J. Inst. Maths Applics (1972) **10**, 134–149

The Numerical Evaluation of *B*-Splines*

M. G. Cox

Division of Numerical Analysis and Computing, National Physical Laboratory, Teddington, Middlesex

[Received 29 October 1971]

The conventional method of evaluating *B*-splines by means of divided differences is numerically unstable. A relation between *B*-splines of consecutive degrees is derived which enables *B*-splines to be evaluated in a stable and efficient fashion. Analyses of error growth in the conventional and the stable methods are carried out. For the stable method an *a priori* relative error bound is obtained.

1. Introduction

A NUMBER of authors (including Anselone & Laurent, 1968; Carasso & Laurent, 1968; Herriot & Reinsch, 1971; LaFata & Rosen, 1970; Schumaker, 1969) have proposed *B*-splines as a convenient basis for problems of interpolation and smoothing by means of spline functions with fixed knots. In forming the linear algebraic equations defining the multipliers of the basis functions and in evaluating the subsequent approximating spline it is necessary to employ an algorithm for evaluating *B*-splines. Most algorithms so far proposed (for example, Schumaker, 1969; Herriot & Reinsch, 1971) have evaluated the *B*-splines from their definition in terms of divided differences. This approach may be satisfactory for splines of low degree with relatively uniform knot spacing. However, for splines of high degree, or splines with relatively non-uniform knot spacing, considerable loss of accuracy may result (see the comments by Schumaker, 1969 and Herriot & Reinsch, 1971). An alternative method described here is shown to be numerically stable and has been used successfully for *B*-splines of low and high degrees with uniform and highly non-uniform knot spacing. *A posteriori* and *a priori* error analyses are carried out and numerical comparisons are made with the conventional approach.

2. Spline Functions

Given a strictly increasing set of real numbers $x_1, x_2, \ldots, x_m$, a *spline function* $s(x)$ of order n (or degree $n-1$) with *knots* $x_1, x_2, \ldots, x_m$ is a function possessing the following two properties.

(1) In each of the intervals

$$x \leqslant x_1;\ x_{j-1} \leqslant x \leqslant x_j\ (j = 2, 3, \ldots, m);\ x_m \leqslant x,$$

$s(x)$ is a polynomial of degree $n-1$ or less.

(2) $s(x)$ and its derivatives of orders $1, 2, \ldots, n-2$ are continuous.

* National Physical Laboratory report DNAC 4, August 1971.

3. B-splines

Let

$$x_+^{n-1} = \begin{cases} x^{n-1}, & x \geqslant 0 \\ 0, & x < 0 \end{cases} \tag{3.1}$$

and define

$$M_n(x; y) = (y-x)_+^{n-1}. \tag{3.2}$$

Suppose $x_{i-n}, x_{i-n+1}, \ldots, x_i$ are $n+1$ real numbers in strictly increasing order. Consider the divided difference of order n of the function (3.2) with respect to the variable y based on the arguments $y = x_{i-n}, x_{i-n+1}, \ldots, x_i$. Using a notation similar to that of Steffensen (1927) this divided difference is conventionally denoted by $M(x; x_{i-n}, x_{i-n+1}, \ldots, x_i)$, which we shall abbreviate to $M_{ni}(x)$. In terms of the function

$$\omega_{ni}(x) = (x-x_{i-n})(x-x_{i-n+1}) \ldots (x-x_i) \tag{3.3}$$

an explicit expression (Greville, 1969) is

$$M_{ni}(x) = \sum_{r=i-n}^{i} \frac{(x_r-x)_+^{n-1}}{\omega'_{ni}(x_r)}, \tag{3.4}$$

where the prime denotes differentiation with respect to x.

The function $M_{ni}(x)$ is a spline of degree $n-1$ having the knots $x_{i-n}, x_{i-n+1}, \ldots, x_i$. Moreover for $x \geqslant x_i$ it is identically zero, by virtue of (3.1), and for $x \leqslant x_{i-n}$, $M_{ni}(x)$ is simply the divided difference of order n of a polynomial of degree $n-1$ and hence vanishes identically. For $x_{i-n} < x < x_i$, $M_{ni}(x)$ has the property that it is strictly positive (Curry & Schoenberg, 1966). $M_{ni}(x)$ is termed a *B-spline* or *fundamental spline* of degree $n-1$ based on the knots $x_{i-n}, x_{i-n+1}, \ldots, x_i$. These splines were first introduced for the case of equally spaced knots by Schoenberg (1946), and for the case of arbitrarily spaced knots by Curry & Schoenberg (1966).

We have departed slightly from convention in our definition of B-splines in two ways. Firstly, the definition (3.4) has the property that

$$\int_{-\infty}^{\infty} M_{ni}(x)\, dx = 1/n, \tag{3.5}$$

whereas the usual definition (see, for example, Curry & Schoenberg, 1966) includes a factor n so that the value of the integral is normalized to unity. We find the inclusion of this factor a hindrance however, particularly when we come to derive a recurrence relation for the values of $M_{ni}(x)$. The factor can always be inserted for computational or other purposes as required. Secondly, we employ a double subscript in our abbreviated notation for B-splines, as opposed to the single subscript preferred by most authors. Our notation is necessary since we need to refer to B-splines of *various degrees* defined on *various knot sets*.

4. Representation of Splines

One representation of a spline of degree $n-1$ based on the strictly increasing set of knots $x_1, x_2, \ldots, x_m$ is

$$s(x) = \sum_{i=1}^{n} a_i x^{i-1} + \sum_{j=1}^{m} b_j (x-x_j)_+^{n-1}. \tag{4.1}$$

The representation (4.1) is very unsuitable for purposes of interpolation and smoothing

(see, for example, Carasso, 1966). In fact its use can give rise to two serious problems. Firstly, the set of linear equations used to derive the coefficients a_i and b_j of the spline tends to be extremely ill-conditioned, even for small values of n. Secondly, the evaluation of $s(x)$, even supposing the coefficients have been found accurately, can suffer from severe loss of accuracy due to cancellation. A further disadvantage is that the number of arithmetic operations needed to evaluate (4.1) can be quite large.

A representation that is much more convenient for practical purposes (see Anselone & Laurent, 1968), whilst still requiring only $m+n$ coefficients in its definition, is given by expressing $s(x)$ as a linear combination of B-splines $M_{ni}(x)$. Firstly, we introduce $2n$ additional knots $x_{-n+1}, x_{-n+2}, \ldots, x_0$ and $x_{m+1}, x_{m+2}, \ldots, x_{m+n}$, where $x_{-n+1} < x_{-n+2} < \ldots < x_1$ and $x_m < x_{m+1} < \ldots < x_{m+n}$. Then for $x_0 \leqslant x \leqslant x_{m+1}$ the spline can be expressed in the form (Schumaker, 1969),

$$s(x) = \sum_{i=1}^{m+n} c_i M_{ni}(x). \tag{4.2}$$

5. The Conventional Method of Calculating B-splines

For any particular value of x, $M_{ni}(x)$ is conventionally calculated by means of the recursive definition for divided differences (Steffensen, 1927). This approach leads to the following algorithm (we assume that $x_{i-n} < x < x_i$, otherwise $M_{ni}(x) = 0$).

