METHODEN DER APPROXIMATIONSTHEORIE, BAND 3
ISNM 30, BIRKHÄUSER, BASEL, 1976

ON CALCULATING WITH B-SPLINES

II. INTEGRATION

Carl de Boor, Tom Lyche, and Larry L. Schumaker

This paper is a continuation of the paper [1] of the same name by the first author in which it is shown how values of B-splines and their derivatives can be computed by stable algorithms based on recursions involving only convex combinations of nonnegative quantities (cf. also Cox [3]). In this paper we consider integrals of B-splines and of B-spline series. In addition, we derive recursions for the computation of integrals of products of B-splines (of possibly different orders and on possibly different knot sequences). As an application, we consider the numerical computation of the Gram matrix which arises in least squares fitting using B-splines.

§ 1. Introduction

We begin by introducing some basic notation. Let k be a positive integer, and suppose $\underline{x}$ denotes a biinfinite sequence of real numbers

$$(1.1) \qquad \ldots\ldots \le x_{-2} \le x_{-1} \le x_0 \le x_1 \le \ldots\ldots$$

with at most k values equal to each other (i.e., $x_i < x_{i+k}$ for all i).

We define

$$(1.2)\quad Q^k_{i,j,\underline{x}}(t) := [x_i,\ldots,x_{i+j}](s-t)^{k-1}_+ .$$

When there is no chance of confusion, we shall usually suppress the $\underline{x}$ subscript on the symbol $Q^k_{i,j,\underline{x}}$. Except for normalization factors, $Q^k_{i,k}$ is the B-spline of order k which is positive on (x_i,x_{i+k}) and is zero outside $[x_i,x_{i+k}]$. Specifically,

$$(1.3)\quad M_{i,k} = k\, Q^k_{i,k}$$

is the B-spline normalized to have unit integral while

$$(1.4)\quad N_{i,k} = (x_{i+k} - x_i)Q^k_{i,k}$$

is the B-spline normalized so that $\sum_i N_{i,k} = 1$.

For $j < k$, $Q^k_{i,j}$ is a polynomial spline which is useful in constructing a basis for natural splines (see e.g. [5,9]). It is easily seen that $Q^k_{i,j}$ is positive on $(-\infty,x_{i+j})$ and is zero on (x_{i+j},∞). We note that

$$(1.5)\quad Q^1_{i,1}(t) = \begin{cases} 1/(x_{i+1}-x_i) & \text{for } x_i \le t < x_{i+1} \\ 0 & \text{otherwise.} \end{cases}$$

Moreover, it is known (cf. [1,9]) that

$$(1.6)\quad Q^k_{i,j}(t) = Q^{k-1}_{i,j-1}(t) + (x_{i+j}-t)Q^{k-1}_{i,j}(t),$$

and if $x_i < x_{i+j}$,

$$(1.7)\quad Q^k_{i,j}(t) = \frac{(t-x_i)Q^{k-1}_{i,j-1}(t)+(x_{i+j}-t)Q^{k-1}_{i+1,j-1}(t)}{(x_{i+j} - x_i)} .$$

These recursions are proved by applying Leibniz's rule for the k^{th} divided difference of a product of two functions to the function $(s-t)^{k-2}_+ (s-t) = (s-t)^{k-1}_+$. Because of the support properties of the $Q^k_{i,j}$, the value of $Q^k_{i,j}(t)$ can be computed recursively by forming only nonnegative combinations of nonnegative quantities.

§ 2. Integrals of B-splines.

In this section we are interested in computing values of $\int_c^d Q^m_{i,j}(t)dt$ for given $0 \le j \le m \le k$ and $c < d$. Since the most interesting case is where $j = m$, we begin with a lemma concerning it. As it is somewhat more convenient to work with a normalized B-spline, we define

$$(2.1)\quad I^m_i(c,d) = m\int_c^d Q^m_{i,m}(t)dt = \int_c^d M_{i,m}(t)dt$$

LEMMA 2.1. For any $1 \le m \le k$,

$$(2.2)\quad I^m_i(c,d) = \sum_{j\ge i} N_{j,m+1}(d) - \sum_{j\ge i} N_{j,m+1}(c).$$

Proof: First, we observe that

$$\begin{aligned}(d/dt)N_{i,m+1}(t) &= (d/dt)([x_{i+1},\ldots,x_{i+m+1}](\cdot -t)^m_+ \\ &\qquad -[x_i,\ldots,x_{i+m}](\cdot -t)^m_+) \\ &= -m([x_{i+1},\ldots,x_{i+m+1}](\cdot - t)^{m-1}_+ \\ &\qquad - [x_i,\ldots,x_{i+m}](\cdot - t)^{m-1}_+) \\ &= M_{i,m}(t) - M_{i+1,m}(t);\end{aligned}$$

and hence,

$$N_{i,m+1}(d) - N_{i,m+1}(c) = \int_c^d [M_{i,m}(t) - M_{i+1,m}(t)]dt.$$

Thus,

$$I_i^m(c,d) = \int_c^d M_{i+1,m}(t)dt + N_{i,m+1}(d) - N_{i,m+1}(c).$$

Repeating this process until the M's no longer have support on (c,d) leads to (2.2). ∎

Concerning the other type of B-splines, we have the following result.

