Computing 30, 189–199 (1983)

Computing

Least Significant Bit Evaluation of Arithmetic Expressions in Single-Precision

S. M. Rump and H. Böhm, Karlsruhe

Received March 24, 1982; revised December 5, 1982

Abstract — Zusammenfassung

Least Significant Bit Evaluation of Arithmetic Expressions in Single-Precision. Single-precision floating-point computations may yield an arbitrary false result due to cancellation and rounding errors. This is true even for very simple, structured arithmetic expressions such as Horner's scheme for polynomial evaluation. A simple procedure will be presented for fast calculation of the value of an arithmetic expression to least significant bit accuracy in single precision computation. For this purpose in addition to the floating-point arithmetic only a precise scalar product (cf. [2]) is required. If the initial floating-point approximation is not too bad, the computing time of the new algorithm is approximately the same as for usual floating-point computation. If not, the essential progress of the presented algorithm is that the inaccurate approximation is recognized and corrected. The algorithm achieves high accuracy, i.e. between the left and the right bound of the result there is at most one more floating-point number. A rigorous estimation of all rounding errors introduced by floating-point arithmetic is given for general triangular linear systems. The theorem is applied to the evaluation of arithmetic expressions.

AMS Subject Classifications: 65G05, 65F99.

Key words: Inclusion, automatic verification of correctness, roundig error, high accuracy, error estimation.

Bestmögliche Einschließung des Wertes eines arithmetischen Ausdrucks in einfachgenauer Rechnung. Durch Auslöschung und Rundungsfehler können in Gleitpunktrechnungen beliebig große Fehler entstehen. Dies trifft bereits zu für sehr einfache, strukturierte Ausdrücke wie etwa das Horner Schema zur Auswertung von Polynomen. Hier wird ein einfacher Algorithmus vorgestellt zur schnellen Berechnung des Wertes eines arithmetischen Ausdrucks. Der Wert wird in einfacher Genauigkeit „eingeschlossen", d. h. es werden Schranken für den Wert berechnet. Zu diesem Zweck wird zusätzlich zu den vier (Gleitpunkt-)Grundrechnungsarten nur ein genaues Skalarprodukt benötigt (s. [2]). Wenn die erste Gleitpunkt-Approximation nicht zu schlecht ist, ist die Rechenzeit des neuen Algorithmus von der gleichen Größenordnung wie die für gewöhnliche Gleitpunktrechnung. Andernfalls wird, und das ist der wesentliche Fortschritt, die Ungenauigkeit der Näherung festgestellt und korrigiert. Die Genauigkeit der Einschließung ist fast bestmöglich, d. h. zwischen linker und rechter Grenze liegt maximal eine weitere Gleitpunktzahl. Eine rigorose Fehlerabschätzung aller Rundungsfehler durch Gleitpunkt-Arithmetik für allgemeine lineare Gleichungssysteme mit Dreiecksmatrix wird gegeben. Der Satz wird auf die Auswertung arithmetischer Ausdrücke angewandt.

0. Introduction

Let S be a subset of $\mathbb{R}$, e.g. the set of single precision floating-point numbers. For the definition of operations in S, VS, (n-tuples over S) and MS (n^2-tuples over S) we refer to [3, 4] (see also [5]). For this purpose a monotone and antisymmetric projection

0010-485X/83/0030/0189/$ 02.20

$\square : \mathbb{R} \to S$ (similar for vectors and matrices) is needed. The operations in S, VS and MS are denoted by $\boxed{*}$, $* \in \{+, -, \cdot, /\}$ with well-known restrictions for division. The main property holding in all spaces is (cf. [3, 4])

$$(RG) \qquad A \boxed{*} B = \square (A * B) \qquad * \in \{+, -, \cdot, /\}$$

where $A, B \in S, VS, MS$ (all meaningful combinations; for details see [3, 4]). The operations $\circledast$, $* \in \{+, -, \cdot, /\}$ in the corresponding interval spaces $\mathbb{I}S$, $\mathbb{I}VS$ and $\mathbb{I}MS$ are defined in [3, 4]. Again, a monotone, antisymmetric, outwards directed rounding $\diamondsuit : \mathbb{P}\mathbb{R} \to \mathbb{I}S$ (similar for vectors and matrices) is required. The fundamental property is

$$(RG) \qquad A \circledast B = \diamondsuit (A * B) \qquad * \in \{+, -, \cdot, /\}.$$

For a thorough discussion see [3, 4], for a compact introduction see [5].

