An algorithm for smoothing, differentiation and integration of experimental data using spline functions

P. Dierckx (*)

ABSTRACT

This paper presents an algorithm for fitting a smoothing spline function to a set of experimental or tabulated data.
The obtained spline approximation can be used for differentiation and integration of the given discrete function.
Because of the ease of computation and the good conditioning properties we use normalised B-splines to represent the smoothing spline.
A Fortran implementation of the algorithm is given.

1. INTRODUCTION

Given the set of data points (x_i, y_i), $i = 1,2,\dots,m$ in the range $[a, b]$, and a set of positive numbers w_i, $i = 1,2,\dots,m$, we determine a spline function $s(x)$ of degree k with knots t_j, $j = k+1, k+2, \dots, n-k$ from the condition that

$$\sum_{r=k+2}^{n-k-1} [d_r]^2 \tag{1.1}$$

is minimal for all $s(x)$ satisfying the constraint

$$\sum_{i=1}^{m} w_i \, [y_i - s(x_i)]^2 \leqslant S$$

where d_r represents the discontinuity jump of $s^{(k)}(x)$ at t_r, i.e.

$$d_r = s^{(k)}(t_r + 0) - s^{(k)}(t_r - 0) \tag{1.2}$$

and where S is a given constant.

The number of knots and their position are chosen automatically by the algorithm.

2. SPLINE FUNCTIONS - B-SPLINES [3]

Given a strictly increasing sequence of real numbers $t_{k+1}, t_{k+2}, \dots, t_{n-k}$ a spline function $s(x)$ of degree k with knots t_j, $j = k+1, k+2, \dots, n-k$ is a function defined on the range $[t_{k+1}, t_{n-k}]$ having the following two properties

In each interval (t_j, t_{j+1}), $j = k+1, \dots, n-k-1$, $s(x)$ is given by some polynomial of degree k or less. (2.1)

$s(x)$ and its derivatives of orders $1, 2, \dots, k-1$ are continuous everywhere in the range $[t_{k+1}, t_{n-k}]$ (2.2)

The class $\eta_k(t_{k+1}, \dots, t_{n-k})$ of spline functions of degree k with knots $t_{k+1}, \dots, t_{n-k}$ is a $(n-k-1)$ dimensional vector space. Let

$$g_k(t;x) = (t-x)_+^k = \begin{cases} (t-x)^k & t \geqslant x \\ 0 & t < x \end{cases} \tag{2.3}$$

then the B-spline $M_{i,k}(x)$ is given as the $(k+1)$th divided difference of $g_k(t;x)$ on $t_i, t_{i+1}, \dots, t_{i+k+1}$ for fixed x, i. e.

$$M_{i,k}(x) = g_k(t_i, t_{i+1}, \dots, t_{i+k+1}; x) \tag{2.4}$$

while the normalised B-spline $N_{i,k}(x)$ is

(*) Applied Mathematics and Programming Division, University of Leuven, Celestijnenlaan 200 B, B-3030 Heverlee (Belgium)
The author is grant-holder of the I.W.O.N.L. (Belgium)

$$N_{i,k}(x) = (t_{i+k+1} - t_i) M_{i,k}(x) \tag{2.5}$$

The B-splines $M_{i,k}(x)$ and $N_{i,k}(x)$ are positive for $t_i < x < t_{i+k+1}$ and zero otherwise, i. e.

$$\begin{aligned} N_{i,k}(x) &= 0 \qquad && x \leqslant t_i \text{ or } x \geqslant t_{i+k+1} \\ N_{i,k}(x) &> 0 && t_i < x < t_{i+k+1} \end{aligned} \tag{2.6}$$

This feature makes the B-splines very attractive for practical computations.

In order to obtain a basis for $\eta_k(t_{k+1}, \ldots, t_{n-k})$ we introduce 2k additional knots

$t_1, t_2, \ldots, t_k, t_{n-k+1}, \ldots, t_n$

$$\begin{aligned} &t_1 < t_2 < \ldots < t_k < t_{k+1} \\ &t_{n-k} < t_{n-k+1} \ \ldots < t_n \end{aligned} \tag{2.7}$$

Now every $s(x) \in \eta_k(t_{k+1}, \ldots, t_{n-k})$ has a unique representation as a linear combination of the normalised B-splines $N_{j,k}(x)$, $j = 1, 2, \ldots, n-k-1$

$$s(x) = \sum_{j=1}^{n-k-1} c_j N_{j,k}(x) \tag{2.8}$$

3. THE SOLUTION OF PROBLEM (1.1)

Using the method of Lagrange we rewrite the problem (1.1) as a minimisation without constraints

$$\min K(p,z,\bar{c}) = \sum_{r=k+2}^{n-k-1} d_r^2 + p \left\{ \sum_{i=1}^{m} w_i [y_i - s(x_i)]^2 + z^2 - S \right\} \tag{3.1}$$

Minimising $K(p,z,\bar{c})$ with respect to p and z gives

$$\sum_{i=1}^{m} w_i [y_i - s(x_i)]^2 = S - z^2 \tag{3.2}$$

$$p \cdot z = 0 \tag{3.3}$$

We only consider the non-trivial case $p \neq 0$, $z = 0$. In section 4 we shall prove that with every positive value of p there corresponds a single spline function s(x) who minimises

$$K(p,0,\bar{c}) = \sum_{r=k+2}^{n-k-1} d_r^2 + p \left\{ \sum_{i=1}^{m} w_i [s(x_i) - y_i]^2 - S \right\} \tag{3.4}$$

We introduce the function $F_n(p)$ defined by

$$F_n(p) = \sum_{i=1}^{m} w_i [s(x_i) - y_i]^2 \tag{3.5}$$

According to (3.2) and taking into account that $z = 0$ we have to find a value of p such that

$$F_n(p) = S \tag{3.6}$$

In section 5 we shall show how to do this.

4. THE SOLUTION OF PROBLEM (3.4)

The problem is to minimise $K(p,0,\bar{c})$ for fixed p value.

From (1.2), (2.3), (2.4), (2.5), (2.6) and (2.8) we get

$$s(x_i) = \sum_{j=1}^{n-k-1} c_j N_{j,k}(x_i) \tag{4.1}$$

$$\text{with } N_{j,k}(x_i) = 0 \text{ if } x_i \leqslant t_j \text{ or } x_i \geqslant t_{j+k+1} \tag{4.2}$$

$$d_r = \sum_{j=1}^{n-k-1} c_j q_{j,r} \tag{4.3}$$

$$\begin{aligned} \text{with } q_{j,r} &= 0 \text{ if } j < r-k-1 \text{ or } j > r \\ &= (-1)^{k+1} k! \frac{(t_{j+k+1} - t_j)}{\pi'_{j,k}(t_r)} \qquad r-k-1 \leqslant j \leqslant r \end{aligned} \tag{4.4}$$

where $\pi_{j,k}(t) = (t - t_j)(t - t_{j+1}) \ldots (t - t_{j+k+1})$

$$K(p,0,\bar{c}) = \sum_{r=k+2}^{n-k-1} \left[\sum_{j=1}^{n-k-1} c_j q_{j,r} \right]^2 + p \left\{ \sum_{i=1}^{m} w_i \left[\sum_{j=1}^{n-k-1} c_j N_{j,k}(x_i) - y_i \right]^2 - S \right\} \tag{4.5}$$

The necessary condition for a minimum $\frac{\partial K}{\partial c_\ell} = 0$ gives

$$\sum_{j=1}^{n-k-1} c_j \left[\sum_{r=k+2}^{n-k-1} q_{j,r} q_{\ell,r} \right] + p \sum_{j=1}^{n-k-1} c_j \left[\sum_{i=1}^{m} w_i N_{j,k}(x_i) N_{\ell,k}(x_i) \right] = p \sum_{i=1}^{m} w_i y_i N_{\ell,k}(x_i) \qquad \ell = 1, 2, \ldots, n-k-1 \tag{4.6}$$

or in matrix form

$$G\,C = Z \quad \text{where} \tag{4.7}$$

$$G = A + p^{-1} B \tag{4.8}$$

$$B_{s,j} = \sum_{r=k+2}^{n-k-1} q_{s,r}\, q_{j,r} \tag{4.9}$$

$$A_{s,j} = \sum_{i=1}^{m} w_i N_{s,k}(x_i) N_{j,k}(x_i) \tag{4.10}$$

$$Z_j = \sum_{i=1}^{m} w_i y_i N_{j,k}(x_i) \tag{4.11}$$

$$C^T = (c_1, c_2, \dots, c_{n-k-1}) \tag{4.12}$$

From (4.2), (4.4), (4.9) and (4.10) we know that

A is a $(n-k-1) \times (n-k-1)$ positive definite matrix, $2k+1$ banded, (4.13)

B is a $(n-k-1) \times (n-k-1)$ positive semidefinite matrix, $2k+3$ banded (rank $n-2k-2$), (4.14)

and consequently $(p > 0)$ that

G is a $(n-k-1) \times (n-k-1)$ positive definite matrix, $2k+3$ banded. (4.15)

Equation (4.7) is easily solved using the method of Cholesky for positive definite bandmatrices [6].

5. SOLUTION OF PROBLEM (3.6)

We wish to find a positive value of p such that $F_n(p) = S$.
From (3.5) and (4.8) it follows that

$$F_n(0) = \sum_{i=1}^{m} w_i \,[y_i - P_k(x_i)]^2 \tag{5.2}$$

where $P_k(x)$ represents the least squares polynomial approximation of degree k, and that

$$F_n(\infty) = \sum_{i=1}^{m} w_i \,[y_i - S_{n,k}(x_i)]^2 \tag{5.3}$$

where $S_{n,k}(x)$ represents the least squares spline approximation of degree k with knots $t_{k+1}, \dots, t_{n-k}$.