LEMMA 2.2. If $0 \le j \le m-1 < k$, then

$$(2.3)\quad m\int_c^d Q_{i,j}^m(t)dt = Q_{i,j}^{m+1}(c) - Q_{i,j}^{m+1}(d) .$$

Proof: Clearly, we have

$$m\int_u^{x_{i+j}} Q_{i,j}^m(t)dt = m[x_i,\ldots,x_{i+j}] \int_u^{x_{i+j}} (s-t)_+^{m-1}dt$$

$$= [x_i,\ldots,x_{i+j}](s-u)_+^m = Q_{i,j}^{m+1}(u) .$$

Subtracting the values for u = c and u = d, we obtain (2.3). ∎

The identity (2.3) is also true for j = m, provided that $x_i < x_{i+m}$, and thus provides an alternative way of computing integrals of the ordinary B-splines. The relation is generally not true for j = m and $x_i = x_{i+m}$, since in this case we want to consider the integral to be 0, while the right-hand-side may not be zero in view of the

fact that

$$(2.4)\quad Q_{i,m}^{m+1}(t) = (x_i - t)_+^0$$

in this case.

To use Lemma 2.2, we may use the recursions of section 1 to compute the two B-splines, and then do the subtraction. It is also possible to derive a direct recursion in which only nonnegative combinations of nonnegative quantities appear, and for which no subtraction is needed at the end. For convenience, we set

$$(2.5)\quad T_i^m(c,d) = Q_{i,m}^{m+1}(c) - Q_{i,m}^{m+1}(d) .$$

LEMMA 2.3. For any c < d,

$$(2.6)\quad T_i^1(c,d) = \begin{cases} I_i^1(c,d) & , \text{ if } x_i < x_{i+1} \\ (x_i-c)_+^0 - (x_i-d)_+^0 & , \text{ if } x_i = x_{i+1} , \end{cases}$$

where

$$(2.7)\quad I_i^1(c,d) = \begin{cases} (d-c)/(x_{i+1}-x_i) & \text{if } x_i \le c \le d \le x_{i+1} \\ (x_{i+1}-c)/(x_{i+1}-x_i) & \text{if } x_i \le c \le x_{i+1} \le d \\ (d-x_i)/(x_{i+1}-x_i) & \text{if } c \le x_i \le d \le x_{i+1} \\ 1 & \text{if } c \le x_i < x_{i+1} \le d \\ 0 & \text{otherwise} . \end{cases}$$

Moreover,

$$(2.8)\quad T_i^m(c,d) = \frac{(c-x_i)T_i^{m-1}(c,d)+(x_{i+m}-c)T_{i+1}^{m-1}(c,d)}{(x_{i+m} - x_i)} +$$

$$+ (d-c)Q_{i,m}^m(d),$$

and hence, interchanging c and d in (2.8), also

$$(2.9)\quad T_i^m(c,d) = \frac{(d-x_i)T_i^{m-1}(c,d)+(x_{i+m}-d)T_{i+1}^{m-1}(c,d)}{(x_{i+m}-x_i)} + (d-c)Q_{i,m}^m(c).$$

Proof: The computation of $I_i^1(c,d)$ is easily carried out using (1.5). Now if we use the recursion (1.7) in (2.5), we obtain

$$\begin{aligned}T_i^m(c,d)(x_{i+m}-x_i) &= (c-x_i)Q_{i,m-1}^m(c)+(x_{i+m}-c)Q_{i+1,m-1}^m(c)\\ &\quad -(d-x_i)Q_{i,m-1}^m(d)-(x_{i+m}-d)Q_{i+1,m-1}^m(d)\\ &= (c-x_i)[Q_{i,m-1}^m(c)-Q_{i,m-1}^m(d)] +\\ &\quad +(x_{i+m}-c)[Q_{i+1,m-1}^m(c)-Q_{i+1,m-1}^m(d)]\\ &\quad +(d-c)[Q_{i+1,m-1}^m(d)-Q_{i,m-1}^m(d)] .\end{aligned}$$

By the definition of divided difference and using (2.5) again, we obtain (2.8). ∎

We emphasize that the coefficients in front of all nonzero quantities in formulas (2.8) and (2.9) are nonnegative. Coupled with (2.7), these recursions permit the stable evaluation of definite integrals of B-splines without performing a subtraction of two other integrals.

