A priori Worst-Case Error Bounds for Floating-Point Computations

Walter Krämer
Institut für Wissenschaftliches Rechnen
und Mathematische Modellbildung (IWRMM)
Universität Karlsruhe
D-76128 Karlsruhe, Germany

Abstract

A new technique for the a priori calculation of rigorous error bounds for floating-point computations is introduced. The theorems given in the paper combined with interval arithmetic lead to the implementation of reliable software-routines, which enables the user to compute the desired error bounds automatically by a suitable computer program.

As a prominent example a table-lookup algorithm for calculating the function exp(x) - 1 that has been published by Tang [15] is analyzed using these new tools. The result shows the high quality of the new approach.

1. Introduction

It is a well known fact, that the results of floating-point computations may be totally wrong: For the numerical solution of the following simple linear system of equations IEEE double arithmetic [4, 8] is used (64 bit data format, best possible (semimorphic) floating-point operations). The components of the true solution of the system $Ax = b$ with

$$\mathrm{A} = (a_{ij}) := \begin{pmatrix} 64919121 & -159018721 \\ 41869520.5 & -102558961 \end{pmatrix}, \; \mathrm{b} := \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

are given by $x_1 = a_{22}/(a_{11}a_{22} - a_{12}a_{21})$, $x_2 = -a_{21}/(a_{11}a_{22} - a_{12}a_{21})$. The corresponding floating-point computations result in the machine numbers $\widetilde{x}_1 = 102558961.0, \widetilde{x}_2 = 41869520.5$. **All** computed **digits** are **wrong!** Although the data are exactly representable and only 5 floating-point operations are done, the computed "solution" has nothing to do with the exact solution[1].

Such "numerical catastrophes" are extremely rare in practice, but to cite Kahan [7]:

> Significant discrepancies [between the computed and the true result] are very rare, too rare to worry about all the time, yet not rare enough to ignore.

[1]The solution of the system is $x_1 = 205117922, x_2 = 83739041$

In the present paper a methodology is described, which allows to decide whether the floating-point computation of a given real valued expression delivers an accurate result. Here, the independent variables as well as the parameters of the expression may vary within given intervals (the width of which may be large). Then the conclusion drawn from the application of the new error-analyzing tools will be valid for floating-point evaluations of the expression with arbitrary point data for the independent variables and the parameters lying in their corresponding intervals.

Moreover, the tools which are discussed enable the user to analyze complete program parts including loops, iterations, and recursion. Also intermediate results in the underflow range are treated in a reliable manner.

As an example the result of a rigorous worst-case error estimation for an implementation of an accurate table-lookup algorithm for exp(x)-1 described by Tang [15] is shown. Finding a reliable worst case error bound for such an algorithm by hand is very cumbersome and error prone. A lot of different cases have to be analyzed separately. One must be sure that **all** different branches are considered when performing the error estimation[2].

2. Some Notations and Conventions

$S = S(b, l, \underline{e}, \overline{e})$ denotes a floating-point system with base b, with l mantissa digits, and exponent e in the range $\underline{e} \le e \le \overline{e}$. $S(2, 53, -1022, 1023)$ corresponds to the IEEE double data format (64 bit). $\circ \in \{+, -, \bullet, /\}$ denotes a real operation and $\boxdot, \circ \in \{+, -, \bullet, /\}$ the corresponding floating-point operation (round to nearest). Notice the implicit priority rules used in the notation

[2]In [13] Tang proposed a table-lookup algorithm for logarithms. The user is supplied with a very small over all error bound. Nevertheless, some constants in the table shown in the cited paper are wrong and must be corrected! Apparently this fact has not been detected by the error analysis done by hand. (The correct values for the constants C_lead(j), j=17, 57, and 73 are `3FBF EC91 31DC 0000`, `3FD7 92A5 5FDD 8000`, and `3FDC E1AF 0B86 0000`, respectively. Here, the same notation as in [13], pages 397 and 398 is used.)

$|a \circ b - a \boxdot b| := |(a \circ b) - (a \boxdot b)|$. MinReal is the smallest positive normalized floating-point number, MaxReal the largest floating-point number. For the IEEE double format these numbers are MinReal:=$2.22\ldots \cdot 10^{-308}$, MaxReal:=$1.78\ldots \cdot 10^{308}$. $\widetilde{a}$, $\widetilde{f}$ are quantities computed in floating-point arithmetic, ε^* means Wilkinson's epsilon $\varepsilon^* := \frac{1}{2}b^{1-l}$, i. e. $\varepsilon^* = \frac{1}{2}2^{1-53} = 1.11022\ldots \cdot 10^{-16}$ for the IEEE double format. $I\!R^+$ denotes the set of all positive real numbers, $I\!I\!R$ the set of all closed real (finite) intervals, $IS = \{[\underline{a}, \overline{a}] | \underline{a}, \overline{a} \in S, \underline{a} \leq \overline{a}\}$ the set of all intervals bounded by floating-point numbers, $|A| := \max_{a \in A} |a|$ the maximum absolute value, $< A >:= \min_{a \in A} |a|$ the minimum absolute value and $\mathrm{diam}(A) := \sup(A) - \inf(A)$ the diameter (width) of an interval $A \in I\!I\!R$.