$F_n(0)$ depends neither on the number of knots nor on their position, i. e.

$$F_n(0) = F(0) \tag{5.4}$$

If $S \geqslant F(0)$ $P_k(x)$ is the trivial solution of problem (1.1).
Let us now suppose that $0 \leqslant S < F(0)$.
We introduce the matrices E and Y defined by

$$E_{s,j} = \sqrt{w_s}\, N_{j,k}(x_s) \tag{5.5}$$

$$Y^T = (\sqrt{w_1}\, y_1, \sqrt{w_2}\, y_2, \dots, \sqrt{w_m}\, y_m) \tag{5.6}$$

From (4.10) and (4.11) we get the expressions

$$A = E^T E \tag{5.7}$$

$$Z = E^T Y \tag{5.8}$$

We now express $F_n(p)$ as the product of two matrices

$$F_n(p) = [EC - Y]^T\, [EC - Y] \tag{5.9}$$

or according to (4.7)

$$\begin{aligned} F_n(p) &= [EG^{-1}Z - Y]^T\, [EG^{-1}Z - Y] \\ &= [EG^{-1}E^TY - Y]^T\, [EG^{-1}E^TY - Y] \\ &= Y^T[EG^{-1}AG^{-1}E^T - 2EG^{-1}E^T \\ &\quad + EA^{-1}E^T]\,Y + Y^TY - Y^TEA^{-1}E^TY \end{aligned} \tag{5.10}$$

For $p = \infty$ we have

$$F_n(\infty) = Y^TY - Y^TEA^{-1}E^Ty \tag{5.11}$$

$F_n(p)$ becomes

$$\begin{aligned} F_n(p) &= Y^T[EG^{-1}AG^{-1}E^T - 2EG^{-1}E^T + EA^{-1}E^T]Y \\ &\quad + F_n(\infty) = p^{-2}Y^TE(A+p^{-1}B)^{-1}BA^{-1}B(A \\ &\quad + p^{-1}B)^{-1}E^TY + F_n(\infty) = Z^TA^{-1}(pI \\ &\quad + BA^{-1})^{-1}B^{1/2}B^{1/2}A^{-1}B^{1/2}B^{1/2}(pI \\ &\quad + A^{-1}B)^{-1}A^{-1}Z + F_n(\infty) \end{aligned}$$

We can easily verify that

$$(pI + BA^{-1})^{-1}B^{1/2} = B^{1/2}(pI + B^{1/2}A^{-1}B^{1/2})^{-1} \tag{5.12}$$

$$B^{1/2}(pI + A^{-1}B)^{-1} = (pI + B^{1/2}A^{-1}B^{1/2})^{-1}B^{1/2} \tag{5.13}$$

Finally $F_n(p)$ becomes

$$F_n(p) = Z^T A^{-1} B^{1/2} (pI + B^{1/2} A^{-1} B^{1/2})^{-1} B^{1/2} A^{-1} B^{1/2}$$
$$(pI + B^{1/2} A^{-1} B^{1/2})^{-1} B^{1/2} A^{-1} Z + F_n(\infty) \tag{5.14}$$

$$F_n(p) = \sum_{i=1}^{n-2k-2} \frac{h_i^2 \lambda_i}{(p+\lambda_i)^2} + F_n(\infty) \tag{5.15}$$

where the λ_i are the non-zero eigenvalues of $B^{1/2} A^{-1} B^{1/2}$. This matrix is positive semidefinite. Consequently the λ_i are all positive and $F_n(p)$ is a convex and strictly decreasing function of p. (fig.1)

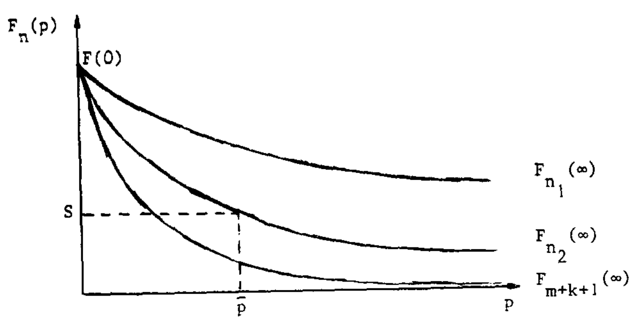


Fig. 1.

If $n = m + k + 1$ the least squares spline function $S_{n,k}(x)$ (5.3) is an interpolating spline function, i. e.

$$F_{m+k+1}(\infty) = 0$$

We now proceed as follows : From the fact that $F_n(p)$ is a strictly decreasing function of p, we know that, once we have found a value of n such that $F_n(\infty) \leqslant S < F(0)$, there exists a unique positive root $\bar{p}$ of $F_n(p) = S$.

Since we also know that $F_n(p)$ is convex, we could use a Newton iteration, starting with $p_0 = 0$ to produce a strictly increasing sequence p_r $r = 1, 2, \ldots$ converging to the correct value of $\bar{p}$.

Unfortunately, this method appears to be too slow to be always applicable. Reinsch [9] suggests that a more efficient and also globally convergent method for determining $\bar{p}$ is to solve $F_n(p)^{-1/2} = S^{-1/2}$ instead of $F_n(p) = S$. He proves that, according to expression (5.15), $F_n(p)^{-1/2}$ is a concave and strictly increasing function.

Starting with $p_0 = 0$, we now produce a strictly increasing sequence

$$p_{r+1} = p_r + \frac{2F_n(p_r)^{1/2}}{S^{1/2}} \left(\frac{S^{1/2} F_n(p_r)^{1/2} - F_n(p_r)}{F_n'(p_r)} \right)$$
$$r = 0, 1, \ldots \tag{5.16}$$

converging more rapidly to the correct value of $\bar{p}$. The quantities $F_n(p_r)$ and $F_n'(p_r)$ needed in (5.16) are easily computed at each step. Indeed, given a solution $s_{p_r}(x)$ of (3.4) with smoothing parameter p_r we can compute directly

$$F_n(p_r) = \sum_{i=1}^{m} w_i \, [s_{p_r}(x_i) - y_i]^2 \tag{5.17}$$

To see how to get $F_n'(p_r)$ we refer to (5.10).

$$F_n(p) = [E(A + p^{-1}B)^{-1} Z - Y]^T [E(A + p^{-1}B)^{-1} Z - Y]$$
$$F_n'(p) = 2[EG^{-1}Z - Y]^T [E(A + p^{-1}B)^{-1} Z - Y]'$$
$$= 2p^{-2} [EG^{-1}Z - Y]^T [E(A + p^{-1}B)^{-1} B (A + p^{-1}B)^{-1}$$
$$= -2p^{-1} (AC - Z)^T G^{-1} (AC - Z)$$
$$= -2p^{-3} (BC)^T G^{-1} (BC) \tag{5.18}$$

The number (5.18) is computable at each step since the main work in computing G^{-1} operating on a vector has been done in the solution of (4.7) for C.
In conclusion of this chapter we still have to say something about our choice of the knots. Let us first state that we haven't really tried to find the exact minimal number of knots, neither their optimal position.
We simply produce a number of increasing values n_i, $3k + 1 \leqslant n_i \leqslant m + k + 1$ and reject or accept n_i whether $F_{n_i}(\infty) > S$ or not.
We take $t_{k+1} = a$ and $t_{n_i - k} = b$. The other knots are chosen rather arbitrarily but we do take care of the fact that if there is a concentration of data points somewhere there will be a concentration of knots too, and if the data points are symmetrical with respect to the centre of the range [a, b] the knots will be as well.

6. EVALUATION OF s(x) - DIFFERENTIATION - INTEGRATION

In this section we mention some formulas for computing $s^{\nu}(x)$ for $0 \leqslant \nu \leqslant k$ and arbitrary $a \leqslant x \leqslant b$

[2] and derive an expression for

$$\int_\alpha^\beta s(x)\,dx \text{ for } a \leqslant \alpha, \beta \leqslant b$$

$$s(x) = \sum_{j=1}^{n-k-1} c_j N_{j,k}(x) \qquad (2.8)$$

Most of the known properties of B-splines can be derived from the simple identity [2]

$$N_{j,k}(x) = \frac{x - t_j}{t_{j+k} - t_j} N_{j,k-1}(x) + \frac{t_{j+k+1} - x}{t_{j+k+1} - t_{j+1}} N_{j+1,k-1}(x) \qquad (6.1)$$

Since $N_{j,0}(x) = 1$ for $t_j \leqslant x < t_{j+1}$ and $= 0$ otherwise, it follows that

$$s(x) = c_j^{[k]}(x) \qquad t_j \leqslant x < t_{j+1} \qquad (6.2)$$

where

$$c_j^{[i]}(x) \begin{cases} = c_j & i = 0 \\ = \dfrac{x - t_j}{t_{j+k+1-i} - t_j} c_j^{[i-1]}(x) + \dfrac{t_{j+k+1-i} - x}{t_{j+k+1-i} - t_j} c_{j-1}^{[i-1]}(x) & i > 0 \end{cases}$$

From (2.3), (2.4), (2.5), (2.8) and (6.1) we also know that

$$s^{\nu}(x) = k(k-1)\ldots(k-\nu+1) \sum_{j=1}^{n-k-1+\nu} c_j^{(\nu)} N_{j,k-\nu}(x) \qquad (6.3)$$

where

$$c_j^{(\nu)} \begin{cases} = c_j & \nu = 0 \\ = \dfrac{c_j^{(\nu-1)} - c_{j-1}^{(\nu-1)}}{t_{j+k+1-\nu} - t_j} & \nu > 0 \end{cases}$$

Finally it is easily verified from (2.3), (2.4), (2.5) that

$$\int_\alpha^\beta s(x)dx = \sum_{j=1}^{n-k-1} c_j \frac{(t_{j+k+1} - t_j)}{(k+1)} [g_{k+1}(t_j, \ldots, t_{j+k+1}; \alpha) - g_{k+1}(t_j, \ldots, t_{j+k+1}; \beta)] \qquad (6.4)$$

where

$$g_{k+1}(t_j, \ldots, t_{j+k+1}; x) = 1 \quad x \leqslant t_j$$
$$= 0 \quad x \geqslant t_{j+k+1}$$

7. A SUBROUTINE PACKAGE FOR SMOOTHING, DIFFERENTIATION AND INTEGRATION OF EXPERIMENTAL DATA

1) **Subroutine smoot (X, Y, W, M, XI, XE, K, S, N, T, NK1, C, IER)**

Purpose :

Given the set of data points (x_i, y_i) in the range $[a, b]$ with weighting factors w_i, $i = 1, \ldots, m$, SMOOT determines a spline $s(x)$ of degree k with knots t_j, $j = 1, \ldots, n$ from the condition that

$$\sum_{r=k+2}^{n-k-1} [s^{(k)}(t_r + 0) - s^{(k)}(t_r - 0)]^2$$

is minimal for all $s(x)$ satisfying the constraint

$$\sum_{i=1}^{m} w_i [y_i - s(x_i)]^2 \leqslant S$$

The user has to provide the data points (x_i, y_i), the weights w_i, the smoothing factor S, the degree of the fitting spline and an over-estimate of the number of knots. The program returns the knots of $s(x)$ and the coefficients c_j of the normalised B-spline representation of $s(x)$.

