0.10)	25	$3\pi/2$	90	1.23(1.62)	0.37(0.11)
10	$9\pi/16$	115	1.08(1.08)	0.11(0.11)	26	$25\pi/16$	115	1.29(1.29)	0.11(0.11)
11	$5\pi/8$	85	1.51(1.95)	0.42(0.11)	27	$13\pi/8$	95	1.30(1.54)	0.34(0.11)
12	$11\pi/16$	110	1.29(1.37)	0.17(0.11)	28	$27\pi/16$	100	1.27(1.46)	0.27(0.11)
13	$3\pi/4$	115	1.23(1.23)	0.10(0.10)	29	$7\pi/4$	105	1.30(1.40)	0.23(0.13)
14	$13\pi/16$	90	1.29(1.66)	0.38(0.13)	30	$29\pi/16$	90	1.32(1.64)	0.39(0.13)
15	$7\pi/8$	85	1.52(1.85)	0.47(0.13)	31	$15\pi/8$	115	1.23(1.23)	0.11(.011)
16	$15\pi/16$	85	1.67(1.93)	0.51(0.13)	32	$31\pi/16$	90	1.21(1.69)	0.36(0.12)

ultimate smoothing factor S_J, the corresponding overall error (5.1) and the total time needed on an IBM 3033 to determine the approximations according to $S_1, S_2, \ldots, S_J$. From the second call of the routine DECOSP we set IOPT = 1 (see Section 5.1). So we always continue with the set of knots found at the last call of the routine. In this example all selected spline approximations $s_i(x)$ have the same knots, i.e., $\{t_j \mid j = 0, 1, \ldots, n+1\} = \{-0.6, -0.45\ (0.075)\ 0.45, 0.6\}$ $(n = 13)$. Consequently, the corresponding convolution matrix has to be calculated only once, i.e., with the first data set. The gain in time for the following data sets clearly appears in Table IV. Each time we proceed with a new data set we must reset IFLAG = 0 (see Section 4). Since we do not store the Givens transformations and have a new right-hand side $\bar{e}$ (3.9) we must also retriangularize the convolution matrix. The computation time for this depends linearly on the number of data points. If we proceed with a new smoothing factor S_j, we keep IFLAG = 1. In that case the convolution matrix must not be retriangularized (see Section 3.2) and computation time now mainly depends on the number of B-spline coefficients and on the number of iterations to find the root of $F(p) = S_j$.

The total time for determining the 32 spline approximations (139 calls of DECOSP) is about 10.5 sec and the mean error is 1.28%. Some of the splines $s_i(x)$ are shown in Fig. 4. To simulate the use of DECOSP in a fully automatic way, we report in Table IV the ERROR and time results according to the same smoothing factor $S = 115$ for all data sets. (It is safer to choose the smoothing factor too large rather than too small.) The total time for 32 calls of DECOSP is about 4.5 sec and the mean error is 1.54%.

5.5. *Example* 5

In a last test taken from $[10, 14]$ $f(x)$ is a sum of two exponentials, i.e.,

$$f(x) = \exp\left(-\left(\frac{x+100}{75}\right)^2\right) + \exp\left(-\left(\frac{x-100}{75}\right)^2\right),$$

and the resolution function $r(x)$ which is rectangular, i.e.,

$$\begin{aligned} r(x) &= 1, \qquad |x| < 125, \\ &= 0, \qquad |x| \geqslant 125, \end{aligned}$$

can be represented as a first-degree spline with knots $\tau_{-1} = \tau_0 = -125$, $\tau_1 = \tau_2 = 125$ and B-spline coefficients $d_{-1} = d_0 = 1$. The distorted signal $y(x)$ is sampled at 250 points $x_q = 4(q - 125)$, $q = 1, 2, \ldots, 250$, and the data values y_q contain pseudo-random noise uniformly distributed on $[-5, 5]$. The weights are all set equal to one. Figure 5 shows the data points (x_q, y_q), the exact function $f(x)$ (dashed line) and the cubic spline approximation $s(x)$ according to a smoothing factor $S = 2110$ (full line). The overall error (5.1) is 2.8% whereas Hunt [10] obtains 2.1% and Verkerk [14] 3.1%.