3. Theoretical Results

Let f denote an expression and $f(a)$ its evaluation at the point $a \in A \in I\!I\!R$. Let $\widetilde{f}(\widetilde{a})$ be the result of the corresponding floating-point computations. One is interested in an a priori error bound $\Delta(f)$ for the maximal absolute error

$$|f(a) - \widetilde{f}(\widetilde{a})| \leq \Delta(f)$$

valid for all $a \in A$ and all $\widetilde{a} = a + \Delta_a$ with $\Delta_a \leq \Delta(a)$. Here the interval A and the error bound $\Delta(a)$ for $\widetilde{a}$ are assumed to be known.

To achieve this, the original expression is broken down into basic operations like $+, -, \cdot, /$ and $\sqrt{x}$ for which the following theorems will give reliable worst case error bounds. These operations are used as basics because they are required by the IEEE Standard for Binary Floating-Point Arithmetic [4]. For the fundamental operations $\circ \in \{+, -, \bullet, /\}$ this standard guarantees:

$$\bigwedge_{\substack{a, b \in S \text{ with} \\ |a \circ b| \in [\mathtt{MinReal}, \mathtt{MaxReal}]}} \left| \frac{a \circ b - a \boxdot b}{a \circ b} \right| \leq \varepsilon^* .$$

However, the methodology is not restricted to this set of basic operations. It can be extended, for example, to a complete set of operations (functions like sin, cos etc.) as it is commonly available in a higher programming language like FORTRAN or C (see Section 6).

It should be emphasized that in contrast to conventional error estimations also higher order error terms and underflow situations are treated rigorously by the following theorems. Using an interval arithmetic as well as directed rounded operations makes it possible to build subroutines preserving these properties. Such software tools are supplied by the PASCAL-XSC module eps_ari (see Section 4).

The following lemma bounds the maximal absolute error for results lying in the underflow range.

Lemma 1 (Underflow Range) For results lying in the underflow range it holds

$$\bigwedge_{\substack{a, b \in S \\ |a \circ b| < \mathtt{MinReal}}} |a \boxdot b - a \circ b| \leq \mathtt{MinReal} . \qquad (1)$$

Proof: sign($a \boxdot b$) = sign($a \circ b$) and diam([0, MinReal]) = MinReal □.

What follows is a discussion of the error propagation for the basic binary operations with arguments that are afflicted by rounding errors.

Let $\circ \in \{+, -, \bullet, /\}$ and $a \in A \in I\!I\!R$, $b \in B \in I\!I\!R$,

$$S \ni \widetilde{a} = a + \Delta_a \text{ with } |\Delta_a| \leq \Delta(a),$$

$$S \ni \widetilde{b} = b + \Delta_b \text{ with } |\Delta_b| \leq \Delta(b).$$

It is assumed that the quantities $A, B, \Delta(a)$ and $\Delta(b)$ are given. The goal is to find a bound $\Delta(\circ) \in I\!R^+$ with

$$\bigwedge_{\substack{a \in A,\ b \in B \\ a \circ b, \widetilde{a} \circ \widetilde{b} \in [-\mathtt{MaxReal}, \mathtt{MaxReal}]}} \bigwedge_{\substack{|a - \widetilde{a}| \leq \Delta(a) \\ |b - \widetilde{b}| \leq \Delta(b)}} \left| a \circ b - \widetilde{a} \boxdot \widetilde{b} \right| \leq \Delta(\circ) .$$

The case of addition is treated in

Theorem 1 (Addition) For the addition $\circ := +$ it holds:

$$\bigwedge_{\substack{a \in A \\ b \in B}} \bigwedge_{\substack{|a - \widetilde{a}| \leq \Delta(a) \\ |b - \widetilde{b}| \leq \Delta(b)}} \left| a + b - \widetilde{a} \boxplus \widetilde{b} \right| \leq \Delta(\mathrm{add})$$

with

$$\Delta(\mathrm{add}) := \mathtt{MinReal} + \varepsilon^* |A + B| + (1 + \varepsilon^*) \Big(\Delta(a) + \Delta(b) \Big)$$

Proof: Let $a \in A, b \in B$, $\widetilde{a} = a + \Delta_a \in A + [-\Delta(a), \Delta(a)], |\Delta_a| \leq \Delta(a)$ and $\widetilde{b} = b + \Delta_b \in B + [-\Delta(b), \Delta(b)], |\Delta_b| \leq \Delta(b)$ arbitrary but fixed.