Description of the parameters :

INPUT PARAMETERS

X, Y :	The set of data points x_i, y_i, $i = 1, \ldots, m$.
W :	The weights w_i, $i = 1, \ldots, m$.
M :	The number of data points m.
XI, XE :	End points a and b of the approximation interval.
K :	Degree of the fitting spline.
S :	Non negative smoothing factor S. If $w_i = (\delta y_i)^{-2}$ where δy_i is an estimate of the standard deviation of y_i, it is recommended that S be chosen in the range $m \pm \sqrt{2m}$. If S = 0 SMOOT returns an interpolating spline (IER = -1). If $S \gg 0$ SMOOT returns the least squares polynomial approximation of degree k (IER = -2).

N : An over-estimate of the number of knots. This parameter must be set by the user to indicate the storage space available to the subroutine. The dimensions of the arrays

T :	(n)	B,G :	(n-k-1,k+2)
C,Z,V :	(n-k-1)	H1,H2 :	(k+1)
A :	(n-k-1,k+1)	H :	(2k+2)

depend on k and n. Since n is unknown at the time the user sets up the dimension information, an over-estimate of these arrays will generally be made.
The following remarks are intended to help the user make an over-estimate of the space required.

(1) $3k+1 \leqslant n \leqslant m+k+1$
(2) The smaller the value of S, the greater n will be.
(3) Normally $n = m/2$ should be an over-estimate.

OUTPUT PARAMETERS

N : The final number of knots of s(x).
T : The position of the knots $t_j, j=1,\ldots,n$
$T(N-NK1) = XI$, $T(NK1+1) = XE$
NK1 : Dimension of s(x) ($NK1 = n-k-1$).
C : The coefficients of the normalised B-spline representation of s(x), i. e. c_j, $j = 1, \ldots, NK1$.
IER : Error code
IER = -2 : SMOOT returns the least squares polynomial approximation of degree k
IER = -1 : SMOOT returns an interpolating spline.
IER = 0 : SMOOT returns a smoothing spline.
IER = 1 : The required storage space exceeds the available storage space, specified by the user. (N too small).
IER = 2 : The maximal number of iterations allowed in the Newton process has been reached.
IER = 3 : A theoretically impossible result was found during the computations : $F_n(p)^{-1/2}$ is not concave.
IER = 4 : A theoretically impossible result was found during the computations : matrix A is not positive definite. (probably $\max_i(w_i)/\min_i(w_i)$ is too large).
IER = 10 : The input data are invalid. (see restrictions).

RESTRICTIONS

$a \leqslant x_1 < x_2 < \ldots < x_m \leqslant b$

$w_i > 0 \quad i = 1, \ldots, m$

$2 \leqslant k \leqslant m/2$

Subroutines and function subprograms required :
BANDET, BANSOL. (These programs are the Fortran IV version of existing Algol procedures [6]).

2) Function Deriv (T, N, C, NK1, NU, ARG, 1)

Purpose : Given a spline function s(x) of degree k with knots t_j, $j=1,\ldots,n$ and normalised B-spline representation (coefficients c_j, $j=1,\ldots,n-k-1$), DERIV produces the value of the ν-th derivative of s(x) for the argument x, $t_\ell \leqslant x < t_{\ell+1}$.

Description of the parameters :

INPUT PARAMETERS

T, N, C, NK1 : See output parameters of SMOOT.
NU : The order of the derivative.
ARG : Value of the argument.
L : Parameter specifying the position of the argument.
$T(L) \leqslant ARG < T(L+1)$ or if $ARG = T(NK1+1)$, $L = NK1$.

OUTPUT PARAMETER

DERIV : Value of the derivative.

RESTRICTIONS

$T(N-NK1) \leqslant ARG \leqslant T(NK1+1)$, $NU \geqslant 0$

Subroutines and functions required : none.

3) Function splint (T, N, C, NK1, ALFA, BETA)

Purpose : Given a spline function s(x) of degree k with knots t_j, $j = 1, \ldots, n$ and normalised B-spline representation (coefficients c_j, $j=1,\ldots,n-k-1$), SPLINT produces the value of the integral of s(x) between α and β.

Description of the parameters :

INPUT PARAMETERS

T, N, C, NK1 : See output parameters of SMOOT.
ALFA, BETA : End points α and β of the integration interval.

OUTPUT PARAMETER

SPLINT : Value of the integral.

RESTRICTIONS

T(N-NK1) ≤ ALFA, BETA ≤ T(NK1 + 1)

Subroutines and function required : BSPLIN.

8. NUMERICAL RESULTS

EXAMPLE 1 :

Using a random generator we generated a set of normally distributed stochastic variates z_i (expected value 0, standard deviation 0.05) and computed

$$\bar{z} = \frac{1}{101}\sum_{i=1}^{101} z_i \qquad \delta^2 = \frac{1}{100}\sum_{i=1}^{101}(z_i - \bar{z})^2$$

Given the set of data points

$$\begin{cases} x_i = (i-1) \times \pi/50 & \\ y_i = \cos x_i + z_i & i = 1, 2, \ldots, 101 \\ w_i = \delta^{-2} & (\delta \text{ is an estimate of the standard deviation of } y_i) \end{cases}$$

we then determined a smoothing spline s(x) of degree K = 5 and smoothing factor S = 98. The knot positions were found to be

$$t_{j+5} = \frac{2\pi}{5}(j-1) \qquad j = 1, 2, \ldots, 6 \ (N = 16)$$

We also computed the first and the second derivative of s(x) and compared it with the exact values of the derivatives of f(x) = cos x. (fig. 2).

EXAMPLE 2

As in example 1 we produced a set of 101 data points

$$\begin{cases} x_i = (i-51)\ 0.04 & i = 1, 2, \ldots, 101 \\ y_i = 20\, e^{-x_i^2} + z_i & (z_i \text{ a normally distributed stochastic variate. E.V.} = 0, \text{ S.D.} = 1) \\ w_i = \delta^{-2} & \end{cases}$$

and determined a smoothing cubic spline (K=3, S=98).

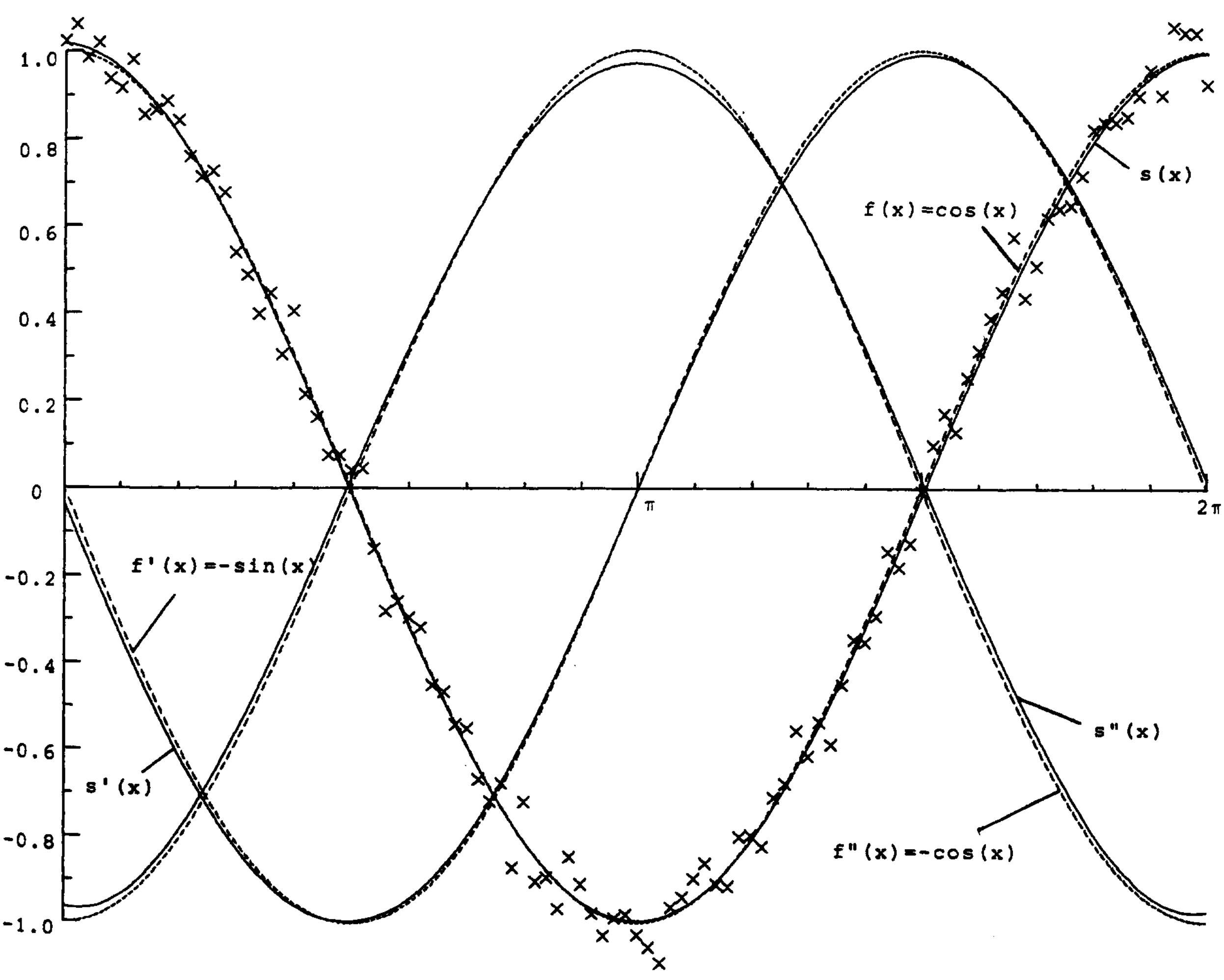


Fig. 2.