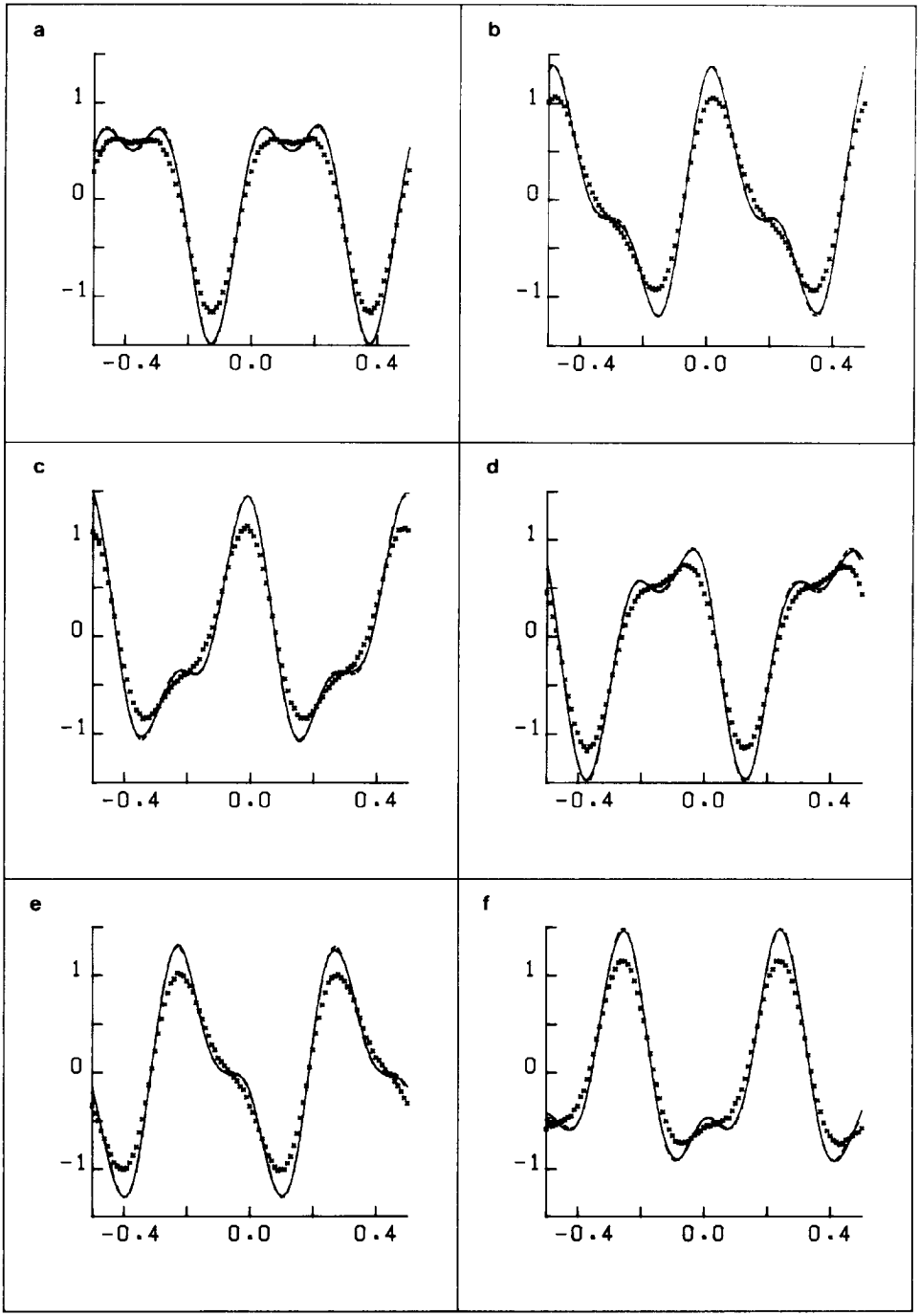

FIG. 4. Deconvolution of simulated data according to $f(x) = \sin(4\pi x + \alpha) + 0.5\cos(8\pi x)$ and $r(x) = \max\{0, 10(1 - 10|x|)\}$: (a) $\alpha = 0$, (b) $\alpha = 5\pi/16$, (c) $\alpha = 5\pi/8$, (d) $\alpha = 15\pi/16$, (e) $\alpha = 5\pi/4$, (f) $\alpha = 25\pi/16$.

