JOURNAL OF RESEARCH of the National Bureau of Standards—B. Mathematics and Mathematical Physics
Vol. 71B, No. 1, January–March 1967

Stable Evaluation of Polynomials

C. Mesztenyi* and C. Witzgall**

(November 3, 1966)

A class of Newton forms

$$P(x)=a_0+a_1(x-x_0)+\ \ldots\ +a_n(x-x_0)\ \ldots\ (x-x_{n-1})$$

are discussed which admit a stable evaluation algorithm in an interval $[A, B]$. Stability is defined in the paper. The estimate

$$\frac{\Delta P}{|P|}\leqslant 2+6\frac{M'(L)L}{M(L)},$$

where $L:=B-A$ and $M(x):=|a_0|+|a_1|x+\ \ldots\ +|a_n|x^n$, is shown to hold for the relative error of evaluation of $P(x)$ in $[A, B]$.

Key Words: Evaluation, Newton form, polynomial, relative error, round-off.

There are algorithms for polynomial evaluation which require fewer operations (Motzkin [4],[1] Belaga [1], Pan [6], Cheney [2], Knuth [3]), but these algorithms are prone to lose significance. The Horner scheme itself leaves much to be desired in this respect, and this fact has led to the study of methods which reduce the loss of significance, but may require additional operations (Rice [7], Lawson [9]).

An evaluation algorithm depends on the form in which the polynomial is given. Forms on which a "stable" evaluation algorithm can be based are called "well-conditioned." This note calls attention to a class of well-conditioned Newton forms.[2] These forms involve more parameters and require more operations for evaluation than the normal form described above. The number of operations is about the same as for evaluating an expansion by Tchebychef polynomials, but in general more parameters are required.

Loosely speaking, an algorithm is "unstable" if one observes for some arguments an excessive loss of significance. A rigorous, but somewhat narrow, definition of stability will be based on an idealized floating point arithmetic and an error analysis similar to the one carried out by v. Neumann and Goldstine [5], however referring to the relative rather than the absolute error. The arithmetic will be finite in that it will restrict itself to numbers which are "representable" with mantissae of a fixed finite length.[3] It will be infinite in that it will permit arbitrarily large positive and negative exponents. Thus there will be infinitely many representable numbers. Some of the theoretical results of this paper hinge on this idealization, and might be considered artificial. Nevertheless, these results point in the direction of desirable improvements, and provide a reasonable classification of algorithms from the view point of stability.

Our definition of condition and stability differs from that given by Rice [7].

1. Minimal Newton forms. The evaluation algorithms to be considered are based on the

Newton form

of polynomials:

$$P(x)=a_0+a_1(x-x_0)+a_2(x-x_0)\,(x-x_1)+\ \ldots\ +a_n(x-x_0)\ \ldots\ (x-x_{n-1}). \qquad (1.1)$$

Polynomials in this form are evaluated by the following adaption of the Horner scheme:

$$\begin{aligned} D_n &:= a_n \\ D_i &:= a_i+(x-x_i)D_{i+1}, \qquad i=n-1,\ \ldots\ ,0, \\ P(x) &:= D_0. \end{aligned} \qquad (1.2)$$

*University of Maryland, College Park, Md.
**Mathematics Research Laboratory, Boeing Scientific Laboratories, Seattle, Washington 98110.

[1] Figures in brackets indicate the literature references at the end of this paper.

[2] Our study of these forms was stimulated by a discussion with J. Rice on polynomial evaluation. It profited from discussions with A. J. Goldman, who also detected a major mistake in the previous version of this paper.

[3] $0.53_{10}15$ is representable in decimal arithmetic of mantissa length $s\geqslant 2$; 0.1_{10} is not representable in any binary arithmetic.

Instability can arise when adding the coefficients a_i and when subtracting the "critical values" x_i. There are three reasons for being less concerned with the effect of the subtractions:

First, if the values x and x_i are both representable in the given finite arithmetic, then forming the differences $x-x_i$ is a "stable" operation. A similar advantage does not obtain for the addition, even if the coefficients a_i are representable. Furthermore, the effect of replacing x and x_i by representable numbers close to them, in other words, the effect of "rounding," can be estimated by means of the differential formula:

$$dP=[D_1]d(x-x_0)+[D_2(x-x_0)]d(x-x_1)+\ldots$$

$$\ldots+[D_n(x-x_0)\ldots(x-x_{n-2})]d(x-x_n).$$

Here the D_i are the intermediate results of (1.2).

