MATHEMATICS OF COMPUTATION, VOLUME 25, NUMBER 115, JULY, 1971

A Generalized Interpolation Algorithm

By A. C. R. Newbery

Abstract. An interpolation algorithm is derived, which will construct an $(n + 1)$-point interpolant based on any sequence of interpolatory functions that can be defined by a three-term linear recursion. By suitable parameter choice, a single algorithm can be made to interpolate in terms of the classical polynomial sequences or in terms of trigonometric or hyperbolic series etc. An analysis of truncation error is included.

Given $n + 1$ points (x_i, f_i) with the x_i distinct, and given $n + 1$ functions $\phi_i(x)$, the interpolation problem consists in calculating $n + 1$ weights w_i such that

$$f_i = \sum_{j=0}^{n} w_j \phi_j(x_i), \qquad i = 0, 1, \cdots, n. \tag{1}$$

Apart from the familiar Lagrangian case, where $\phi_j(x) = x^j$, the commonest cases involve basis functions ϕ_j which obey a three-term recursion of the form

$$\phi_{j+1} = (g(x) - \alpha_j)\phi_j - \beta_j \phi_{j-1}, \qquad j = 0, 1, \cdots, \tag{2}$$

where the α_j, β_j are given scalars, $g(x)$ and $\phi_0(x)$ are given functions, and $\phi_{-1}(x) \equiv 0$. This case includes direct interpolation in terms of the classical polynomials of Legendre, Chebyshev, etc.; it also includes interpolation in terms of sine and cosine polynomials and their hyperbolic analogs. We now wish to construct an algorithm for which the input data are vectors with components $x_i, f_i, \alpha_j, \beta_j, \phi_0(x_i), g(x_i)$ and the output vector has components w_j in accordance with (1), (2). The subscripts i, j run through the range $[0, n]$ with $\beta_0 = 0$.

The development of the algorithm will follow the same pattern as used in [1] for the special case of sine series interpolation. First, we develop a sequence of functions $\{\pi_k(x)\}$, $k = 1, 2, \cdots$, with the properties that (a) $\pi_k(x)$ is a linear combination of the $\phi_j(x)$ for $j \leqq k$ with the coefficient of ϕ_k being unity, (b) $\pi_k(x_i) = 0$ for $i < k$ and for no other subscript i. In order to meet these requirements, we set $\pi_1(x) = \phi_1 - (g(x_0) - \alpha_0)\phi_0 = (g(x) - g(x_0))\phi_0$. (It follows from (2) that these two formulations are equivalent.) Subsequently, $\pi_{k+1}(x) = (g(x) - g(x_k))\pi_k(x)$. The fact that (2) defines a way of expressing $g\phi_j$ as a linear combination of $\phi_{j+1}, \phi_j, \phi_{j-1}$ implies that π_{k+1} is of the required form. At this stage, therefore, we may conclude that for each k there exist coefficients $p_j(k)$ such that

$$\pi_k(x) = \prod_{i=0}^{k-1} (g(x) - g(x_i))\phi_0(x) = \sum_{j=0}^{k} p_j \phi_j(x), \qquad p_k = 1. \tag{3}$$

In order to meet the requirement that $\pi_k(x)$ should not vanish at any node x_i other than those for which $i < k$, it is sufficient to impose the restrictions that $\phi_0(x_i) \neq 0$

Received July 23, 1970, revised December 14, 1970.
AMS 1969 *subject classifications.* Primary 6520, 6580; Secondary 4130.
Key words and phrases. Interpolation, osculatory interpolation, Bürmann series.

at all nodes and $g(x_i) \neq g(x_j)$ for all distinct nodes x_i, x_j. These are the minimum restrictions under which the interpolant is constructible. Let $L_k(x)$ denote a linear combination of the basis functions $\phi_0, \phi_1, \cdots, \phi_k$ such that $L_k(x_i) = f_i$ for $i \leqq k$. The L_k are constructible, as is easily verified, by the following recursion:

$$L_0(x) = f_0\phi_0(x)/\phi_0(x_0), \qquad b_k = [f_{k+1} - L_k(x_{k+1})]/\pi_{k+1}(x_{k+1}),$$
$$L_{k+1}(x) = L_k(x) + b_k\pi_{k+1}(x), \qquad k = 0, 1, \cdots, n-1. \tag{4}$$

