Bergische Universität
Wuppertal

Computing a priori Error Bounds for Floating-point Evaluations of Arithmetic Expressions

Frithjof Blomquist, Walter Krämer

Preprint
BUW-WRSWT 2006/1

Wissenschaftliches Rechnen/
Softwaretechnologie

wr
swt

Impressum

Herausgeber:	Prof. Dr. W. Krämer, Dr. W. Hofschuster Wissenschaftliches Rechnen/Softwaretechnologie Fachbereich C (Mathematik und Naturwissenschaften) Bergische Universität Wuppertal Gaußstr. 20 42097 Wuppertal, Germany

Internet-Zugriff

Die Berichte sind in elektronischer Form erhältlich über die World Wide Web Seiten

`http://www.math.uni-wuppertal.de/wrswt/literatur.html`

Autoren-Kontaktadressen

Frithjof Blomquist
Adlerweg 6
66436 Püttlingen, Germany

E-mail: `blomquist@math.uni-wuppertal.de`

Walter Krämer
Bergische Universität Wuppertal
Gaußstr. 20
42097 Wuppertal, Germany

E-mail: `kraemer@math.uni-wuppertal.de`

Computing a priori Error Bounds for Floating-point Evaluations of Arithmetic Expressions

Frithjof Blomquist [a], Walter Krämer [b]

[a] *Adlerweg 6, 66346 Püttlingen, Germany*

[b] *Bergische Universität Wuppertal,Gaußstr. 20, 42119 Wuppertal, Germany*

Abstract

For the implementation of mathematical functions on a computer often a large number of different expressions are used in different parts of the domain of the function under consideration. Deriving corresponding a-priori error bounds for the numerical evaluation of these expressions by hand is a very tedious and error-prone task. We discuss software tools implemented in C-XSC which can be used to automatically compute a priori error bounds for the numerical evaluation (using IEEE-754 floating-point operations) of arithmetic expressions. The tools are available as open source software and have been used with great success to find reliable and tight worst-case error bounds for mathematical function expressions. For a very simple example numerical results will be shown.

Key words:
computing a priori error bounds, open-source software, C-XSC, interval functions

MSC Subject Classifications: 65G30, 65K05.

1 Introduction

In general the computer evaluation of a function expression $f()$ leads to a defective machine value $\tilde{f}(\tilde{x})$ for IEEE machine numbers $\tilde{x} \in [a, b]$ as arguments. With the exact function value $f(\tilde{x})$ the absolute error can be written as $\tilde{f}(\tilde{x}) - f(\tilde{x}) = \varepsilon_x \cdot f(\tilde{x})$, where ε_x is the relative error depending on the argument $\tilde{x}$, the used expression term (mathematically equivalent expressions in general lead to different error bounds), and the actual rounding mode of the

Email addresses: blomquist@math.uni-wuppertal.de (Frithjof Blomquist), kraemer@math.uni-wuppertal.de (Walter Krämer).

CPU. If $|\varepsilon_x| \leq \varepsilon(f) \;\; \forall \tilde{x} \in [a,b]$ and if f is, for example, a positive, monotonic function then for all real arguments $x \in [a,b]$ the function values $f(x)$ are included by

$$\frac{\tilde{f}(a)}{1+\varepsilon(f)} \leq f(x) \leq \frac{\tilde{f}(b)}{1-\varepsilon(f)} \quad \forall x \in [a,b] \tag{1}$$

Hence, for the implementation of interval standard functions we need the relative error bounds $\varepsilon(f)$, which have been calculated e. g. in [5, 11, 4, 13, 7, 9, 10]. In [1] these $\varepsilon(f)$-values are computed automatically by reading the actual approximation term of the function under consideration. Here we use a more expensive but more flexible method, where each of the basic operations in the function term will be handled by an appropriate function call in a C-XSC program [2]. In some cases the $\varepsilon(f)$-values can be improved by choosing different function terms in different regions of the domain and by using refined error bounds for elementary operations with special operands. We demonstrate these techniques With numerical examples. For the algorithms of the functions $\sqrt{x^2-1}$ and e^{-x^2} we give some hints for getting small error bounds.