Second, if the terms of the sum $D_i:=a_i+(x-x_i)D_{i+1}$ are of the same sign, then the effect of the relative error of $(x-x_i)D_{i+1}$ on the relative error of D_i will be weighed by the ratio $|(x-x_i)D_{i+1}/D_i|$. The relative error of $x-x_i$ tends to be large when $|x-x_i|$ becomes small. Thus the relative error caused by forming the difference is toned down by the above ratio precisely when it tends to be high. On the other hand, if x and $-x_i$ have the same sign, then one does not observe a similar toning down of the relative error incurred by the preceding addition.

Third, we give a theoretical reason. We shall be able to show that $x-x_i$ can be evaluated "stably" if additional coefficients and operations are introduced, provided x is representable, even if x_i is not representable.

Recognizing the additions as the main cause of instability, the idea is to neutralize them by choosing the critical values x_i so as to ensure that a_i has always the same sign as $(x-x_i)D_{i+1}$.

This is indeed possible in any given interval of evaluation $[A, B]$. The first step of constructing such a Newton form is to divide out all zeros of the polynomial $P(x)$ in $[A, B]$:

$$P(x)=:(x-x_0)\ldots(x-x_m)P^*(x).$$

One then defines $a_0=a_1=\ldots=a_m=0$, and the problem is clearly reduced to finding a suitable Newton form for $P^*(x)$.

Let us therefore assume that $P(x)\neq 0$ in $[A, B]$. If $P(x)$ is given in Newton form (1.1), and if a_0 has the same sign as the sum of the remaining terms for all x in $[A, B]$, then sign $(a_0)=$ sign $(P(x))$, and $|a_0|\leq|P(x)|$ in $[A, B]$. This suggests the following definition of a_0:

$$|a_0|:=\min\{|P(x)|\mid x\epsilon[A, B]\}$$

$$\text{sign}\,(a_0):=\text{sign}\,(P(x)).$$

Then $P(x)-a_0$ has zeroes

$$x_0, x_1, \ldots, x_{k-1}$$

in $[A, B]$. Zeroes are represented as often as their multiplicity indicates. If, for instance, y is a double zero of $P(x)$ in $[A, B]$ then y occurs twice among the x_i. We have

$$P(x)=D_0(x)=a_0+(x-x_0)\ldots(x-x_{k-1})D_k(x).$$

The polynomial $D_k(x)$ does not vanish in $[A, B]$, and can therefore be treated in the same way as $P(x)$. The coefficient a_k is defined by

$$|a_k|:=\min\{|D_k(x)|\mid x\epsilon[A, B]\}$$
$$\text{sign}\,(a_k):=\text{sign}\,(D_k(x)),$$

and

$$x_k, \ldots, x_{l-1}$$

are the zeroes of $D_k(x)-a_k$. We have again

$$D_k(x)=a_k+(x-x_k)\ldots(x-x_{l-1})D(x),$$

and so on. We call the Newton form so constructed the

minimal Newton form

of the polynomial $P(x)$ in $[A, B]$.

If all zeros of $P(x)$ are in $[A, B]$, then the minimal Newton form of $P(x)$ in $[A, B]$ coincides with the "product form" $a_n\Pi(x-x_i)$ of the polynomial. If $P(x)$ does not vanish in $[A, B]$, then all zeros of odd multiplicity encountered in the construction of the minimal Newton form are at the ends of the interval $[A, B]$.

We shall show later that a minimal Newton form can be evaluated in a "stable" manner if the argument x is representable.

2. On transformations into stable forms. It is quite clear that the process of determining the minimal Newton form is, in itself, a very unstable process, which has to be carried out in double or even triple precision. This appears to be a principle: the constants of a better conditioned form contain more "information," and if one wants this additional information, then one has to work for it.