In executing this algorithm, it is recommended that the evaluation of $\pi_{k+1}(x_{k+1})$ in the expression for b_k should be performed using the product form (3), while one would need to use the summation form in (3) when deriving the ϕ-polynomial form of L_{k+1} from that of L_k. This summation form has been stated explicitly for π_1; thereafter, each one may be computed as follows: Let $\pi_k = \sum_{i=0}^{k} p_i\phi_i(x)$ and $\pi_{k+1} = \sum_{i=0}^{k+1} p_i'\phi_i(x)$, then

$$p_i' = (\alpha_i - g(x_k))p_i + p_{i-1} + \beta_{i+1}p_{i+1}, \qquad j = 0, 1, \cdots, k+1, \tag{5}$$

where the undefined coefficients p_{k-1}, p_{k+1}, and p_{k+2} are set equal to zero. The evaluation of $L_k(x) \equiv \sum_{i=0}^{k} q_i\phi_i(x)$ for $x = x_{k+1}$, which is needed to compute b_k in (4), may be done in either of two ways: First, if a complete matrix of quantities $\phi_i(x_i)$ is available, then the computation is simply a scalar product of two $(k+1)$-dimensional vectors. On the other hand, if this matrix is not available and is not needed for other purposes, it would be a waste of time and storage to compute it specifically for this job. Instead, one should recognize that evaluating L_k is formally equivalent to evaluating a weighted sum of orthogonal polynomials, and a suitable algorithm adapted from [3, p. 70] is:

$$t_k = q_k, \qquad t_{k-1} = q_{k-1} + t_k(g(x) - \alpha_{k-1}),$$
$$t_r = q_r + t_{r+1}(g(x) - \alpha_r) - \beta_{r+1}t_{r+2}, \qquad r = k-2, k-3, \cdots, 0. \tag{6}$$

The value of $L_k(x)$ is then $t_0\phi_0(x)$.

In order to derive an error term associated with our $(n+1)$-point generalized interpolation algorithm, it will be necessary to make some more restrictive assumptions on the nature of $g(x)$, $\phi_0(x)$. Let I be the continuous interval on the x-axis over which interpolation is required; thus I includes all the x_i and any other x-value at which the interpolant may be evaluated. We require that f, g, ϕ_0 be differentiable at least $n+1$ times in I and that neither g' nor ϕ_0 vanishes in I. It will be observed that these conditions imply the weaker conditions that were earlier stated to be necessary and sufficient for the construction of the interpolant. The $(n+1)$-point interpolant $L_n(x)$ is therefore constructible, and by examination of algorithm (6), we see that $L_n(x) \equiv P_n(g)\phi_0(x)$, where P_n is an algebraic polynomial of degree n in the variable $g = g(x)$. Writing $g_i = g(x_i)$, we have therefore implicitly constructed a polynomial P_n such that, at all nodes x_i,

$$f(x_i) = L_n(x_i) = P_n(g_i)\phi_0(x_i).$$

Equivalently, writing $g(x) = z$, we have constructed a polynomial P_n such that

$$P_n(z) = f(g^{-1}(z))/\phi_0(g^{-1}(z)) \tag{7}$$

at every point $z_i = g(x_i)$. The fact that this is a well-defined expression follows from

our assumption that g', ϕ_0 do not vanish in I. The expression for the internodal error $E(z)$ in (7) is the standard Lagrangian error term [2, p. 63]:

$$
\begin{aligned}
E(z) &= f(g^{-1}(z))/\phi_0(g^{-1}(z)) - P_n(z) \\
&= \frac{1}{(n+1)!}\prod_{i=0}^{n}(z - z_i)\left(\frac{d}{dz}\right)^{n+1}[f(g^{-1}(z))/\phi_0(g^{-1}(z))]_{z=\zeta},
\end{aligned} \tag{8}
$$

where $g^{-1}(\zeta) \in I$.

In order to investigate the differential expression in (8), we define $Q_r(x) = (d/dz)^r[f(x)/\phi_0(x)]$. We have $Q_0 = f(x)/\phi_0(x)$, $Q_1 = (\phi_0 f' - \phi_0' f)/g'\phi_0^2$. If we let $Q_r = N_r/g'^{2r-1}\phi_0^{r+1}$, $r = 1, 2 \cdots$, then

$$N_{r+1} = N_r' g'\phi_0 - N_r(2r-1)g''\phi_0 + (r+1)g'\phi_0'.$$

The differential expression N_r is linear in f but its coefficients involve nonlinear combinations of ϕ_0, g' and their derivatives. We conclude that Q_{n+1}, which is the differential expression in (8), is $N_{n+1}/(g'^{2n+1}\phi_0^{n+2})$. Actually, the error in our interpolation is not $E(z)$ but $\tilde{E}(x)$, where

$$\tilde{E}(x) = \phi_0(x)E(z) = \frac{\phi_0(x)}{(n+1)!}\prod_{i=0}^{n}(g(x) - g(x_i))N_{n+1}/[g'^{2n+1}(\eta)\phi_0^{n+2}(\eta)], \tag{9}$$

where N_{n+1} is evaluated at some point η in I. Although the structure of N_{n+1} is too complicated to permit any clear practical insights, there is still something to be learned from the form of the error term (9). At first, it appears that if g' or ϕ_0 becomes small anywhere in I, then large errors are to be expected; however, it may be noted that if ϕ_0 and g are multiplied by constants p, q, the error remains invariant. This can be seen by following our derivation, and it is also evident geometrically. We conclude that large errors may be expected if g' is small relative to g or relative to its higher derivatives (which are implicit in N_{n+1}) anywhere in I; similarly, the smallness of ϕ_0 relative to its derivatives has to be seen as a danger signal. For instance, if one is constructing a cosine series interpolant, one normally takes $\phi_0 = 1$, $g = 2\cos x$. If the interval I contains points η such that $\sin \eta$ is small, then we run into one of the above mentioned problems. If we try to construct a sine series interpolant, then we encounter *both* the above problems.