2 Calculating error bounds

2.1 Error bounds of the elementary operations

Let $S(2,53)$ denote the system of IEEE-numbers. In general the sum $\tilde{a}+\tilde{b}$ of two machine numbers $\tilde{a}$ and $\tilde{b}$ is not a machine number itself and must be rounded to $\tilde{a} \oplus \tilde{b} \in S(2,53)$. If $\varepsilon_{\tilde{a},\tilde{b}}$ is the relative error, the absolute error is given by $(\tilde{a} \oplus \tilde{b}) - (\tilde{a}+\tilde{b}) = \varepsilon_{\tilde{a},\tilde{b}} \cdot (\tilde{a}+\tilde{b})$ depending on $\tilde{a}$ and $\tilde{b}$ and on the actual rounding mode of the CPU. If we allow the rounding to any of the two neighbouring machine numbers (in this case we use the term high accuracy) then for the four elementary operations with results in the normalized range $|\varepsilon_{\tilde{a},\tilde{b}}| \leq 2^{-52} =: \varepsilon^*$,and the absolute error of our sum can be estimated in high accuracy:

$$|(\tilde{a} \oplus \tilde{b}) - (\tilde{a}+\tilde{b})| \leq \varepsilon^* \cdot |\tilde{a}+\tilde{b}| \tag{2}$$

If all error estimations are done with the rounding of the elementary operations in high accuracy, the calculated error bounds are valid independently from the actul rounding mode of the CPU.

2.2 Calculating error bounds in C-XSC

We assume real values a and $b \in I\!R$ with $a \in \boldsymbol{A}$ and $b \in \boldsymbol{B}$, where $\boldsymbol{A}, \boldsymbol{B}$ are machine intervals and interpret a as an exact result value, which has been evaluated on a computer with the machine result $\tilde{a} = a + \delta_a$. Furthermore we assume $|\delta_a| \leq \Delta(a) \; \forall a \in \boldsymbol{A}$ (the bound $Delta(a)$ is a known quantity). If we calculate the machine sum $\tilde{a} \oplus \tilde{b}$ of the defective function values $\tilde{a}$ and $\tilde{b}$ then the absolute error is defined by $\delta_+ := \tilde{a} \oplus \tilde{b} - (a+b)$. With $\Delta := \Delta(a) + \Delta(b)$

the absolute error $|\delta_+|$ is bounded by

$$|\delta_+| \leq \Delta(+) := \Delta + \varepsilon^* \cdot (\Delta + |\boldsymbol{A} \diamond \boldsymbol{B}|) \quad a \in \boldsymbol{A}, b \in \boldsymbol{B}, \tag{3}$$

where $|\boldsymbol{A} \diamond \boldsymbol{B}|$ is the maximum of the absolute values of the interval sum $\boldsymbol{A} \diamond \boldsymbol{B}$. Here $\diamond$ denotes interval addition. Appropriate bounds $\Delta(\circ), \varepsilon(\circ)$ for the absolute and relative errors of the elementary operations $\circ \in \{+, -, \cdot, /\}$ are given in [1]. Our simple example of the addition demonstrates that for calculating the error bound of an elementary operation we only need the inclusion intervals $\boldsymbol{A}, \boldsymbol{B}$ of the exact operands a and b and their absolute or relative error bounds. Hence, for any elementary operation of a given expression we need an appropriate function call with the input data $\boldsymbol{A}$and $\boldsymbol{B}$ together with the error bounds of the operands which deliver the error bound of the operation and an inclusion interval $\boldsymbol{R}$ for all exact operation results $a \circ b$. The described functions are available in C-XSC, and an error bound of a given expression can be calculated by a sequence of appropriate function calls [2]. The following program fragment shows the calculation of a relative error bound for the simple term $T_1(x) = 1 - x^2,\ \tilde{x} \in [2^{-12}, 0.658]$:

```
#include "abs_relh.hpp" //basic C-XSC routines for the computation
                        //  of worst case relative or absolute
                        //  error bounds for the elemntary
                        //  operations and functions
#include "bnd_util.hpp" //function Max_bnd_Xi() based on subdivision
#include <iostream>     //input/output
using namespace cxsc;
using namespace std;

//Compute error bound for the function term T1 over the interval x
real T1(const interval& x, const real& errx)
{  // T1(x)= 1 - x^2;
   interval x2, res;  real errx2, relerr;
   abs_mulh1(x,errx,x,errx,x2,errx2);  //x2= x^2
   rel_1mx_delh(x2,errx2,res,relerr);  //r = 1-x^2
   return relerr;                      //relative error bound
}

int main() {
   interval X(comp(0.5,-11),0.658);  //interval [2**(-12),0.658]
   real bound; //output parameter (worst case rel. error bound)
   real diam=1e-5; //diameter of subintervals
   real errx=0;    //function arguments are error free flp numbers
   Max_bnd_Xi(T1,X,errx,diam,bound); //compute the maximum error
                                     //  bound for T1() over X
                                     //  using subintervals of X
                                     //  of diameter <= 1e-5
   cout << "Rel. error bound eps(T1,X) = " << RndUp << bound << endl;
}
```

The computed relative error bound is $2.67...E-16$.

