MATHEMATICS OF COMPUTATION
VOLUME 43, NUMBER 167
JULY 1984, PAGES 205–217

Polynomial Interpolation: Lagrange versus Newton

By Wilhelm Werner

Abstract. We show that the Lagrangian form of the interpolating polynomial can be calculated with the same number of arithmetic operations as the Newtonian form.

1. Introduction. It is well known that

$$p(t) = \sum_{i=0}^{n} f_i L_i(t), \qquad L_i(t) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{t - t_j}{t_i - t_j}, \qquad i = 0(1)n, \tag{1.1}$$

is a representation of the unique interpolating polynomial $p \in \Pi_n$ (= space of nth degree complex polynomials) corresponding to the data (t_i, f_i), $i = 0(1)n$, $t_i \neq t_j$ for $i \neq j$.

If we set

$$w_i := 1 \Big/ \prod_{\substack{j=0 \\ j \neq i}}^{n} (t_i - t_j), \qquad i = 0(1)n, \tag{1.2}$$

then (1.1) can be written as

$$p(t) = \sum_{i=0}^{n} w_i f_i \prod_{\substack{j=0 \\ j \neq i}}^{n} (t - t_j). \tag{1.1'}$$

At first glance it seems that about n^2 arithmetic operations (i.e. multiplications or divisions) are necessary to compute the coefficients w_i, $i = 0(1)n$, of this Lagrangian form of p (cf. Knuth [5, p. 484], Winrich [12]), whereas the coefficients of the Newtonian form of p,

$$p(t) = \sum_{i=0}^{n} a_i \prod_{j=0}^{i-1} (t - t_j), \tag{1.3}$$

can be computed with about $n^2/2$ operations via the algorithm of divided differences. It is shown below that $n^2/2$ operations are sufficient for the computation of the coefficients w_i, $i = 0(1)n$, too.

The form (1.1), (1.1') of the interpolating polynomial is of theoretical interest only; for practical purposes it is advisable to use the barycentric representation,

Received February 9, 1983; revised August 1, 1983.
1980 *Mathematics Subject Classification*. Primary 65D05.

0025-5718/84 $1.00 + $.25 per page

which is "eminently suitable for machine computation" (Henrici [3, p. 237]):

$$p(t) = \begin{cases} \dfrac{\sum_{i=0}^{n} \dfrac{w_i}{t - t_i} f_i}{\sum_{i=0}^{n} \dfrac{w_i}{t - t_i}} & \text{if } t \neq t_k, k = 0(1)n, \\ f_k & \text{if } t = t_k, k = 0(1)n. \end{cases} \tag{1.4}$$

This form of the interpolating polynomial which is attributed to Taylor [10] exhibits a remarkable numerical stability (cf. Henrici [3, p. 237 ff.]) which is mainly due to the fact that the interpolation property is preserved even if the coefficients w_i, $i = 0(1)n$, are perturbed; this is not the case for other representations of the interpolating polynomial, e.g. (1.1′). For various important distributions of the interpolating points t_i, $i = 0(1)n$, it is possible to compute the coefficients w_i, $i = 0(1)n$, analytically (cf. Henrici [3, p. 237 ff.]).

One evaluation of (1.4) requires $2n + 3$ multiplications (resp. divisions) and $3n + 1$ additions (resp. subtractions); one evaluation of (1.3) requires but n multiplications and $2n$ additions (resp. subtractions), so that there remains a slight advantage for the Newtonian form of the interpolating polynomial which may be significant if the interpolating polynomial must be evaluated many times. If, however, different interpolating polynomials using the same node points are to be evaluated, then the barycentric form (1.4) of the interpolating polynomial should be preferred.

2. Conversion of the Newtonian Form of the Interpolating Polynomial into the Lagrangian Form. For $f_i = 1$, $i = 0(1)n$, (1.1′) reads

$$1 = \sum_{i=0}^{n} w_i \prod_{\substack{j=0 \\ j \neq i}}^{n} (t - t_j), \tag{2.1}$$

so that for $t \neq t_j, j = 0(1)n$,

$$\frac{1}{\prod_{j=0}^{n}(t - t_j)} = \sum_{i=0}^{n} \frac{w_i}{t - t_i}. \tag{2.2}$$

If we consider the left-hand side of (2.1) as a polynomial in Newtonian form, we may generalize the problem of computing the coefficients w_i, $i = 0(1)n$, as follows:

Given $p \in \Pi_n$ in Newtonian form (1.3); compute the Lagrangian form of p,

$$p(t) = \sum_{i=0}^{n} \sigma_i \prod_{\substack{j=0 \\ j \neq i}}^{n} (t - t_j), \tag{2.3}$$

where $\sigma_i := w_i p(t_i)$, $i = 0(1)n$.