For this reason, we shall not concern ourselves with the difficulties of finding the constants of algorithms. These difficulties are bound to increase with the quality of the end-product. Besides, minimal Newton forms will be used only if a polynomial has to be evaluated repeatedly, as for instance in function subroutines.

3. Evaluation of derivatives. The Horner scheme is frequently extended to determine the value of the first derivative of a polynomial, using the algorithm:

$$E_n:=D_n$$

$$E_i:=D_i+xD_{i+1},\qquad i=n-1,\ldots,1,\qquad (3.1)$$

$P'(x) := E_1,$

where the D_i denote the intermediate results of the Horner scheme for $P(x)$. We shall use this fact in section 11 for deriving an error estimate.

Similar algorithms exist for Newton forms. If the D_i are the intermediate results of algorithm (1.2) applied for a given x, then:

$$P'(x) = D_1 + D_2(x - x_0) + \ldots + D_n(x - x_0) \ldots (x - x_{n-2}),$$

which suggests the algorithm:

$$E_n := D_n$$

$$E_i := D_i + (x - x_{i-1})E_{i+1}, \qquad i = n-1, \ldots, 1$$

$$P'(x) := E_1.$$

This algorithm is in general not stable, even if the original polynomial P is in minimal Newton form.

4. Numerical example.[4] Consider the ill-conditioned polynomial $a_0 + \ldots + x^5$ with the coefficients:

$$\begin{aligned} a_0 &= 4.\ 10074\ 70239\ 8387 \\ a_1 &= -11.\ 29173\ 84073\ 737 \\ a_2 &= 8.\ 42475\ 03796\ 1924 \\ a_3 &= 0.\ 92113\ 31318\ 58071 \\ a_4 &= -3.\ 05937\ 81605\ 8204. \end{aligned}$$

Its minimal Newton form in the interval [0, 1] has the following coefficients and critical values:

$b_0 = 0.00103\ 19917\ 44066\ 05$
$b_1 = 0.0$
$b_2 = 3.41269\ 84126\ 9841$
$b_3 = -1.87912\ 08791\ 2088$
$b_4 = 0.60784\ 31372\ 54902$
$b_5 = 1.0$
$x_0 = 0.83361\ 06489\ 18469$
$x_1 = x_0$
$x_2 = x_3 = 1.0$
$x_4 = 0.0.$

To compare the loss of significance incurred in evaluating the polynomial in normal form and minimal Newton form, the interval [0, 1] was divided into 50 subintervals, in each of which 50 random arguments were generated. The maximum number of significant digits lost was plotted for each subinterval (fig. 1).

5. Representable numbers. This and the following sections will contain some theoretical examinations. An arithmetic will be specified, with respect to which we shall define stability. Actual error estimates are then derived for the evaluation of minimal Newton forms.

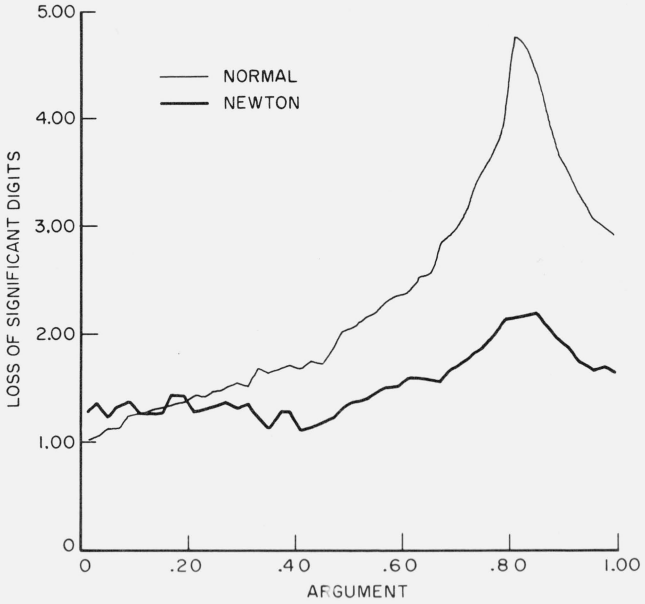

FIGURE 1

Consider a floating point arithmetic with number base β and mantissa length s, and assume that there are no restrictions on the size of the exponents. Denote by