I) Error bound for $|\widetilde{a} + \widetilde{b} - \widetilde{a} \boxplus \widetilde{b}|$:

a) $\widetilde{a} + \widetilde{b} \in U$, the exact sum of the disturbed arguments lies in the underflow range:

$$\widetilde{a} + \widetilde{b} \in U \Longrightarrow \widetilde{a} \boxplus \widetilde{b} \in U$$

$$\overset{\text{Lemma 1}}{\Longrightarrow} |\widetilde{a} + \widetilde{b} - \widetilde{a} \boxplus \widetilde{b}| \leq \mathtt{MinReal}$$

b) $\widetilde{a}+\widetilde{b} \notin U$, i.e. the exact sum $|\widetilde{a}+\widetilde{b}|$ is an element of [MinReal, MaxReal] :

$$\begin{aligned}\widetilde{a}+\widetilde{b} \notin U &\Longrightarrow \widetilde{a} \boxplus \widetilde{b} \notin U \\ &\Longrightarrow |\widetilde{a}+\widetilde{b}-\widetilde{a} \boxplus \widetilde{b}| \\ &\leq \varepsilon^* |\widetilde{a}+\widetilde{b}| \\ &\leq \varepsilon^* \Big(|A+B| + \triangle(a) + \triangle(b) \Big).\end{aligned}$$

This holds, because from $\widetilde{a} \in A + [-\triangle(a), \triangle(a)]$ it follows $|\widetilde{a}| \leq |A| + \triangle(a)$.

II) Error bound for $|a+b-(\widetilde{a}+\widetilde{b})|$:

It holds:

$$\begin{aligned}&|a+b-(\widetilde{a}+\widetilde{b})| \\ &\quad \leq |a-\widetilde{a}| + |b-\widetilde{b}| \\ &\quad \leq \triangle(a) + \triangle(b)\end{aligned}$$

Over all error bound:

$$\begin{aligned}&|a+b-\widetilde{a} \boxplus \widetilde{b}| \\ &\leq |a+b-(\widetilde{a}+\widetilde{b})| + |\widetilde{a}+\widetilde{b}-\widetilde{a} \boxplus \widetilde{b}| \\ &\overset{\text{I),II)}{\leq} \triangle(a) + \triangle(b) + \\ &\qquad \begin{cases} \texttt{MinReal}, & \text{Case I) a)} \\ \varepsilon^* \Big(|A+B| + \triangle(a) + \triangle(b) \Big), & \text{Case I) b)} \end{cases} \\ &\leq \\ &\begin{cases} \triangle(a) + \triangle(b) + \texttt{MinReal}, & \text{Case I) a)} \\ \varepsilon^* |A+B| + (1+\varepsilon^*) \Big(\triangle(a) + \triangle(b) \Big), & \text{Case I) b)} \end{cases} \\ &\leq \varepsilon^* |A+B| + (1+\varepsilon^*) \Big(\triangle(a) + \triangle(b) \Big) + \texttt{MinReal} \\ &= \triangle(\text{add}).\end{aligned}$$

The quantities $a, b, \widetilde{a}, \widetilde{b}$ have been chosen arbitrarily. This proves Theorem 1 □

Theorem 2 (Subtraction) The error bound $\triangle(\text{sub})$ for the subtraction is very similar to the error bound $\triangle(\text{add})$ for the addition. It holds:

$$\triangle(\text{sub}) := \texttt{MinReal} + \varepsilon^* |A-B| + (1+\varepsilon^*) \Big(\triangle(a) + \triangle(b) \Big)$$

Proof: Use the representation $a - b = a + (-b)$ and the proof of Theorem 1 □

Theorem 3 (Multiplication) For the multiplication $\circ := \cdot$ it holds:

$$\bigwedge_{\substack{a \in A \\ b \in B}} \bigwedge_{\substack{|a-\widetilde{a}| \leq \triangle(a) \\ |b-\widetilde{b}| \leq \triangle(b)}} \Big| a \cdot b - \widetilde{a} \boxdot \widetilde{b} \Big| \leq \triangle(\text{mul})$$

with

$$\triangle(\text{mul}) := \texttt{MinReal} + |A||B|\varepsilon^* + \Big(|A| \triangle(b) + |B| \triangle(a) + \triangle(a) \triangle(b) \Big) \cdot (1+\varepsilon^*)$$

Proof: Let $a \in A, b \in B$, $\widetilde{a} = a + \triangle_a \in A + [-\triangle(a), \triangle(a)], |\triangle_a| \leq \triangle(a)$ and $\widetilde{b} = b + \triangle_b \in B + [-\triangle(b), \triangle(b)], |\triangle_b| \leq \triangle(b)$ arbitrary but fixed.