By a well-known result on divided differences (cf. Milne-Thompson [7, p. 7], Steffensen [9, p. 15]) one has:

$$a_k = \sum_{i=0}^{k} \frac{p(t_i)}{\prod_{j=0; j \neq i}^{k}(t_i - t_j)} = \sum_{i=0}^{k} \sigma_i \prod_{j=k+1}^{n} (t_i - t_j), \qquad k = 0(1)n,$$

i.e. our problem is equivalent to the solution of the following triangular system of linear equations:

$$(2.4)\qquad \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \\ a_n \end{pmatrix} = \begin{pmatrix} \prod_{j=1}^{n}(t_0 - t_j) & 0 & 0 & \cdots & 0 \\ \prod_{j=2}^{n}(t_0 - t_j) & \prod_{j=2}^{n}(t_1 - t_j) & 0 & \cdots & 0 \\ \prod_{j=3}^{n}(t_0 - t_j) & \prod_{j=3}^{n}(t_1 - t_j) & \prod_{j=3}^{n}(t_2 - t_j) & 0 & 0 \\ \vdots & \vdots & & & \vdots \\ t_0 - t_n & t_1 - t_n & t_2 - t_n & \cdots & 0 \\ 1 & 1 & 1 & \cdots & 1 \end{pmatrix} \begin{pmatrix} \sigma_0 \\ \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_{n-1} \\ \sigma_n \end{pmatrix}.$$

This system may be solved by Gauss-elimination; as will become more clear later, it is however reasonable to perform the elimination process in a different order of succession:

1. Divide the first equation of (2.4) by $t_0 - t_1$; set
$$a_0^{(1)} := a_0/(t_0 - t_1).$$
2. Subtract the first equation from the second; set
$$a_1^{(1)} := a_1 - a_0^{(1)}.$$
3. Divide the first equation by $t_0 - t_2$; set
$$a_0^{(2)} := a_0^{(1)}/(t_0 - t_2).$$
4. Subtract the first equation from the third; set
$$a_2^{(1)} := a_2 - a_0^{(2)}.$$
5. Divide the second equation by $t_1 - t_2$; set
$$a_1^{(2)} := a_1^{(1)}/(t_1 - t_2).$$
6. Subtract the second equation from the third; set
$$a_2^{(2)} := a_2^{(1)} - a_1^{(2)}.$$

Continuing in this way we get the algorithm

$$(2.5)\qquad \begin{aligned} &a_k^{(0)} := a_k, \qquad k = 0(1)n, \\ &\left.\begin{aligned} a_k^{(i)} &:= a_k^{(i-1)}/(t_k - t_i) \\ a_i^{(k+1)} &:= a_i^{(k)} - a_k^{(i)} \end{aligned}\right\}, \qquad k = 0(1)i-1,\ i = 1(1)n, \\ &\sigma_i := a_i^{(n)}, \qquad i = 0(1)n. \end{aligned}$$

Remarks. (a) Algorithm (2.5) requires $\frac{1}{2}n(n+1)$ divisions and $n(n+1)$ additions for the transformation of a Newtonian polynomial into its Lagrangian representation.

(b) (2.5) was used in [11] for an economical realization of the Durand-Kerner method for the simultaneous determination of polynomial roots. □

The system (2.4) may also be used to solve the converse problem

"Given $p \in \Pi_n$ in Lagrangian form (2.3); compute the coefficients a_i, $i = 0(1)n$, of the Newtonian representation (1.3)"

if $t_i \neq t_j$ for $i \neq j$; this problem is solved by the algorithm

$$
\left.\begin{aligned}
&\sigma_i^{(0)} := \sigma_i, \qquad i = 0(1)n,\\
&\sigma_{n-k}^{(k+1)} := \sum_{i=0}^{n-k} \sigma_i^{(k)}\\
&\sigma_i^{(k+1)} := (t_i - t_{n-k})\sigma_i^{(k)} \qquad i = n-k(-1)0
\end{aligned}\right\}, \qquad k = 0(1)n,
\tag{2.6}
$$

$$
a_i := \sigma_i^{(n+1-i)}, \qquad i = 0(1)n.
$$

3. Efficient Computation of the Lagrangian Form of the Interpolating Polynomial. An application of algorithm (2.5) to (2.1) immediately yields an algorithm for the efficient computation of the quantities w_i, $i = 0(1)n$, which were defined in (1.2):

$$
\left.\begin{aligned}
&a_0^{(0)} := 1, \quad a_k^{(0)} := 0, \qquad k = 1(1)n,\\
&a_k^{(i)} := a_k^{(i-1)}/(t_k - t_i)\\
&a_i^{(k+1)} := a_i^{(k)} - a_k^{(i)}
\end{aligned}\right\}, \qquad k = 0(1)i-1,\ i = 1(1)n,
\tag{3.1}
$$

$$
w_i := a_i^{(n)}, \qquad i = 0(1)n.
$$

Thus we have shown that the coefficients of the Lagrangian form of the interpolating polynomial can be calculated with approximately $n^2/2$ divisions and n^2 additions (resp. subtractions); note that a similar operation count is valid for the algorithm of divided differences which computes the coefficients of the Newtonian form of the interpolating polynomial; cf. Winrich [12].

There is one objection against the Lagrangian form not mentioned so far: it is the assertion that one must know all the data in advance and repeat the computations if interpolation points are added subsequently. On the other hand, it is easy to add a row in the scheme of divided differences if an interpolation point is added; in the following, a direct derivation of (3.1) reveals that (3.1) has the same desirable property. Let $p_n \in \Pi_n$ be the unique solution of the interpolation problem

"Given (t_i, f_i), $i = 0(1)n$, $t_i \neq t_j$ for $i \neq j$; compute $p_n \in \Pi_n$,

$$
p_n(t) = \sum_{m=0}^{n} a_m^{(n)} f_m \prod_{\substack{j=0\\ j\neq m}}^{n} (t - t_j),
\tag{$\mathbf{P}_n$}
$$

such that $p_n(t_i) = f_i$, $i = 0(1)n$."