We close this section by mentioning another formula for computing the definite integral of a usual B-spline. If we put $c = x_i$ in (2.8), we obtain

$$T_i^m(x_i,d) = T_{i+1}^{m-1}(x_i,d) + (d-x_i)Q_{i,m}^m(d).$$

Now, if we continue this process of reduction, we obtain

$$(2.10)\quad T_i^m(x_i,d) = \sum_{r=0}^{m-1} (d-x_{i+r})Q_{i+r,m-r}^{m-r}(d) .$$

When $x_i < x_{i+m}$ this gives the value of $I_i^m(x_i,d)$, and formula (2.10) is exactly that of Gaffney [4], which, incidentally, provided the impetus for this paper.

§ 3. Integrals of B-spline expansions.

In this section, we consider the question of finding the indefinite or definite integral of a B-spline expansion of the form

$$(3.1)\quad s(t) = \sum_{i=1}^{N} c_i\, Q_{i,k}^k(t) .$$

Define

$$(3.2)\quad S(t) = \int_{x_1}^t s(u)du .$$

It follows that S is a polynomial spline of order k+1 with the same knot sequence $\underline{x}$ as s. Thus, it can also be written as a linear combination of B-splines over the knot sequence $\underline{x}$, but of order k+1, of course.

LEMMA 3.1. If s is given by (3.1), then its indefinite integral S as in (3.2) is

$$(3.3)\quad S(t) = \sum_{i=1}^{N} c_i^{(-1)} Q_{i,k+1}^{k+1}(t) , \quad x_1 \le t \le x_{N+k}$$

where

$$(3.4)\quad c_i^{(-1)} = \frac{(x_{i+k+1}-x_i)}{k} \sum_{j=1}^{i} c_j \ .$$

Proof: If S is given by (3.3), then

$$S'(t)=k \sum_{i=1}^{N} c_i^{(-1)} \frac{[Q^k_{i,k}(t)-Q^k_{i+1,k}(t)]}{(x_{i+k+1} - x_i)} = s(t)=\sum_{i=1}^{N} c_i Q^k_{i,k}(t).$$

By the linear independence of the Q's, the coefficients must agree, and (3.4) follows. ■

This lemma permits the evaluation of the definite integral $\int_a^b s(t)dt$ for any $a < b$ in terms of two evaluations of the indefinite integral $S(t)$. An alternate approach to computing definite integrals is to use the formulae of section 2. We have

LEMMA 3.2. Let p and r be such that $x_p \le a < x_{p+1}$ and $x_r < b \le x_{r+1}$.

Then

$$(3.5)\quad \int_a^b s(t)dt=\frac{1}{k}\Big[\sum_{\max\{1,p+1-k\}}^{p} c_i I_i^k(a,b)+ \sum_{p+1}^{r-k} c_i + \sum_{\max\{p+1,r+1-k\}}^{\min\{r,N\}} c_i I_i^k(a,b)\Big] \ .$$

Proof: By the support properties of the Q's and the fact that

$$\int_{x_i}^{x_{i+k}} Q^k_{i,k}(t)dt = 1/k$$

we immediately obtain (3.5). ■

The required $I_i^k(a,b)$ can be computed recursively by the formula (2.8) for those in the first sum and formula (2.9) for those in the third sum. The total amount of calculation is essentially equivalent to two evaluations of the indefinite integral.

§ 4. Inner products of B-splines.

We begin with the observation that

$$I_i^m(y_j,y_{j+1}) = m \int_{y_j}^{y_{j+1}} Q^m_{i,m,\underline{x}}(t)dt$$

$$= (y_{j+1}-y_j)m \int_{-\infty}^{\infty} Q^m_{i,m,\underline{x}}(t)Q^1_{j,1,\underline{y}}(t)dt \ ,$$

where the $\underline{x}$ and $\underline{y}$ subscripts on the Q's indicate the knot sequences over which they are defined (cf. section 1). This shows that the definite integral of a B-spline is the integral of the product of two B-splines, and suggests that we consider, more generally, the evaluation of inner products of B-splines of arbitrary orders on two (possibly) different knot sequences.

Suppose $\underline{x}$ is a biinfinite sequence as in (1.1) with $x_i < x_{i+k}$, and that $\underline{y}$ is a similar sequence with $y_j < y_{j+h}$. Then for arbitrary i,j and $1 \le m \le k$, $1 \le n \le h$, we are interested in the inner-products

$$(4.1)\quad I_{i,j}^{m,n} = \frac{(m+n-1)!}{(m-1)!(n-1)!}\int_{-\infty}^{\infty} Q_{i,m,\underline{x}}^{m}(t)Q_{j,n,\underline{y}}^{n}(t)dt\ .$$

If $x_i = x_{i+m}$ or $y_j = y_{j+n}$, we take $I_{i,j}^{m,n} = 0$. We have introduced the factorials in the definition (4.1) in order to simplify certain recursions to be obtained for their stable computation. The special case where the knot sequences $\underline{x}$ and $\underline{y}$ are one and the same is discussed in greater detail in the following section.