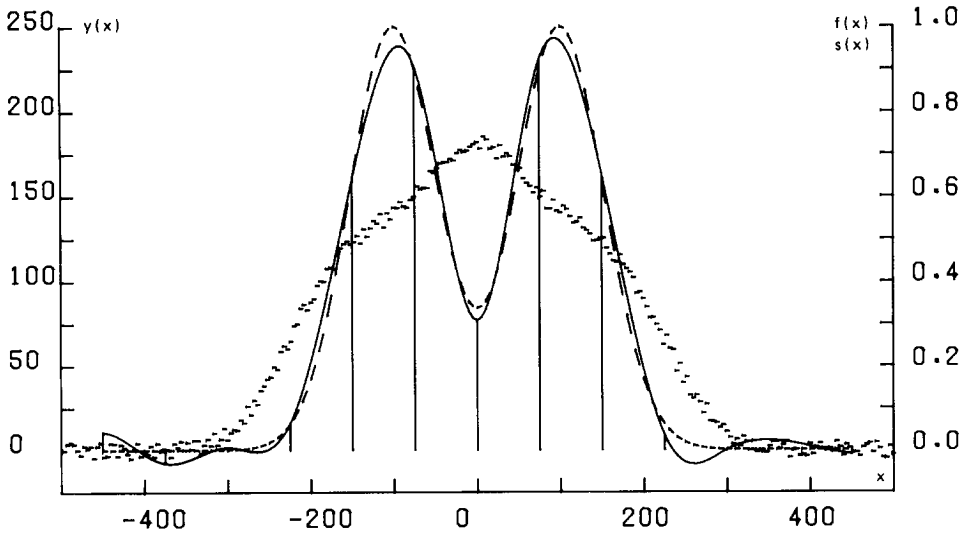

FIG. 5. Deconvolution of simulated data according to $f(x) = \exp(-((x + 100)/75)^2) + \exp(-((x - 100)/75)^2)$ and $r(x) = 1$, $|x| < 125$, $r(x) = 0$, $|x| \geqslant 125$.

6. CONCLUDING REMARKS

In the preceding sections we have described an algorithm for data deconvolution using spline functions. It is assumed that the response function of the measuring device is translation invariant and that it can be approximated properly by a spline $R(x)$. A second spline function $s(x)$ is then determined such that its convolution with $R(x)$ fits to the given data measurements. $R(x)$ and $s(x)$ are represented by B-splines. Hence, evaluation, differentiation and integration can be performed in a rapid and accurate way while using the recurrence schemes for evaluating [2, 4], differentiating [2] and integrating [8] B-splines.

The deconvolution method uses no a priori knowledge of the function to be aproximated and as opposed to methods which apply discrete and fast fourier transform techniques no restrictions on the number or position of the data points are imposed. The method also allows individual weighting of the data points.

Besides $R(x)$, the set of data points with the corresponding weights and the degree of $s(x)$, the user merely has to provide a positive parameter S to control the smoothness of $s(x)$. The number of knots and their position are then determined automatically, trying to take account of the specific behaviour of the signal underlying the data. Normally the number of knots will be considerably less than the number of data points, saving a lot of computation time and memory.

The smoothing factor S has to be chosen carefully: too large S-values will result in an underfitting of the data, and too small S-values will yield a spline which is highly influenced by the errors on the data points. Confidence limits for S are available if

the statistical errors on the data points can be estimated. However, the user should examine the deconvolution results graphically before accepting the fit as satisfactory. The computer program DECOSP which is based on the deconvolution algorithm provides a mode of computation through which a good S-value can be found in a very economical way. As for the degree of $s(x)$, cubic splines are recommended. This will usually give a good compromise between efficiency (computation time) and quality of fit. The proposed method could easily be extended to deal with other than spline approximations for the resolution function or even with cases where the response function is not translation invariant. In fact, only the direct computation of the elements of the convolution matrix involved (Section 3.3.a) should then be adjusted.

ACKNOWLEDGMENTS

I would like to thank Prof. Dr. R. Piessens for valuable comments and suggestions. This research is supported by the FKFO, Belgium, under Grant 2.0021.75.

REFERENCES

1. W. BOEHM, *Computer Aided Design* **12**(1980), 199–201.
2. C. DE BOOR, *J. Approx. Theory* **6** (1972), 50–62.
3. C. DE BOOR, "A Practical Guide to Splines," Springer-Verlag, New York/Berlin, 1978.
4. M. G. COX, *J. Inst. Math. Appl.* **10** (1972), 134–149.
5. P. DIERCKX, *J. Comp. Appl. Math.* **1** (1975), 165–184.
6. P. DIERCKX, "An Improved Algorithm for Curve Fitting with Spline Functions," Report TW 54, Dept. of Computer Science, K. U. Leuven, Belgium, 1981.
7. P. DIERCKX, "An Algorithm for Experimental Data Deconvolution Using Spline Functions," Report TW 59, Dept. of Computer Science, K. U. Leuven, Belgium, 1982.
8. P. W. GAFFNEY, *J. Inst. Math. Appl.* **17** (1976), 37–41.
9. W. M. GENTLEMAN, *J. Inst. Math. Appl.* **12** (1973), 329–336.
10. B. R. HUNT, "An Improved Technique for Using the Fast Fourier Transform to Solve Convolution-Type Integral Equations," Report LA–4515–MS, Los Alamos Scientific Laboratory, Los Alamos, N. Mex., 1970.
11. M. W. JOHNSON, "The 'Moments' Method for the Deconvolution of Experimental Spectra," Report RL–77–095/A, Rutherford Laboratory Chilton, Didcot, Oxon, England, 1977.
12. A. E. MCKINNON, A. G. SZABO, AND D. R. MILLER, *J. Phys. Chem.* **81** (1977), 1564–1570.
13. C. REINSCH, *Numer. Math.* **10** (1976), 177–183.
14. P. VERKERK, *Comput. Phys. Comm.* **25** (1982), 325–345.