A natural question arises: How can one compute efficiently the solution $a_k^{(i)}$, $k = 0(1)i$, of $\mathbf{P}_i$, if the coefficients $a_k^{(i-1)}$, $k = 0(1)i-1$, are known?

By the definition of $a_k^{(i-1)}$ (cf. (1.1′), (1.2)) it is obvious that

$$
a_k^{(i)} = a_k^{(i-1)}/(t_k - t_i), \qquad k = 0(1)i-1.
\tag{3.2}
$$

For the computation of $a_i^{(i)}$ use (2.2):

$$\begin{aligned} a_i^{(i)} &= 1 \Big/ \prod_{j=0}^{i-1} (t_i - t_j) && \text{(by definition)} \\ &= \sum_{k=0}^{i-1} \frac{a_k^{(i-1)}}{t_i - t_k} && \text{(by (2.2))} \\ &= -\sum_{k=0}^{i-1} a_k^{(i)} && \text{(by (3.2))}. \end{aligned}$$

The successive solution of the problems $\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_n$ via these formulas then obviously corresponds to algorithm (3.1); this derivation clearly exhibits the fact that the addition of interpolation points presents no difficulties.

Of some practical importance is the special Hermite interpolating polynomial p which satisfies

$$p(t_j) = f_j, \quad p'(t_j) = f_j', \qquad j = 0(1)n.$$

The barycentric form of the interpolating polynomial corresponding to these data is

$$(3.3) \quad p(t) = \begin{cases} \dfrac{\displaystyle\sum_{i=0}^{n} \left\{ \frac{w_i}{t - t_i} \left(\frac{w_i}{t - t_i} - v_i \right) f_i + w_i \frac{w_i}{t - t_i} f_i' \right\}}{\displaystyle\sum_{i=0}^{n} \frac{w_i}{t - t_i} \left(\frac{w_i}{t - t_i} - v_i \right)} & \text{if } t \neq t_j, j = 0(1)n, \\ f_j & \text{if } t = t_j, j = 0(1)n, \end{cases}$$

where the quantities w_i are defined as in (1.2) and

$$v_i := 2w_i \sum_{\substack{j=0 \\ j \neq i}}^{n} \frac{1}{t_i - t_j}, \qquad i = 0(1)n$$

(cf. Henrici [3, p. 255], Bulirsch and Rutishauser [1, p. 248–250]). Using algorithm (3.1) one may compute the quantities w_i, v_i, $i = 0(1)n$, *simultaneously* with about $n^2/2$ multiplications and $n^2/2$ divisions:

$$(3.4) \quad \begin{aligned} &a_0^{(0)} := 1, \quad a_k^{(0)} := 0, \qquad k = 1(1)n, \\ &b_k^{(0)} := 0, \qquad k = 0(1)n, \\ &\left.\begin{aligned} q_{ik} &:= 1/(t_k - t_i) \\ a_k^{(i)} &:= q_{ik} a_k^{(i-1)} \\ a_i^{(k+1)} &:= a_i^{(k)} - a_k^{(i)} \\ b_k^{(i)} &:= b_k^{(i-1)} + q_{ik} \\ b_i^{(k+1)} &:= b_i^{(k)} - q_{ik} \end{aligned}\right\}, \qquad k = 0(1)i - 1, i = 1(1)n, \\ &\left.\begin{aligned} w_i &:= a_i^{(n)} \\ v_i &:= 2w_i b_i^{(n)} \end{aligned}\right\}, \qquad i = 0(1)n. \end{aligned}$$

One evaluation of p given by (3.3) then requires about $4n$ multiplications and n divisions. Note that the algorithm of divided differences (which must be

slightly modified to include the case of confluent interpolating points: cf. Bulirsch and Rutishauser [1, p. 245 ff.]) requires about $2n^2$ divisions for the computation of the coefficients of the corresponding Newtonian form of p and thus is much less efficient than the above algorithm (3.4).

4. On the Numerical Stability of the Barycentric Form of the Interpolating Polynomial. Due to rounding errors algorithm (3.1) usually produces coefficients $\tilde{w}_i$, $i = 0(1)n$, which differ more or less from the exact coefficients w_i, $i = 0(1)n$, defined in (1.2). It is therefore reasonable to study the influence of these errors on the interpolation process. In contrast to other interpolation formulas, the interpolation property of the barycentric formula (1.4) is *not* destroyed by perturbations of the coefficients w_i, $i = 0(1)n$, as long as the computed values of w_i, $i = 0(1)n$, are different from zero. The perturbation of the exact coefficients, however, has the effect that the function

$$p(t) := \begin{cases} \dfrac{\sum_{i=0}^{n} \dfrac{\tilde{w}_i}{t - t_i} f_i}{\sum_{i=0}^{n} \dfrac{\tilde{w}_i}{t - t_i}} & \text{if } t \neq t_j, j = 0(1)n, \\ f_j & \text{if } t = t_j, j = 0(1)n \end{cases}$$

(which is continuous in t_j, $j = 0(1)n$), is usually no longer a polynomial, but a rational function, i.e. the evaluation of the barycentric formula with perturbed coefficients corresponds to an exact evaluation of some rational interpolation formula. The error of this (rational) interpolation process is described in the following