I) Estimation for $|\widetilde{a} \cdot \widetilde{b} - \widetilde{a} \boxdot \widetilde{b}|$:

a) $\widetilde{a} \cdot \widetilde{b} \in U \Longrightarrow \widetilde{a} \boxdot \widetilde{b} \in U$
$\overset{\text{Lemma 1}}{\Longrightarrow} |\widetilde{a} \cdot \widetilde{b} - \widetilde{a} \boxdot \widetilde{b}| \leq \texttt{MinReal}$

b) $\widetilde{a} \cdot \widetilde{b} \notin U \Longrightarrow |\widetilde{a} \boxdot \widetilde{b}| \notin U$

$$\begin{aligned}&\Longrightarrow |\widetilde{a} \cdot \widetilde{b} - \widetilde{a} \boxdot \widetilde{b}| \leq \varepsilon^* |\widetilde{a} \cdot \widetilde{b}| \\ &\leq \varepsilon^* (|A| + \triangle(a))(|B| + \triangle(b)) \\ &= \varepsilon^* (|A||B| + |A|\triangle(b) + |B|\triangle(a) + \triangle(a)\triangle(b)).\end{aligned}$$

II) Error bound for $|a \cdot b - \widetilde{a} \cdot \widetilde{b}|$:

$$\begin{aligned}&|a \cdot b - \widetilde{a} \cdot \widetilde{b}| \\ &\quad = |a \cdot b - (a + \triangle_a)(b + \triangle_b)| \\ &\quad = |a\,\triangle_b + b\,\triangle_a + \triangle_a\,\triangle_b| \\ &\quad \leq |A| \triangle(b) + |B| \triangle(a) + \triangle(a) \triangle(b).\end{aligned}$$

Over all error bound:

$$\begin{aligned}&|a \cdot b - \widetilde{a} \boxdot \widetilde{b}| \leq |a \cdot b - \widetilde{a} \cdot \widetilde{b}| + |\widetilde{a} \cdot \widetilde{b} - \widetilde{a} \boxdot \widetilde{b}| \\ &\overset{\text{I),II)}{\leq} \\ &\begin{cases} \texttt{MinReal} \\ \quad + |A| \triangle(b) + |B| \triangle(a) + \triangle(a) \triangle(b), & \text{I) a)} \\ (|A| \triangle(b) + |B| \triangle(a) + \triangle(a) \triangle(b))(1+\varepsilon^*) \\ \quad + |A||B|\varepsilon^*, & \text{I) b)} \end{cases} \\ &\leq \texttt{MinReal} + |A||B|\varepsilon^* \\ &\qquad + (|A| \triangle(b) + |B| \triangle(a) + \triangle(a) \triangle(b))(1+\varepsilon^*) \\ &= \triangle(\text{mul}) \quad \square\end{aligned}$$

To prove Theorem 4 below, the following lemma will be useful:

Lemma 2 (Inverse)

$$\bigwedge_{|\eta| \le \eta^* < 0.5} \left| \frac{1}{1+\eta} \right| \le 1 + \varepsilon(\text{inv})$$

with $\varepsilon(\text{inv}) := (1 + 2\eta^*) \cdot \eta^*$.

Proof: With $|\eta| \le \eta^* < 0.5$ one gets

$$1 - \eta^* \le \frac{1}{1+\eta} \le 1 + (1 + 2\eta^*) \cdot \eta^* \quad \square$$

Theorem 4 (Division) In case of the division $\circ := /$ it holds:

$$\bigwedge_{\substack{a \in A \\ b \in B}} \bigwedge_{\substack{|a - \tilde{a}| \le \Delta(a) \\ |b - \tilde{b}| \le \Delta(b)}} \left| a/b - \tilde{a} \not\square \tilde{b} \right| \le \Delta(\text{div}) \text{ with}$$

$$\Delta(\text{div}) := \texttt{MinReal} + \frac{1}{<B> - \Delta(b)} \cdot \left(\Delta(a) + (|A| + \Delta(a)) \cdot (\varepsilon^* + \frac{\Delta(b)}{<B>} + 2(\frac{\Delta(b)}{<B>})^2) \right)$$

Here we have to make the additional assumption

$$\Delta(b) < \; 0.5 \cdot <B> \qquad (2)$$

Assumption (2) means, that the interval B is sufficiently far away fom zero. In practice this is not a serious restriction.

Proof: Let $a \in A, b \in B$,
$\tilde{a} = a + \Delta_a \in A + [-\Delta(a), \Delta(a)], \Delta_a \le \Delta(a)$ and
$\tilde{b} = b + \Delta_b \in B + [-\Delta(b), \Delta(b)], \Delta_b \le \Delta(b)$ arbitrary but fixed.