We now show that the integrals $I_{i,j}^{m,n}$ are closely connected with certain divided differences. Let

$$(4.2)\quad T_{i,j}^{m,n} = (-1)^m[x_i,\ldots,x_{i+m}]_t[y_j,\ldots,y_{j+n}]_s(s-t)_+^{m+n-1}\ .$$

$T_{i,j}^{m,n}$ is well-defined as long as not both $x_i = x_{i+m}$ and $y_j = y_{j+n}$. In fact, we observe that

$$(4.3)\quad T_{i,j}^{m,n} = \begin{cases} \binom{m+n-1}{n} Q_{i,m}^{m}(y_j)\ , & y_j = y_{j+n}\ ,\ n \geq 0\ , \\ \binom{m+n-1}{n} Q_{j,n}^{n}(x_i)\ , & x_i = x_{i+m}\ ,\ n \geq 0\ . \end{cases}$$

To see the connection with the integrals $I_{i,j}^{m,n}$, suppose we insert the function $f(t) = (m-1)!(n-1)!Q_{j,n,\underline{y}}^{m+n}(t)/(m+n-1)!$ in the well known formula (cf. e.g. [5])

$$[x_i,\ldots,x_{i+m}]f = \int_{-\infty}^{\infty} Q_{i,m,\underline{x}}^{m}(t)f^{(m)}(t)dt/(m-1)!\ .$$

When then obtain

$$(4.4)\quad I_{i,j}^{m,n} = (-1)^m[x_i,\ldots,x_{i+m}]Q_{j,n,\underline{y}}^{m+n}$$

$$= T_{i,j}^{m,n}$$

when $x_i < x_{i+m}$ and $y_j < y_{j+n}$.

We now give some recursions for the quantities $T_{i,j}^{m,n}$.

LEMMA 4.1. Suppose $1 \leq m \leq k$ and $1 \leq n \leq h$. Then

$$(4.5)\quad T_{i,j}^{m,n} = \frac{(x_{i+m}-y_j)T_{i,j}^{m,n-1}+(y_{j+n}-x_{i+m})T_{i,j+1}^{m,n-1}}{(y_{j+n}-y_j)} + T_{i,j}^{m-1,n}\ ,$$

(suitable when $x_{i+m} \leq y_{j+n}$ and $y_j < y_{j+n}$),

$$(4.6)\quad T_{i,j}^{m,n} = \frac{(x_i-y_j)T_{i,j}^{m,n-1}+(y_{j+n}-x_i)T_{i,j+1}^{m,n-1}}{(y_{j+n}-y_j)} + T_{i+1,j}^{m-1,n}\ ,$$

(suitable when $y_j \leq x_i$ and $y_j < y_{j+n}$),

$$(4.7)\quad T_{i,j}^{m,n} = \frac{(y_j-x_i)T_{i,j}^{m-1,n}+(x_{i+m}-y_j)T_{i+1,j}^{m-1,n}}{(x_{i+m}-x_i)} + T_{i,j+1}^{m,n-1}\ ,$$

(suitable when $x_i \leq y_j$ and $x_i < x_{i+m}$),

$$(4.8)\quad T_{i,j}^{m,n} = \frac{(y_{j+n}-x_i)T_{i,j}^{m-1,n}+(x_{i+m}-y_{j+n})T_{i+1,j}^{m-1,n}}{(x_{i+m}-x_i)} + T_{i,j}^{m,n-1}\ ,$$

(suitable when $y_{j+n} \leq x_{i+m}$ and $x_i < x_{i+m}$).

(We emphasize that the factors in front of the T's are always nonnegative whenever the T is nonzero, so that only nonnegative combinations of nonnegative quantities need ever be computed).

Proof: It will be enough to prove the first formula, as the others are proved similarly. Using (1.7), we have.

$$(-1)^m T^{m,n}_{i,j} = [x_i,\dots,x_{i+m}]Q^{m+n}_{j,n,\underline{y}}$$

$$= [x_i,\dots,x_{i+m}]_t\left\{\frac{(t-y_j)Q^{m+n-1}_{j,n-1}(t)+(y_{j+n}-t)Q^{m+n-1}_{j+1,n-1}(t)}{(y_{j+n}-y_j)}\right\}.$$

Now, applying Leibniz's rule for the divided difference of a product, (cf. [1]), we obtain

$$(-1)^m(y_{j+n}-y_j)T^{m,n}_{i,j} =$$

$$= (x_{i+m}-y_j)[x_i,\dots,x_{i+m}]Q^{m+n-1}_{j,n-1}+[x_i,\dots,x_{i+m-1}]Q^{m+n-1}_{j,n-1}$$

$$+ (y_{j+n}-x_{i+m})[x_i,\dots,x_{i+m}]Q^{m+n-1}_{j+1,n-1}-[x_i,\dots,x_{i+m-1}]Q^{m+n-1}_{j+1,n-1}$$

Dividing through by $(-1)^m(y_{j+n}-y_j)$ and identifying terms leads to equation (4.5). ■

The recursions in Lemma 4.1 can be used to reduce the upper indices on $T^{m,n}_{i,j}$ until we reach one of the cases in (4.3), where values of ordinary B-splines are needed. These in turn can be computed conveniently using the recursions given in section 1. We discuss the details of carrying out this recursion for computing $T^{m,n}_{i,j}$ in the next section for the case $\underline{x} = \underline{y}$. We close this section by mentioning that recursions similar to those in Lemma 4.1 can also be obtained for the expressions

$$T^{m,n}_{i,j,r} = (-1)^m[x_i,\dots,x_{i+m}]_t[y_j,\dots,y_{j+n}]_s(s-t)^{r-1}_+,$$

for general r.