PROPOSITION (C. SCHNEIDER [8]). *Let*
(i) $\alpha_i \in \mathbf{R} \setminus \{0\}$, $i = 0(1)n$,
(ii) $f \in C(\mathbf{R})$,
(iii) $q \in \Pi_n$, $q(t) := \sum_{i=0}^{n} \alpha_i \prod_{j=0; j\neq i}^{n} (t - t_j)$,
(iv)

$$p(t) := \begin{cases} \dfrac{\sum_{i=0}^{n} \dfrac{\alpha_i}{t - t_i} f_i}{\sum_{i=0}^{n} \dfrac{\alpha_i}{t - t_i}} & \text{if } t \neq t_j, j = 0(1)n, \\ f_j & \text{if } t = t_j, j = 0(1)n. \end{cases}$$

If $q(t) \neq 0$, *then*

$$f(t) - p(t) = \frac{\prod_{j=0}^{n} (t - t_j)}{q(t)} (fq)[t_0, t_1, \ldots, t_n, t].$$

(*Here* $h[t_0, \ldots, t_k]$ *denotes the kth divided difference of a function* h *with respect to* $t_0, \ldots, t_k$); *if, furthermore,* $q \in \Pi_m$ *for some* $m < n$, *then the interpolating function* p *is exact for any* $f \in \Pi_{n-m}$.

Proof. If $q(t) \neq 0$, then

$$p(t) = \frac{\sum_{i=0}^{n} q(t_i) f_i \prod_{j=0;\, j\neq i}^{n} \frac{t - t_j}{t_i - t_j}}{q(t)},$$

so that

$$f(t) - p(t) = \frac{q(t)f(t) - \sum_{i=0}^{n} q(t_i) f_i \prod_{j=0;\, j\neq i}^{n} \frac{t - t_j}{t_i - t_j}}{q(t)}$$

$$= \frac{\prod_{j=0}^{n}(t - t_j)}{q(t)} (fq)[t_0, t_1, \ldots, t_n, t]$$

(by the usual remainder formula of polynomial interpolation). The interpolation formula under consideration therefore is exact for any function f such that $fq \in \Pi_n$. □

Remark. (a) Note that algorithm (3.1) preserves the property

$$\sum_{k=0}^{n} a_k^{(n)} = 0,$$

even if the individual coefficients $a_k^{(n)}$, $k = 0(1)n$, are perturbed by rounding errors; the corresponding polynomial q introduced in the above proposition thus is of degree $n - 1$ at most. Algorithm (3.1) combined with the barycentric formula (1.4) thus represents an interpolation process which is exact at least in Π_1 even in the presence of rounding errors.

(b) The numerical stability of the algorithm (3.1) itself strongly depends on an appropriate arrangement of the interpolating points $t_0, t_1, \ldots, t_n$; extensive numerical experiments indicate that one should arrange the nodes t_i, $i = 0(1)n$, such that

$$|\mu - t_0| \geqslant |\mu - t_1| \geqslant \cdots \geqslant |\mu - t_n|, \quad \text{where } \mu := \left(\sum_{i=0}^{n} t_i \right) \Big/ (n + 1).$$

5. Numerical Results. In order to demonstrate effects caused by rounding errors we use two examples where the interpolation process converges *very slowly*; it should be pointed out that our results are not typical for everyday interpolation problems where usually all algorithms which we discuss below yield satisfactory results. We use the following notations:

(a) Algorithm "LAGRANGE (1.2)" consists of

1. computation of the quantities w_i, $i = 0(1)n$, according to the definition (1.2);
2. evaluation of the barycentric form of the interpolating polynomial (1.4).

(b) Algorithm "LAGRANGE (3.1)" consists of

1. computation of the quantities w_i, $i = 0(1)n$, according to (3.1);
2. evaluation of the barycentric form of the interpolating polynomial (1.4).

(c) Algorithm "NEWTON/HORNER" consists of

1. computation of the Newtonian form of the interpolating polynomial by the algorithm of divided differences;

2. evaluation of the interpolating polynomial by Horner's algorithm.

A summary of our numerical experience with the above algorithms is the following:

1. The algorithm LAGRANGE (3.1) is in general not as stable as LAGRANGE (1.2) (or e.g. the Neville-Aitken algorithm).

2. As is the case for the NEWTON/HORNER-algorithm, the numerical stability of LAGRANGE (3.1) strongly depends on an appropriate arrangement of the interpolating points t_i, $i = 0(1)n$. If this aspect, however, is taken into account, then LAGRANGE (3.1) is of similar quality as LAGRANGE (1.2). It is noteworthy in this context that the favorable arrangements of the node points for the algorithms LAGRANGE (3.1) and NEWTON/HORNER are different.

To demonstrate the influence of the arrangement of the interpolating points on the numerical stability we use four different arrangements:

1. "Arrangement 1": $t_0 < t_1 < \cdots < t_n$.

2. "Arrangement 2": $|t - t_0| \leqslant |t - t_1| \leqslant \cdots \leqslant |t - t_n|$, where t is that point where the interpolating polynomial is to be evaluated.

3. "Arrangement 3": Replace " $\leqslant$ " by " $\geqslant$ " in Arrangement 2.