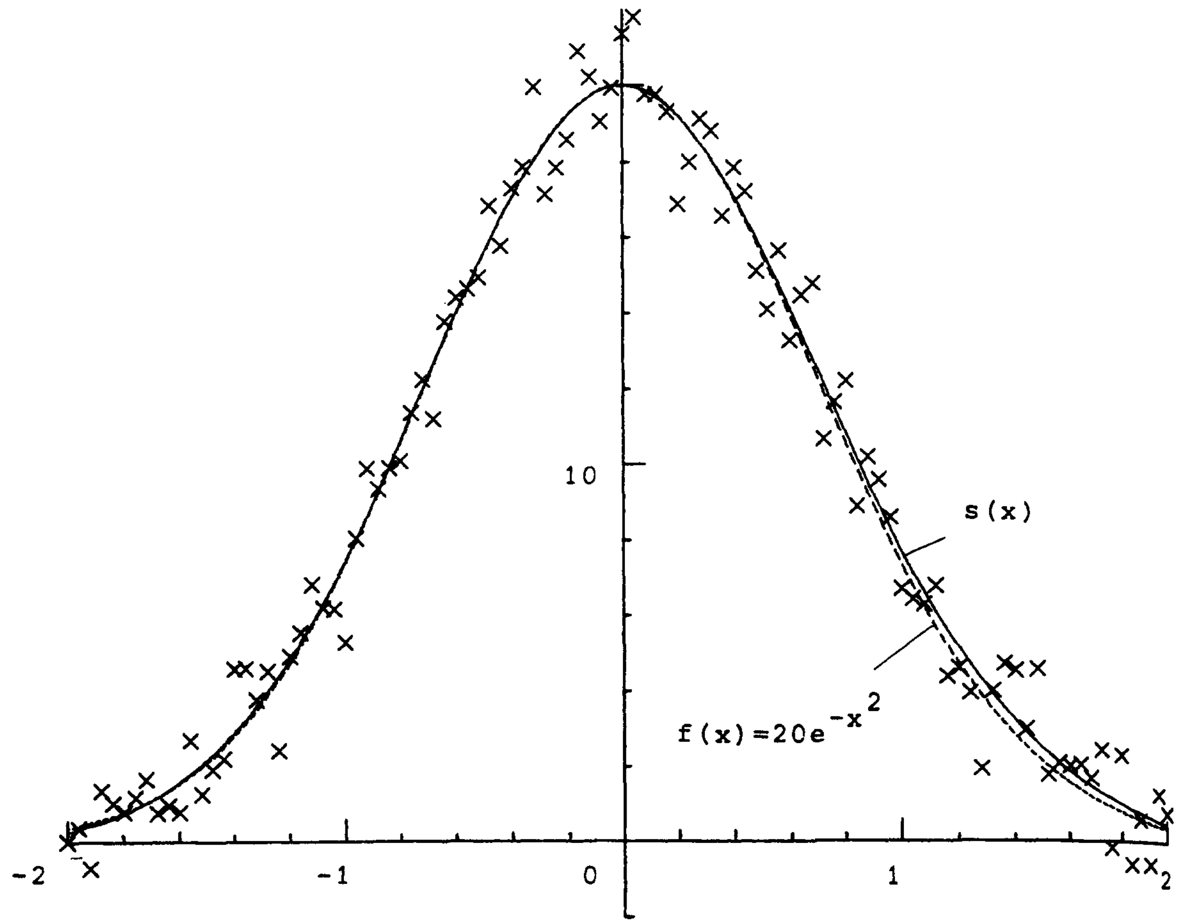


Fig. 3a.

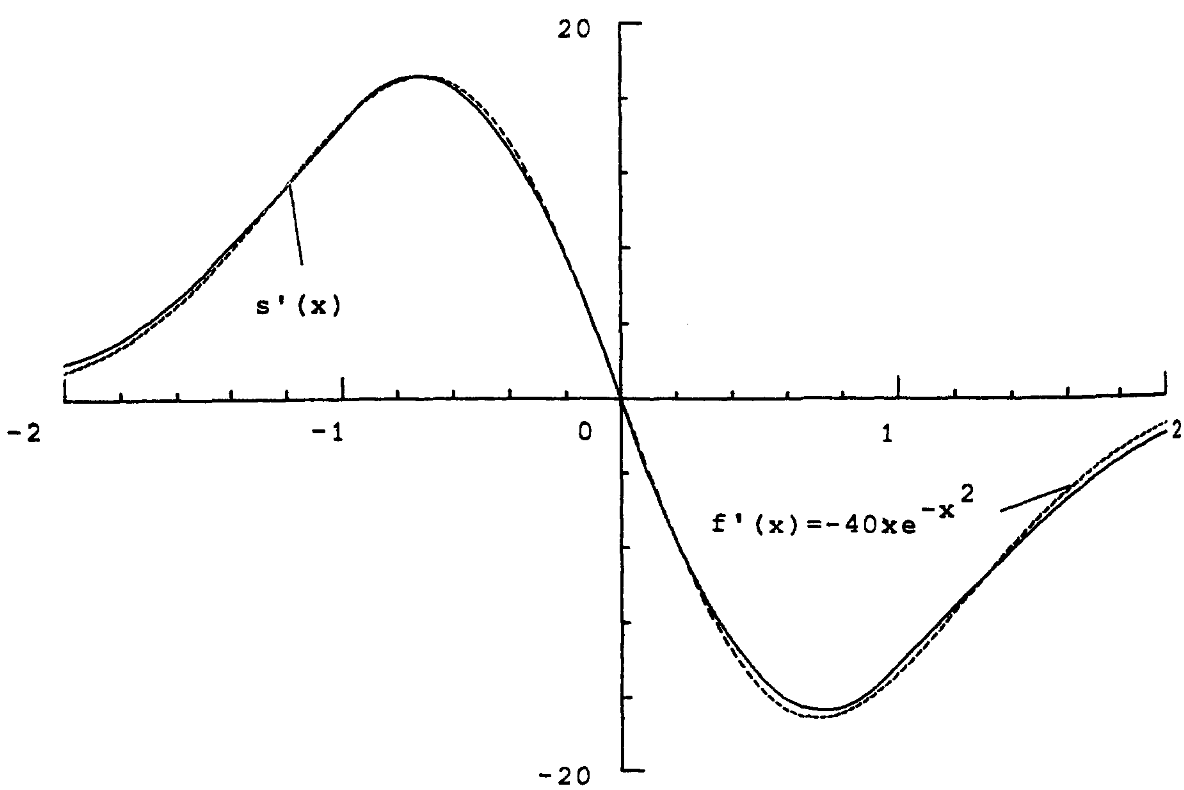


Fig. 3b.

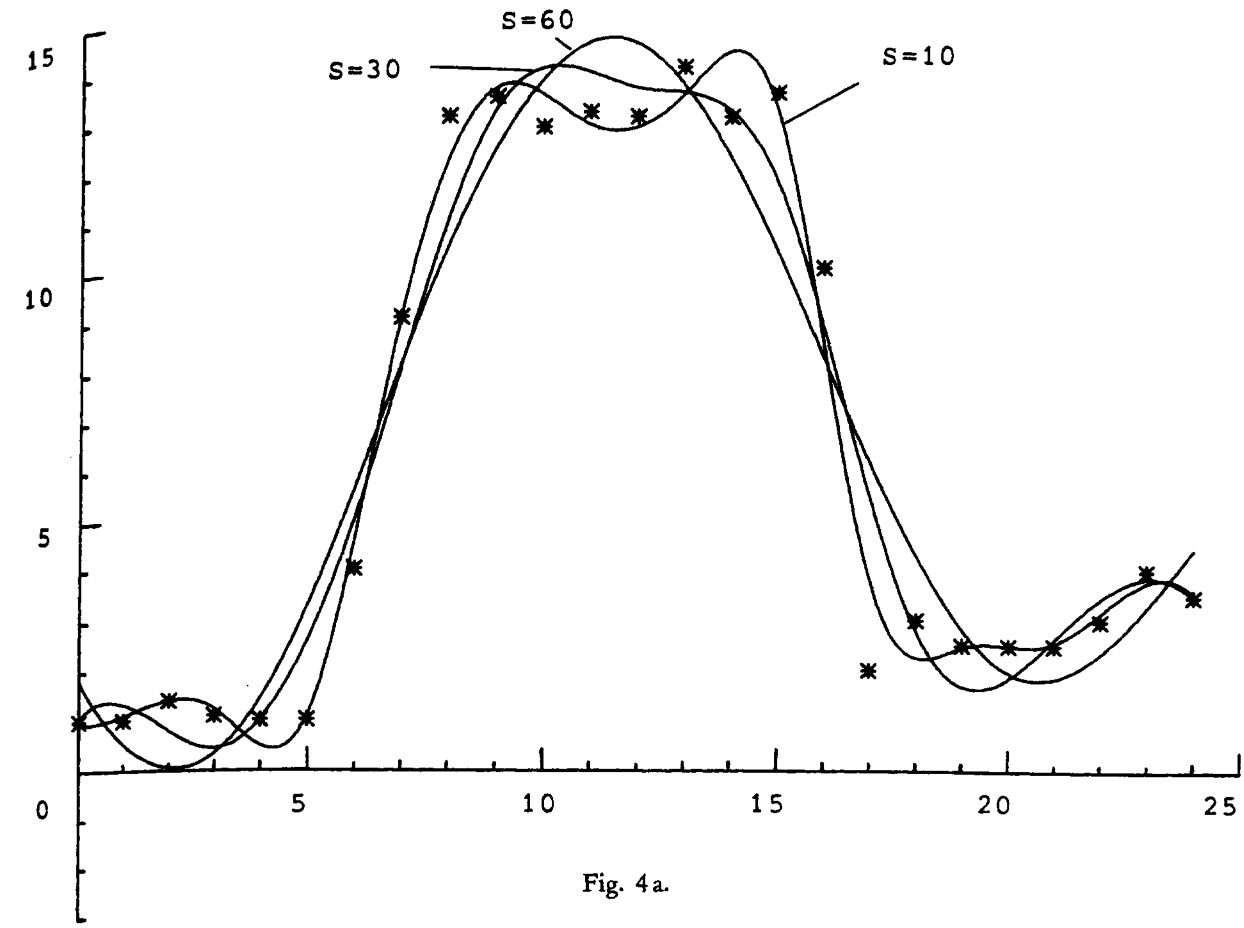


Fig. 4a.