3 Improvements of the error bounds

To get error bounds as small as possible there are two essential facilities. Firstly, in special cases of the elementary operations we should try to find error bounds in different regions of the operands compared to the right hand side of (2). Secondly, for a given function expression a mathematically equivalent expression should be found to minimize the error bound.

3.1 Special cases of elementary operations

In [1] several improvements for the elementary operations have already been treated. Here for the addition $\tilde{a}+\tilde{b}$ we consider the special case $\tilde{a}=1$ and for the term $1+\tilde{x}$ with $\tilde{x} \in [\frac{-1}{4}, \frac{-1}{8}]$ the inequality (2) delivers the error bound: $|(1 \oplus \tilde{x}) - (1+\tilde{x})| \leq \varepsilon^* \cdot (1-\frac{1}{8}) = \frac{7}{8} \cdot \varepsilon^*$. But rounding the bit pattern of the exact sum $1+\tilde{x}$ to the neighbouring machine number in high accuracy we get for all $\tilde{x} \in [\frac{-1}{4}, \frac{-1}{8}]$ the smaller upper bound $|(1 \oplus \tilde{x}) - (1+\tilde{x})| \leq \frac{3}{8} \cdot \varepsilon^* < \frac{7}{8} \cdot \varepsilon^*$. For some intervals $\boldsymbol{X} \ni \tilde{x}$ Table 1 lists the improved error bounds α and the error bounds β calculated with the right hand side of (2).

$\tilde{x} \in \boldsymbol{X}$	α	β	$\tilde{x} \in \boldsymbol{X}$	α	β	$\tilde{x} \in \boldsymbol{X}$	α	β
$[-2^{54}, -2^{53})$	1	4	$(\frac{-1}{4}, \frac{-1}{8}]$	$\frac{3}{8} \cdot \varepsilon^*$	$\frac{7}{8} \cdot \varepsilon^*$	$[0, \frac{1}{2})$	ε^*	$\frac{3}{2} \cdot \varepsilon^*$
$[-2^{53}, \frac{-1}{2}]$	0	2	$(\frac{-1}{8}, -2^{-4}]$	$\frac{7}{16} \cdot \varepsilon^*$	$\frac{15}{16} \cdot \varepsilon^*$	$[\frac{1}{2}, 1)$	$\varepsilon^*/2$	$2 \cdot \varepsilon^*$
$(\frac{-1}{2}, \frac{-1}{4}]$	$\varepsilon^*/4$	$\frac{3}{4} \cdot \varepsilon^*$	$[-2^{-4}, 0]$	$\varepsilon^*/2$	ε^*	$[1, 2)$	ε^*	$3 \cdot \varepsilon^*$

Table 1: Improved error bounds α of the term $1+\tilde{x},\ x \in \boldsymbol{X}$

The improvement of the error bounds α compared to the β-values emphasizes that the right hand side of (2) delivers only a general upper bound, which can be reduced for the term $1+x$ in special intervals $\boldsymbol{X}$. The complete table and the proof of the error bounds α are given in [2].

3.2 Appropriate function terms

For the simple function $f(x) = 1 - x^2$ let us calculate an error bound for all machine values $\tilde{x} \in [2^{-12}, 1]$. First we use the two equivalent expressions $T_1(x) := 1-x^2$ and $T_2(x) := (1-x)(1+x)$. In C-XSC the relative error bounds $\varepsilon(T_1)$ and $\varepsilon(T_2)$ are calculated for different subintervals $\boldsymbol{X} \subset [2^{-12}, 1]$ with help

of the error calculus described above. Results are listed in the following table:

$\tilde{x} \in \boldsymbol{X}$	$\varepsilon(T_1)$	$\varepsilon(T_2)$	$\tilde{x} \in \boldsymbol{X}$	$\varepsilon(T_1)$	$\varepsilon(T_2)$
$[2^{-12}, 2^{-11}]$	$0.5001 \cdot \varepsilon^*$	$2.4999 \cdot \varepsilon^*$	$[2^{-1}, 0.6755]$	$1.2991 \cdot \varepsilon^*$	$1.6667 \cdot \varepsilon^*$
$[2^{-11}, 2^{-1}]$	$0.8334 \cdot \varepsilon^*$	$2.1667 \cdot \varepsilon^*$	$[0.6755, 0.9999]$	$5000 \cdot \varepsilon^*$	$1.2985 \cdot \varepsilon^*$

Table 2: Relative error bounds $\varepsilon(T_1)$ and $\varepsilon(T_2)$ for different subintervals $\boldsymbol{X}$

In the $\varepsilon(T_1)$-column the $\varepsilon(T_1)$-values are increasing with $\tilde{x}$ coming nearer to 1 because of the well known cancellation effects, whereas the $\varepsilon(T_2)$-values are decreasing. Hence, if we chose $T_1(x)$ in $\tilde{x} \in [2^{-12}, 0.6755]$ and $T_2(x)$ in $[0.6755, 1]$ then the relative error is $\varepsilon(T_1, T_2) = 1.2991 \cdot \varepsilon^*$. In the $\varepsilon(T_1)$-column with a first error free-summand 1 the error bounds become smaller with decreasing values of the second defective summand $\tilde{x}^2$ compared to 1.

However, we can still improve the relative error bound if we again use $T_1(x)$ in $[2^{-12}, 0.658]$. In the remaining interval $[0.658, 1]$ we set $\tilde{x} = 1 - d$. Then the difference $d := 1 \ominus \tilde{x} \equiv 1 - \tilde{x}$ can be calculated error-free, and $f(x)$ can be written in the form $f(\tilde{x}) = 1 - \tilde{x}^2 = 2d - d^2 =: T_3(d)$. Now again with our error calculus we find the smaller error bound $\varepsilon(T_1, T_3) = 1.2063 \cdot \varepsilon^*$. With this simple example we demonstrated that proper expressions in different subintervals are essential for a small error bound in the total interval.

3.3 Hints for getting small error bounds

- Use the improved C-XSC tools for calculating error bounds of the elementary operations described in [2], where many application examples can be found.
- Use in the domain of a given function in different subintervals the proper expression to get error bounds as small as possible.
- Use sums with an error-free first summand and a defective second summand, which should be as small as possible.
- Avoid products with defective factors. Decompose such factors in two summands, and evaluate the product with simulated higher precision [7].

4 Hints for small error bounds of additional standard functions

In Table 3 we list new enhanced standard functions, implemented in C-XSC, together with their relative error bounds, for machine arguments in the do-

mains of the respective functions.

$f(x)$	$\varepsilon(f)$	$f(x)$	$\varepsilon(f)$	$f(x)$	$\varepsilon(f)$
$\sqrt{x+1}-x$	$2.2501 \cdot \varepsilon^*$	$\sqrt{x^2+1}$		$\sqrt{x^2-1}$	$1.0004 \cdot \varepsilon^*$
$\sqrt{1-x^2}$	$1.6667 \cdot \varepsilon^*$	$\ln(\sqrt{x^2+y^2})$	$2.3242 \cdot \varepsilon^*$	$\sqrt{x^2+y^2}$	
e^{-x^2}	$2.0801 \cdot \varepsilon^*$				

Table 3: Some C-XSC functions with improved relative error bounds

In the following two subsections we give some hints for getting small error bounds for the functions $\sqrt{x^2-1}$ and e^{-x^2}.

4.1 $\boldsymbol{f(x) = \sqrt{x^2-1},\ x \geq 1}$

For $1 \leq \tilde{x} \leq 44000$ we use Newton's method starting with the machine number $\tilde{y}_0 \approx \sqrt{x^2-1}$, where in different subintervals different function expressions should be evaluated. With a sufficient good approximation $\tilde{y}_0$ Newton's method converges quadratically, and the next iterative refinement is

$$y_1 := \frac{1}{2}\left(\tilde{y}_0 + \frac{\tilde{x}^2-1}{\tilde{y}_0}\right) \equiv \tilde{y}_0 + \frac{1}{2} \cdot \frac{(\tilde{x}^2-1) - \tilde{y}_0^2}{\tilde{y}_0} \in \mathbb{IR}. \tag{4}$$

If we evaluate the first term in (4) then because of the factor 1/2 both summands in the parentheses must be of the same order, and the calculated error bound $\varepsilon(1) = 1.9996 \cdot \varepsilon^*$ cannot be optimal. However, if we use the second term in (4) then because of the quadratic convergence property the second summand must be very small compared to the error-free first summand $\tilde{y}_0$. This is the ideal condition for getting small error bounds, and we get $\varepsilon(2) = 1.0004 \cdot \varepsilon^*$. The new problem, how to evaluate the small numerator $(\tilde{x}^2-1) - \tilde{y}_0^2$ with a sufficient accuracy, is discussed in [2].