§ 5. Computing the Gram matrix.

Let $x_{n_1} \le \dots \le x_{n_2}$ be a sequence of real numbers with at most k repetitions, and let $\{Q^k_{i,k}\}_{i=n_1}^{n_2-k}$ be the corresponding B-splines. Then in least squares approximation by splines using the basis $\{Q_i\}_{n_1}^{n_2-k}$ is necessary to compute the matrix

$$(5.1)\quad G = (G_{ij})_{i,j=n_1}^{n_2-k},\quad G_{ij}=\int_{-\infty}^{\infty}Q^k_{i,k}(t)Q^k_{j,k}(t)dt\ .$$

(This matrix also arises in an algorithm for computing natural interpolating splines; cf. [6,7,8]). Clearly G is symmetric, and in view of the support properties of the Q's, it is also banded, with $G_{ij} = 0$ whenever $|i-j| \ge k$.

In this section we want to discuss several methods for computing the matrix G, or what is equivalent, the matrix $I = (I_{ij})$

$$I_{ij} := I^{k,k}_{i,j} = \frac{(2k-1)!}{(k-1)!^2}\,G_{i,j}\ .$$

In view of (4.2), it is clear that the matrix I can be computed directly using divided differences. If the computation is properly arranged, this can be accomplished

in order Nk^2 operations (cf.,e.g., [7]). This will be perfectly suitable if the order is not too high, and if the spacing of the knots is relatively uniform. If not, the definition of divided difference implies that numerical difficulties may be encountered, (cf. the example in section 6).

As a second approach to computing the matrix I, we may utilize the recursions of section 4 with $\underline{x} = \underline{y}$. In view of (1.5), we observe that if $x_i < x_{i+1}$, then

$$I^{1,0}_{i,i} = I^{1,1}_{i,i} = 1/(x_{i+1} - x_i) .$$

The recursion may be carried out using only (4.7) and (4.8), where the first of these is used if $j > i$ and the second if $j < i$. For $j = i$ the choice depends on the multiplicity of the knots; we use (4.8) in case $x_{i+1} < x_{i+m}$, and (4.7) otherwise.

With some care, it is possible to write a program to carry out this algorithm. In section 6 we include an ALGOL procedure for computing I. Since only positive combinations of nonnegative quantities are computed, this method is extremely stable. The number of operations involved is of order Nk^3 however. On the other hand, the computation provides the Gram matrices $(I^{m,m}_{i,j})$ of all orders $1 \le m \le k$ in the process of computing the Gram matrix of order k. (A similar phenomenon occurs in the stable computation of the collocation matrix $(Q^k_i(x_j))$ using the recurrence (1.7). Computation by divided differences can be carried out with order Nk operations while the recursions require Nk^2. On the other hand, all of the collocation matrices of lower orders are produced as intermediate results if the recursions are used).

Finally, we want to mention a third method for computing the matrix I which is also stable, and also involves order Nk^3 operations. It is well-known that there exist points $-1 \le \tau_1 < \ldots < \tau_k \le 1$ and positive coefficients $(A_i)_1^k$ so that the Gaussian quadrature formula

$$\sum_{i=1}^{k} A_i g(\tau_i) \approx \int_{-1}^{1} g(t)dt$$

is exact for all polynomials $g \in \mathcal{P}_{2k}$. Now if $j \ge i$, then

$$A_{ij} = \int_{x_j}^{x_{i+k}} Q^k_{i,k}(t)Q^k_{j,k}(t)dt = \sum_{v=j}^{i+k-1} \int_{x_v}^{x_{v+1}} Q^k_{i,k}(t)Q^k_{j,k}(t)dt .$$

But in each of these subintervals both $Q^k_{i,k}$ and $Q^k_{j,k}$ are polynomials of order k, so the product is a polynomial of order 2k-1. Thus each of these pieces can be computed exactly using the Gaussian quadrature formula obtained by converting the interval $[x_v, x_{v+1}]$ into $[0,1]$. The values of the Q's at the k points in this interval needed for the quadrature formula can be computed using the B-spline recursion (1.7); cf. the packages in [2]. For convenience, we include an ALGOL procedure for computing the least squares matrix I using Gaussian quadrature in the following section.