$$\Sigma$$

the set of all numbers which are representable in this arithmetic. To each real number x, we assign a number $\bar{x} \epsilon \Sigma$ such that no other number in Σ is closer to x. Thus

$$|x - \bar{x}| \leq |x - y| \text{ for all } y \epsilon \Sigma. \tag{5.1}$$

In particular, $\bar{x} = x$ if $x \epsilon \Sigma$. We assume that $\overline{(-x)} = -\bar{x}$, therefore $-\Sigma = \Sigma$.

With the notation

$$\alpha(x) := \begin{cases} 0 \text{ if } x \epsilon \Sigma \\ 1 \text{ otherwise} \end{cases}$$

one verifies that[5]

$$|x - \bar{x}| \leq |\bar{x}|\alpha(x)\epsilon \leq |\bar{x}|\epsilon \text{ where } \epsilon := \frac{1}{2}\beta^{-s+1} \tag{5.2}$$

(Scarborough's Theorem I [8] p. 3). Note that (5.2) holds only if the arithmetic does not restrict the size of the exponents. Indeed, it follows from (5.2) that:

$$\bar{x} = 0 \text{ implies } x = 0. \tag{5.3}$$

[4] The computer time for this calculation was supported by the Computer Science Center of the University of Maryland under Grant NsG 398 from the National Aeronautics and Space Administration.

[5] Actually a sharper estimate holds: $|x - \bar{x}| \leq \epsilon\beta^t$ where $t := \lfloor \log_\beta |\bar{x}| \rfloor$, that is, t is the greatest integer smaller than $\log_\beta |\bar{x}|$.

In other words, zero is the only number which gives zero when rounded. This is clearly only true if negative exponents of arbitrarily large modulus are admitted.

6. Stable algorithms. We now assume that each addition, subtraction, multiplication, and division of two numbers in Σ is carried out exactly, and that the result is rounded afterwards. Furthermore, each given number which is not in Σ must be rounded before it enters an operation. These rules appear to be a reasonable idealization of a floating point arithmetic which combines two registers to hold the result of each operation, normalizing and rounding to single precision subsequently.

An algorithm aims at computing a result r as a function of given parameters $a, b, c, \ldots$, but in reality computes a quantity $\hat{r}$. The error that is generated in the course of the algorithm then is defined by

$$\Delta r := |r - \hat{r}|.$$

Consider, for instance, the algorithm which consists of multiplying two given numbers u and v:

$$p := uv$$

Then $\hat{p} := \overline{\overline{u}\,\overline{v}}$, and (5.2) gives

$$|p - \hat{p}| = |uv - u\bar{v} + u\bar{v} - \bar{u}\bar{v} + \bar{u}\bar{v} - \hat{p}|$$

$$\leqslant |u\bar{v}|\epsilon + |\bar{u}\bar{v}|\epsilon + |\hat{p}|\epsilon.$$

In view of

$$|u| \leqslant |\bar{u}| + |u - \bar{u}| \leqslant (1+\epsilon)|\bar{u}|,$$

$$|\bar{u}\bar{v}| \leqslant |\hat{p}| + |\bar{u}\bar{v} - \hat{p}| \leqslant (1+\epsilon)|\hat{p}|,$$

one has

$$\frac{\Delta p}{\epsilon|\hat{p}|} = \frac{|p - \hat{p}|}{\epsilon|\hat{p}|} \leqslant 3 + 3\epsilon + \epsilon^2. \tag{6.1}$$

The upper bound of the above error term does not depend on u and v. Hence we say that the formation of the product uv from given numbers u and v is "stable."