All these operations are well-defined and effectively implementable on computers. This is demonstrated in [3, 4]. For this purpose especially a precise inner product is necessary (cf. [2]).

As long as no overflow or underflow occurs, for $\square$ the following error estimation is satisfied.

$$(E) \qquad \forall a, b \in S: |a \boxed{*} b - a * b| \leqq \varepsilon |a * b| \qquad * \in \{+, -, \cdot, /\}.$$

This holds not only for floating-point operations in S but also for vector and matrix operations (cf. [1, 3, 4]). Moreover

$$(E') \qquad \forall A, B \in \mathbb{I}S: A \circledast B \subseteq (A * B) \cdot [1 - \varepsilon, 1 + \varepsilon], \; * \in \{+, -, \cdot, /\}.$$

This is also true in case of multiplication for $A, B \in \mathbb{I}VS$.

For $A = [a, b] \in \mathbb{I}\mathbb{R}$ with $a, b \in \mathbb{R}$ the diameter $d(A)$, the absolute value $|A|$ and the midpoint $m(A)$ are defined by

$$d(A) := b - a; \; |A| = \max(|a|, |b|); \; m(A) := \tfrac{1}{2}(a + b).$$

For $A = [a, b] \in \mathbb{I}S$ with $a, b \in S$ the midpoint of A is defined by

$$\boxed{m}(A) := a \boxplus (b \boxminus a) \boxslash 2.$$

Because $\square$ is a monotone projection we have $\boxed{m}(A) \geqq a$. If $b \boxminus a$ or $(b \boxminus a) \boxslash 2$ causes underflow or if $a = b$, then $a = \boxed{m}(A) \leqq b$. Otherwise the error estimation (E) states for $c, d \in S$ with $c * d \geqq 0$

$$c \boxed{*} d \leqq (c * d) \cdot (1 + \varepsilon) \qquad \text{for } * \in \{+, -, \cdot, /\}.$$

If $\varepsilon \leqq \sqrt{2} - 1$ then

$$(b \boxminus a) \boxslash 2 \leqq (b \boxminus a)/2 \cdot (1 + \varepsilon) \leqq \frac{b - a}{2} \cdot (1 + \varepsilon)^2 \leqq b - a,$$

$$a + (b \boxminus a) \boxslash 2 \leqq a + b - a = b \qquad \text{and}$$

$$a \boxplus (b \boxminus a) \boxslash 2 \underset{(RG)}{=} \square \left(a + (b \boxminus a) \boxslash 2\right) \leqq \square b = b.$$

Therefore

$$a \leqq \boxed{m}([a, b]) \leqq b.$$

This is not true in general for $\tilde{m}([a,b]) := (a \boxplus b) \boxslash 2$.

In case $2 \notin S$ the estimations remain valid when replacing 2 by some $\alpha \in S$ with $\alpha \geqq (1+\varepsilon)^2$.

1. Arithmetic Expression

Under an arithmetic expression we understand a finite formula consisting of constants and $+, -, \cdot, /, (,)$. Here constants are elements of $S \subseteq \mathbb{R}$ which is a set of floating-point numbers. The arithmetic expression can be transformed to a quotient of two expressions, where in the numerator and denominator quotients may occur but only with constants in the denominator. Example (for constants a, b):

$$a^2 - b + \frac{4a^2}{b(b-a)} \to \frac{(a^2-b)(b-a)+4a^2/b}{b-a}. \tag{1}$$

Further an expression can be altered in such a way, that at most one factor in every product is an expression itself (the other are constants). Example (numerator of (1)):

$$(a^2-b)(b-a)+4a^2/b \to (a^2-b)b-(a^2-b)a+4a^2/b. \tag{2}$$

Therefore we consider restricted expressions which can be obtained by applying the following rules:

1. A constant is a restricted expression.
2. The sum and difference of two restricted expressions is a restricted expression.
3. The product of a restricted expression and a constant is a restricted expression.
4. A restricted expression divided by a constant is a restricted expression.