4. "Arrangement 4": $|\mu - t_0| \geqslant |\mu - t_1| \geqslant \cdots \geqslant |\mu - t_n|$, where

$$\mu := \left(\sum_{i=0}^{n} t_i \right) / (n + 1).$$

Remarks. (a) Arrangement 2 is recommended by Hildebrand [4, p. 50], and (with reference to [4]) by Krogh [6] for Newton interpolation; Arrangement 3 is included to demonstrate the effects of an inappropriate arrangement of the interpolating points. As was mentioned already extensive numerical experiments indicated that Arrangement 4 is the appropriate one for our algorithm (3.1).

(b) Note that a change of the arrangement of the node points strongly influences the condition number of the matrix in (2.4). It is well-known that the condition number can be diminished by *scaling*; since the coefficients w_i, $i = 0(1)n$, can be replaced by γw_i, $i = 0(1)n$ (if $\gamma \neq 0$), in the barycentric formula (1.4), one may replace the interpolating points t_i, $i = 0(1)n$, by θt_i, $i = 0(1)n$, where $\theta \neq 0$ is chosen in such a way that the matrix in (2.4) contains no elements of too different orders of magnitude. This device actually (slightly) improved some results in our experiments; the influence of this kind of scaling, however, was by far not as significant as the influence due to the different arrangements of the interpolating points.

Example 1. Approximate

$$\lim_{t \to 0} \frac{\sin(t) + \cos(t) - 1}{t} \quad \text{by } p_n(0),$$

where $p_n \in \Pi_n$ interpolates the function

$$f(t) := \frac{\sin(t) + \cos(t) - 1}{t} \qquad (t \neq 0)$$

at the points

$$t_i = \frac{i - m}{2}\pi \ (i = 0(1)m - 1, m + 1(1)n), \qquad t_m = \frac{\pi}{4} \quad \text{with } m := \left[\frac{n + 1}{2} \right] - 1;$$

it can be shown that

$$\lim_{\substack{t \to 0 \\ t \neq 0}} f(t) = 1 = \lim_{n \to \infty} p_n(0).$$

All computations were done in single as well as in double precision (27 bit resp. 62 bit).

The following tables contain the errors $|p_n(0) - 1|$.

TABLE 1 (*Arrangement* 1)

n	Single precision			Double precision		
	LAGRANGE (1.2)	LAGRANGE (3.1)	NEWTON/ HORNER	LAGRANGE (1.2)	LAGRANGE (3.1)	NEWTON/ HORNER
5	1.94(-3)	1.94(-3)	1.94(-3)	1.94(-3)	1.94(-3)	1.94(-3)
10	1.44(-3)	1.44(-3)	1.44(-3)	1.44(-3)	1.44(-3)	1.44(-3)
15	8.02(-6)	8.05(-6)	7.23(-6)	8.06(-6)	8.06(-6)	8.06(-6)
20	1.66(-5)	1.57(-5)	1.14(-5)	1.67(-5)	1.67(-5)	1.67(-5)
25	8.94(-8)	3.52(-6)	3.75(-5)	8.56(-8)	8.56(-8)	8.56(-8)
30	2.24(-7)	1.20(-5)	2.19(-4)	2.88(-7)	2.88(-7)	2.88(-7)
35	8.94(-8)	5.78(-4)	1.98(-3)	1.27(-9)	1.27(-9)	1.04(-7)
40	0.00	1.56(-3)	1.07(-2)	5.89(-9)	5.89(-9)	1.17(-7)
45	2.98(-8)	1.38(-2)	9.13(-2)	2.24(-11)	2.30(-11)	1.36(-6)
50	5.96(-8)	1.13(-2)	6.71(-1)	1.32(-10)	1.13(-10)	7.87(-6)

TABLE 2 (*Arrangement* 2 *with* $t := 0$)

n	Single precision			Double precision		
	LAGRANGE (1.2)	LAGRANGE (3.1)	NEWTON/ HORNER	LAGRANGE (1.2)	LAGRANGE (3.1)	NEWTON/ HORNER
5	1.94(-3)	1.94(-3)	1.94(-3)	1.94(-3)	1.94(-3)	1.94(-3)
10	1.44(-3)	1.44(-3)	1.44(-3)	1.44(-3)	1.44(-3)	1.44(-3)
15	8.03(-6)	8.05(-6)	8.06(-6)	8.06(-6)	8.06(-6)	8.06(-6)
20	1.66(-5)	1.66(-5)	1.67(-5)	1.67(-5)	1.67(-5)	1.67(-5)
25	8.94(-8)	5.96(-8)	8.94(-8)	8.56(-8)	8.56(-8)	8.56(-8)
30	2.38(-7)	4.32(-7)	2.83(-7)	2.88(-7)	2.88(-7)	2.88(-7)
35	5.22(-8)	0.00	1.12(-7)	1.27(-9)	1.27(-9)	1.18(-7)
40	1.49(-8)	1.86(-7)	1.12(-7)	5.89(-9)	5.89(-9)	1.18(-7)
45	1.49(-8)	2.83(-7)	1.12(-7)	2.24(-11)	2.24(-11)	1.18(-7)
50	1.12(-7)	1.18(-6)	1.12(-7)	1.32(-10)	1.32(-10)	1.18(-7)