I) Estimation for $|\tilde{a}/\tilde{b} - \tilde{a} \not\square \tilde{b}|$:

a) $\tilde{a}/\tilde{b} \in U \Longrightarrow \tilde{a} \not\square \tilde{b} \in U$
$\overset{\text{Lemma 1}}{\Longrightarrow} |\tilde{a}/\tilde{b} - \tilde{a} \not\square \tilde{b}| \le \texttt{MinReal}$

b) $\tilde{a}/\tilde{b} \notin U \Longrightarrow |\tilde{a} \not\square \tilde{b}| \notin U$
$\Longrightarrow |\tilde{a}/\tilde{b} - \tilde{a} \not\square \tilde{b}| \le \varepsilon^* |\tilde{a}/\tilde{b}|$
$\le \varepsilon^* (|A| + \Delta(a)) \cdot \frac{1}{<B> - \Delta(b)}$

Here it is used that $\tilde{b} \in B + [-\Delta(b), \Delta(b)]$ yielding
$$\frac{1}{|\tilde{b}|} \le \frac{1}{<B> - \Delta(b)}.$$

II) Error bound for $|a/b - \tilde{a}/\tilde{b}|$:

$$\begin{aligned} |a/b - \tilde{a}/\tilde{b}| &= |\frac{a}{b} - \frac{(a+\Delta_a)}{b} \cdot \frac{1}{1 + \frac{\Delta_b}{b}}| \\ &= |\frac{a}{b} - \frac{(a+\Delta_a)}{b} \cdot (1 + \varepsilon_{\text{inv}})| \\ &= \frac{1}{|b|} |\Delta_a + (a + \Delta_a) \cdot \varepsilon_{\text{inv}}| \\ &\overset{(*)}{\le} \frac{1}{<B>} \Big(\Delta(a) + (|A| + \Delta(a)) \cdot \varepsilon(\text{inv}) \Big) \end{aligned}$$

with $|\varepsilon_{\text{inv}}| \le \varepsilon(\text{inv}) := (1 + 2 \cdot \frac{\Delta(b)}{<B>}) \cdot \frac{\Delta(b)}{<B>}$. The last inequality (*) results from Lemma 2 and assumption (2).

Over all error bound:

$$\begin{aligned} & |a/b - \tilde{a} \not\square \tilde{b}| \le |a/b - \tilde{a}/\tilde{b}| + |\tilde{a}/\tilde{b} - \tilde{a} \not\square \tilde{b}| \\ \overset{\text{I),II)}{\le} & \frac{1}{<B>} \Big(\Delta(a) + (|A| + \Delta(a)) \cdot \varepsilon(\text{inv}) \Big) + \\ & \begin{cases} \texttt{MinReal}, & \text{Case I) a)} \\ \varepsilon^* (|A| + \Delta(a)) \cdot \frac{1}{<B> - \Delta(b)}, & \text{Case I) b)} \end{cases} \\ \le & \texttt{MinReal} + \frac{1}{<B> - \Delta(b)} \cdot \\ & \Big(\Delta(a) + (|A| + \Delta(a)) \cdot (\varepsilon^* + \varepsilon(\text{inv})) \Big) \\ = & \Delta(\text{div}) \quad \square \end{aligned}$$

In the following theorem the square root function (a unary operation) is considered.

Theorem 5 (Square root) Let the auxiliary function $h(a)$ be defined by

$$h(a) = \varepsilon^* \sqrt{a} + \frac{\Delta(a)(1 + \varepsilon^*)}{2\sqrt{a - \Delta(a)}} \qquad (3)$$

with $a \in A$ and $a > \Delta(a)$. Then it holds:

$$\bigwedge_{\Delta(a) < a \in A} \bigwedge_{|a - \tilde{a}| \le \Delta(a)} \left| \sqrt{a} - \widetilde{\sqrt{\tilde{a}}} \right| \le \Delta(\text{sqrt})$$

where

$$\Delta(\text{sqrt}) := \max\{h(\inf(A)), h(\sup(A))\}$$

Proof: Let $a \in A, \tilde{a} = a + \Delta_a$, with $|\Delta_a| \le \Delta(A)$ arbitrary but fixed. Using the Mean-Value Theorem one finds the following representation

$$\begin{aligned} \widetilde{\sqrt{\tilde{a}}} &= \widetilde{\sqrt{a + \Delta_a}} = \sqrt{a + \Delta_a}(1 + \varepsilon_{\sqrt{}}) \\ &= \left(\sqrt{a} + \frac{\Delta_a}{2\sqrt{\xi}} \right) (1 + \varepsilon_{\sqrt{}}) \end{aligned}$$

with

$$\xi = \xi(a, \tilde{a}) \in [\min\{a, \tilde{a}\},\ \max\{a, \tilde{a}\}] \subseteq A + [-\Delta(a),\ \Delta(a)]\,.$$

The square root function is assumed to conform to the IEEE-754 standard, i.e. $|\varepsilon_{\surd}| \leq \varepsilon^*$, yielding

$$\left|\sqrt{a} - \widetilde{\sqrt{\tilde{a}}}\right| \leq h(a)\,.$$

Now the auxiliary function h has exactly one minimum in the range $(0, \infty)$, i. e. $\max\{h(x) | x \in A\} = \max\{h(\inf(A)),\ h(\sup(A))\}$ □