When evaluating a restricted expression, each rule means computing an intermediate result. Let $a, b, c, \ldots$ be constants and x, y values of subterms. Then a new intermediate result z is obtained in one of the following ways:

1. $z = a$
2. $z = x \pm y$
3. $z = x \cdot a$
4. $z = x/a$ or $a \cdot z = x$.

So a restricted arithmetic expression can be regarded as a system of linear equations. The variables are the intermediate results.

The final division of the including intervals for numerator and denominator of the original arithmetic expression introduces an additional relative error of the magnitude of the relative rounding error unit.

The transformation of an arithmetic expression into a restricted arithmetic expression resp. the quotient of two restricted expressions can be performed automatically. An effective algorithm for this process has been implemented.

Applying only 1., 2., 3. and 4. may result in many variables. The number of variables can be reduced. Example for (2):

$$\begin{aligned} x_1 &= a & x_4 &= x_3 \cdot b \\ x_2 &= a \cdot x_1 & b \cdot x_5 &= x_2 \\ x_3 &= x_2 - b & x_6 &= -a \cdot x_3 + x_4 + 4 \cdot x_5 \end{aligned} \tag{3}$$

For calculating the value of a polynomial $p(\xi) = \sum_{i=0}^{n} a_i \cdot \xi^{n-i}$ we obtain the linear system

$$x_0 = a_0;\ x_{i+1} = \xi \cdot x_i + a_{i+1} \text{ for } 0 \leqq i \leqq n-1. \tag{4}$$

Obviously the linear system of a restricted arithmetic expression is lower triangular. So it is easy to solve by forward substitution and, which is most important, this process can be iterated. Moreover, not only approximations can be achieved but an inclusion of the value of the expression with high accuracy.

2. The Algorithm

Let a linear system with lower triangular matrix be given:

$$L \cdot x = b \text{ with } L \in MS,\ b \in VS \text{ and } L_{ij} = 0 \text{ for } i < j. \tag{5}$$

Here L_{ij} denotes the ij-th component of L.

An approximate solution of (5) can be obtained by

$$x_k = \left(b_k \boxminus \boxed{\sum_{i=1}^{k-1}} L_{ki} \boxdot x_i \right) \boxslash L_{kk},\ 1 \leqq k \leqq n. \tag{6}$$

Applying the Bohlender algorithm (cf. [2]) we get instead of (6)

$$x_k = \square \left\{ b_k - \sum_{i=1}^{k-1} L_{ki} \cdot x_i \right\} \boxslash L_{kk}, \quad 1 \leqq k \leqq n.$$

with only two roundings in the computation of each component. Let $(\tilde{x}_1, \ldots, \tilde{x}_n) =: \tilde{x}$ be any approximate solution of (5). Then the residue of (5) with respect to that approximation can be included with least significant bit accuracy using Bohlender's algorithm:

$$\Delta_k = \diamondsuit \left\{ b_k - \sum_{i=1}^{k} L_{ki} \cdot \tilde{x}_i \right\}, \quad 1 \leqq k \leqq n.$$

Then with

$$Y_k = \diamondsuit \left\{ \Delta_k - \sum_{i=1}^{k-1} L_{ki} \cdot \tilde{x}_i \right\} \diamondslash L_{kk}, \quad 1 \leqq k \leqq n \tag{7}$$

and $Y := (Y_1, \ldots, Y_n)$ we have

$$L^{-1} \cdot b \in \tilde{x} \oplus Y.$$

Therefore the value of the arithmetic expression is included in $\tilde{x}_n \oplus Y_n$.

If the diameter of $\tilde{x}_n \oplus Y_n$ is small enough, we are ready with an inclusion of the value of the given arithmetic expression. If not, an iteration is performed with a residue of higher order: Let $\tilde{x}^0 := \tilde{x}$ and $\tilde{x}^1 := \boxed{m}(Y)$.