In general, we call an algorithm which computes r from given parameters $a, b, c, \ldots$

stable,

if there exists an error estimate of the form

$$\frac{\Delta r}{\epsilon|\hat{r}|} \leqslant k,$$

where k does *not depend* on the values of the parameters $a, b, c, \ldots$

We turn now to the addition of two given numbers u and v:

$$s = u + v.$$

Then $\hat{s} = \overline{\bar{u} + \bar{v}}$, and one has after a similar calculation:

$$\frac{\Delta s}{\epsilon|\hat{s}|} \leqslant 1 + \frac{|\bar{u}|}{|\hat{s}|}\alpha(u) + \frac{|\bar{v}|}{|\hat{s}|}\alpha(v). \tag{6.2}$$

The estimate (6.2) can be improved. For instance, the first term in (6.2) can be deleted if $|s| \leqslant \min\{|u|, |v|\}$. However, no matter how refined the estimate, it will always depend on u and v. Indeed, $u + v \neq 0$ and $\hat{s} = 0$ may both hold simultaneously, precluding any finite bound. The addition of two numbers is therefore not stable.

In the special case $\alpha(u) = \alpha(v) = 0$, that is, if both u and v are representable, then (6.2) proves the addition to be stable. This is also true if u and v have the same sign.

7. Simplifying error estimates. Henceforth we shall assume that ϵ is very small. Thus we shall delete ϵ in expressions of the form $k + l\epsilon$, if both k and l are *parameter independent:*

$$k + l\epsilon \cong k. \tag{7.1}$$

This leads us, for instance, to equate $|\hat{r}|$ with $|r|$, since

$$|r| \geqslant |\hat{r}| - |r - \hat{r}| \geqslant |\hat{r}| - k\epsilon|\hat{r}| = |\hat{r}|(1 - k\epsilon)$$

and similarly $|\hat{r}| \geqslant |r|(1 - k\epsilon)$, provided the algorithm producing $\hat{r}$ is stable.

We state without proof that, if an algorithm is shown stable with the help of rule (7.1), then it could have been shown stable without it.

8. Error propagation. Suppose that p, u, and v are intermediate results of some algorithm, and that

$$p = uv$$

according to this algorithm. We want to estimate the error of p in terms of Δu and Δv, and the error generated by rounding and multiplying. Now $\hat{p} = \overline{\hat{u}\hat{v}}$, and analogous to the derivation of (6.1) we have

$$\Delta p \leqslant |v|\Delta u + |\hat{u}|\Delta v + |\hat{p}|\epsilon.$$

If the intermediate results u and v were arrived at in a stable manner, then this is also true for p. By (7.1) we can therefore consider $|u|$ and $|\hat{u}|$, as well as $|p|$ and $|\hat{p}|$, to be equal. Similar considerations hold for the division of two intermediate results. We conclude: If u and v are intermediate results of the algorithm which are obtained in a stable manner, then

$$\frac{\Delta uv^{\pm 1}}{\epsilon|uv^{\pm 1}|} \leqslant 1 + \frac{\Delta u}{\epsilon|u|} + \frac{\Delta v}{\epsilon|v|}. \tag{8.1}$$

The estimate (8.1) is usually postulated without assumptions about the algorithm. It is argued that $\frac{\Delta u}{|u|}$ and $\frac{\Delta v}{|v|}$ are comparable to ϵ in magnitude, . . . a claim that comes close to assuming stability of the algorithm . . . , and that therefore the term $\frac{\Delta u}{\epsilon|u|}\cdot\frac{\Delta v}{\epsilon|v|}\epsilon$ may be neglected.

The following estimate can be derived without stability assumptions:

$$\frac{\Delta(u\pm v)}{\epsilon|u\pm v|}\leqslant 1+\frac{|u|}{|u\pm v|}\cdot\frac{\Delta u}{\epsilon|u|}+\frac{|v|}{|u\pm v|}\cdot\frac{\Delta v}{\epsilon|v|}. \tag{8.2}$$

9. Stability of minimal Newton forms. Let us now turn to algorithms for the evaluation of polynomials. Once and for all we will assume that

(9.1) *the argument* x *for which a polynomial* P(x) *is to be evaluated is representable.*

Consider the evaluation of a polynomial in minimal Newton form by the adapted Horner scheme (1.2). All operations in this algorithm are stable except the subtractions $x-x_i$. The existence of a stable evaluation thus depends on the possibility of evaluating the differences $x-x_i$ in a stable manner.