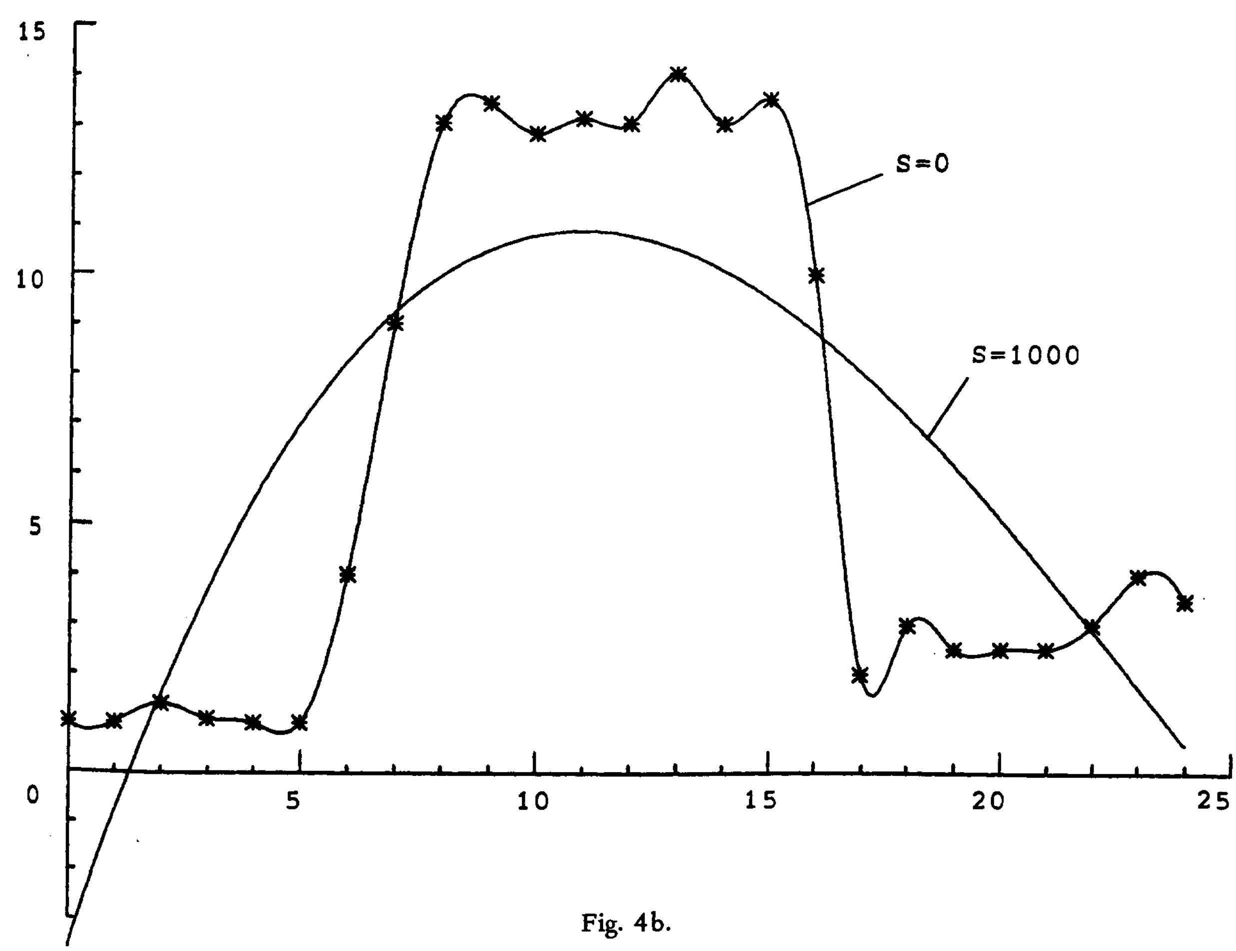


Fig. 4b.

The position of the knots t_j, $j = 4, \ldots, 12 (N = 15)$ are

$-2.(0.52) - 0.96 (0.48) 0.96 (0.52) 2.$

Fig. 3 shows the set of data points (x_i, y_i) and the graphs of $s(x)$ and its first derivative $s'(x)$ compared with the exact functions

$$f(x) = 20\, e^{-x^2} \qquad f'(x) = -40\, x\, e^{-x^2}$$

EXAMPLE 3

Fig. 4 shows the influence of the smoothing factor S. The data : (x_i, y_i), w_i, $i = 1, \ldots, 25$, $K = 3$

Fig. 4-a

i) $S = 10$

t_{j+3}, $j = 1, \ldots, 12$ $(N = 18)$: $0,3\,(2)\,21,24$

ii) $S = 30$

t_{j+3}, $j = 1, \ldots, 9$ $(N = 15)$: $0\,(3)\,24$

iii) $S = 60$

t_{j+3}, $j = 1, \ldots, 7$ $(N = 13)$: $0\,(4)\,24$

Fig. 4-b (borderline case)

i) $S = 0$ interpolating spline

t_{j+3}, $j = 1, \ldots, 23$ $(N = 29)$: $0,2\,(1)\,22, 24$

ii) $S = 1000$ least squares polynomial approximation

t_{j+3}, $j = 1, \ldots, 4$ $(N = 10)$: $0\,(8)\,24$

EXAMPLE 4

Fig. 5 shows the graphs of some smoothing splines of different degree K, and the influence of the weighting factors w_i.

The data : (x_i, y_i) $i = 1, \ldots, 15$

Fig. 5-a $w_i = 1$, $i = 1, \ldots, 15$, $S = 0.2$

i) $K = 2$

t_{j+2}, $j = 1, \ldots, 11$ $(N = 15)$: $-8(1.5)-5, -4(2)4,5(1.5)8$

ii) $K = 4$

t_{j+4}, $j = 1, \ldots, 12$ $(N = 20)$: $-8, -6.5, -6(1) -4(2)4(1)6, 6.5, 8$

iii) $K = 6$

t_{j+6}, $j = 1, \ldots, 10$ $(N = 22)$: $-8(1.5)-5, -4, -2, 2, 4, 5(1.5)8$

Fig. 5-b $S = 12$, $K = 2$, $w_i = 1$, $i = 1, \ldots, 15$ $i \neq 7, 9$

i) $w_7 = w_9 = 1$

t_{j+2}, $j = 1, 2, 3$ $(N = 7)$: $-8, 0, 8$

ii) $w_7 = w_9 = 10$

t_{j+2}, $j = 1, \ldots, 8$ $(N = 12)$: $-8(1.5)-5, -2, 2,5(1.5)8$

iii) $w_7 = w_9 = 100$

t_{j+2}, $j = 1, \ldots, 8$ $(N = 12)$: $-8(1.5)-5, -2, 2,5(1.5)8$

EXAMPLE 5

Let $f(x)$ a given function, α and β the end points of the integration interval and I the exact value of the integral

$$I = \int_{\alpha}^{\beta} f(x)\, dx$$

Using a random generator we generate a set of uniformly distributed stochastic variates r_i over the range -1 to $+1$, and compute

$$err_i = \frac{ACC}{100} f(x_i)\, r_i \qquad \overline{err} = \sum_{i=1}^{m} err_i / m$$