Compute

$$\Delta_k = \diamondsuit \left\{ b_k - \sum_{i=1}^{k} L_{ki} \cdot \tilde{x}_i^0 - \sum_{i=1}^{k} L_{ki} \cdot \tilde{x}_i^1 \right\}, \quad 1 \leqq k \leqq n$$

and then Y_k again by (7). Then

$$L^{-1} \cdot b \in \tilde{x}^0 \oplus \tilde{x}^1 \oplus Y \tag{8}$$

and the value of the arithmetic expression is included in $\tilde{x}_n^0 \oplus \tilde{x}_n^1 \oplus Y_n$.

Applying again Bohlender's algorithm we obtain a sharper inclusion $\diamondsuit (\tilde{x}^0 + \tilde{x}^1 + Y)$ instead of (8).

We summarize this procedure in an algorithm.

Algorithm 1: Evaluation of arithmetic expressions.

A) {Initialization}
for $k := 1$ **to** n **do** $\tilde{x}_k^0 := \square \left\{ b_k - \sum_{i=1}^{k-1} L_{ki} \cdot \tilde{x}_i^0 \right\} \boxslash L_{kk}$; $p := 0$; **goto** C);

B) {Computation of midpoint}
for $k := 1$ **to** n **do** $\tilde{x}_k^p := \boxed{m}\,(Y_k)$;

C) {Computation of residue}
for $k := 1$ **to** n **do** $\Delta_k := \diamondsuit \left\{ b_k - \sum_{j=0}^{p} \sum_{i=1}^{k} L_{ki} \cdot \tilde{x}_i^j \right\}$;

D) {New inclusion}
for $k := 1$ **to** n **do** $Y_k := \diamondsuit \left\{ \Delta_k - \sum_{i=1}^{k-1} L_{ki} \cdot Y_i \right\} \diamondsuit L_{kk}$;

E) {End of loop}
$U := \diamondsuit \left\{ \sum_{j=0}^{p} \tilde{x}_n^j + Y_n \right\}$; $p := p + 1$;
if {succ(succ(inf(U))) $\geqq$ sup(U) **or** ($p > 10$ **and** underflow)}
then goto F) **else goto** B);

F) {Result}
U is an inclusion of the value of the arithmetic expression described by $Lx = b$;

inf(U) resp. sup(U) denotes the lower resp. upper bound of U. Therefore inf(U), sup(U) $\in S$. Further succ(s), $s \in S$ denotes the successor of s in the floating-point screen S. Therefore the condition

$$\text{succ}(\text{succ}(\inf(U))) \geqq \sup(U)$$

bounds the diameter of U to two units of the last digit of the mantissa of U. In case of underflow the succeeding estimations are not valid, therefore in this case p is limited to 10 to assure termination of the algorithm.

In algorithm 1 it is important to use Bohlender's algorithm in the steps A), C), D) and E).

3. Accuracy of the Result

Next we estimate the diameter of the intervals Y_i, $1 \leqq i \leqq n$ and therefore especially the maximum error of the final inclusion

$$\diamondsuit\left(\sum_{j=0}^{p} \tilde{x}_n^j + Y_n\right)$$

of the value of the arithmetic expression after the p-th iteration. As long as no over- or underflow occurs these estimations include *all* rounding errors during computation as well as the overestimation due to dependecies of intervals. We start the discussion with some definitions.

Define

$$M_i := \max\left\{\frac{1}{|L_{ii}|} \cdot \sum_{v=1}^{i-1} |L_{iv}|, 1\right\} \text{ for } 1 \leqq i \leqq n \tag{9}$$

and

$$N_i := \prod_{v=1}^{i} M_v \quad \text{for } 1 \leqq i \leqq n. \tag{10}$$

To distinguish the Δ_i resp. Y_i for different values of p we introduce a superindex p to the Δ_i resp. Y_i. To avoid misunderstandings we repeat the steps B), C) and D) of algorithm 1 in short (vector) notation. Here, for instance, $\tilde{x}^p$ denotes the vector $(\tilde{x}_1^p, \ldots, \tilde{x}_n^p)$.