To evaluate the difference $x-y$, we split y into its representable part and remainder

$$y=\bar{y}+r,$$

and evaluate $(x-\bar{y})-r$. Since x and $\bar{y}$ are both representable, and therefore $\Delta x=\Delta\bar{y}=0$, we have by (8.2):

$$\frac{\Delta(x-\bar{y})}{\epsilon|x-y|}\leqslant 1.$$

Then again by (8.2):

$$\frac{\Delta(x-y)}{\epsilon|x-y|}\leqslant 1+\frac{|x-\bar{y}|}{|x-y|}\cdot\frac{\Delta(x-\bar{y})}{\epsilon|x-\bar{y}|}+\frac{|r|}{|x-y|}\cdot\frac{\Delta r}{\epsilon|r|}\leqslant 1+\frac{|x-\bar{y}|+|r|}{|x-y|}.$$

In view of $|x-\bar{y}|=|x-y+r|\leqslant|x-y|+|r|$ this gives

$$\frac{\Delta(x-y)}{\epsilon|x-y|}\leqslant 2+\frac{2|r|}{|x-y|}.$$

Now according to (5.1)

$$|r|=|\bar{y}-y|\leqslant|x-y|,$$

since x is representable. Hence

$$\frac{\Delta(x-y)}{\epsilon|x-y|}\leqslant 4. \tag{9.2}$$

The algorithms $((\bar{x}-\bar{y})+s)-r$ and $(\bar{x}-\bar{y})+(s-r)$, where x is not required to be representable and where s denotes the remainder term $x-\bar{x}$ of x, are not stable. Indeed, if $\bar{x}=\bar{y}$, then both algorithms consist of simply subtracting r from s, which was seen to be unstable in section 6.

Consider the algorithm (1.2) for evaluating Newton forms. Since all other operations of (1.2) are stable, the entire algorithm (1.2) is stable provided the differences $x-x_i$ are evaluated in the stable manner described above.

For the step

$$D_i:=a_i+(x-x_i)D_{i+1},$$

we have by (9.2), (8.1), and (8.2)

$$\frac{\Delta(x-x_i)D_{i+1}}{\epsilon|(x-x_i)D_{i+1}|}\leqslant 1+\frac{\Delta(x-x_i)}{\epsilon|x-x_i|}+\frac{\Delta D_{i+1}}{\epsilon|D_{i+1}|}\leqslant 5+\frac{\Delta D_{i+1}}{\epsilon|D_{i+1}|}.$$

$$\frac{\Delta D_i}{\epsilon|D_i|}\leqslant 1+\frac{|a_i|}{|D_i|}\cdot\frac{\Delta a_i}{\epsilon|a_i|}+\frac{|(x-x_i)D_{i+1}|}{|D_i|}\cdot\frac{\Delta(x-x_i)D_{i+1}}{\epsilon|(x-x_i)D_{i+1}|}$$

$$\leqslant 1+\frac{|a_i|}{|D_i|}+\frac{|(x-x_i)D_{i+1}|}{|D_i|}\left(5+\frac{\Delta D_{i+1}}{\epsilon|D_{i+1}|}\right)$$

$$\leqslant 1+\frac{|a_i|+|(x-x_i)D_{i+1}|}{|D_i|}+\frac{|(x-x_i)D_{i+1}|}{|D_i|}\left(4+\frac{\Delta D_{i+1}}{\epsilon|D_{i+1}|}\right).$$

Since minimal Newton forms are constructed so as to ensure that $|a_i|+|(x-x_i)D_{i+1}|=|D_i|$, we finally have

$$\frac{\Delta D_i}{\epsilon|D_i|}\leqslant 2+\frac{|(x-x_i)\Delta_{i+1}|}{|D_i|}\left(4+\frac{\Delta D_{i+1}}{\epsilon|D_{i+1}|}\right), \tag{9.3}$$

from which one immediately concludes that

$$\frac{\Delta D_i}{\epsilon|D_i|}\leqslant 6+\frac{\Delta D_{i+1}}{\epsilon|D_{i+1}|}.$$

Since $\Delta D_n\leqslant|D_n|\epsilon$, we have finally

$$\frac{\Delta P}{\epsilon|P|}\leqslant 6n+1, \tag{9.4}$$

where n is the degree of the polynomial $P(x)$.