So far we have assumed that we were required to construct an explicit approximation to $f(x)$ in terms of the basis functions ϕ_i. In the simpler case where all that is needed is an algorithm for pointwise interpolation, the simplest procedure would be to use the fact that, at nodes x_i, $f(x_i) = P_n(g_i)\phi_0(x_i)$; when interpolating, we assume that the same relation holds, at least approximately, also at nonnodal points. At every node x_i, we compute g_i and $\tilde{f}_i = f(x_i)/\phi_0(x_i)$. Then, implicitly or explicitly, we construct the algebraic polynomial $P_n(g)$ such that $P_n(g_i) = \tilde{f}_i$; then we assume that, at any nonnodal point x, $P_n(g(x)) = \tilde{f}(x) = f(x)/\phi_0(x)$. For the given x, we compute $g(x)$ and evaluate $P_n(g)$ for this argument, using any reputable form of polynomial interpolation. This yields an interpolated value for $\tilde{f}(x)$, which must be multiplied by $\phi_0(x)$ to yield the interpolated value of $f(x)$. This procedure, which is mathematically equivalent to the construction and evaluation of $L_n(x)$, will generally be considered computationally preferable when pointwise interpolation is all that is required. The user can apply the efficient and well-understood algorithms for poly-

nomial interpolation; he only needs to pre-edit the data and post-edit the output.

It is also possible to write generalized algorithms of this type for osculatory and other confluent cases; however, we do encounter difficulties which do not arise in the distinct-node case. We will investigate here only the Taylor-series analog, in which the function $f(x)$ is specified by its ordinate and successive derivatives at a single node x_1. The product form of (3) is replaced by

$$\pi_k(x) = (g(x) - g(x_1))^k \phi_0(x). \tag{3'}$$

$L_k(x)$ is defined as that linear combination of $\phi_0, \phi_1, \cdots, \phi_k$ which matches derivatives through the kth with $f(x)$ at x_1. It is defined as in (4), except that

$$b_k = (f^{(k+1)} - L_k^{(k+1)}(x_1))/\pi_{k+1}^{(k+1)}(x_1). \tag{4'}$$

The denominator in (4') simplifies to $(k + 1)!g'^{k+1}(x_1)\phi_0(x_1)$, but we have no simple way, in general, of evaluating the $(k + 1)$th derivative of L_k at x_1. There will be cases where this is a trivial problem, as when $g(x)$, $\phi_0(x)$, α_i, β_i are chosen to yield a sine or cosine series, but there appears to be no simple way of proceeding in the absence of such special information. A more serious objection is that the algorithm may fail to solve solvable problems of confluent type. For instance, it is a well-posed problem to construct a cosine series approximation to an even function whose Maclaurin expansion is known; however, the expression (4') would be undefined because $g(x) = 2 \cos x$ and $g'(0) = 0$. It is possible to bypass this difficulty by writing $\phi_0(x) = 1$, $g(x) = 2 \sin (x/2)$. This generates an expansion in the space of $1, \sin (x/2), \cos x, \sin (3x/2), \cos 2x, \cdots$, and since all odd-order derivatives of the function are zero, so will all coefficients of the sine terms be zero, and we therefore have the required cosine expansion.

It may be noted that when $\phi_0(x) \equiv 1$, the confluent form of our algorithm is mathematically equivalent to the Bürmann series [2, p. 25], although the computational pattern has been simplified and made more adaptable for automatic computation.

In summary, it seems that, in the discrete-node case, we can achieve a high level of versatility without great degradation of performance. This versatility should be valuable in situations where some searching or experimentation is needed before deciding on the exact form of interpolant. Where confluent cases are concerned, the coverage is less good, and *ad hoc* remedies are sometimes needed; nevertheless, there are probably useful applications in this area too.

University of Kentucky
Lexington, Kentucky 40506

1. A. C. R. NEWBERY, "Interpolation by algebraic and trigonometric polynomials," *Math. Comp.*, v. 20, 1966, pp. 597–599. MR 34 #3752.
2. F. B. HILDEBRAND, *Introduction to Numerical Analysis*, McGraw-Hill, New York, 1956. MR 17,788.
3. J. HART ET AL., *Computer Approximations*, Wiley, New York, 1968.