B) $\tilde{x}^p := \boxed{m}(Y^p)$;

C) $\Delta^p := \diamondsuit\left\{b - \sum_{j=0}^{p} L \cdot \tilde{x}^j\right\}$;

D) **for** $k := 1$ **to** n **do** $Y_k^{p+1} := \diamondsuit\left\{\Delta_k^p - \sum_{i=1}^{k-1} L_{ki} \cdot Y_i^{p+1}\right\} \diamondslash L_{kk}$;

We assume in the following that during computation no over- or underflow occurs.

Define

$$A_0^p := 0;\ A_i^p := \max_{1 \leqq v \leqq i} \frac{|Y_v^p|}{N_v} \quad \text{for } 1 \leqq i \leqq n, p \geqq 1 \tag{11}$$

$$D_0^p := 0;\ D_i^p := \max_{1 \leqq v \leqq i} \frac{d(Y_v^p)}{N_v} \quad \text{for } 1 \leqq i \leqq n,\ p \geqq 1. \tag{12}$$

To prove the succeeding lemma we need the following relation.

$$\begin{aligned} Y_i^{p+1} &= \left\{\diamondsuit\left(\Delta_i^p - \sum_{v=1}^{i-1} L_{iv} \cdot Y_v^{p+1}\right)\right\} \diamondslash L_{ii} \\ &\subseteq \left\{\left(\Delta_i^p - \sum_{v=1}^{i-1} L_{iv} \cdot Y_v^{p+1}\right) / L_{ii}\right\} \cdot [1-\varepsilon, 1+\varepsilon]^2 \text{ for } 1 \leqq i \leqq n, p \geqq 0. \end{aligned} \tag{13}$$

Lemma 2: *For A_i^p, D_i^p from definitions* (11), (12) *the following estimations hold:*

$$A_i^{p+1} \leqq \sum_{v=1}^{i} \left\{ \frac{|\Delta_v^p|}{|L_{vv}| \cdot N_v} \cdot (1+\varepsilon)^{2(i-v+1)} \right\} \tag{14}$$

$$D_i^{p+1} \leqq \sum_{v=1}^{i} \left\{ \frac{d(\Delta_v^p)}{|L_{vv}| \cdot N_v} (1+\varepsilon)^2 + \frac{|Y_v^{p+1}|}{N_v} \cdot 4\varepsilon \right\} \cdot (1+\varepsilon)^{2(i-v)}, \tag{15}$$

both for $1 \leqq i \leqq n$ and $p \geqq 0$.

Proof: From (13) we get (cf. [13]) for $1 \leqq i \leqq n$ and $p \geqq 0$

$$|Y_i^{p+1}| \leqq \left\{ \frac{|\Delta_i^p|}{|L_{ii}|} + M_i \cdot \max_{1 \leqq v \leqq i-1} |Y_v^{p+1}| \right\} \cdot (1+\varepsilon)^2.$$

Therefore

$$\frac{|Y_i^{p+1}|}{N_i} \leqq \left\{ \frac{|\Delta_i^p|}{|L_{ii}| \cdot N_i} + A_{i-1}^{p+1} \right\} \cdot (1+\varepsilon)^2 \quad \text{for } 1 \leqq i \leqq n, p \geqq 0.$$

This implies (14) by induction. Again from (13) we get (cf. [1]) for $1 \leqq i \leqq n$, $p \geqq 0$

$$d(Y_i^{p+1}) \leqq \left\{ \frac{d(\Delta_i^p)}{|L_{ii}|} + M_i \cdot \max_{1 \leqq v \leqq i-1} d(Y_v^{p+1}) \right\} \cdot (1+\varepsilon)^2 + |Y_i^{p+1}| \cdot 4\varepsilon.$$

Therefore

$$\frac{d(Y_i^{p+1})}{N_i} \leqq \left\{ \frac{d(\Delta_i^p)}{|L_{ii}| \cdot N_i} + D_{i-1}^{p+1} \right\} \cdot (1+\varepsilon)^2 + \frac{|Y_i^{p+1}|}{N_i} \cdot 4\varepsilon$$

for $1 \leqq i \leqq n$ and $p \geqq 0$. This implies (15). □

The next lemma estimates the diameter and absolute value of Δ_i^{p+1} dependent on $d(Y^{p+1})$.