4. Usage of the PASCAL-XSC Module eps_ari

The theorems of the preceding section are coded in PASCAL-XSC [9] using interval operations. Of course any other programming language that provides an interval arithmetic can be used. Five functions named `DeltaAdd`, `DeltaSub`, `DeltaMul`, `DeltaDiv`, and `DeltaSqrt`, respectively are supplied by the module eps_ari. Calling the function `DeltaAdd` in the following assignment

```
DeltaRes:=
   DeltaAdd( A, B, DeltaA, DeltaB );
```

gives as result a machine number `DeltaRes` $\in S$ with

$$\bigwedge_{\substack{a \in A \\ b \in B}} \bigwedge_{\substack{|a - \tilde{a}| \leq \Delta(a) \\ |b - \tilde{b}| \leq \Delta(b)}} \left|a + b - \tilde{a} \boxplus \tilde{b}\right| \leq \Delta(\text{add})$$

$$\leq \texttt{DeltaRes}$$

(see Theorem 1). Here $A \subseteq \texttt{A} \in IS$, $B \subseteq \texttt{B} \in IS$, $\Delta(a) \leq \texttt{DeltaA} \in S$, and $\Delta(b) \leq \texttt{DeltaB} \in S$. The basic function `DeltaAdd` supplied by the module eps_ari looks like

```
global const Epsilon = 1.110224E-16;
{ Upper bound for 2**(-53)  which is }
{    Wilkinson's epsilon for        }
{    the IEEE double format         }

global const
   MinReal = 2.2250738585072013E-308;
{ Smallest positive normalized      }
{    floating-point number          }

global function DeltaAdd(
          A, B          : interval;
          DeltaA, DeltaB: real      ): real;
var u, v: real;
begin
   if { ... some  special cases ... }
      ...
   else begin     { See Theorem 1  }
      u:= Epsilon *> MaxAbs( A + B );
      v:= (DeltaA +> DeltaB)*>(1 +> Epsilon);
      DeltaAdd:= MinReal +> u +> v;
   end;
end;
```

The symbols +> and *> denote the machine operations addition and multiplication with rounding towards $+\infty$. The operator + in the expression `A + B` means interval addition. All these operators are predefined in PASCAL-XSC.

The following simple example shows, how the basic routines `DeltaAdd`, `DeltaSub`, ... can be combined to compute a worst-case error bound for the evaluation of the polynomial $p(x) = \sum_{i=0}^{n} p_i x^i$. As usual the computation is done according to Horner's scheme. It is assumed that $x \in \texttt{X} \in IS$, $\tilde{x} = x + \Delta_x$ with $|\Delta_x| \leq \Delta(x) \leq \texttt{DeltaX} \in S$, p_i, $\tilde{p}_i \in \texttt{p[i]} \in IS$, where `X`, `DeltaX`, and `p[i]` are known quantities. Of course p_i, $\tilde{p}_i \in \texttt{p[i]}$ implies $|\tilde{p}_i - p_i| \leq$ diam (p[i]), i. e. $\Delta(p_i) \leq$ diam (p[i]).

```
H:= p[n]; DeltaH:= diam( p[n] );
for i:= n-1 downto 0 do begin
   DeltaH:=
     DeltaMul(H, X, DeltaH, DeltaX);
   H:= H * X;
   DeltaH:=
     DeltaAdd(H,p[i],DeltaH,diam(p[i]));
   H:= H + p[i];
end;
PolX   := H;
AbsErr:= DeltaH;
```

Now the inequality

$$|p(x) - \tilde{p}(\tilde{x})| \leq \texttt{AbsErr}$$

holds for all $x \in \texttt{X}$ and all $\tilde{x} = x + \Delta_x$ with $|\Delta_x| \leq \texttt{DeltaX}$. Moreover $p(x) \in \texttt{PolX}$ for $x \in \texttt{X}$ and $\texttt{PolX} \ni \tilde{p}(x) := (\ldots((\tilde{p}_n \boxdot x \boxplus \tilde{p}_{n-1}) \boxdot x \boxplus \tilde{p}_{n-2}) \boxdot x \boxplus \ldots \boxplus \tilde{p}_1) \boxdot x \boxplus \tilde{p}_0$ for any set of polynomial coefficients $\tilde{p}_i \in \texttt{p[i]}, i = 0(1)n$. Examples producing numerical results are given in [5].

5. A priori Error Bound for an Accurate Table Method Realizing $e^x - 1$

To show that the methods discussed in this paper are relevant for practical purposes an error estimation for an algorithm implemented in ANSI-C which computes the function $e^x - 1$ is performed. It is assumed that the implementation of $\exp(x) - 1$ uses operations which comply with the IEEE standard [4].

Some remarks on the table-lookup algorithm: Only some parts of the algorithm are described informally. This will

be sufficient to show that an error estimation by hand is very expensive (many different cases) and error prone. The detailed description as well as the complete ANSI-C source code can be found in [5]. The complete algorithm is described in [5, 15].