When performing least squares fitting of a given function f by polynomial splines of order k, it is also necessary to compute the integrals

$$\int_{x_i}^{x_{i+k}} Q^k_{i,k}(t)f(t)dt, \quad i = 1,\ldots,N.$$

As these usually would have to be computed by a quadrature formula anyway, it seems likely that the use of Gauss quadrature to compute the least squares matrix is preferable to the two methods metioned earlier for general knot sequences.

We conclude this section by mentioning that with equally spaced knots it is relatively easy to compute the least squares entries by hand in view of the relative simplicity of the B-splines in this case. For comparison and checking purposes, we list the results for orders k = 2,3, with $x_i = i$, i = 1,..., N+k.

(5.3) $I^{2,1}_{i,i} = I^{2,1}_{i,i+1} = 1/2,\ I^{2,2}_{i,i} = 1,\ I^{2,2}_{i,i+1} = 1/4$

(5.4) $I^{3,1}_{i,i} = I^{3,1}_{i,i+2} = 1/6,\ I^{3,1}_{i,i+1} = 2/3$

$I^{3,2}_{i,i+1} = I^{3,2}_{i,i} = 11/12 = 9.16666666666,-1$

$I^{3,2}_{i,i+2} = 1/12 = 8.33333333333,-2$

$I^{3,3}_{i,i} = 11/6 = 1.83333333333$

$I^{3,3}_{i,i+1} = 13/18 = 7.22222222222,-1$

$I^{3,3}_{i,i+2} = 1/36 = 2.77777777777,-2$.

§ 6. ALGOL procedures for the Gram matrix.

In this section we give ALGOL procedures for computing the Gram matrix defined in section 5 using the recurrence method and the Gaussian quadrature method outlined there.

We begin with the recurrence method. The input to the following procedure is:

n1 integer , the index of the first knot

n2 integer , the index of the last knot (with n2-n1≥k)

k integer , the order (degree +1) of the spline (k≥2)

x array[n1:n2], the knot sequence in increasing order with $x_i < x_{i+k}$, i = n1,...,n2-k.

The output of the procedure is the array c[n1:n2-k,0:k-1], where

$$c_{ij} = \frac{(2k-1)!}{[(k-1)!]^2}\begin{cases}\int Q^k_{ik}(t)Q^k_{i+j,k}(t)dt, & i=n1,\ldots,n2-k \\ & j=0,1,\ldots,\min(k-1,n2-k-i) \\ 0 & \text{otherwise}\end{cases}$$

```
procedure RR(n1,n2,k,x,c);
value  n1,n2,k; integer n1,n2,k; array x,c;
begin
  integer k1;k1:=k-1;
  begin
    integer i,j,l,m,i1,i2,j1,l1,l2,m1,m2,n2m;
    real xi,e;
    array A[n1:n2,-k:k,1:k], Q[n1:n2,1:k],D[0:k1];
    D[0] := 0;
    for i := n1 step 1 until n2 do
    for l := 1 step 1 until k do
```

```
begin
  Q[i,1] := 0;
  for j := -k step 1 until k do A[i,j,1] := 0
  end zero;
  for m := 2 step 1 until k do
  begin
    m1 := m-1 ; m2 := m-2; n2m := n2-m;
    for i := n1 step 1 until n2m do
    if x[i+m]  > x[i] then
    begin
      xi := x[i+m]-x[i]; i1 := i+1;
      for j := 1 step 1 until m1 do
        D[j] := (x[i+j]-x[i])/xi;
      i2 := if x[i+m] > x[i+1] then -1 else 0;
      A[i,0,1] := Q[i,1] := if m = 2 then 1/xi
                            else D[1] * Q[i,1];
      for j := 1 step 1 until m2 do
      begin j1 := j+1; e := A[i1,j-1,1];
        Q[i,j1] := Q[i1,j]+D[j1]*(Q[i,j1]-Q[i1,j]);
        A[i,j,1] := e+Q[i,j1]+D[j]*(A[i,j,1]-e)
      end j;
      A[i,m1,1] := Q[i,m1];
      for l := 2 step 1 until m1 do
      begin l1 := l-1; A[i,-l1,l] := D[1]*A[i,-l1,l];
        for j := 1-l1 step 1 until i2 do
        begin e := A[i1,j-1,l];
          A[i,j,l] := e+A[i,j,l1]+D[j+l]*(A[i,j,l]-e)
        end j;
        for j := i2+1 step 1 until m1 do
        begin e := A[i1,j-1,l];
          A[i,j,l] := e+A[i,j+1,l1]+D[j]*(A[i,j,l]-e)
        end j
      end l;
```



```
      l2:= n1-i; if l2 < -m1 then l2 := -m1;
      for j := l2 step 1 until -1 do
      begin e := A[i+j,1-j,m1];
        A[i,j,m] := e+A[i,j,m1]+D[j+m]*(A[i+j,-j,m1]-e)
      end j;
      A[i,0,m] := if i2 = -1 then 2*A[i,1,m1]
                  else 2*A[i,0,m1]
    end i;
    if m < k then
    begin
      for i := n1 step 1 until n2m do
      for j := 1 step 1 until m1 do
        A[i,j,m] := A[i+j,-j,m]
    end
  end m;
  for i := n1 step 1 until n2m do
  for j := 0 step 1 until k1 do c[i,j] := A[i+j,-j,k]
 end
end RR;
```