Lemma 3: *For $1 \leqq i \leqq n$ and $p \geqq 0$ the following is true:*

$$|\Delta_i^{p+1}| \leqq (1+\varepsilon) \cdot \sum_{v=1}^{i} |L_{iv}| \cdot d(Y_v^{p+1}) \tag{16}$$

and

$$d(\Delta_i^{p+1}) \leqq \varepsilon \cdot |\Delta_i^{p+1}|. \tag{17}$$

Proof: Steps C) and B) of the algorithm yield for $p \geqq 0$ and x^* with $L \cdot x^* = b$

$$\begin{aligned} \Delta^{p+1} &= \diamondsuit \left\{ b - \sum_{j=0}^{p+1} L \cdot \tilde{x}^j \right\} \\ &\subseteq \diamondsuit \left\{ L \cdot x^* - \sum_{j=0}^{p} L \cdot \tilde{x}^j - L \cdot Y^{p+1} \right\} \\ &\subseteq \left\{ L \cdot \left(x^* - \sum_{j=0}^{p} \tilde{x}^j - Y^{p+1} \right) \right\} \cdot [1-\varepsilon, 1+\varepsilon]. \end{aligned}$$

This implies (16) because of

$$x^* \in \sum_{j=0}^{p} \tilde{x}^j + Y^{p+1}.$$

Estimation (17) is clear because $\diamondsuit: \mathbb{R} \to \mathbb{I}S$ rounds to the interval of smallest possible diameter.

□

The following theorem gives an estimation of $d(Y_i^{p+2})$ depending on $d(Y_i^{p+1})$.

Theorem 4: *For $1 \leqq i \leqq n$ and $p \geqq 0$ the following estimation holds:*

$$D_i^{p+2} \leqq 5\varepsilon \cdot (1+\varepsilon)^{2i+1} \cdot i^2 \cdot D_i^{p+1}. \tag{18}$$

Proof:

$$\begin{aligned}
D_i^{p+2} &\overset{(15)}{\leqq} \sum_{v=1}^{i} \left\{ \frac{d(\Delta_v^{p+1})}{|L_{vv}| \cdot N_v} (1+\varepsilon)^2 + \frac{|Y_v^{p+2}|}{N_v} \cdot 4\varepsilon \right\} (1+\varepsilon)^{2(i-v)} \\
&\overset{(17,14)}{\leqq} \sum_{v=1}^{i} \left\{ \frac{\varepsilon \cdot |\Delta_v^{p+1}|}{|L_{vv}| \cdot N_v} (1+\varepsilon)^2 + \sum_{\mu=1}^{v} \frac{|\Delta_\mu^{p+1}|}{|L_{\mu\mu}| \cdot N_\mu} (1+\varepsilon)^{2(v-\mu+1)} \cdot 4\varepsilon \right\} \cdot (1+\varepsilon)^{2(i-v)} \\
&\leqq \sum_{v=1}^{i} \sum_{\mu=1}^{v} \frac{|\Delta_\mu^{p+1}|}{|L_{\mu\mu}| \cdot N_\mu} \cdot (1+\varepsilon)^{2(i-\mu+1)} \cdot 5\varepsilon \\
&\overset{(16)}{\leqq} \sum_{v=1}^{i} \sum_{\mu=1}^{v} \sum_{\alpha=1}^{\mu} \frac{|L_{\mu\alpha}| \cdot d(Y_\alpha^{p+1})}{|L_{\mu\mu}| \cdot N_\mu} \cdot (1+\varepsilon)^{2i-2\mu+3} \cdot 5\varepsilon \\
&\overset{(9)}{\leqq} 5\varepsilon \cdot \sum_{v=1}^{i} \sum_{\mu=1}^{v} \left\{ \frac{M_\mu}{N_\mu} \cdot \max_{1 \leqq \alpha \leqq \mu-1} d(Y_\alpha^{p+1}) + \frac{d(Y_\mu^{p+1})}{N_\mu} \right\} \cdot (1+\varepsilon)^{2i-2\mu+3} \\
&\overset{(12)}{\leqq} 5\varepsilon \cdot \sum_{v=1}^{i} \sum_{\mu=1}^{v} \left\{ D_{\mu-1}^{p+1} + \frac{d(Y_\mu^{p+1})}{N_\mu} \right\} \cdot (1+\varepsilon)^{2i-2\mu+3}.
\end{aligned}$$