TABLE 3 (*Arrangement* 3 *with* $t := 0$)

n	Single precision			Double precision		
	LAGRANGE (1.2)	LAGRANGE (3.1)	NEWTON/ HORNER	LAGRANGE (1.2)	LAGRANGE (3.1)	NEWTON/ HORNER
5	1.94(-3)	1.94(-3)	1.94(-3)	1.94(-3)	1.94(-3)	1.94(-3)
10	1.44(-3)	1.44(-3)	1.44(-3)	1.44(-3)	1.44(-3)	1.44(-3)
15	7.97(-6)	8.05(-6)	7.81(-6)	8.06(-6)	8.06(-6)	8.06(-6)
20	1.67(-5)	1.67(-5)	1.64(-5)	1.67(-5)	1.67(-5)	1.67(-5)
25	1.64(-7)	1.04(-7)	1.13(-6)	8.56(-8)	8.56(-8)	8.56(-8)
30	2.83(-7)	2.53(-7)	2.98(-7)	2.88(-7)	2.88(-7)	2.88(-7)
35	5.96(-8)	7.45(-8)	9.69(-4)	1.27(-9)	1.27(-9)	9.67(-4)
40	1.49(-8)	7.45(-8)	6.06	5.89(-9)	5.89(-9)	6.06
45	5.96(-8)	1.64(-7)	8.46(+1)	2.24(-11)	2.24(-11)	8.46(+1)
50	9.69(-8)	9.98(-7)	3.10(+3)	1.32(-10)	1.32(-10)	3.10(+3)

TABLE 4 (*Arrangement* 4)

n	Single precision LAGRANGE (1.2)	LAGRANGE (3.1)	NEWTON/ HORNER	Double precision LAGRANGE (1.2)	LAGRANGE (3.1)	NEWTON/ HORNER
5	1.94(-3)	1.94(-3)	1.94(-3)	1.94(-3)	1.94(-3)	1.94(-3)
10	1.44(-3)	1.44(-3)	1.44(-3)	1.44(-3)	1.44(-3)	1.44(-3)
15	8.oo(-6)	8.o5(-6)	7.91(-6)	8.o6(-6)	8.o6(-6)	8.o6(-6)
20	1.67(-5)	1.67(-5)	1.64(-5)	1.67(-5)	1.67(-5)	1.67(-5)
25	1.49(-7)	5.96(-8)	2.68(-7)	8.56(-8)	8.56(-8)	8.56(-8)
30	2.24(-7)	2.83(-7)	5.59(-7)	2.88(-7)	2.88(-7)	2.88(-7)
35	2.98(-8)	3.73(-8)	3.25(-4)	1.27(-9)	1.27(-9)	3.24(-4)
40	2.98(-8)	1.49(-8)	2.25	5.89(-9)	5.89(-9)	2.25
45	1.49(-8)	0.00	3.55	2.24(-11)	2.24(-11)	3.55
50	7.45(-8)	2.38(-7)	7.95(+2)	1.32(-10)	1.32(-10)	7.95(+2)

As was noticed very often algorithm NEWTON/HORNER reacts much more sensitively to an inappropriate arrangement of the interpolating points than does the algorithm LAGRANGE (3.1). The conclusion that Arrangement 4 is the appropriate one for LAGRANGE (3.1) is confirmed by the following table which contains the numbers

$$\max\left\{\left|\ln\left|\frac{\tilde{a}_i^{(n)}}{a_i^{(n)}}\right|\right| \, i = 0(1)n\right\},$$

where $a_i^{(n)}$, $i = 0(1)n$, are the exact coefficients in the barycentric representation of p_n, and $\tilde{a}_i^{(n)}$, $i = 0(1)n$, denote those computed by algorithm (3.1).

TABLE 5 (*Influence of different arrangements on algorithm* (3.1))

n	Single precision Arrangement 1	2	4	Double precision Arrangement 1	2	4
5	1.11(-7)	2.23(-7)	5.57(-8)	1.2o(-17)	1.24(-17)	1.24(-17)
10	5.oo(-6)	1.83(-5)	8.44(-8)	1.15(-15)	3.70(-16)	1.28(-17)
15	3.7o(-3)	5.9o(-4)	2.5o(-7)	3.33(-14)	2.59(-15)	7.37(-18)
20	3.10(-1)	7.49(-2)	7.4o(-7)	3.17(-11)	1.oo(-12)	1.64(-17)
25	5.42	1.61	2.16(-6)	6.24(-9)	4.15(-11)	2.87(-17)
30	1.o9(+1)	4.88	4.35(-6)	2.o7(-6)	8.41(-9)	3.22(-16)
35	1.55(+1)	9.35	2.43(-5)	3.64(-4)	9.52(-8)	1.85(-15)
40	2.23(+1)	1.5o(+1)	1.17(-3)	4.22(-2)	7.54(-6)	5.83(-15)
45	2.71(+1)	1.77(+1)	1.24(-3)	2.78	5.32(-4)	3.17(-14)
50	3.64(+1)	2.39(+1)	3.66(-2)	7.92	3.o6(-1)	5.63(-13)

Even though the computed values of $a_i^{(n)}$, $i = 0(1)n$ (at least in part), differ by orders of magnitude from the exact coefficients for $n \geqslant 30$ (for Arrangements 1, 2 and single-precision arithmetic), we achieved reasonable results in Tables 1, 2; this is due to the fact that the barycentric formula represents an interpolation formula even for "false" coefficients.