The following procedure computes the original Gram matrix G defined in (5.1), (without the factorials in I). The input consists of the same quantities as before, (except that now we limit $k \le 6$). The output in this case is the array a[n1:n2-k,0:k-1] given by

$$a_{ij} = \begin{cases} \int_{-\infty}^{\infty} Q^k_{i,k}(t) Q^k_{i+j,k}(t)\,dt, & i = n1,\ldots,n2-k \\ & j = 0,..,\min(k-1,N2-k-1) \\ 0 & \text{, otherwise.} \end{cases}$$

```
procedure GQ(n1,n2,k,x,a);
value n1,n2,k; integer n1,n2,k; array x,a;
begin
 integer i,j,u,v,i1,i2,n,n3,k1;
```

```
  real t,t1,t2,h,xv;
  array Q[n1:n2],w,z[1:k];
  switch s := L2,L3,L4,L5,L6;
  goto s[k-1];
L2: w[2] := 1; z[2] := 0.5773502692; goto next;
L3: w[3] := 0.5555555556; w[2] := 0.8888888889;
    z[3] := 0.7745966692; z[2] := 0; goto next;
L4: w[4] := 0.3478548451; w[3] := 0.6521451549;
    z[4] := 0.8611363116; z[3] := 0.3399810436;
    goto next;
L5: w[5] := 0.2369268851; w[4] := 0.4786286705;
    w[3] := 0.5688888889; z[5] := 0.9061798459;
    z[4] := 0.5384693101; z[3] := 0; goto next;
L6: w[6] := 0.1713244924; w[5] := 0.3607615730;
    w[4] := 0.4679139346; z[6] := 0.9324695142;
    z[5] := 0.6612093865; z[4] := 0.2386191861;
next: n := n2-k; n3 := n2-1; i1 := k÷2; k1 := k-1;
  for i := 1 step 1 until i1 do
  begin
    w[i] := w[k+1-i]; z[i] := -z[k+1-i]
  end weights and nodes;
  for i := n1 step 1 until n do
  for j := 0 step 1 until k1 do a[i,j] := 0;
  for v := n1 step 1 until n3 do
  if x[v+1] > x[v] then
  begin
    h:=(x[v+1]-x[v])/2; xv:=(x[v]+x[v+1])/2; Q[v+1]:=0;
    for u := 1 step 1 until k do
    begin
      Q[v] := 0.5/h; t := h*z[u]+xv; i1 := i2 := v;
      for n := 2 step 1 until k do
      begin
        if i1 > n1 then begin i1 := i1-1; Q[i1] := 0 end;
        if i2 > n2-n then i2 := i2-1;
```



```
        for i := i1 step 1 until i2 do
        Q[i]:=Q[i+1]+(t-x[i])*(Q[i]-Q[i+1])/(x[i+n]-x[i])
      end n;
      t1 := h*w[u];
      for i := i1 step 1 until i2 do
      begin n := i2-i; t2:= t1*Q[i];
        for j := 0 step 1 until n do
          a[i,j] := a[i,j]+t2*Q[i+j]
      end i
    end u
  end v
end GQ;
```

If the procedure GQ is to be used for k > 6, more quadrature weights and nodes must be included. Moreover, if the machine has more than 10 decimal digits of accuracy, then the appropriate number of figures for these constants should be given.

Both of the procedures RR and GQ were written by the second named author, and have been tested at the University of Oslo (on a CDC 3300) and at the University of Munich (on a Telefunken TR 440). It was found that RR and GQ both use essentially the same amount of time, and also produced numbers which never differed by more than one unit in the 10^{th} figure (after scaling the output of GQ with the appropriate factorials). On the other hand, with nonuniform knots, we found that the divided difference scheme for computing the matrix I was quite inaccurate when the knot spacing ratio became large.