For $1 \leqq \mu \leqq v \leqq i$,

$$D_{\mu-1}^{p+1} \leqq D_{v-1}^{p+1} \quad \text{and} \quad \frac{d(Y_\mu^{p+1})}{N_\mu} \leqq D_v^{p+1} \quad \text{and} \quad D_{v-1}^{p+1} \leqq D_v^{p+1} \leqq D_i^{p+1}.$$

Then $D_0^{p+1} = 0$ and $\sum_{v=1}^{i} (2v-1) = i^2$ finishes the proof. □

Corollary 5: For $p \geqq 0$ the $(p+1)$-st inclusion of the value of the arithmetic expression satisfies:

$$d(Y_n^{p+1}) \leqq \{\varepsilon \cdot 5 \cdot (1+\varepsilon)^{2n+1} \cdot n^2\}^p \cdot N_n \cdot \max_{1 \leqq v \leqq n} \frac{d(Y_v^1)}{N_v}. \tag{19}$$

As far as no over- or underflow occurs (19) is a rigorous estimation of the diameter of the inclusion of the value of the arithmetic expression including all rounding errors.

It is a worst case estimation. The estimation remains valid when using any $\tilde{x}^p \in Y^p$ instead of $\boxed{m}(Y^p)$ in step B) of Algorithm 1.

Theorem 4 and Corollary 5 have been derived (and are valid) for arbitrary triangular linear systems. In this context they are applied to an algorithm for evaluating arithmetic expressions.

4. Computational Results

The algorithm described to compute the value of arithmetic expressions has been implemented on a mini-computer based on the Z80 with decimal arithmetic and 12-digit mantissa, on a UNIVAC 1108 and on an IBM 370/168. Following we give some sample tests for the algorithm.

1. $a+b-a$ for $a=1_{10}30$, $b=1$
2. $b^2(4a^4+b^2-4a^2)-8a^6$ for $a=470832$, $b=665857$
3. $\sum_{i=1}^{5} a_i^2-\frac{1}{5}\left(\sum_{i=1}^{5} a_i\right)^2$ for $a_i=7.951_{10}7+i-3$, $1\leqq i\leqq 5$

   Remark: Expressions like this occur in least square approximation.
4. $f(t)=((543339720\,t-768398401)\,t-1086679440)\,t+1536796802$
   a) $t=1.4142$
   b) $t=1.41421356238$
   c) $t=1.414213561$
5. $(f(1-h)-2f(1)+f(1+h))/h^2$ for

   $$f(t)=\frac{4970\,t-4923}{4970\,t^2-9799\,t+4830},$$

   a) $h=1_{10}-4$
   b) $h=1_{10}-5$
   c) $h=1_{10}-12$

   This formula has to be treated as a single expression.

The following table shows the computational results on the minicomputer of the Institute for Applied Mathematics at the University of Karlsruhe. On this computer a 12-decimal-digit floating-point arithmetic with maximum accuracy (cf. [2]) is implemented. In the columns of the table are displayed from left to right:

The number of the example.
The floating-point approximation $\tilde{x}$.
The correct value x^* of the expression rounded to 12 decimal digits.
The number p of iterations in the algorithm.
The final result U of the algorithm.

	$\tilde{x}$	x^*	p	U
1.	0.0	+1.0	1	+1.0
2.	$+5.0_{10}23$	+1.0	2	+1.0
3.	−100000.0	+10.0	1	+10.0
4. a)	+0.280	+0.282673919360	1	+0.282673919360
4. b)	+0.00695	$+7.32719247117_{10}-14$	2	$+7.3271924711\,{}^{8}_{7}{}_{10}-14$
4. c)	−0.01	$+2.89746134369_{10}-9$	2	$+2.8974613436\,{}^{9}_{8}{}_{10}-9$
5. a)	90.00895	70.7881908792	1	$70.788190879\,{}^{3}_{1}$
5. b)	4500.004	93.7679047546	2	$93.767904754\,{}^{7}_{5}$
5. c)	0.0	94.0000000000	3	94.0000000001 93.9999999999

The displayed final results U in the fifth column of the table are either points, i.e. exact floating-point numbers, or intervals where between the left and the right bound there is at most one floating-point number. As seen from the fourth column in the table typically one iteration is necessary to achieve least significant bit accuracy of the result.