$$\delta^2 = \sum_{i=1}^{m} (err_i - \overline{err})^2 / (m-1)$$

Given the set of data points

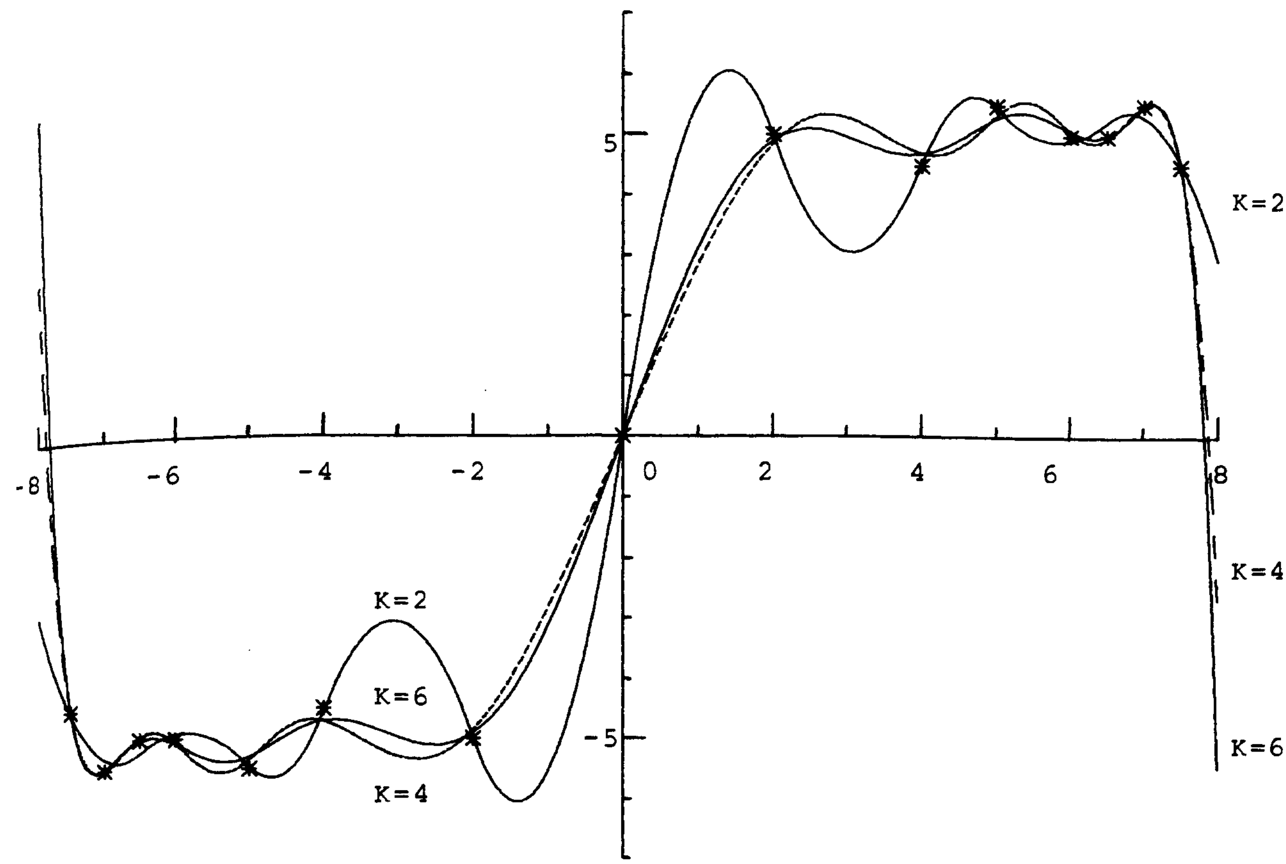
$$\left[\begin{array}{ll} x_i = \frac{\beta - \alpha}{m-1}(i-1) + \alpha & \\ y_i = f(x_i) + err_i & i = 1, 2, \ldots, m \\ w_i = \delta^{-2} & \end{array}\right.$$

we then determine $s(x)$, a smoothing spline approximation of $f(x)$ (degree K, number of knots N, smoothing facter $s = m$, approximation interval α, β), and compute the integral

$$J = \int_{\alpha}^{\beta} s(x)\, dx \approx I$$

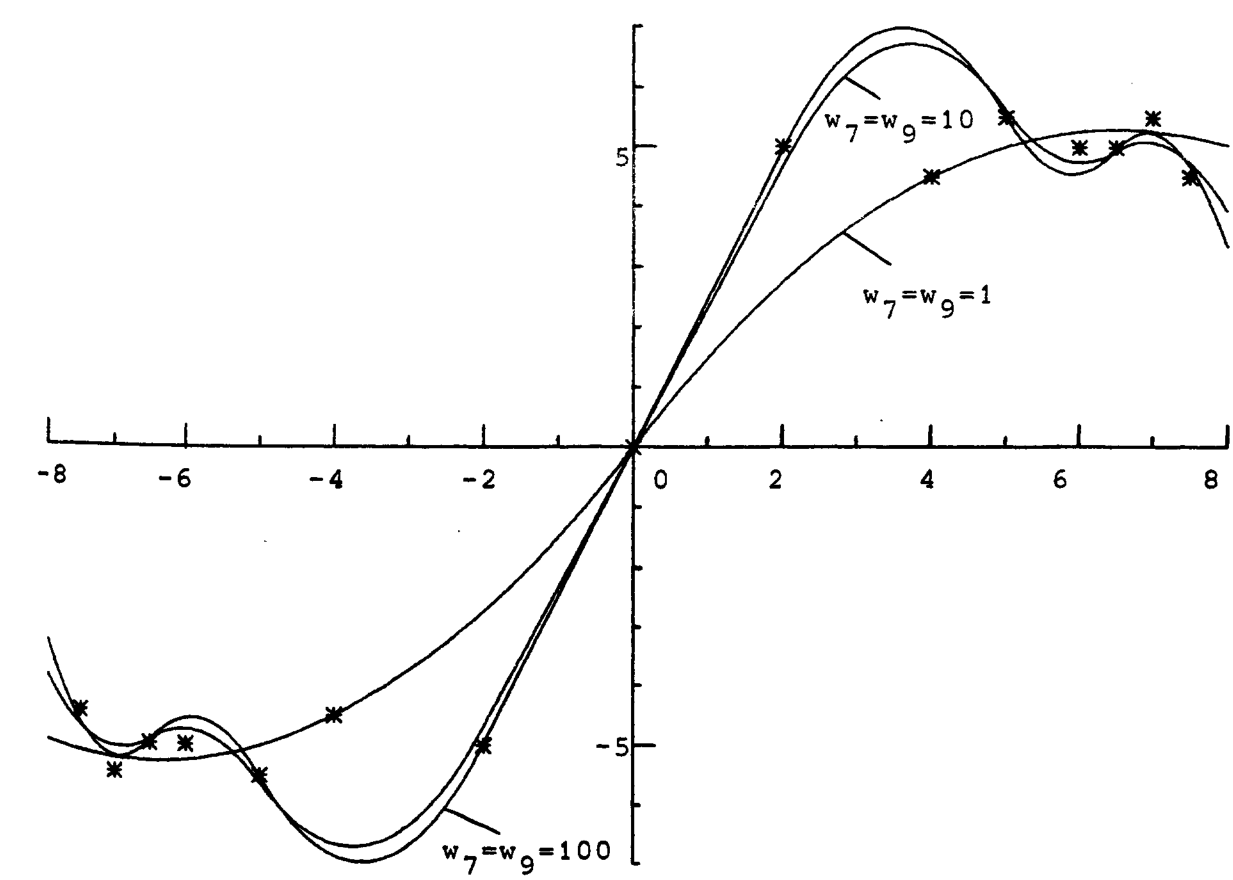
(relative error ERREL $= \left|\frac{I-J}{I}\right| 100\ \%$)

The computations are carried out for $f(x) = \sqrt{x(1-x)}$ and $f(x) = 1/(1 + 0.5 \cos x)$ and the results are presented in tables 1 and 2.



5
-5
-8
-6
-4
-2
0
2
4
6
8
K=2
K=4
K=6
K=2
K=6
K=4

Fig. 5a.



5
-5
-8
-6
-4
-2
0
2
4
6
8
w7=w9=10
w7=w9=1
w7=w9=100

Fig. 5b.

TABLE 1

$f(x) = \sqrt{x(1-x)}$; $\alpha = 0$; $\beta = 1$; $K = 3$;
$I = 0.3926991$

ACC %	m = S	N	J	ERREL %
1	26	23	0.3901680	0.64
	51	44	0.3919165	0.20
	101	44	0.3924994	0.05
	201	65	0.3925147	0.05
5	26	13	0.3898125	0.74
	51	23	0.3911605	0.39
	101	23	0.3907945	0.48
	201	23	0.3940969	0.35
10	26	13	0.3919327	0.19
	51	15	0.3920243	0.17
	101	15	0.3918650	0.21
	201	23	0.3943436	0.42

9. CONCLUSION

In the preceding sections we have described an algorithm for smoothing, differentiation and integration of functions defined by a set of data points. Our method determines a smoothing spline function of degree k. It chooses automatically the number of knots n and their position. Normally n is quite less than m, the number of data points.
A small value for the number of knots has several advantages : storage requirement and computing time are reduced, a possible better knot spacing improves the numerical stability.
As opposed to other methods [4, 5, 8, 10] which determine a natural spline ($k = 2\ell - 1$; $s^{\nu}(a) = s^{\nu}(b) = 0, \nu = \ell, \ell+1, \ldots, k$), we do not impose restrictions on the fitting spline function.
The smoothing spline in our algorithm is represented in terms of normalised B-splines. The coefficients are obtained by solution of a banded linear system. Evaluation, differentiation and integration of the smoothing spline is performed in a rapid and accurate way using some stable recursion relations. The user has to provide a positive, constant smoothing factor S. This parameter, however, has a direct meaning, at least when the weighting factors are $w_i = (\delta y_i)^{-2}$ and δy_i is an estimate of the standard deviation of y_i.
Reinsch [8] suggests choosing S in the range $m \pm \sqrt{2m}$.
If $S \gg 0$ the algorithm returns the least squares polynomial approximation of degree k. If $S = 0$ it returns an interpolating spline.

TABLE 2

$f(x) = 1/(1 + 0.5 \cos x)$; $\alpha = 0$; $\beta = \pi/2$; $K = 5$;
$I = 1.209199$

ACC %	m = S	N	J	ERREL %
1	26	16	1.206084	0.26
	51	16	1.208869	0.03
	101	16	1.209832	0.05
	201	16	1.208930	0.02
5	26	16	1.205881	0.27
	51	16	1.211060	0.15
	101	16	1.205363	0.32
	201	16	1.214111	0.41
10	26	16	1.215603	0.53
	51	16	1.211824	0.22
	101	16	1.211402	0.18
	201	16	1.212788	0.30

ACKNOWLEDGEMENT

The author is greatly indebted to Dr. R. Piessens and Dr. P. Dewilde, Miss A. Haegemans and Miss E. De Doncker for valuable comments and suggestions.