For $\tilde{x} \geq 44000$ we use the approximation $f(x) \approx x - \frac{1}{2x}$. Again the second summand is quite small compared to the error-free first summand x,so we get the small error bound $\varepsilon(3) = 1.0004 \cdot \varepsilon^*$. [2] includes a discussion of the approximation error in the two intervals $[1, 44000)$ and $[44000, \infty)$.

4.2 $\boldsymbol{f(x) = e^{-x^2},\ x \geq 0}$

The simplest algorithm $\tilde{f}(\tilde{x}) := \exp(\tilde{x} \odot \tilde{x})$ leads to an unacceptable error bound of order 10^{-13} (it should be smaller by a factor of about 1/500), so a more sophisticated algorithm is needed. Here `exp()` is a call of the C-XSC

exponential function with the error bound $\varepsilon(exp) = 2.357962556 \cdot 10^{-16} = 1.0620 \cdot \varepsilon^*$. We use the approximation:

$$e^{-x^2} \approx \begin{cases} g_1(x) := 1, & 0 \le x \le 2^{-26} \\ g_2(x) := 1 - x^2 + \frac{x^4}{2!} - \frac{x^6}{3!}, & 2^{-26} \le x \le 2^{-6} \\ g_3(u,v) := e^{-u} - v \cdot e^{-u}, & 2^{-6} \le x \le x_0 = 26.61\ldots \\ g_4(x) := 0, & x > x_0 \end{cases} \tag{5}$$

For machine values $\tilde{x} > x_0$ the function values $f(\tilde{x})$ lie in the denormalized range, so an error estimation is only needed in $0 \le \tilde{x} \le x_0$.

$g_1(\tilde{x}) \equiv 1$ is a machine number, so only the approximation error must be considered. The Taylor series of $f(x)$ is an alternating Leibniz series whose relative error can be estimated by

$$\left| \frac{e^{-x^2} - 1}{e^{-x^2}} \right| \le \frac{x^2}{1!} \cdot e^{x^2} =: r_1(x).$$

To find an upper bound of the relative error the monotonic function $r_1(x)$ must be evaluated at the boundary point $\tilde{x} = 2^{-26}$; $r_1(2^{-26}) < 1.000001 \cdot \varepsilon^*$.

In the second interval the relative approximation error can be estimated by the monotonic function $\varepsilon(app_x) := x^8 e^{x^2}/4!$, evaluated at the boundary point $\tilde{x} = 2^{-6}$. We use Horner's scheme for $g_2(x)$:

$$g_2(x) = 1 - x^2 \cdot \left(1 - \frac{x^2}{2} \cdot \left(1 - \frac{x^2}{3}\right)\right) \quad \text{, and an error estimation delivers}$$
$$\left| \frac{f(\tilde{x}) - \tilde{g}_2(\tilde{x})}{f(\tilde{x})} \right| \le \varepsilon(f_{g_2}) = 1.1051 \cdot \varepsilon^*.$$

The remaining interval is $2^{-6} \le \tilde{x} \le x_0$, with $e^{-x^2} \approx g_3$. With an auxiliary function we use the exact decomposition $\tilde{x}^2 \equiv u + v$ with the machine numbers $u := \tilde{x} \odot \tilde{x}$ and $v := \tilde{x}^2 - u$. e^{-x^2} can then be written in the form

$$\begin{aligned} e^{-x^2} &= e^{-u} \cdot e^{-v} = e^{-u} \cdot (1 - v + r_v) \\ &= e^{-u} - (v - r_v) \cdot e^{-u} = e^{-u} - h(v) \cdot e^{-u}. \end{aligned}$$