In the range $x \leq \ln(1-\frac{1}{4})$ or $\ln(1+\frac{1}{4}) \leq x$ one computes the value $e^x - 1$ in three steps. First the argument is reduced to the interval $[-\ln(2)/64, \ln(2)/64]$ using the formula

$$x = (32m + j) \cdot \frac{\ln(2)}{32} + r$$

with $m \in \mathbf{Z}$, $j \in \{0, 1, \ldots, 31\}$ and $|r| \leq \ln(2)/64$. Here m, j are determined and then r is computed simulating a higher precision arithmetic (staggered arithmetic). Second, for the reduced argument r an approximation to $e^r - 1$ is computed by the polynomial approximation $e^r - 1 \approx p(r) := r + r^2 \cdot (a_0 + a_1 r + a_2 r^2 + a_3 r^3 + a_4 r^4)$.
Third, the reconstruction of the result is done in accordance to the formula

$$e^x - 1 = 2^m(2^{j/32} + 2^{j/32}(e^r - 1)) - 1.$$

The powers $2^{j/32}$, $j \in \{0, 1, 2, \ldots, 31\}$ are precomputed values and stored in a table (table method) using two double numbers for each entry.

Automatic error analysis: Applying the routines of the module `eps_ari` gives a worst case error bound for the algorithm described above. To avoid overestimations in interval calculations, i.e. to get sharp error bounds, the argument range has to be subdivided into thousands of small subintervals (which in fact can be done using an additional simple loop statement). Again, the complete program for the automatic error estimation can be found in [5].

With the abbreviation $f(x) := \exp(x) - 1$ one finds the following over all upper error bound $\varepsilon(f)$ for the relative error of the accurate table method[3]:

$$\bigwedge_{\substack{0 \neq x \in S \\ x \text{ causes no overflow}}} \left| \frac{\left(\exp(x) - 1\right) - \widetilde{f(x)}}{\exp(x) - 1} \right|$$

$$\leq \varepsilon(f) := 1.302 \cdot 10^{-16} \leq 1.18 \cdot \varepsilon^*. \qquad (4)$$

Notice, that this bound holds simultaneously for **all** floating-point arguments which do not cause an overflow error. It must be emphasized that, in general, already **one** simple floating-point operation leads to the bound $1.0 \cdot \varepsilon^*$. This demonstrates the high quality of the factor 1.18 in (4). Indeed, as already stated by Tang [15], the algorithm leads to an implementation with near-perfect accuracy. In comparison with the result (4) Tang's original analysis gives a slightly better error bound in ulps. But see Footnote 2.

[3]The approximation errors of the different polynomial approximations have been estimated rigorously using an interval tool described in [11].

6. Conclusions

The new technique can be used to get tight and reliable worst-case error bounds for floating-point computations (almost) automatically. In doing a priori error estimations a lot of error prone hand calculations can be omitted. The described implementation of the "absolute error arithmetic" using interval operations leads to bounds that are rigorous, realistic, and of high quality.
Some improvements and extensions are planned:

— Theoretical foundation and implementation of functions like `DeltaSin`, `DeltaArcSin`, `DeltaCos`, `DeltaArcCos`, `DeltaExp`, `DeltaLog`, ... corresponding to the functions sin, arcsin, cos, arccos, exp, log ... commonly supplied by higher programming languages[4].

— Theoretical foundation and realization of a worst-case error arithmetic for *relative* error bounds.

— A C++ implementation using an object oriented approach will simplify the usage of the described error arithmetic tools even more. Such an approach can also be done in PASCAL-XSC. Introducing a new data type, e.g.

```
eps_ari_type =
   record
      Enclosure : interval;
      ErrorBound: real;
   end;
```

allows the overloading of operators and functions for this type. If `A` denotes a variable of type `eps_ari_type` then the record component `A.Enclosure` represents an enclosure (floating-point interval) of $A \ni a$ and the floating-point number `A.ErrorBound` is an upper bound for $|a - \widetilde{a}|$, i. e. for $\Delta(a)$.

For quantities of the new data type the routines `DeltaAdd`, `DeltaSub`, ... can be replaced by operator calls +, −, ..., respectively. This preserves the mathematical notation of expressions when using the error bound arithmetic. Horner's Scheme example from above could be simplified to (`H` and `X` have to be of type `eps_ari_type`):

```
{ Initialization:                }
H.Enclosure := p[n];
```

[4]The numerical quality of the functions as they are supplied by the compiler or run-time library must be known (for example in the form: their accuracy is better than k units in the last place)

```
H.ErrorBound:= diam( p[n] );
{ Horner's scheme:              }
for i:= n-1 downto 0 do begin
   H:= H * X + p[i];
end;
{ Range of the polynomial       }
{   over X.Enclosure:           }
PolX:= H.Enclosure;
{ Worst-case error bound:       }
AbsErr:= H.ErrorBound;
```

The presented error arithmetic can also be used to control the numerical error of multi-precision computations [10]. If, for example, error bounds are known for the steps of an iterative process done in the field of the real numbers, then the additional numerical errors introduced by the multi-precision computations can be estimated rigorously in advance. Statements like "An arithmetic using $(k+n)$-digits with respect to base b is sufficient to get a relative accuracy better than b^{-k} " are possible. Such statements are important for reliable multi-precision calculations for example in the field of computer algebra.