Algorithm 5.1

Set

$$D_{0j} = (x_j - x)_+^{n-1}, \qquad j = i-n, i-n+1, \ldots, i. \tag{5.1}$$

Compute

$$D_{rj} = \frac{D_{r-1,j} - D_{r-1,j-1}}{x_j - x_{j-r}} \tag{5.2}$$

for $j = i-n+r, i-n+r+1, \ldots, i$; $r = 1, 2, \ldots, n$.

Then

$$M_{ni}(x) = D_{ni}. \tag{5.3}$$

For example, if $n = 6$ the elements in the following triangular array are computed:

$$\begin{array}{ccccccc}
D_{0,i-6} & & & & & & \\
& D_{1,i-5} & & & & & \\
D_{0,i-5} & & D_{2,i-4} & & & & \\
& D_{1,i-4} & & D_{3,i-3} & & & \\
D_{0,i-4} & & D_{2,i-3} & & D_{4,i-2} & & \\
& D_{1,i-3} & & D_{3,i-2} & & D_{5,i-1} & \\
D_{0,i-3} & & D_{2,i-2} & & D_{4,i-1} & & D_{6i} \\
& D_{1,i-2} & & D_{3,i-1} & & D_{5i} & \\
D_{0,i-2} & & D_{2,i-1} & & D_{4i} & & \\
& D_{1,i-1} & & D_{3i} & & & \\
D_{0,i-1} & & D_{2i} & & & & \\
& D_{1i} & & & & & \\
D_{0i} & & & & & &
\end{array}$$

In practice advantage can be taken of the property

$$D_{0j} = 0, \qquad x_j \leqslant x, \tag{5.4}$$

in order to reduce the number of applications of (5.2). Thus if for example $x_{l-3} \leqslant x < x_{l-2}$, then the above array takes the form:

$$
\begin{array}{ccccccccc}
0 & & & & & & & & \\
 & 0 & & & & & & & \\
0 & & 0 & & & & & & \\
 & 0 & & 0 & & & & & \\
0 & & 0 & & D_{4,l-2} & & & & \\
 & 0 & & D_{3,l-2} & & D_{5,l-1} & & & \\
0 & & D_{2,l-2} & & D_{4,l-1} & & D_{6l} & & \\
 & D_{1,l-2} & & D_{3,l-1} & & D_{5l} & & & \\
D_{0,l-2} & & D_{2,l-1} & & D_{4l} & & & & \\
 & D_{1,l-1} & & D_{3l} & & & & & \\
D_{0,l-1} & & D_{2l} & & & & & & \\
 & D_{1l} & & & & & & & \\
D_{0l} & & & & & & & &
\end{array}
$$

In general it is necessary to compute and store only a trapezoidal array of non-zero elements. The modified version of the algorithm, taking advantage of (5.4), can be stated as follows:

Algorithm 5.2

Set

$$D_{r,l-1} = 0, \qquad r = 0, 1, \ldots, n-k-1 \tag{5.5}$$

and

$$D_{0j} = (x_j - x)^{n-1}, \qquad j = l, l+1, \ldots, i, \tag{5.6}$$

where $l = i-k$ is the unique integer satisfying

$$x_{l-1} \leqslant x < x_l. \tag{5.7}$$

Compute (5.2) for $j = \mathrm{i}, i-1, \ldots, p$; $r = 1, 2, \ldots, n$, where $p = \max(i-n+r, l)$. Then $M_{nl}(x)$ is given by (5.3).

It is unnecessary in practice to store the whole trapezoidal array, since as soon as D_{rj} is computed from (5.2), it may conveniently overwrite $D_{r-1,j}$, the latter being no longer required. It follows that the number of storage locations required is at most $n+1$.

The elements D_{rj} are all theoretically non-negative (Greville, 1969) and cancellation may therefore take place in computing (5.2) if $D_{r-1,j}$ and $D_{r-1,j-1}$ are of similar size. Hence the possibility exists of significant relative error growth in the computed values of the D_{rj} and hence of appreciable error in the computed value of $M_{nl}(x)$.

We expect therefore this algorithm to be unstable; this expectation is observed in practice, even for relatively "simple" examples. In Section 8 we use a "running" error analysis to give *a posteriori* bounds for the errors in the computed values of D_{rj}.

6. A Stable Method for Calculating B-splines

We now prove that the recurrence relation

$$M_{nl}(x) = \frac{(x-x_{l-n})M_{n-1,l-1}(x) + (x_l - x)M_{n-1,l}(x)}{x_l - x_{l-n}} \tag{6.1}$$

holds for all values of x.

Upon making use of (3.4), the right-hand side of (6.1) becomes

$$\frac{x-x_{i-n}}{x_i-x_{i-n}}\sum_{r=i-n}^{i-1}\frac{(x_r-x)_+^{n-2}}{\omega'_{n-1,i-1}(x_r)}+\frac{x_i-x}{x_i-x_{i-n}}\sum_{r=i-n+1}^{i}\frac{(x_r-x)_+^{n-2}}{\omega'_{n-1,i}(x_r)}$$

$$=\frac{(x-x_{i-n})(x_{i-n}-x)_+^{n-2}}{(x_i-x_{i-n})\omega'_{n-1,i-1}(x_{i-n})}+\sum_{r=i-n+1}^{i-1}\frac{(x_r-x)_+^{n-2}}{x_i-x_{i-n}}\left(\frac{x-x_{i-n}}{\omega'_{n-1,i-1}(x_r)}+\frac{x_i-x}{\omega'_{n-1,i}(x_r)}\right)$$

$$+\frac{(x_i-x)(x_i-x)_+^{n-2}}{(x_i-x_{i-n})\omega'_{n-1,i}(x_i)}. \tag{6.2}$$

Use of the identities

$$\omega'_{ni}(x_j)=(x_j-x_i)\omega'_{n-1,i-1}(x_j), \qquad (j\neq i) \tag{6.3}$$

and

$$\omega'_{ni}(x_j)=(x_j-x_{i-n})\omega'_{n-1,i}(x_j), \qquad (j\neq i-n) \tag{6.4}$$

derived from (3.3), enables the first and last terms on the right-hand side of (6.2) to be expressed as

$$\frac{(x_{i-n}-x)_+^{n-1}}{\omega'_{ni}(x_{i-n})} \quad \text{and} \quad \frac{(x_i-x)_+^{n-1}}{\omega'_{ni}(x_i)}, \tag{6.5}$$

respectively, and the rth term ($r=i-n+1, i-n+2, \ldots, i-1$) of the summation as

$$\frac{(x_r-x)_+^{n-2}\{(x_r-x_i)(x-x_{i-n})+(x_r-x_{i-n})(x_i-x)\}}{(x_i-x_{i-n})\omega'_{ni}(x_r)}=\frac{(x_r-x)_+^{n-1}}{\omega'_{ni}(x_r)}. \tag{6.6}$$

Combining (6.6) summed over all values of r from $i-n+1$ to $i-1$ with (6.5) gives

$$\sum_{r=i-n}^{i}\frac{(x_r-x)_+^{n-1}}{\omega'_{ni}(x_r)}, \tag{6.7}$$

which, by virtue of definition (3.4), is equal to $M_{ni}(x)$, and hence the required result (6.1) is proved.