Example 2 (*Runge's Example*). As was proved by Runge (cf. Davis [2, p. 78], Steffensen [9, p. 35 ff]), convergence of the sequence $\{p_n(t)\}_{n\in\mathbf{N}}$, where $p_n \in \Pi_n$ interpolates the function

$$f(t) := \frac{1}{1+t^2}$$

at the equidistant nodes $t_i := -5 + 10i/n$, $i = 0(1)n$, takes place for $t \leqslant 3.63\ldots$. Since we want to study numerical effects of various interpolation algorithms we do not take advantage of the fact that the coefficients w_i, $i = 0(1)n$, of the barycentric formula (1.4) can be computed explicitly (cf. Henrici [3, p. 239]). In the following tables we present the errors of the three interpolation algorithms under consideration at

$$t = 2.51234567.$$

As in the preceding example all computations were performed in 27-bit and 62-bit arithmetic for four different arrangements of the interpolating points which are described at the beginning of this section.

TABLE 6 (*Arrangement* 1)

	Single precision			Double precision		
n	LAGRANGE (1.2)	LAGRANGE (3.1)	NEWTON/ HORNER	LAGRANGE (1.2)	LAGRANGE (3.1)	NEWTON/ HORNER
5	5.13(-1)	5.13(-1)	5.13(-1)	5.13(-1)	5.13(-1)	5.13(-1)
10	8.54(-1)	8.54(-1)	8.53(-1)	8.54(-1)	8.54(-1)	8.54(-1)
15	1.12(-1)	1.12(-1)	1.12(-1)	1.12(-1)	1.12(-1)	1.12(-1)
20	1.30(-2)	1.30(-2)	1.35(-2)	1.30(-2)	1.30(-2)	1.30(-2)
25	1.88(-2)	2.56(-2)	2.58(-2)	1.88(-2)	1.88(-2)	1.88(-2)
30	3.28(-2)	1.16(-2)	8.o8(-2)	3.28(-2)	3.28(-2)	3.28(-2)
35	4.65(-3)	3.oo(-5)	2.o8(-1)	4.65(-3)	4.65(-3)	4.65(-3)
40	1.o1(-3)	o.oo	2.12	1.o1(-3)	1.o1(-3)	1.o1(-3)
45	6.6o(-4)	o.oo	1.62(+1)	6.62(-4)	6.62(-4)	6.62(-4)
50	1.26(-3)	1.36(-8)	1.34(+2)	1.26(-3)	1.27(-3)	1.26(-3)
55	1.92(-4)	1.36(-8)	1.4o(+3)	1.92(-4)	2.29(-4)	1.92(-4)
60	5.92(-5)	4.o9(-8)	1.53(+4)	5.87(-5)	2.67(-6)	5.88(-5)

TABLE 7 (*Arrangement* 2 *with* $t := 2.51234567$)

	Single precision			Double precision		
n	LAGRANGE (1.2)	LAGRANGE (3.1)	NEWTON/ HORNER	LAGRANGE (1.2)	LAGRANGE (3.1)	NEWTON/ HORNER
5	5.13(-1)	5.13(-1)	5.13(-1)	5.13(-1)	5.13(-1)	5.13(-1)
10	8.54(-1)	8.54(-1)	8.54(-1)	8.54(-1)	8.54(-1)	8.54(-1)
15	1.12(-1)	1.12(-1)	1.12(-1)	1.12(-1)	1.12(-1)	1.12(-1)
20	1.30(-2)	1.30(-2)	1.30(-2)	1.30(-2)	1.30(-2)	1.30(-2)
25	1.88(-2)	1.88(-2)	1.88(-2)	1.88(-2)	1.88(-2)	1.88(-2)
30	3.28(-2)	3.28(-2)	3.28(-2)	3.28(-2)	3.28(-2)	3.28(-2)
35	4.65(-3)	4.74(-3)	4.65(-3)	4.65(-3)	4.65(-3)	4.65(-3)
40	1.o1(-3)	8.4o(-4)	1.o1(-3)	1.o1(-3)	1.o1(-3)	1.o1(-3)
45	6.86(-4)	3.16(-3)	6.62(-4)	6.62(-4)	6.62(-4)	6.62(-4)
50	1.19(-3)	3.82(-2)	1.26(-3)	1.26(-3)	1.26(-3)	1.26(-3)
55	4.15(-5)	1.35(+2)!	1.93(-4)	1.92(-4)	1.92(-4)	1.92(-4)
60	1.41(-4)	2.53(-1)	5.96(-5)	5.87(-5)	5.87(-5)	5.87(-5)

TABLE 8 (*Arrangement* 3 *with* $t := 2.51234567$)