The new tools are also of great advantage when constructing algorithms according to Ziv [16].

Future work will show whether the presented concept of a reliable error arithmetic will give tight results in situations as discussed in [2, 3] [5].

For the implementation of the new tools (machine) interval operations are of great advantage. At least directed rounded operations must be available. It is a great pity that such features of IEEE 754 lack support in programming languages and compilers (see the discussion in [8]). In the so called XSC-languages [9] interval operations are available via their usual mathematical operator notations. These languages also support directed rounded operations.

References

[1] Adams, E., Kulisch, U.: *Scientific Computing with Automatic Result Verification*, Mathematics in Science and Engineering, Vol. 189, Academic Press, 1993.

[2] W. E. Ferguson, T. Brightman: *Accurate and Monotone Approximations of Some Transcendental Functions*, Proceedings to the 10th Symposium on Computer Arithmetic, IEEE Computer Society Press, pp. 237–244, 1991.

[3] W. E. Ferguson: *Exact Computation of a Sum or Difference with Applications to Argument Reduction*, Proceedings to the 12th Symposium on Computer Arithmetic, IEEE Computer Society Press, pp. 216–221, 1995.

[4] American National Standards Institute / Institute of Electrical and Electronics Engineers: *A Standard for Binary Floating-Point Arithmetic.* ANSI/IEEE Std. 754-1985, New York, 1985 (reprinted in SIGPLAN **22**, 2, pp 9–25, 1987).

[5] Hofschuster, W., Krämer, W.: *Ein rechnergestützter Fehlerkalkül mit Anwendung auf ein genaues Tabellenverfahren*, Preprint 96/5 des Instituts für Wissenschaftliches Rechnen und Mathematische Modellbildung, 1996:
`ftp://iamk4515.mathematik.uni-karlsruhe.de`,
Directory: `/pub/iwrmm/preprints`

[6] Hofschuster, W. and Krämer, W. : *A Computer Oriented Approach to Get Sharp Reliable Error Bounds,* Reliable Computing, Issue 3, Volume 3, 1997.

[7] Kahan, W. M. : *The Regrettable Failure of Automated Error Analysis.* A Mini-Course prepared for the conference at MIT on Computers and Mathematics, 1989.

[8] Kahan, W. M. : *Lecture Notes on the Status of IEEE Standard 754 for Binary Floating-Point Arithmetic*, June 5, 1995.

[9] Klatte, R., Kulisch, U., Neage, M., Ratz, D., Ullrich, Ch.: *PASCAL-XSC, Language Reference with Examples*, Springer, 1992.

[10] Krämer, W.: *Multiple-Precision Computations with Result Verification*, in: *Scientific Computing with Automatic Result Verification* ed. by E. Adams and U. Kulisch, Academic Press, 1992.

[11] Krämer, W.: *Sichere und genaue Abschätzung des Approximationsfehlers bei rationalen Approximationen*, Bericht des Instituts für Angewandte Mathematik, Universität Karlsruhe, 1996:
`ftp://iamk4515.mathematik.uni-karlsruhe.de`,
Directory: `/pub/documents/reports`

[12] Tang, P.T.P.: *Table-Driven Implementation of the Exponential Function in IEEE Floating-Point Arithmetic.* ACM Trans. on Math. Software, Vol 15, No 2, 144-157, 1989.

[13] Tang, P.T.P.: *Table-Driven Implementation of the Logarithm Function in IEEE Floating-Point Arithmetic.* ACM Trans. on Math. Software, Vol 16, No 4, 378-400, 1990.

[5] Due to pessimism on the part of interval arithmetic, which is caused by insufficient use of the correlation of different parts of the expression, the computed bounds may be not tight enough. In such cases more intelligence (e. g. monotonicity considerations) has to be added to the interval expression evaluation and/or a suitable subdivision strategy must be applied.

[14] Tang, P.T.P.: *Table-Lookup Algorithms for Elementary Functions and Their Error Analysis*, Proceedings to the 10th Symposium on Computer Arithmetic, IEEE Computer Society Press, pp. 232–236, 1991.

[15] Tang, P.T.P.: *Table-Driven Implementation of the Expm1 Function in IEEE Floating-Point Arithmetic.* ACM Trans. on Math. Software, Vol 18, No 2, 211-222, 1992.

[16] Ziv, A.: *Fast Evaluation of Elementary Mathematical Functions with Correctly Rounded Last Bit.* ACM Trans. on Math. Software, Vol. 17, N0. 3, pp 410-423, 1991.