We may therefore state the following algorithm for evaluating $M_{ni}(x)$ (again we assume that $x_{i-n}<x<x_i$, otherwise $M_{ni}(x)=0$).

Algorithm 6.1

Set

$$M_{1j}=M_{1j}(x), \qquad j=i-n+1, i-n+2, \ldots, i. \tag{6.8}$$

Compute

$$M_{rj}=\frac{(x-x_{j-r})M_{r-1,j-1}+(x_j-x)M_{r-1,j}}{x_j-x_{j-r}} \tag{6.9}$$

for $j=i-n+r, i-n+r+1, \ldots, i$; $\quad r=2, 3, \ldots, n$.

Then

$$M_{ni}(x)=M_{ni}. \tag{6.10}$$

For example, if $n = 6$, the elements in the following triangular array are computed:

$$
\begin{array}{ccccccc}
M_{1,i-5} & & & & & & \\
& M_{2,i-4} & & & & & \\
M_{1,i-4} & & M_{3,i-3} & & & & \\
& M_{2,i-3} & & M_{4,i-2} & & & \\
M_{1,i-3} & & M_{3,i-2} & & M_{5,i-1} & & \\
& M_{2,i-2} & & M_{4,i-1} & & M_{6i} & \\
M_{1,i-2} & & M_{3,i-1} & & M_{5i} & & \\
& M_{2,i-1} & & M_{4i} & & & \\
M_{1,i-1} & & M_{3i} & & & & \\
& M_{2i} & & & & & \\
M_{1i} & & & & & &
\end{array}
$$

As with the conventional algorithm advantage can be taken of zero elements in the array in order to reduce the number of applications of (6.9). By making use of the relation,

$$M_{1j}(x) = \begin{cases} 1/(x_j - x_{j-1}), & x_{j-1} \leqslant x < x_j \\ 0, & \text{otherwise} \end{cases} \tag{6.11}$$

the above array takes, if for example $x_{i-3} \leqslant x < x_{i-2}$, the following form:

$$
\begin{array}{ccccccc}
0 & & & & & & \\
& 0 & & & & & \\
0 & & 0 & & & & \\
& 0 & & M_{4,i-2} & & & \\
0 & & M_{3,i-2} & & M_{5,i-1} & & \\
& M_{2,i-2} & & M_{4,i-1} & & M_{6i} & \\
M_{1,i-2} & & M_{3,i-1} & & M_{5i} & & \\
& M_{2,i-1} & & M_{4i} & & & \\
0 & & M_{3i} & & & & \\
& 0 & & & & & \\
0 & & & & & &
\end{array}
$$

In general it is necessary to compute and store only a rhomboidal array of non-zero elements. The modified version of the algorithm, taking advantage of (6.11), can be stated as follows:

Algorithm 6.2

Let $l = i-k$ be the unique integer satisfying (5.7). Set $M_{r,l-1} = 0$ for $r = 1, 2, \ldots, n-k+1$ and $M_{r,l+r} = 0$ for $r = 1, 2, \ldots, k$. Compute $M_{r,l+j}$ from (6.9) for $r = j+1, j+2, \ldots, j+n-k$; $j = 0, 1, \ldots, k$, except for $r = 1, j = 0$, in which case M_{1l} is $1/(x_l - x_{l-1})$ from (6.11). Finally $M_{nl}(x)$ is given by (6.10).

We need not store the complete rhomboidal array since as soon as $M_{r,l+j}$ has been computed it may overwrite $M_{r-1,l+j-1}$. The number of storage locations required is at most n.

Since $M_{r-1,j-1}$, $M_{r-1,j}$ and the terms $x - x_{j-r}$, $x_j - x$ and $x_j - x_{j-r}$ are all positive (or at least non-negative) for the above algorithm, it follows that there will be no cancellation in evaluating the right-hand side of (6.9). Consequently we expect the recurrence relation to be numerically stable. In Sections 9 and 10 we carry out analyses of error growth in this relation.

7. Floating-point Arithmetic

We assume that computations are carried out in floating-point binary arithmetic with a t-digit mantissa. Following Wilkinson (1965: 110 *et seq.*) let x and y be two standard floating-point numbers and $*$ denote any one of the arithmetic operations $+$, $-$, $\times$, $\div$. If $fl(x * y)$ denotes the computed value of $x * y$ then

$$fl(x * y) = (1+\varepsilon)(x * y), \tag{7.1}$$

where

$$|\varepsilon| \leqslant 2^{-t}. \tag{7.2}$$

There is also the additional relation

$$fl(x \pm y) = (x \pm y)/(1+\varepsilon), \tag{7.3}$$

where ε satisfies (7.2), given by Kahan, which is sometimes more convenient than the relation

$$fl(x \pm y) = (1+\varepsilon)(x \pm y). \tag{7.4}$$

We shall use (7.3) or (7.4) as convenient.

Some computers have less accurate rounding procedures than those which give the above results, but we assume (as do Peters & Wilkinson, 1971, in a different context) that the differences are not of great consequence.

We shall also make use of the relations

$$(1+2^{-t})^s < 1+1{\cdot}06s2^{-t}, \tag{7.5}$$

$$(1-2^{-t})^{-s} < 1+1{\cdot}12s2^{-t}, \tag{7.6}$$

where s is a positive integer. These relations hold as long as s and t satisfy the mild restriction

$$s2^{-t} < 0{\cdot}1. \tag{7.7}$$

We assume throughout this paper that the inequality (7.7) is satisfied for all values of s that arise. Relation (7.5) is given by Wilkinson (1965: 113). Relation (7.6) is proved as follows:

$$\begin{aligned}
(1-2^{-t})^{-s} &= 1+\sum_{k=1}^{\infty} \frac{s(s+1)(s+2)\ldots(s+k-1)}{k!}2^{-kt} \\
&\leqslant 1+\sum_{k=1}^{\infty} \frac{s(2s)(3s)\ldots(ks)}{k!}2^{-kt} \\
&= 1+\sum_{k=1}^{\infty} (s2^{-t})^k \\
&< 1+s2^{-t}\sum_{k=0}^{\infty} (0{\cdot}1)^k \\
&< 1+1{\cdot}12s2^{-t}.
\end{aligned}$$

Following Wilkinson (1965: 114) we shall sometimes use relation (7.5) in the form

$$(1+2^{-t})^s < 1+s2^{-t_1}, \tag{7.8}$$

where

$$2^{-t_1} = (1{\cdot}06)2^{-t}. \tag{7.9}$$

We observe that relation (7.7) is equivalent to the inequality

$$s2^{-t_1} < 0{\cdot}106. \tag{7.10}$$

All numerical results given in this paper were obtained using the KDF9 computer, which has a floating-point word with 39 binary bits in the mantissa.