REFERENCES

1. Cox M. G., "A data-fitting package for the non-specialist user". Report NAC 40 (July 1973). National Physical Laboratory, Teddington, Middlesex.
2. de Boor C., "On calculating with B-splines". J. Approximation Theory 6, 1972, pp. 50-62.
3. Greville T. N. E., "Introduction to spline functions, theory and applications of spline functions", T.N.E. Greville ed., Academic Press, New York, 1969, pp. 1-35.
4. Lyche T. and Schumaker L., "Computation of smoothing and interpolating natural splines via local bases", SIAM J. of analysis 10, 1973, pp. 1027-1038.
5. Lyche T. and Schumaker L., "Procedures for computing smoothing and interpolating natural splines", Communications of the A.C.M. 17, 1974, pp. 463-467.
6. Martin R. S. and Wilkinson J. H., "Symmetric decomposition of positive definite bandmatrices", Num. Math. 7, 1965, pp. 355-361.
7. Powell M. J. D., "Curve fitting by splines in one variable, numerical approximation to functions and data", Hayes J. G. ed., The Athlone Press, 1970, pp. 65-83.
8. Reinsch C. H., "Smoothing by spline functions", Num. Math. 10, 1967, pp. 177-183.
9. Reinsch C. H., "Smoothing by spline functions II", Num. Math. 16, 1971, pp. 451-454.
10. Woodford Ch., "An algorithm for data smoothing using spline functions", BIT 10, 1971, pp. 501-510.

```
      SUBROUTINE SMOOT(X,Y,W,M,XI,XE,K,S,N,T,NK1,C,IER)
C   SMOOT DETERMINES A SMOOTHING SPLINE APPROXIMATION (NORMALISED
C   B-SPLINE REPRESENTATION) OF A GIVEN DISCRETE FUNCTION.
      DIMENSION X(M),Y(M),W(M),T(N),C(100),A(100,6),
     < B(100,7),G(100,7),Z(100),V(100),H(12),H1(6),H2(6)
C   DATA INITIALISATION STATEMENT TO SPECIFY
C     TOL : THE REQUESTED RELATIVE ACCURACY FOR THE ROOT OF F(P)=S
C     MAX : THE MAXIMAL NUMBER OF ITERATIONS ALLOWED IN THE
C           NEWTON PROCESS
      DATA TOL/0.01/,MAX/20/
C   BEFORE STARTING COMPUTATIONS A DATACHECK IS MADE
C   IF THE INPUTDATA ARE INVALID CONTROLE IS REPASSED TO
C   THE DRIVER PROGRAM (IER=10)
      IER = 10
      IF(K.LT.2 .OR. M.LT.2*K) RETURN
      IF(XI.GT.X(1) .OR. X(M).GT.XE) RETURN
      WMAX = W(1)
      IF(WMAX.LE.0.) RETURN
      DO 20 I=2,M
        IF(X(I-1).GE.X(I) .OR. W(I).LE.0.) RETURN
        IF(W(I).GT.WMAX) WMAX = W(I)
  20  CONTINUE
C   COMPUTATIONS ARE STARTED
      IF(S.LT.0.) S = 0.
      S2 = SQRT(S)
      K1 = K+1
      M1 = M-1
C   WE CHOOSE THE INITIAL VALUE FOR THE NUMBER OF KNOTS, I.E.
C      S.NE.0 : N = 3K+1
C      S.EQ.0 : N = M+K+1 (INTERPOLATION)
      IFIB1 = 1
      IFIB2 = 2
      NMIN = 3*K+1
      NMAX = M+K1
      NN = N
      N = NMIN
      IF(S.EQ.0.) N = NMAX
C   WE CHECK WHETHER THE REQUIRED STORAGE SPACE EXCEEDS THE
C   AVAILABLE STORAGE SPACE
 100  IF(N.GT.NN) GO TO 930
C   WE CHOOSE THE KNOTS T(K1),...T(NK1+1) IN THE RANGE (XI,XE)
      NK1 = N-K1
      M2 = N-2*K1+1
      L = M1/M2+1
      IR = M1-M2*(L-1)
      MI = L+1
      ME = M-L
      NI = K1+1
      NE = NK1
      DO 160 JJ=1,2
        IR1 = IR/2
        IF(IR1.EQ.0) GO TO 140
        DO 120 J=1,IR1
          T(NI) = X(MI)
          T(NE) = X(ME)
          NI = NI+1
          MI = MI+L
          NE = NE-1
          ME = ME-L
 120    CONTINUE
```

```
140     IF(JJ.EQ.2) GO TO 180
        IR2 = 2*IR1-IR+1
        L = L-1
        IR = M2-IR-IR2
        MI = MI-1
        ME = ME+1
160   CONTINUE
180    IF( IR1*2.EQ. IR) GO TO 200
      T(NI) = X(MI)
      IF(IR2.EQ.1) GO TO 200
      T(NI) = (X(MI)+X(MI+1))*0.5
200   T(K1) = XI
      T(NK1+1) = XE
C    WE CHOOSE 2K ADDITIONAL KNOTS FOR OUR B-SPLINE REPRESENTATION
      F1 = T(K1+1)-XI
      F2 = XE-T(NK1)
      DO 220 J=1,K
        I = K1-J
        IN = N-I
        T(I) = T(I+1)-F1
        T(IN+1) = T(IN)+F2
220   CONTINUE
C   THE ELEMENTS OF A AND Z ARE COMPUTED
      DO 240 IR=1,NK1
        Z(IR) = 0.
        DO 240 IK=1,K1
          A( IR, IK) = 0.
240   CONTINUE
      L = K1
      DO 360 IT=1,M
        ARG = X( IT)
        IF(ARG.GE.T(L+1) .AND. L.NE.NK1) L = L+1
C    IF T(L) <=X< T(L+1) ONLY THE NORMALISED B-SPLINES NL-K,K(X),
C    ...NL,K(X) HAVE A VALUE DIFFERENT FROM ZERO.WE  COMPUTE
C    H1(K1-I) = NL-I,K(X) ,I=0,1,...K
        H1(1) = 1.
        DO 300 J=1,K
          H2(1) = 0.
          J1 = J+1
          DO 260 I=1,J
            LI = L+I
            LJ = LI-J
            F1 = H1(I)/(T(LI)-T(LJ))
            H2( I) = H2( I)+F1*(T(L I)-ARG)
            H2(I+1) = F1*(ARG-T(LJ))
260       CONTINUE
          DO 280 I=1,J1
            H1( I) = H2(I)
280       CONTINUE
300     CONTINUE
C    A IS A (2K+1) BANDED POSITIVE DEFINITE MATRIX.THE ELEMENTS
C    ARE STORED IN COMPACT FORM
        LK = L-K
        DO 340 L1=LK,L
          K5 = L1-L+K1
          Z(L1) = Z(L1)+H1(K5)*Y(IT)*W(IT)
          DO 320 L2=L1,L
            K6 = L2-L+K1
            IR = L2
            IK = K1-L2+L1
```

```
            A(IR,IK) = A(IR,IK)+H1(K5)*H1(K6)*W(IT)
 320      CONTINUE
 340    CONTINUE
 360  CONTINUE
C   WE FIRST DETERMINE THE LEAST SQUARES SPLINE FUNCTION (P=INFIN.)
      ITER = 0
      DO 380 IR=1,NK1
        DO 380 IK=1,K1
          G(IR,IK) = A(IR,IK)
 380  CONTINUE
      LS = K1
C   DECOMPOSITION OF THE POSITIVE DEFINITE BANDMATRIX G=A+B/P
 400  CALL BANDET(G,LS,NK1,IER)
      IF(IER.EQ.0) GO TO 420
C   G WAS FOUND TO BE NOT POSITIVE DEFINITE
C      ITER=0 (G=A) THIS RESULT IS THEORETICALLY IMPOSSIBLE
C      ITER=1 OUR INITIAL CHOISE OF P WAS TOO SMALL (B SINGULAR)
      IF( ITER.EQ.0) GO TO 960
      P = P*10.
      GO TO 700
C   WE SOLVE THE SYSTEM OF EQUATIONS G*C=Z AND FIND THE
C   COEFFICIENTS OF THE B-SPLINE REPRESENTATION
 420  CALL BANSOL(G,LS,NK1,Z,C)
      IF(S.EQ.0.) GO TO 910
C   WE COMPUTE FP = F(P)
      FP = 0.
      L = K1
      DO 440 IT=1,M
        ARG = X(IT)
        IF(ARG.GE.T(L+1) .AND. L.NE.NK1) L = L+1
        DO 425 I=1,K1
          IK = L+I-K1
          H(I) = C(IK)
 425    CONTINUE
        DO 435 J=2,K1
          DO 435 JJ=J,K1
            I = J+K1-JJ
            LI = L+I-K1
            LJ = L+I-J+1
            H(I) = ((ARG-T(LI))*H(I)+(T(LJ)-ARG)*H(I-1))/(T(LJ)-T(LI))
 435    CONTINUE
        FP = FP+W(IT)*(Y(IT)-H(K1))**2
 440  CONTINUE
C   TEST ON CONVERGENCE
      IF(ABS((FP-S)/S).LT.TOL) RETURN
      IF(ITER.NE.0) GO TO 600
C   TEST WHETHER THE NUMBER OF KNOTS HAS TO BE INCREASED
      IF(FP.GT.S) GO TO 800
C   THE ELEMENTS OF B ARE COMPUTED.B IS A (2K+3) BANDED POSITIVE
C   SEMIDEFINITE MATRIX.THE ELEMENTS ARE STORED IN COMPACT FORM
      LS = LS+1
      DO 460 IR=1,NK1
        DO 460 IK=1,LS
          B(IR,IK) = 0.
 460  CONTINUE
      DO 560 L=LS,NK1
        DO 480 J=1,K1
          L1 = L+J
          L2 = L1-LS
          K5 = K1+J
```

```
            H(J) = T(L)-T(L2)
            H(K5) = T(L)-T(L1)
 480      CONTINUE
          F1 = -H(LS)*H(K1)
          DO 540 J=1,LS
            DO 520 I=J,LS
              F = 1.
              DO 500 LL=1,K1
                L1 = J+LL-1
                L2 = I+LL-1
                F2 = H(L1)*H(L2)/F1