To give a specific example, we computed the values of I_{55}^{44} using the knots $5,6,6+10^{-r},8,9$; i.e.

$$I_{55}^{44} = 140 \int_{-\infty}^{\infty} [Q_{5,4}(t)]^2 dt$$

$$= [5,6,6+10^{-r},8,9]_s[5,6,6+10^{-r},8,9]_t(s-t)_+^7$$

for values of r = 0,1,...,9. As the following table shows, the values produced by RR and GQ agreed to 10 figures, while the values found by divided differences, (using essentially the program in [7]), became increasingly inaccurate with r. We have underlined the first figure in error in the table of values produced by divided differences. For further details and other numerical tests, see Lyche[11].

r	RR and GQ	DD
0	4.194444445	4.194444445
1	4.066497736	4.066497716
2	4.040109644	4.040109621
3	4.037345542	4.037344000
4	4.037067900	4.037048708
5	4.037040124	4.036818046
6	4.037037346	4.037239721
7	4.037037068	4.020766567
8	4.037037040	4.041030623
9	4.037037037	2.016235838

We should also remark that it is easy to modify the procedure GQ to compute approximations to the integrals

$$r_i = \int_{-\infty}^{\infty} f(t)Q_{i,k}(t)dt, \quad i = n1,...,n2-k$$

needed in performing least squares using the B-splines $\{Q_{i,k}\}_{n1}^{n2-k}$. Indeed, if f is a real procedure, then the quantities $r_{n1},...,r_{n2-k}$ will be stored in the k^{th} column of the matrix a provided we modify the procedure GQ as follows:

i) change k1 to k in the statement which is 6 lines below the label " next"; ie., to

```
for j := 0 step 1 until k do a[i,j] := 0;
```

ii) include an extra statement 6 lines above "end GQ;", namely,

```
begin n:=i2-i;t2:=t1*Q[i]; A[i,k] := A[i,k]+t2*f(t);
```

References.

1. de Boor, C., On calculating with B-splines, J. Approx. Th. 6 (1972), 50-62.

2. de Boor, C., Subroutine package for calculating with B-splines, MRC TSR 1333, to appear, SIAM J. Numer. Anal.

3. Cox, M.G., The numerical evaluation of B-splines, J. Inst. Math. and Appl. 10(1972), 134-149.

4. Gaffney, P.W., The calculation of indefinite integrals of B-splines, CSS 10, AERE Harwell, Oxfordshire, England, July, 1974.

5. Greville, T.N.E., Introduction to spline functions, in Theory and Application of Spline Functions, T.N.E. Greville, ed., Academic Press, New York, 1969, 1-35.

6. Greville, T.N.E., Numerical procedures for interpolation by spline functions, SIAM J. Numer. Anal. 1(1964), 53-68.

7. Herriot, J.G. and C.H. Reinsch, Procedures for natural spline interpolation, Algorithm 472, Comm. A.C.M. 16(1973), 763-768.

8. Jerome, J.W. and L.L. Schumaker, A note on obtaining natural spline functions by the abstract approach of Atteia and Laurent, SIAM J. Numer. Anal. 5(1968), 657-663.

9. Lyche, Tom and Larry L. Schumaker, Computation of smoothing and interpolating natural splines via local bases, SIAM J. Numer. Anal.10(1973),1027-1038.

10. Lyche, Tom and Larry L. Schumaker, Procedures for computing smoothing and interpolating natural splines, Comm. A.C.M. 17(1974), 463-467.

11. Lyche, Tom, Computation of B-spline Gram matrices, ISBN 82-553-0226-3, No. 7, Mathematics Institute, University of Oslo, 1975.

Acknowledgment.

The last named author was supported in part by the Deutsche Forschungsgemeinschaft and by the United States Air Force under grant AFOSR 74-2598A.

Carl de Boor
Mathematics Research Center
University of Wisconsin
Madison, Wisconsin 53706

Tom Lyche
Department of Mathematics
University of Oslo
Oslo 3, Norway

Larry L. Schumaker
Department of Mathematics
University of Texas
Austin, Texas 78712

SIMULTANAPPROXIMATION BEI RANDWERTAUFGABEN

VON

Elsbeth Bredendiek und Lothar Collatz, Hamburg

Zusammenfassung:

Für verschiedene Typen von Randwertaufgaben kann man mit Hilfe von Simultanapproximation Näherungslösungen aufstellen. Man hat bei der Simultanapproximation die Möglichkeit, durch geeignete Wahl des Typs der Näherung und durch verschiedene Gewichtung in den einzelnen Komponenten dem zugrundeliegenden realen Problem weitgehend Rechnung zu tragen und oft relativ schnell und einfach gute Näherungen zu erhalten. Dies wird an verschiedenen numerischen Beispielen unter Verwendung von H-Mengen illustriert.
Diese Beispiele sind ausführlich durchgerechnet und durchweg sehr einfach gewählt, um die Methode besser hervortreten zu lassen; es sind meist lineare (aber auch eine nichtlineare) Randwertaufgaben bei partiellen Differentialgleichungen und eine Integralgleichung 1. Art.

Darüber hinaus sollen hier weitere Kriterien für die Wahl der Gewichtsfaktoren angegeben werden. Im Falle der Gültigkeit von Monotoniesätzen gelangt man dann sogar zu exakten Einschließungen für die Lösungen der Randwertaufgaben.

1. Problemstellung:

Es sei B ein Gebiet des n-dimensionalen Punktraumes R^n. Für eine Funktion $u(x_j)$ der Koordinaten $x_1,\ldots x_n$ sei eine lineare oder nichtlineare Differentialgleichung