8. *A Posteriori* Error Bounds for the Conventional Method

In this section we indicate how a "running" error analysis (Peters & Wilkinson, 1971) may be used to obtain *a posteriori* error bounds for the value of $M_{nt}(x)$ computed by means of divided differences. For purposes of our analysis we assume that the knots $x_{i-n}, x_{i-n+1}, \ldots, x_i$ and the argument x can be represented exactly as standard floating-point numbers. In Section 11 we discuss the implications of this assumption.

For a specific value of x let

$$\bar{D}_{rj} = D_{rj} + E_{rj}, \tag{8.1}$$

where $\bar{D}_{rj}$ is the computed value of D_{rj}. Then by virtue of (5.2),

$$\begin{aligned}\bar{D}_{rj} &= fl\left[\frac{\bar{D}_{r-1,j}-\bar{D}_{r-1,j-1}}{x_j-x_{j-r}}\right] \\ &= (1+\varepsilon_1)(1+\varepsilon_2)(1+\varepsilon_3)\left[\frac{\bar{D}_{r-1,j}-\bar{D}_{r-1,j-1}}{x_j-x_{j-r}}\right],\end{aligned} \tag{8.2}$$

where

$$|\varepsilon_1|, |\varepsilon_2|, |\varepsilon_3| \leqslant 2^{-t}. \tag{8.3}$$

Thus

$$\bar{D}_{rj} = (1+3e)\left[\frac{\bar{D}_{r-1,j}-\bar{D}_{r-1,j-1}}{x_j-x_{j-r}}\right], \tag{8.4}$$

where

$$|e| < 2^{-t_1}. \tag{8.5}$$

Making use of (8.1), equation (8.4) may be expressed as

$$D_{rj}+E_{rj} = \frac{D_{r-1,j}+E_{r-1,j}-D_{r-1,j-1}-E_{r-1,j-1}+3e(\bar{D}_{r-1,j}-\bar{D}_{r-1,j-1})}{x_j-x_{j-r}}. \tag{8.6}$$

Application of (5.2) then gives

$$E_{rj} = \frac{E_{r-1,j}-E_{r-1,j-1}+3e(\bar{D}_{r-1,j}-\bar{D}_{r-1,j-1})}{x_j-x_{j-r}}. \tag{8.7}$$

It follows from (8.7) that if we define F_{rj} by the relation

$$F_{rj} = \frac{(F_{r-1,j}+F_{r-1,j-1})+3|\bar{D}_{r-1,j}-\bar{D}_{r-1,j-1}|}{x_j-x_{j-r}}, \tag{8.8}$$

then

$$|E_{rj}| \leqslant 2^{-t_1}F_{rj}. \tag{8.9}$$

To initiate the recurrence we set

$$F_{0j} = (n-1)\bar{D}_{0j}, \tag{8.10}$$

which is obtained from a simple error analysis of the computation of (5.1). (We assume that D_{0j} is evaluated by forming $x_j-x = h$, say, and computing h^{n-1} by repeated multiplication. This approach is in accordance with the definition of exponentiation given in the ALGOL 60 report (Naur, 1963). If n is sufficiently large

it may be more accurate (and perhaps faster) to compute h^{n-1} from $\exp((n-1)\ln h)$; we do not consider this approach here.

The computer itself can be made to determine the values of F_{rj}, since they depend solely on previous values together with the *computed* values of D_{rj}. In particular $2^{-t_1}F_{nl}$ is a bound for the error in the computed value of $M_{nl}(x)$. In practice the computer makes rounding errors in computing the F_{rj}. However, we see from a simple error analysis that the contribution to the error incurred in computing F_{rj} from (8.8) is at most a factor $(1-2^{-t})^{-5}$. Since the contribution to the error incurred in computing F_{0j} from (8.10) is at most a factor $(1-2^{-t})^{-2}$, it follows that $(1-2^{-t})^{-5r-2}2^{-t_1}\bar{F}_{rj}$, where $\bar{F}_{rj}$ is the computed value of F_{rj}, gives a rigorous error bound for $|E_{rj}|$. Now $(1-2^{-t})^{-5r-2} < 1+1{\cdot}12(5r+2)2^{-t} < 1{\cdot}112$ by virtue of (7.6) and (7.7). Hence a simpler but slightly cruder bound for $|E_{rj}|$ is $(1{\cdot}112)2^{-t_1}\bar{F}_{rj}$ or $(1{\cdot}179)2^{-t}\bar{F}_{rj}$. These *a posteriori* error bounds prove to be reasonably realistic (see the examples in Section 12). Moreover, since advantage is taken of any cancellations that occur during the course of the computation, these bounds are much better than those obtained from an *a priori* error analysis (but see the special case discussed in Section 12).

9. *A Posteriori* Error Bounds for the Stable Method

We show in this section, again by means of a "running" error analysis, how *a posteriori* error bounds for the stable method can be determined. Again we assume that the knots $x_{l-n}, x_{l-n+1}, \ldots, x_l$ and the argument x can be represented exactly as standard floating-point numbers, and in Section 11 return to the consequences of this assumption.

For the given argument x let

$$\bar{M}_{rj} = M_{rj}+G_{rj}, \tag{9.1}$$

where $\bar{M}_{rj}$ is the computed value of M_{rj}. Using relation (6.9),

$$\bar{M}_{rj} = fl\left(\frac{(x-x_{j-r})\bar{M}_{r-1,j-1}+(x_j-x)\bar{M}_{r-1,j}}{x_j-x_{j-r}}\right) \tag{9.2}$$

$$= \frac{(1+\varepsilon_5)(1+\varepsilon_6)(1+\varepsilon_7)\{(1+\varepsilon_1)(1+\varepsilon_2)(x-x_{j-r})\bar{M}_{r-1,j-1}+(1+\varepsilon_3)(1+\varepsilon_4)(x_j-x)\bar{M}_{r-1,j}\}}{x_j-x_{j-r}}, \tag{9.3}$$

where

$$|\varepsilon_k| \leqslant 2^{-t}, \qquad k = 1, 2, \ldots, 7. \tag{9.4}$$

Thus

$$\bar{M}_{rj} = \frac{(1+5e_1)(x-x_{j-r})\bar{M}_{r-1,j-1}+(1+5e_2)(x_j-x)\bar{M}_{r-1,j}}{x_j-x_{j-r}}, \tag{9.5}$$

where

$$|e_1|, |e_2| < 2^{-t_1}. \tag{9.6}$$

Upon making use of (9.1) and (6.9), equation (9.5) gives

$$G_{rj} = \frac{(x-x_{j-r})(G_{r-1,j-1}+5e_1\bar{M}_{r-1,j-1})+(x_j-x)(G_{r-1,j}+5e_2\bar{M}_{r-1,j})}{x_j-x_{j-r}}. \tag{9.7}$$

If H_{rj} is defined by

$$H_{rj} = \frac{(x-x_{j-r})(H_{r-1,j-1}+5\bar{M}_{r-1,j-1})+(x_j-x)(H_{r-1,j}+5\bar{M}_{r-1,j})}{x_j-x_{j-r}}, \tag{9.8}$$

with

$$H_{1j} = 2\bar{M}_{1j}, \tag{9.9}$$

then

$$|G_{rj}| \leqslant 2^{-t_1}H_{rj}. \tag{9.10}$$

Relation (9.9) is obtained from a simple error analysis of (6.11).