To provide overestimations by interval calculations we use a sufficiently small subinterval $\boldsymbol{X}$. Then $u \in \boldsymbol{X_2} := \boldsymbol{X} \diamond \boldsymbol{X}$ and $e^{-u} \in \mathtt{exp}(-\boldsymbol{X_2})$. With $d := \mathtt{Sup}(\boldsymbol{X_2}) - \mathtt{pred}(\mathtt{Sup}(\boldsymbol{X_2})) \ \rightsquigarrow \ v \in \boldsymbol{V} := [-d, d]$. If we define $\tilde{h}(v) := v$ then

the evaluation error

$$r_v := e^{-v} - (1-v) = \frac{v^2}{2!} - \frac{v^3}{3!} + \frac{v^4}{4!} \pm \ldots \quad \text{can be estimated by}$$
$$|r_v| \leq \frac{v^2}{2} \cdot \frac{1}{1-|v|} \leq \frac{d^2}{2} \cdot \frac{1}{1-|d|} =: R.$$

Then $h(v)$ is included by $h(v) := v - r_v \in \boldsymbol{V} + [-R, +R]$. Herewith all necessary inclusions for an error estimation are known. With $g_3(u,v) := e^{-u} - v \cdot e^{-v}$ our error calculus delivers

$$\left| \frac{f(\tilde{x}) - \tilde{g}_3(\tilde{x})}{f(\tilde{x})} \right| < \varepsilon(f) := 2.0802 \cdot \varepsilon^* \quad \forall \tilde{x} < x_0.$$

Our algorithm is fast, has a small error bound and doesn't need any additional constants. An underflow is possible in evaluating $v \cdot e^{-v}$ in $g_3(u,v)$ for $\tilde{x} \to x_0$. This can be avoided by an appropriate scaling of $g_3(u,v)$ (again see [2]).

References

[1] A. Bantle, W. Krämer, *Automatic Forward Error Analysis for Floating Point Algorithms*, Reliable Computing, Vol. 7, No. 4, pp. 321-340 (2001).

[2] F. Blomquist, A priori Fehlerabschätzungen in C-XSC für Grundoperationen, Hornerschema und Funktionen, Bergische Universität Wuppertal (2004)
`http://www.math.uni-wuppertal.de/wrswt/xsc/cxsc_new.html`

[3] K. Braune, W. Krämer, High Accuracy Standard Functions for Real and Complex Intervals. In E. Kaucher, U. Kulisch, and Ch. Ullrich, editors, Computer Arithmetic: Scientific Computation and Programming Languages, pages 81-114. Teubner, Stuttgart (1987).

[4] K. Braune, Standard Functions for Real and Complex Point and Interval Arguments with Dynamic Accuracy,. Computing Supplementum, 6:159-184 (1988).

[5] W. v. Gudenberg, Einbettung allgemeiner Rechnerarithmetik in PASCAL mittels eines Operatorkonzeptes und Implementierung der Standardfunktionen mit optimaler Genauigkeit. Dissertation, Universität Karlsruhe (1980).

[6] N. J. Higham, Accuracy and Stability of Numerical Algorithms, 2nd ed., SIAM (2002).

[7] W. Hofschuster, W. Krämer, Ein rechnergestützter Fehlerkalkül mit Anwendungen auf ein genaues Tabellenverfahren, Preprint Nr. 96/5, Institut für Wissenschaftliches Rechnen und Mathematische Modellbildung, Karlsruhe (1996).

[8] W. Hofschuster, W. Krämer, *A Computer Oriented Approach to Get Sharp Reliable Error Bounds*, Reliable Computing 3, pp. 239-248 (1997).

[9] W. Hofschuster, W. Krämer, FI_LIB, eine schnelle und portable Funktionsbibliothek für reelle Argumente und reelle Intervalle im IEEE-

double-Format, Preprint 98/7 des Instituts für Wissenschaftliches Rechnen und Mathematische Modellbildung (IWRMM) an der Universität Karlsruhe(1998).

[10] W. Hofschuster, Zur Berechnung von Funktionswerteinschließungen bei speziellen Funktionen der mathematischen Physik, Dissertation, Universität Karlsruhe (2000).

[11] W. Krämer, Inverse Standard Functions for Real and Complex Point and Interval Arguments with Dynamic Accuracy. Computing Supplementum, 6:185-212 (1988).

[12] W. Krämer *A priori Worst Case Error Bounds for Floating-Point Computations*, IEEE Transactions on Computers, Vol. 47, No. 7 (1998).

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