Note that application of the estimate (8.2) is justified since the factors of the products $(x-x_i)D_{i+1}$ have been arrived at in a stable manner.

10. Product forms. Every polynomial with real coefficients can be written in product form:

$$P(x)=a_n\prod_{i=0}^{k-1}(x-x_i)\prod_{i=k}^{k+l-1}(d_i+(x-x_i)^2),$$

$$d_i>0, \qquad k+2l=n. \tag{10.1}$$

A stable evaluation algorithm results if the differences $x-x_i$ are evaluated in the stable manner described in section 9. The algorithm involves not more than $2n+1$ parameters, $2n$ additions, and n multiplications. The following error estimate follows from (9.2), (8.1), and (8.2):

$$\frac{\Delta P}{\epsilon|P|}\leq 5k+11l+1\leq 6n. \tag{10.2}$$

11. Improved estimate. The reader observes not only, that the error estimate (10.2) is better than the estimate (9.4) for minimal Newton forms, but also that fewer operations are required. It is therefore questionable whether minimal Newton forms should be considered at all, unless the error estimate (9.4) can be improved. We proceed to show that this is indeed possible.

Let L denote the length of the interval $[A, B]$ and define

$$B_i:=|a_i|+|a_{i+1}|L+\ldots+|a_n|L^{n-i}, \qquad i=0, \ldots, n.$$

Then $|D_i|\leq B_i$ in $[A, B]$ since

$$D_i=a_i+a_{i+1}(x-x_i)+\ldots+a_n(x-x_i)\ldots(x-x_{n-1}).$$

Therefore

$$\frac{|(x-x_i)D_{i+1}|}{|D_i|}=1-\frac{|a_i|}{|D_i|}\leq 1-\frac{|a_i|}{B_i}=\frac{B_{i+1}}{B_i}L$$

and (9.3) reduces to

$$\frac{\Delta D_i}{\epsilon|D_i|}\leq 2+\frac{B_{i+1}}{B_i}\left(4+\frac{\Delta D_{i+1}}{\epsilon|D_{i+1}|}\right)L.$$

Consequently

$$\frac{\Delta P}{\epsilon|P|}\leq 2+\frac{6B_1}{B_0}L+\frac{6B_1B_2}{B_0B_1}L^2+\ldots+\frac{B_1\ldots B_n}{B_0\ldots B_{n-1}}L^n,$$

and therefore

$$\frac{\Delta P}{\epsilon|P|}\leq 2+\frac{6}{B_0}(B_1L+B_2L^2+\ldots+B_nL^n). \tag{11.1}$$

If we introduce the polynomial

$$M(L):=|a_0|+|a_1|L+|a_2|L^2+\ldots+|a_n|L^n,$$

Then the B_i are the intermediate results of the Horner scheme applied to $M(L)$. Thus $B_0=M(L)$. Moreover, we obtain from (3.1) that $B_1+\ldots+B_nL^{n-1}=M'(L)$, and we may rewrite (11.1) as follows:

$$\frac{\Delta P}{\epsilon|P|}\leq 2+6\frac{M'(L)L}{M(L)}. \tag{11.2}$$

This error estimate establishes a preference for minimal Newton forms in small intervals.

Polynomials with nonnegative coefficients are minimal Newton forms for all intervals $[0, x]$ with $x>0$. For such polynomials one derives analogously

$$\frac{\Delta P}{\epsilon|P|}\leq 2+2\frac{P'(L)L}{P(L)}.$$

12. Approximately minimal Newton forms. Each sequence of n critical values x_i determines a unique Newton representation of a given polynomial $P(x)$. If the values x_i^* approximate the points x_i which determine the minimal Newton form, then the Newton form

$$P(x)=a_0^*+a_1^*(x-x_0^*)+\ldots+a_n^*(x-x_0^*)\ldots(x-x_{n-1}^*),$$

which is determined by the values x_i^* is approximately minimal, and for all practical purposes, it is stable.