                F = F*F2
 500          CONTINUE
              IR = I-1+L-K1
              IK = LS-I+J
              L1 = L+J-1
              L2 = L+I-1
              L3 = L1-K1
              L4 = L2-K1
              B(IR,IK) = B(IR,IK)+(T(L1)-T(L3))*(T(L2)-T(L4))/(F*F1)
 520        CONTINUE
 540      CONTINUE
 560    CONTINUE
C   WE CHOOSE THE INITIAL VALUE OF P
        ITER = 1
        P = 0.0001/WMAX
        GO TO 700
C   TEST WHETHER S > F(P)
C     ITER = 1: THE LEAST SQUARES POLYNOMIAL OF DEGREE K IS THE
C                 TRIVIAL SOLUTION OF OUR SMOOTHING PROBLEM
C     ITER > 1: F(P)**-0.5 IS NOT CONCAVE.(THEORETICALLY IMPOSSIBLE)
 600    IF(S.GT.FP) IF(ITER-2) 920,950,950
C   TEST ON THE NUMBER OF ITERATIONS
        IF(ITER.EQ.MAX) GO TO 940
C   COMPUTATION OF DFP = FIRST DERIVATIVE OF F(P)
        DO 680 I=1,NK1
          L1 = I-K1
          IF(L1.LT.1) L1 = 1
          F = 0.
          DO 620 I2=L1,I
            I3 = I2-I+LS
            F = F+C(I2)*B(I,I3)
 620      CONTINUE
          IF(I.EQ.NK1) GO TO 660
          L1 = I+K1
          IF(L1.GT.NK1) L1 = NK1
          I1 = I+1
          DO 640 I2=I1,L1
            I3 = I+LS-I2
            F = F+C(I2)*B(I2,I3)
 640      CONTINUE
 660      V(I) = F
 680    CONTINUE
        CALL BANSOL(G,LS,NK1,V,C)
        DFP = 0.
        DO 690 I=1,NK1
          DFP = DFP+C(I)*V(I)
 690    CONTINUE
        DFP = -2.*DFP*PINV**3
C   WE CARRY OUT ONE MORE STEP OF THE NEWTON PROCESS
```

```
      FP2 = SQRT(FP)
      P = P+2.*FP2/S2*(S2*FP2-FP)/DFP
      ITER = ITER+1
C   THE ELEMENTS OF G=A+B/P ARE COMPUTED.G IS A (2K+3) BANDED
C   POSITIVE DEFINITE MATRIX.THE ELEMENTS ARE STORED IN COMPACT FORM
 700  PINV = 1./P
      DO 720 IR=1,NK1
        G(IR,1) = PINV*B(IR,1)
        DO 720 IK=2,LS
          G( IR, IK) = A( IR, IK-1)+PINV*B( IR, IK)
 720  CONTINUE
      GO TO 400
C   WE INCREASE THE NUMBER OF KNOTS AND RESTART THE COMPUTATIONS
 800  IF(N.EQ.NMAX) GO TO 910
      IFIB3 = IFIB1+IFIB2
      IFIB1 = IFIB2
      IFIB2 = IFIB3
      N = NMIN+IFIB3
      IF(N.GT.NMAX) N = NMAX
      GO TO 100
C   BORDERLINE CASES
 910  IER = -1
      RETURN
 920  IER = -2
      RETURN
C   ERROR CODES
 930  IER = 1
      RETURN
 940  IER = 2
      RETURN
 950  IER = 3
      RETURN
 960  IER = 4
      RETURN
      END


      SUBROUTINE BANDET(G,LS,NK1,IER)
C   BANDET DECOMPOSES THE 2*LS-1 BANDET NK1*NK1 POSITIVE DEFINITE
C   MATRIX G IN AN UPPER TRIANGULAR MATRIX AND ITS TRANSPOSE USING
C   THE METHOD OF CHOLESKY.THE TRIANGULAR MATRIX IS RETURNED IN G
      INTEGER P,Q,R,S
      DIMENSION G(100,7)
C   ATTENTION : MATRIX G MUST HAVE THE SAME DIMENSIONS AS
C               SPECIFIED IN THE DRIVER PROGRAM
      DO 500 I=1,NK1
        P = 1
        IF(I.LE.LS) P = LS-I+1
        R = I-LS+P
        DO 300 J=P,LS
          S = J-1
          Q = LS-J+P
          Y = G(I,J)
          IF(P.GT.S) GO TO 200
          DO 100 K=P,S
            Y = Y-G(I,K)*G(R,Q)
            Q = Q+1
 100      CONTINUE
 200      IF(J.EQ.LS) GO TO 400
          G(I,J) = Y*G(R,LS)
```

```
         R = R+1
 300     CONTINUE
 400     IF(Y.LE.0.) GO TO 600
         G(I,J) = 1./SQRT(Y)
 500   CONTINUE
C   NORMAL RETURN TO THE DRIVER PROGRAM
       IER = 0
       RETURN
C   MATRIX G WAS FOUND TO BE NOT POSITIVE DEFINITE
 600   IER = -1
       RETURN
       END


       SUBROUTINE BANSOL(G,LS,NK1,Z,C)
C   BANSOL SOLVES THE SYSTEM DECOMPOSED BY BANDET WITH RIGHT
C   HAND SIDE Z.THE SOLUTION IS RETURNED IN C
       INTEGER P,Q,R
       DIMENSION G(100,7),Z(NK1),C(NK1)
C   ATTENTION : MATRIX G MUST HAVE THE SAME DIMENSIONS AS
C               SPECIFIED IN THE DRIVER PROGRAM
       L = LS-1
       DO 300 I=1,NK1
         Y = Z(I)
         IF(I.EQ.1) GO TO 200
         P = 1
         IF(I.LE.LS) P = LS-I+1
         Q = I
         DO 100 J=P,L
           K = P+L-J
           Q = Q-1
           Y = Y-G(I,K)*C(Q)
 100     CONTINUE
 200     C(I) = Y*G(I,LS)
 300   CONTINUE
       DO 600 I=1,NK1
         R = NK1+1-I
         Y = C(R)
         IF(R.EQ.NK1) GO TO 500
         P = 1
         IF(NK1-R.LT.LS) P = LS-NK1+R
         Q = R
         DO 400 J=P,L
           K = P+L-J
           Q = Q+1
           Y = Y-G(Q,K)*C(Q)
 400     CONTINUE
 500     C(R) = Y*G(R,LS)
 600   CONTINUE
       RETURN
       END


       FUNCTION DERIV(T,N,C,NK1,NU,ARG,L)
C   GIVEN THE NORMALISED B-SPLINE REPRESENTATION OF A SPLINE FUNCTION
C   S(X), DERIV COMPUTES THE NU TH DERIVATIVE OF S(X) FOR X=ARG.
       DIMENSION T(N),C(NK1),H(6)
C   H MUST HAVE THE DIMENSION AT LEAST N-NK1 (=K+1)
       DERIV = 0.
       K1 = N-NK1
```

```
      IF(NU.LT.0 .OR. NU.GE.K1) RETURN
      DO 100 I=1,K1
        IK = L+I-K1
        H(I) = C(IK)
  100 CONTINUE
      IF(NU.EQ.0) GO TO 300
      NU1 = NU+1
      DO 200 J=2,NU1
        DO 200 JJ=J,K1
          I = J+K1-JJ
          LI = L+I-K1
          LJ = L+I-J+1
          H( I) = (H( I)-H( I-1))/(T(LJ)-T(L I))
  200 CONTINUE
      IF(NU.EQ.K1-1) GO TO 500
  300 NU2 = NU+2
      DO 400 J=NU2,K1
        DO 400 JJ=J,K1
          I = J+K1-JJ
          LI = L+I-K1
          LJ = L +I-J +1
          H(I) = ((ARG-T(LI))*H(I)+(T(LJ)-ARG)*H(I-1))/(T(LJ)-T(LI))
  400 CONTINUE
  500 DERIV = H(K1)
      IF(NU.EQ.0) RETURN
      DO 600 I=1,NU
        DERIV = DERIV*FLOAT(K1-I)
  600 CONTINUE
      RETURN
      END


      FUNCTION SPLINT(T,N,C,NK1,ALFA,BETA)
C   GIVEN THE NORMALISED B-SPLINE REPRESENTATION OF A SPLINE FUNCTION
C   S(X), SPLINT COMPUTES THE INTEGRAL OF S(X) BETWEEN ALFA AND BETA.
      DIMENSION T(N),C(NK1)
      SPLINT = 0.
      K1 = N-NK1
      A = ALFA
      B = BETA
      MIN = 0
      IF(A-B) 200,700,100
  100 A = BETA
      B = ALFA
      MIN = 1
  200 IF(A.LT.T(K1)) A = T(K1)
      IF(B.GT.T(NK1+1)) B = T(NK1+1)
      DO 500 J=1,NK1
        TJ = T(J)
        JK = J+K1
        TK = T(JK)
        IF(B.LE.TJ) GO TO 600
        IF(A.GE.TK) GO TO 500
        H1 = TK-TJ
        IF(A.LE.TJ) GO TO 300
        H1 = BSPLIN(T,N,K1,J,A)
  300   H2 = 0.
        IF(B.GE.TK) GO TO 400
        H2 = BSPLIN(T,N,K1,J,B)
  400   SPLINT = SPLINT+(H1-H2)*C(J)
```

```
500   CONTINUE
600   SPLINT = SPLINT/FLOAT(K1)
      IF(MIN.EQ.0) RETURN
      SPLINT = -SPLINT
700   RETURN
      END


      FUNCTION BSPLIN(T,N,K1,L,Y)
C     BSPLIN PRODUCES THE VALUE OF THE INTEGRAL OF K1*NL,K1-1(X)
C     BETWEEN Y AND T(L+K1) WHERE NL,K1-1(X) IS THE NORMALISED
C     B-SPLINE, DEFINED ON THE KNOTS T(L),...T(L+K1)
      DIMENSION T(N),H(6)
C     H MUST HAVE THE DIMENSION AT LEAST K1 (=K+1)
      DO 100 I=1,K1
        LI = L+I
        H(I) = 0.
        IF(T(LI).LE.Y) GO TO 100
        H(I) = T(LI)-AMAX1(T(LI-1),Y)
100   CONTINUE
      DO 200 J=2,K1
        DO 200 JJ=J,K1
          I = J+K1-JJ
          LI = L+I
          LJ = LI-J
          H(I) = H(I-1)*(Y-T(LJ))/(T(LI-1)-T(LJ))+H(I)*(T(LI)-Y)/
     <    (T(LI)-T(LJ+1))
200   CONTINUE
      BSPLIN = H(K1)
      RETURN
      END
```