Once again the computer can be used to determine the values of H_{rj} and in particular the value $2^{-t_1}H_{nl}$, which is a bound for the error in the computed value of $M_{nl}(x)$. The computer makes rounding errors in computing H_{rj}. However, as before we can show that the contribution to the error incurred in computing H_{rj} from (9.8) is at most a factor $(1-2^{-t})^{-7}$. Since the contribution to the error incurred in computing H_{1j} from (9.9) is at most a factor $(1-2^{-t})^{-1}$ it follows that $(1-2^{-t})^{6-7r}2^{-t_1}\bar{H}_{rj}$, where $\bar{H}_{rj}$ is the computed value of H_{rj}, gives a rigorous error bound for $|G_{rj}|$. Again the simpler bound $|G_{rj}| \leqslant (1{\cdot}179)2^{-t}H_{rj}$ may be preferred. These bounds prove to be quite realistic. However, we show in Section 10 that it is unnecessary in practice to carry out computation of these bounds by means of a running error analysis, since realistic *a posteriori* bounds for the absolute errors can be deduced immediately from the computed results. Moreover we obtain in Section 10 *a priori* bounds for the relative errors.

10. *A Priori* Error Bounds for the Stable Method

We now show that H_{rj} satisfies the inequality,

$$H_{rj} \leqslant (1-2^{-t})^{5(1-r)}(5r-3)\bar{M}_{rj}. \tag{10.1}$$

In order to establish this result we first assume it is true for $H_{r-1,j}$ where $r > 1$. Then substitution of (10.1) into the right-hand side of (9.8) gives

$$H_{rj} \leqslant \{(1-2^{-t})^{5(2-r)}(5r-8)+5\}\left(\frac{(x-x_{j-r})\bar{M}_{r-1,j-1}+(x_j-x)\bar{M}_{r-1,j}}{x_j-x_{j-r}}\right). \tag{10.2}$$

But it follows from (9.3) and (9.4) that

$$\bar{M}_{rj} \geqslant (1-2^{-t})^5\left(\frac{(x-x_{j-r})\bar{M}_{r-1,j-1}+(x_j-x)\bar{M}_{r-1,j}}{x_j-x_{j-r}}\right). \tag{10.3}$$

Hence

$$H_{rj} \leqslant \{(1-2^{-t})^{5(2-r)}(5r-8)+5\}(1-2^{-t})^{-5}\bar{M}_{rj}, \tag{10.4}$$

the simplification of which yields (10.1). Hence (10.1) is true for H_{rj}. But it is true for H_{1j} by virtue of (9.9). Hence by induction it is true for all r.

A slightly cruder bound, obtained by means of (7.6) and (7.7) is

$$H_{rj} \leqslant 1{\cdot}112(5r-3)\bar{M}_{rj}. \tag{10.5}$$

It follows from (9.10) that the computed value of $M_{nl}(x)$ differs from the true value by an amount not exceeding $1{\cdot}112(5n-3)2^{-t_1}\bar{M}_{nl}$, or, equivalently, $1{\cdot}179(5n-3)2^{-t}\bar{M}_{nl}$. We observe that this bound is *computable*, since it involves the value of $\bar{M}_{nl}$, rather than the true (unknown) value M_{nl}.

We now give a bound for the relative error R_{ni} in the computed value of $M_{ni}(x)$. Now

$$R_{ni} = |\bar{M}_{ni} - M_{ni}|/M_{ni} = |G_{ni}|/M_{ni}. \tag{10.6}$$

Since

$$|G_{ni}| \leqslant 1{\cdot}179(5n-3)2^{-t}\bar{M}_{ni} \tag{10.7}$$

$$= 1{\cdot}179(5n-3)2^{-t}(M_{ni}+G_{ni}), \tag{10.8}$$

it follows that

$$|G_{ni}| \leqslant \frac{1{\cdot}179(5n-3)2^{-t}M_{ni}}{1-1{\cdot}179(5n-3)2^{-t}}. \tag{10.9}$$

But from (7.7) it follows that the denominator is bounded from below by $1-0{\cdot}1179 = 0{\cdot}8821$. Hence

$$|G_{ni}| \leqslant 1{\cdot}337(5n-3)2^{-t}M_{ni}. \tag{10.10}$$

So

$$R_{ni} \leqslant 1{\cdot}337(5n-3)2^{-t}, \tag{10.11}$$

which is an *a priori* bound for the relative error in the computed value of $M_{ni}(x)$. This result is remarkable in that it is *independent of the positions of the knots* (but see the comments in Section 11 on the effects of decimal to binary conversion). It follows for example that *B*-splines of degree 14 or less can be evaluated with a loss of accuracy not exceeding 100 units in the least significant binary place. Such a result compares extremely favourably with the conventional method for which non-pathological examples of degree very much smaller than 14 (see Section 12) can be constructed that yield no correct figures (on the KDF9 computer) in the results.

11. The Effects of Perturbations in the Data

There is one aspect of the problem that our analysis has so far not covered, viz. the sensitivity of the computed values of $M_{ni}(x)$ with respect to perturbations in the data. By *data* in this context we mean the given values of the knots $x_{i-n}, x_{i-n+1}, \ldots, x_i$ and the argument x. A particular reason why the study of such perturbations is important is that our analyses are rigorous only for data that can be represented exactly as standard floating-point numbers. However, we may be comforted by the fact that our analyses *do* apply to the problem defined by the data *stored in the computer*. So it follows that the stable method solves accurately a problem with data perturbed slightly from that given. For most computers the perturbed (stored) data differs from that given by relative errors bounded in modulus by 2^{-t}.

A posteriori bounds relating to the *given* data, rather than the *stored* data, can be found if required by an extension of the running error analyses described in Sections 8 and 9. The manner in which this analysis is carried out is straightforward but tedious and is not given here. One consequence of this analysis is that the bounds are now no longer independent of the knot spacing. However, unless the knot spacing is highly non-uniform, the bounds for the stable method appear to depend only mildly upon the positions of the knots, although it is now no longer possible to quote *a priori* bounds. On the other hand the bounds (as well as the computed values themselves) for the conventional method seem to be very sensitive to small perturbations in the knots, which is a reflection of the inherent instability of that method.