For certain polynomials of low degree approximately minimal Newton forms may be even rigorously stable in the sense of section 6. Consider for instance the quadratic polynomial in minimal Newton form

$$Q(x)=a_0+(x-z)^2, \qquad a_0>0.$$

The Newton representation of $Q(x)$ determined by the rounded value $\bar{z}$ takes the form:

$$Q(x)=b_0+b_1(x-\bar{z})+(x-\bar{z})^2, \tag{12.1}$$

where $b_0>r^2$, $b_1=-2r$, with $r:=z-\bar{z}$. We proceed to prove that the evaluation of (12.1) by algorithm (1.2) is stable. We put $s:=x-\bar{z}$. Then

$$D:=b_1+(x-\bar{z})=-2r+s$$

$$Q=b_0+(x-\bar{z})D=b_0+sD.$$

We have

$$\Delta D\leq 2|r|\epsilon+s|\epsilon|+|D|\epsilon$$

$$\Delta sD\leq 2|sD|\epsilon+|s|\Delta D\leq 3|sD|\epsilon+2|rs|\epsilon+s^2\epsilon$$

and

$$\Delta Q \leq |b_0|\epsilon + 3|sD|\epsilon + 2|rs|\epsilon + s^2\epsilon + Q\epsilon \qquad (12.2)$$

By (5.1), $|s-r| = |x-z| \geq |\bar{z}-z| = |r|$. Hence

$$Q = a_0 + (x-z)^2 = (b_0 - r^2) + (s-r)^2 \geq b_0$$

for all representable x. This in turn implies $Q \geq sD \geq 0$. As a consequence, we may delete most of the absolute bars in (12.2):

$$\frac{\Delta Q}{\epsilon} \leq b_0 + 3sD + 2|rs| + s^2 + Q \leq 2Q + 2sD + 2|rs| + s^2.$$

If $rs \leq 0$, then $2|rs| + s^2 = s^2 - 2rs = sD$. Hence

$$\frac{\Delta Q}{\epsilon} \leq 2Q + 3sD \leq 5Q.$$

If $rs > 0$, then $|s| \geq 2|r|$. Indeed $(s-r)^2 \geq r^2$ implies $2rs \leq s^2$, and if $rs > 0$, then we have also $2|r||s| \leq |s|^2$. Furthermore, $sD + 2|rs| = sD + 2rs = s^2$. Finally, we note that

$$\frac{s^2}{Q} \leq \frac{s^2}{(s-r)^2} \leq 4 \text{ for } |s| \geq |2r|.$$

Hence

$$\frac{\Delta Q}{\epsilon |Q|} \leq 2 + \frac{sD}{Q} + \frac{2s^2}{(s-r)^2} \leq 11.$$

This estimate is as good as the one derived from (10.2) taking into account the fact that $a_2 = 1$.

References

[1] Belaga, E. G., Some problems involved in the computation of polynomials, Dokl. Adad. Nauk SSR **123**, 775–777 (1958).

[2] Cheney, E. W., Algorithms for the evaluation of polynomials using a minimum number of multiplications, Tech. Note 2, Computation and Data Processing Center, Aerospace Corp., 1962.

[3] Knuth, D. E., Evaluation of polynomials by computer, Comm. Assoc. Comp. Mach. **5**, 595–599 (1962).

[4] Motzkin, T. S., Evaluation of polynomials, Bull. Amer. Math. Soc. **61**, 163, Abstract 315B (1955).

[5] v. Neumann, J., and Goldstine, Numerical inverting of matrices of high order, Bull. Amer. Math. Soc. **53**, 1021–1099 (1947).

[6] Pan, V. Ya., Certain schemes for the calculation of values of polynomials with real coefficients, Prob. Kibernetiki **5**, 17–29 (1959).

[7] Rice, J. R., On the Conditioning of Polynomial and Rational Forms.

[8] Scarbrough, J. B., Numerical Mathematical Analysis (Johns Hopkins Press, Baltimore, Md., 1930).

[9] Lawson, C. L., Study of methods of polynomial evaluations, Jet Propulsion Laboratory, Interim Report No. 1, Feb. 1964.

FIGURE 1

(Paper 71B1–191)