	Single precision			Double precision		
n	LAGRANGE (1.2)	LAGRANGE (3.1)	NEWTON/ HORNER	LAGRANGE (1.2)	LAGRANGE (3.1)	NEWTON/ HORNER
5	5.13(-1)	5.13(-1)	5.13(-1)	5.13(-1)	5.13(-1)	5.13(-1)
10	8.54(-1)	8.54(-1)	8.54(-1)	8.54(-1)	8.54(-1)	8.54(-1)
15	1.12(-1)	1.12(-1)	1.12(-1)	1.12(-1)	1.12(-1)	1.12(-1)
20	1.30(-2)	1.30(-2)	1.30(-2)	1.30(-2)	1.30(-2)	1.30(-2)
25	1.88(-2)	1.96(-2)	1.83(-2)	1.88(-2)	1.88(-2)	1.88(-2)
30	3.28(-2)	1.44(-2)	3.76(-2)	3.28(-2)	3.28(-2)	3.28(-2)
35	4.65(-3)	1.o1(-5)	1.53(-2)	4.65(-3)	4.65(-3)	4.65(-3)
40	1.o1(-3)	0.00	1.61(-1)	1.o1(-3)	1.o1(-3)	1.o1(-3)
45	6.6o(-4)	0.00	5.96(-1)	6.62(-4)	6.62(-4)	6.62(-4)
50	1.26(-3)	0.00	7.1o	1.26(-3)	1.27(-3)	1.26(-3)
55	2.o2(-4)	0.00	3.60(+2)	1.92(-4)	2.23(-4)	1.92(-4)
60	5.15(-5)	0.00	1.82(+3)	5.87(-5)	7.38(-6)	5.87(-5)

TABLE 9 (*Arrangement* 4)

	Single precision			Double precision		
n	LAGRANGE (1.2)	LAGRANGE (3.1)	NEWTON/ HORNER	LAGRANGE (1.2)	LAGRANGE (3.1)	NEWTON/ HORNER
5	5.13(-1)	5.13(-1)	5.13(-1)	5.13(-1)	5.13(-1)	5.13(-1)
10	8.54(-1)	8.54(-1)	8.54(-1)	8.54(-1)	8.54(-1)	8.54(-1)
15	1.12(-1)	1.12(-1)	1.12(-1)	1.12(-1)	1.12(-1)	1.12(-1)
20	1.30(-2)	1.30(-2)	1.30(-2)	1.30(-2)	1.30(-2)	1.30(-2)
25	1.88(-2)	1.88(-2)	1.88(-2)	1.88(-2)	1.88(-2)	1.88(-2)
30	3.28(-2)	3.28(-2)	3.28(-2)	3.28(-2)	3.28(-2)	3.28(-2)
35	4.65(-3)	4.65(-3)	4.65(-3)	4.65(-3)	4.65(-3)	4.65(-3)
40	1.o1(-3)	1.o1(-3)	1.o1(-3)	1.o1(-3)	1.o1(-3)	1.o1(-3)
45	6.66(-4)	6.51(-4)	6.62(-4)	6.62(-4)	6.62(-4)	6.62(-4)
50	1.26(-3)	1.30(-3)	1.26(-3)	1.26(-3)	1.26(-3)	1.26(-3)
55	1.82(-4)	1.62(-4)	1.92(-4)	1.92(-4)	1.92(-4)	1.92(-4)
60	6.77(-5)	8.o9(-5)	5.90(-5)	5.87(-5)	5.87(-5)	5.87(-5)

Conclusions. For small degrees of the interpolating polynomial (say $n \leqslant 20$) the new interpolation algorithm introduced in this note works as satisfactorily as Newton interpolation does. In critical situations, however, where interpolation polynomials of very high degree must be evaluated, both algorithms require a special arrangement of the interpolating points to avoid numerical instabilities. One should note that the favorable arrangement of the nodes for Newton interpolation depends on the argument where the interpolation polynomial is to be evaluated, whereas the arrangement appropriate for our new algorithm depends on the interpolating points only. It was observed very often that Newton interpolation reacts much more sensitively to an inappropriate arrangement of the node points than does our new algorithm; this is certainly due to the fact that the barycentric formula (1.4) represents a reasonable interpolation formula even for perturbed coefficients.

Fachbereich Mathematik
Johannes Gutenberg-Universität
65 Mainz, Federal Republic of Germany

1. R. BULIRSCH & H. RUTISHAUSER, "Interpolation und genäherte Quadratur," *Mathematische Hilfsmittel des Ingenieurs*, Teil III (R. Sauer, I. Szabó, eds.), Springer, Berlin, 1968, pp. 232–319.

2. P. J. DAVIS, *Interpolation and Approximation*, Blaisdell, Waltham, Mass., 1961.

3. P. HENRICI, *Essentials of Numerical Analysis*, Wiley, New York, 1982.

4. F. B. HILDEBRAND, *Introduction to Numerical Analysis*, McGraw-Hill, New York, 1956.

5. D. E. KNUTH, *The Art of Computer Programming*, Vol. 2, Addison-Wesley, Reading, Mass., 1981.

6. F. T. KROGH, "Efficient algorithms for polynomial interpolation and numerical differentiation," *Math. Comp.*, v. 24, 1970, pp. 185–190.

7. L. M. MILNE - THOMPSON, *The Calculus of Finite Differences*, Macmillan, London, 1933.

8. C. SCHNEIDER, Private communication.

9. J. F. STEFFENSEN, *Interpolation*, Chelsea, New York, 1950.

10. W. J. TAYLOR, "Method of Lagrangian curvilinear interpolation," *J. Res. Nat. Bur. Standards*, v. 35, 1945, pp. 151–155.

11. W. WERNER, "On the simultaneous determination of polynomial roots," *Iterative Solution of Nonlinear Systems of Equations* (R. Ansorge, Th. Meis, W. Törnig, eds.), Lecture Notes in Math., Vol. 953, Springer, Berlin, 1982, pp. 188–202.

12. L. B. WINRICH, "Note on a comparison of evaluation schemes for the interpolating polynomial," *Comput. J.*, v. 12, 1969, pp. 154–155.