Finally we demonstrate the estimation on the accuracy of the result by example 4. b). The linear system is the following:

$$\begin{aligned} x_1 &= a \\ -t\cdot x_1 + x_2 &= b \\ -t\cdot x_2 + x_3 &= c \\ -t\cdot x_3 + x_4 &= d \end{aligned} \quad \text{with} \quad \begin{aligned} a &= 543339720 \\ b &= -768398401 \\ c &= -1086679440 \\ d &= 1536796802 \end{aligned}$$

Then $M_1 = 1$ and $M_2 = M_3 = M_4 = t$ and $N_i = t^{i-1}$ for $1 \leqq i \leqq 4$.

Computing with 12 decimal digits in the mantissa yields

$$\begin{aligned} \tilde{x}_1^0 &= 543339720 \\ \tilde{x}_2^0 &= 0.0037517336 \\ \tilde{x}_3^0 &= -1086679439.99 \\ \tilde{x}_4^0 &= 0.0066386684238 \end{aligned}$$

and

$$Y_4^1 = [-0.00663866842374,\ -0.00663866842372].$$

The first approximation $\tilde{x}_4^0$ for the value of the polynomial is the result computed by Horner's scheme with a maximum accurate arithmetic. The magnitude of the approximation is completely incorrect. The first inclusion is

$$\tilde{x}_4^0 \oplus Y_4^1 = [6\cdot 10^{-14},\ 8\cdot 10^{-14}]$$

which demonstrates the magnitude of the correct solution. The second iteration yields

$$\tilde{x}_4^1 = -0.00663866842373$$

and

$$Y_4^2 = [3.27192471173 \cdot 10^{-15},\ 3.27192471174 \cdot 10^{-15}].$$

Therefore the second inclusion is the final result:

$$\diamondsuit(\tilde{x}_4^0 + \tilde{x}_4^1 + Y_4^2) = [7.32719247117 \cdot 10^{-14},\ 7.32719247118 \cdot 10^{-14}]. \quad (20)$$

Estimation (19) yields

$$d(Y_4^2) \leqq 1.7 \cdot 10^{-23}, \text{ whereas in fact } d(Y_4^2) = 10^{-26}.$$

Furthermore estimation (19) tells that

$$d(Y_4^{p+1}) \leqq (8.01 \cdot 10^{-10})^p \cdot 2 \cdot 10^{-14}$$

which corresponds to at least 18 correct figures of $\diamondsuit(\tilde{x}_4^0 + \tilde{x}_4^1 + \tilde{x}_4^2 + Y_4^3)$.

References

[1] Alefeld, G., Herzberger, J.: Einführung in die Intervallrechnung. Mannheim-Wien-Zürich: Bibliographisches Institut 1974.
[2] Bohlender, G.: Floating-point computation of functions with maximum accuracy. IEEE Trans. on Computers *1977*.
[3] Kulisch, U.: Grundlagen des numerischen Rechnens. (Reihe Informatik, 19.) Mannheim-Wien-Zürich: Bibliographisches Institut 1976.
[4] Kulisch, U., Miranker, W. L.: Computer arithmetic in theory and practice. Academic Press 1982.
[5] Rump, S. M.: Solving non-linear systems with least significant bit accuracy. Computing *29*, 183–200 (1982).
[6] Stoer, J.: Einführung in die Numerische Mathematik I. (Heidelberger Taschenbücher, Bd. 105.) Berlin-Heidelberg-New York: Springer 1972.

Dr. S. M. Rump
Dipl.-Math. H. Böhm
Institut für Angewandte Mathematik
Universität Karlsruhe
Kaiserstrasse 12
D-7500 Karlsruhe
Federal Republic of Germany

Printed in Austria
Druck: Paul Gerin, A-1021 Wien