12. Numerical Examples

In order to compare numerically the conventional and stable methods we give a number of examples. For each case we consider the B-spline of a prescribed degree $n-1$ based on a given set of knots $x_{i-n}, x_{i-n+1}, \ldots, x_i$. The B-spline is evaluated by both methods (using Algorithms 5.2 and 6.2) at the positions of the interior knots $x_{i-n+1}, x_{i-n+2}, \ldots, x_{i-1}$. For the conventional method the error bound $(1{\cdot}179)2^{-t}\bar{F}_{ni}$ and for the stable method the error bound $1{\cdot}179(5n-3)2^{-t}\bar{M}_{ni}$ are also quoted.

Example 1

Degree 5. Knots 0, 1, 2, 3, 4, 5, 6 (Table 1). The values produced by the conventional method differ only slightly from those given by the stable method.

TABLE 1

Degree 5. *Knots* 0, 1, 2, 3, 4, 5, 6

x	Conventional method Value	Conventional method Error bound	Stable method Value	Stable method Error bound
1	$1{\cdot}38888\ 88892_{10}-3$	$3{\cdot}52490_{10}-10$	$1{\cdot}38888\ 88889_{10}-3$	$8{\cdot}04221_{10}-14$
2	$3{\cdot}61111\ 11111_{10}-2$	$9{\cdot}18241_{10}-11$	$3{\cdot}61111\ 11111_{10}-2$	$2{\cdot}09097_{10}-12$
3	$9{\cdot}16666\ 66667_{10}-2$	$1{\cdot}81933_{10}-11$	$9{\cdot}16666\ 66667_{10}-2$	$5{\cdot}30786_{10}-12$
4	$3{\cdot}61111\ 11111_{10}-2$	$2{\cdot}22799_{10}-12$	$3{\cdot}61111\ 11111_{10}-2$	$2{\cdot}09097_{10}-12$
5	$1{\cdot}38888\ 88889_{10}-3$	$6{\cdot}85077_{10}-14$	$1{\cdot}38888\ 88889_{10}-3$	$8{\cdot}04221_{10}-14$

TABLE 2

Degree 21. *Knots* 0, 1, 2, ..., 22

x	Conventional method Value	Conventional method Error bound	Stable method Value	Stable method Error bound
1	$-2{\cdot}90493\ 62704_{10}-4$	$1{\cdot}07302_{10}-1$	$8{\cdot}89679\ 13924_{10}-22$	$2{\cdot}04156_{10}-31$
2	$1{\cdot}70851\ 18477_{10}-4$	$2{\cdot}87569_{10}-2$	$1{\cdot}86577\ 28133_{10}-15$	$4{\cdot}28141_{10}-25$
3	$-5{\cdot}09073\ 29585_{10}-5$	$7{\cdot}22242_{10}-3$	$9{\cdot}26531\ 08069_{10}-12$	$2{\cdot}12612_{10}-21$
4	$8{\cdot}78471\ 73001_{10}-6$	$1{\cdot}69058_{10}-3$	$3{\cdot}70854\ 13543_{10}-9$	$8{\cdot}51003_{10}-19$
5	$5{\cdot}18082\ 90829_{10}-8$	$3{\cdot}66527_{10}-4$	$3{\cdot}40296\ 26271_{10}-7$	$7{\cdot}80881_{10}-17$
6	$1{\cdot}07095\ 90738_{10}-5$	$7{\cdot}30865_{10}-5$	$1{\cdot}10732\ 92030_{10}-5$	$2{\cdot}54100_{10}-15$
7	$1{\cdot}59735\ 14258_{10}-4$	$1{\cdot}32968_{10}-5$	$1{\cdot}59595\ 80785_{10}-4$	$3{\cdot}66226_{10}-14$
8	$1{\cdot}15687\ 81534_{10}-3$	$2{\cdot}18684_{10}-6$	$1{\cdot}15690\ 83302_{10}-3$	$2{\cdot}65477_{10}-13$
9	$4{\cdot}55429\ 00755_{10}-3$	$3{\cdot}21645_{10}-7$	$4{\cdot}55428\ 59425_{10}-3$	$1{\cdot}04508_{10}-12$
10	$1{\cdot}01945\ 49442_{10}-2$	$4{\cdot}17744_{10}-8$	$1{\cdot}01945\ 49722_{10}-2$	$2{\cdot}33935_{10}-12$
11	$1{\cdot}33010\ 31228_{10}-2$	$4{\cdot}71872_{10}-9$	$1{\cdot}33010\ 31238_{10}-2$	$3{\cdot}05220_{10}-12$
12	$1{\cdot}01945\ 49726_{10}-2$	$4{\cdot}55067_{10}-10$	$1{\cdot}01945\ 49722_{10}-2$	$2{\cdot}33935_{10}-12$
13	$4{\cdot}55428\ 59422_{10}-3$	$3{\cdot}66011_{10}-11$	$4{\cdot}55428\ 59425_{10}-3$	$1{\cdot}04508_{10}-12$
14	$1{\cdot}15690\ 83302_{10}-3$	$2{\cdot}37665_{10}-12$	$1{\cdot}15690\ 83302_{10}-3$	$2{\cdot}65477_{10}-13$
15	$1{\cdot}59595\ 80785_{10}-4$	$1{\cdot}18145_{10}-13$	$1{\cdot}59595\ 80785_{10}-4$	$3{\cdot}66226_{10}-14$
16	$1{\cdot}10732\ 92030_{10}-5$	$4{\cdot}08175_{10}-15$	$1{\cdot}10732\ 92030_{10}-5$	$2{\cdot}54100_{10}-15$
17	$3{\cdot}40296\ 26271_{10}-7$	$8{\cdot}26650_{10}-17$	$3{\cdot}40296\ 26271_{10}-7$	$7{\cdot}80881_{10}-17$
18	$3{\cdot}70854\ 13543_{10}-9$	$7{\cdot}37998_{10}-19$	$3{\cdot}70854\ 13543_{10}-9$	$8{\cdot}51003_{10}-19$
19	$9{\cdot}26531\ 08068_{10}-12$	$1{\cdot}73796_{10}-21$	$9{\cdot}26531\ 08069_{10}-12$	$2{\cdot}12612_{10}-21$
20	$1{\cdot}86577\ 28133_{10}-15$	$3{\cdot}48119_{10}-25$	$1{\cdot}86577\ 28133_{10}-15$	$4{\cdot}28141_{10}-25$
21	$8{\cdot}89679\ 13924_{10}-22$	$1{\cdot}65996_{10}-31$	$8{\cdot}89679\ 13924_{10}-22$	$2{\cdot}04156_{10}-31$

Example 2

Degree 21. Knots 0, 1, 2, ..., 22 (Table 2). For values of $x > 13$ the conventional method produces results of comparable accuracy to those of the stable method. However, as x is decreased from 13 to unity the conventional method becomes less and less accurate. Indeed, all values for $x < 6$ have no correct figures at all. (Note that since this B-spline is symmetric about $x = 11$ the conventional method could be used to give reliable results by replacing x by $22-x$ if $x < 11$.)

Example 3

Degree 3. Knots -10000, -9999, 0, 9999, 10000 (Table 3). This example is included to illustrate how erroneous the results of the conventional method can be for a degree as low as three, if the knot spacing is highly non-uniform. Such a case is important in practice since it is often of interest to investigate the case of near-coincident knots. We see that at $x = -9999$ the conventional method produces a negative value. Even at $x = 0$, the peak of the B-spline, three figures have been lost. At $x = 9999$ the result agrees with that of the stable method.

TABLE 3

Degree 3. *Knots* -10000, -9999, 0, 9999, 10000

x	Conventional method Value	Conventional method Error bound	Stable method Value	Stable method Error bound
-9999	$-1{\cdot}50012\,05611_{10}-12$	$2{\cdot}57409_{10}-11$	$2{\cdot}50012\,50063_{10}-13$	$9{\cdot}11496_{10}-24$
0	$2{\cdot}50012\,49812_{10}-5$	$3{\cdot}21865_{10}-12$	$2{\cdot}50012\,50062_{10}-5$	$9{\cdot}11496_{10}-16$
9999	$2{\cdot}50012\,50063_{10}-13$	$8{\cdot}04261_{10}-24$	$2{\cdot}50012\,50063_{10}-13$	$9{\cdot}11496_{10}-24$

Example 4

Degree 9. Knots 2^j, $j = 0, 1, \ldots, 10$ (Table 4). Again the conventional method produces very inaccurate results (from the point of view of relative errors) for the small values of x. However, in terms of absolute error (measured with respect to the peak height), the conventional method appears perfectly adequate. Since in many applications such results would be quite acceptable we might expect that for examples

TABLE 4

Degree 9. *Knots* 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024

x	Conventional method Value	Conventional method Error bound	Stable method Value	Stable method Error bound
2	$-1{\cdot}60028\,39327_{10}-14$	$9{\cdot}53785_{10}-13$	$9{\cdot}60166\,98247_{10}-17$	$9{\cdot}67807_{10}-27$
4	$1{\cdot}80654\,27507_{10}-12$	$9{\cdot}11019_{10}-13$	$1{\cdot}79167\,15893_{10}-12$	$1{\cdot}80593_{10}-22$
8	$1{\cdot}97471\,48111_{10}-9$	$8{\cdot}33280_{10}-13$	$1{\cdot}97471\,78228_{10}-9$	$1{\cdot}99043_{10}-19$
16	$3{\cdot}81224\,92651_{10}-7$	$7{\cdot}03203_{10}-13$	$3{\cdot}81224\,94453_{10}-7$	$3{\cdot}84258_{10}-17$
32	$1{\cdot}72139\,97270_{10}-5$	$5{\cdot}14271_{10}-13$	$1{\cdot}72139\,97251_{10}-5$	$1{\cdot}73510_{10}-15$
64	$1{\cdot}95187\,17159_{10}-4$	$2{\cdot}95574_{10}-13$	$1{\cdot}95187\,17160_{10}-4$	$1{\cdot}96740_{10}-14$
128	$5{\cdot}17660\,42895_{10}-4$	$1{\cdot}14874_{10}-13$	$5{\cdot}17660\,42895_{10}-4$	$5{\cdot}21779_{10}-14$
256	$2{\cdot}40474\,09004_{10}-4$	$2{\cdot}19807_{10}-14$	$2{\cdot}40474\,09004_{10}-4$	$2{\cdot}42387_{10}-14$
512	$6{\cdot}59821\,72615_{10}-6$	$5{\cdot}51868_{10}-16$	$6{\cdot}59821\,72615_{10}-6$	$6{\cdot}65072_{10}-16$

similar to this the conventional method is satisfactory. Example 5 shows that this is not the case.

Example 5

Degree 9. Knots -2^{10-j}, $j = 0, 1, \ldots, 10$ (Table 5). This example is identical to Example 4, except that the knots have been reflected about the origin. The stable method gives identical results in each case. The conventional method again gives its most inaccurate results for the smaller (i.e. more negative) values of x. However, these values not only have large relative errors, but they also possess large absolute errors.

TABLE 5

Degree 9. *Knots* $-1024, -512, -256, -128, -64, -32, -16, -8, -4, -2, -1$

x	Conventional method Value	Conventional method Error bound	Stable method Value	Stable method Error bound
-512	$-1{\cdot}96549\ 35593_{10}-4$	$2{\cdot}03640_{10}-2$	$6{\cdot}59821\ 72615_{10}-6$	$6{\cdot}65072_{10}-16$
-256	$2{\cdot}41473\ 72045_{10}-4$	$3{\cdot}80279_{10}-5$	$2{\cdot}40474\ 09004_{10}-4$	$2{\cdot}42387_{10}-14$
-128	$5{\cdot}17659\ 22318_{10}-4$	$6{\cdot}81279_{10}-8$	$5{\cdot}17660\ 42895_{10}-4$	$5{\cdot}21779_{10}-14$
-64	$1{\cdot}95187\ 16988_{10}-4$	$1{\cdot}13289_{10}-10$	$1{\cdot}95187\ 17160_{10}-4$	$1{\cdot}96740_{10}-14$
-32	$1{\cdot}72139\ 97255_{10}-5$	$1{\cdot}67259_{10}-13$	$1{\cdot}72139\ 97251_{10}-5$	$1{\cdot}73510_{10}-15$
-16	$3{\cdot}81224\ 94452_{10}-7$	$2{\cdot}18970_{10}-16$	$3{\cdot}81224\ 94453_{10}-7$	$3{\cdot}84258_{10}-17$
-8	$1{\cdot}97471\ 78228_{10}-9$	$2{\cdot}63580_{10}-19$	$1{\cdot}97471\ 78228_{10}-9$	$1{\cdot}99043_{10}-19$
-4	$1{\cdot}79167\ 15893_{10}-12$	$1{\cdot}54897_{10}-22$	$1{\cdot}79167\ 15893_{10}-12$	$1{\cdot}80593_{10}-22$
-2	$9{\cdot}60166\ 98247_{10}-17$	$8{\cdot}03074_{10}-27$	$9{\cdot}60166\ 98247_{10}-17$	$9{\cdot}67807_{10}-27$

In all the above examples the error bounds for the stable method are realistic. For the conventional method they are somewhat pessimistic, but are considerably better than could be achieved using some form of *a priori* error analysis (but see the special case $x_{i-1} \leqslant x < x_i$ considered below).

In every case the accuracy of the conventional method falls off as x ranges from x_i to x_{i-n}. However, for values of x sufficiently close to x_i, and always for values of x between x_{i-1} and x_i, values comparable in accuracy to those given by the stable method are produced. This result is easily explained by means of a running error analysis along the lines of Section 9 followed by an inductive proof similar in nature to that in Section 10. In fact the *a priori* bound,

$$|F_{ni}| \leqslant 1{\cdot}179(4n-1)2^{-t}\bar{D}_{ni}, \qquad x_{i-1} \leqslant x < x_i, \tag{12.1}$$

is readily derived. The computed value of $M_{ni}(x)$, *viz* $\bar{D}_{ni}$, is positive and can be shown to have a maximum relative error of $1{\cdot}337(4n-1)2^{-t}$. (The slightly improved bound with the term $4n-1$ replaced by $3n-1$ can be obtained if further advantage is taken of the zeros in the D_{rj} array in this special case.) Note that this bound is even better for $n \geqslant 3$ than (10.11) for the stable method. However this bound applies only for $x_{i-1} \leqslant x < x_i$, whereas the bound for the stable method applies for $x_{i-n} < x < x_i$.

13. Discussion

For B-splines of low degree with relatively uniform knot spacing the conventional method is adequate for certain purposes in that the *absolute* errors in the computed values are usually small (as in Examples 1 and 4). However, in a common situation

to be examined below it is seen that the stable method is faster in that fewer arithmetic operations are required and hence should be preferred on grounds of both accuracy *and* speed.

In many problems of interpolation and smoothing by B-splines it is necessary to compute, not an isolated value, but the set of values $M_{ni}(x)$ for $i = j, j+1, \ldots, j+n-1$, where $x_{j-1} \leqslant x \leqslant x_j$. It is much more efficient to compute all n of the required values of $M_{ni}(x)$ together, rather than determine them individually by means of one of the algorithms described. Either the divided difference algorithm or the stable algorithm can be extended very easily to enable this computation to be carried out. For instance, if $n = 4$ and the stable algorithm is used the elements in the following array are computed:

$$\begin{array}{ccccc}
 & & & & M_{4j} \\
 & & M_{3j} & & \\
 & M_{2j} & & & M_{4,j+1} \\
M_{1j} & & M_{3,j+1} & & \\
 & M_{2,j+1} & & & M_{4,j+2} \\
 & & M_{3,j+2} & & \\
 & & & & M_{4,j+3}
\end{array}$$

If the divided difference algorithm is employed the following array is formed:

$$\begin{array}{ccccccc}
 & & & & & & D_{4j} \\
 & & & & D_{3j} & & \\
 & & D_{2j} & & & & D_{4,j+1} \\
 & D_{1j} & & & D_{3,j+1} & & \\
D_{0j} & & D_{2,j+1} & & & & D_{4,j+2} \\
 & D_{1,j+1} & & & D_{3,j+2} & & \\
D_{0,j+1} & & D_{2,j+2} & & & & D_{4,j+3} \\
 & D_{1,j+2} & & & D_{3,j+3} & & \\
D_{0,j+2} & & D_{2,j+3} & & & & \\
 & D_{1,j+3} & & & & & \\
D_{0,j+3} & & & & & &
\end{array}$$

For the case $n = 4$ the conventional method requires 32 additions, 8 multiplications and 16 divisions to compute the required values; the stable method requires 25 additions, 12 multiplications and 10 divisions (the number of additions for the stable method can be reduced to 19 by careful programming; it does not appear possible to achieve a corresponding reduction for the method of divided differences). Since on any computing machine multiplication is at least as fast as division it is evident that the stable method is faster for cubic B-splines. In fact this result applies in general; for a B-spline of degree $n-1$, $n > 2$, the number of additions, multiplications and divisions are $2n^2$, $n(n-2)$ and n^2, respectively, if the conventional method is used, whereas for the stable method these numbers are $2n^2-2n+1$, $n(n-1)$ and $\frac{1}{2}n(n+1)$, respectively. For large n the stable method has a saving of $\frac{1}{2}n^2$ divisions over the conventional method (if terms of order n are ignored in comparison with those of order n^2).

A number of authors including LaFata & Rosen (1970) and Powell (1970) have suggested the use of formula (3.4) to evaluate $M_{ni}(x)$. At first sight this approach

seems attractive since it gives $M_{nl}(x)$ in explicit form. Unfortunately this method is also unstable, as a running error analysis along the lines of those carried out for the conventional and stable methods indicates. This analysis is not given here, but we make the following comment. The computation of the individual terms of the summation (3.4) can be carried out accurately, but the evaluation of the sum itself frequently involves heavy cancellation between the individual terms, with the result that the computed value of $M_{nl}(x)$ may suffer from appreciable loss of accuracy. In our experience this loss of accuracy is comparable to that incurred using divided differences.

14. Conclusions

A method based on a simple recurrence relation for evaluating B-splines has been derived. Error analyses have been used to show that the method is numerically stable and that the computed values possess extremely small relative errors. This result applies even if the knots are unequally spaced and is in fact independent of the knot positions. For B-splines of degree 10 or less the maximum relative error in the computed values of the spline cannot exceed $(70)2^{-t}$, where t is the number of binary bits in the mantissa of the floating-point representation. For splines of degree 100 or less this figure cannot exceed $(700)2^{-t}$. In general the bound for the relative error depends linearly upon the degree of the spline. These error bounds prove to be quite realistic. The stable method possesses advantages in both speed and accuracy over the conventional method. Although the conventional method is adequate in certain circumstances it is difficult to say in advance when these circumstances obtain.

I should like to thank Mr E. L. Albasiny, Mr J. G. Hayes and Professor V. E. Price for reading the manuscript and making many helpful suggestions.

References

ANSELONE, P. M. & LAURENT, P. J. 1968 *Num. Math.* **12**, 66–82.

CARASSO, C. 1966 *Méthodes numérique pour l'obtention de fonctions-spline.* Thèse de 3ème Cycle, Université de Grenoble.

CARASSO, C. & LAURENT, P. J. 1968 *On the numerical construction and the practical use of interpolating spline-functions.* Proc. IFIP Congress, Edinburgh, A6–9.

CURRY, H. B. & SCHOENBERG, I. J. 1966 *J. Analyse math.* **17**, 71–107.

GREVILLE, T. N. E. 1969 In T. N. E. Greville (Ed.), *Theory and application of spline functions*, 1–35. New York: Academic Press.

HERRIOT, J. G. & REINSCH, C. H. 1971 *ALGOL 60 procedures for the calculation of interpolating natural spline functions.* Computer Science Department, Stanford University, Report STAN–CS–71–200.

LAFATA, P. & ROSEN, J. B. 1970 *Communs Ass. comput. Mach.* **13**, 651–9.

NAUR, P. (Ed.) 1963 *Comput. J.* **5**, 349–367.

PETERS, G. & WILKINSON, J. H. 1971 *J. Inst. Maths Applics* **8**, 16–35.

POWELL, M. J. D. 1970 In J. G. Hayes (Ed.), *Numerical approximation to functions and data*, 65–83. London: Athlone Press.

SCHOENBERG, I. J. 1946 *Q. appl. Maths* **4**, 45–99.

SCHUMAKER, L. L. 1969 In T. N. E. Greville (Ed.), *Theory and application of spline functions*, 65–102. New York: Academic Press.

STEFFENSEN, J. F. 1927 *Interpolation.* New York: Chelsea Publishing Co.

WILKINSON, J. H. 1965 *The algebraic eigenvalue problem.* Oxford: Clarendon Press.