BIT 41(3), pp. 540–562, 2001

RIGOROUS AND PORTABLE STANDARD FUNCTIONS

SIEGFRIED M. RUMP

Inst. of Computer Science III, Technical University Hamburg-Harburg, Schwarzenbergstr. 95, 21071 Hamburg, Germany. rump@tu-harburg.de

Abstract. Today's floating point implementations of elementary transcendental functions are usually very accurate. However, with few exceptions, the actual accuracy is not known. In the present paper we describe a rigorous, accurate, fast and portable implementation of the elementary standard functions based on some existing approximate standard functions. The scheme is outlined for IEEE 754, but not difficult to adapt to other floating point formats. A Matlab implementation is available on the net. Accuracy of the proposed algorithms can be rigorously estimated. As an example we prove that the relative accuracy of the exponential function is better than 2.07eps in a slightly reduced argument range (eps denoting the relative rounding error unit). Otherwise, extensive computational tests suggest for all elementary functions and all suitable arguments an accuracy better than about 3eps.

1. A general approach for rigorous standard functions. Todays libraries for the approximation of elementary transcendental functions are very fast and the results are mostly of very high accuracy. For a good introduction and summary of state-of-the-art methods cf. [19]. The achieved accuracy does not exceed one or two ulp for almost all input arguments; however, there is no proof for that.

Today computers are more and more used for so-called computer-assisted proofs, where assumptions of mathematical theorems are verified on the computer in order to draw anticipated conclusions. Famous examples are the celebrated Kepler conjecture [9], the enclosure of the Feigenbaum constant [8], bounds for the gravitational constant [11, 17], or a computer assisted proof of the existence of eigenvalues below the essential spectrum of the Sturm-Liouville problem [7]. For such applications the use of standard functions which are "most likely" accurate but do not deliver rigorous error bounds contradicts the whole philosophy.

There are a number of libraries supporting rigorous computation of elementary functions, among them [13, 14]. That means the algorithms deliver rigorous lower and upper bounds for the correct function value. Usually, specific polynomial or rational approximations are used, possibly combined with a table look up. The approximation, truncation and computational errors of those approximation formulas are estimated individually. To the authors knowledge, all such existing libraries are developed for a specific number system, specific format or they hardly make use of built-in (approximate) standard functions. This is a real pity because todays standard function libraries are very accurate (see, for example, [23, 24]). However, accuracy estimates are *most likely* but *not rigorously proved to be correct* for all arguments.

We intend to utilize the marvelous accuracy of the built-in standard functions by a table approach. To calculate $f(x)$ for some elementary function f and floating point argument $x \in \mathbb{F}$, we approximate x by some $\widetilde{x} \in \mathbb{F}$ with few nonzero leading bits in the mantissa. The maximal error of the computed approximation by the built-in function to the true value of $f(\widetilde{x})$ is calculated in advance by some initialization procedure. This is executed *once* for an architecture and some given floating point standard function library. Then, the difference between $f(x)$ and $f(\widetilde{x})$ is estimated and bounded for every standard function individually by formulas given below. Note that $\widetilde{x}$ is chosen such that the relative error between x and $\widetilde{x}$ is always small.

This approach is simple and natural. Although it seems not to have been described in the literature, we feel that it would not be enough for a scientific paper. We justify this paper by the success to develop fast algorithms producing very sharp bounds (relative accuracy better than about 3eps) for the result of all elementary standard functions for all possible and suitable arguments. This is achieved by using special correction formulas producing high-accuracy results in double precision floating point arithmetic. Those correction formulas, once established, are not difficult to adapt to other floating point formats.

We note that the presented algorithms are fast in terms of number of operations; the Matlab implementation (see Section 11) is slow due to severe interpretation overhead.

Our new standard function library delivers *rigorous bounds* for the result for *all floating point input arguments.* The order of evaluation of the formulas is carefully chosen to diminish accumulation of rounding errors. This is crucial and is done individually for every standard function.

In the appendix we give an example of a proof for rigorous estimation of the worst case accuracy of the exponential in a slightly reduced argument range. As a result we obtain a worst case accuracy less than 2.07eps. This is proved rather than "guessed" by many test cases.

The accuracy of the bounds for the other standard functions can also be rigorously estimated, quite a tedious work. Instead of doing this we ran quite extensive computational tests including billions of samples, especially focussing on critical areas. The maximum relative error of the bounds with respect to the midpoint in all those tests for all input arguments and all functions were about 3eps. This, of course, does not prove that this is the actual maximum relative accuracy for all floating point arguments. However, the maximum 2eps obtained for the exponential by our extensive test set is not too far from the rigorously estimated upper bound 2.07eps. We think that there is evidence in the formulas for the bounds of the other functions that the true worst case accuracy for *all* suitable floating point input is not too far from the maximum obtained for the extensive test sets, which is 3.01eps. Note, however, even if there is a case with 4eps accuracy of the bounds, say, then the accuracy of the bounds is less than expected, but nevertheless the bounds are always *rigorous.*

After introducing some notation and the discussion of the initialization procedure, we continue with the individual elementary functions. We begin with more detailed descriptions; later on we give hints for an implementation along the previously given lines. As mentioned before, the (pure) Matlab source code is available, see Section 11.

2. Notation.

We describe our approach for IEEE 754 arithmetic standard [12] and double precision format (1 sign bit, 11 exponent bits, 52 mantissa bits plus implicit one). However, we stress that along the following lines it is not difficult to adapt our approach to other formats (such as extended), or other number systems (such as decimal or hexadecimal).

Denote the set of IEEE 754 floating point numbers (including gradual underflow and $\pm\infty$) by $\mathbb{F}$. We use three different rounding modes

$\Box$ rounding to nearest,
∇ rounding downwards (towards $-\infty$), and
Δ rounding upwards (towards $+\infty$).

Throughout this paper we use the following notation:

(1) Operations captured within $\Box(\cdot), \nabla(\cdot), \Delta(\cdot)$ shall *all* be performed with the depicted rounding mode to nearest, downwards or upwards, respectively.

All results produced by our standard functions will be correct including possible underflow and overflow. Operations not preceeded by a rounding symbol depict the true (real) operations with result in $\mathbb{R}$. For example, let $a, b, c \in \mathbb{F}$. Then

$$d_1 = \nabla(a * b - c)$$
$$d_2 = \Delta(a * b - c)$$

produces $d_1, d_2 \in \mathbb{F}$ such that the true result $d = a * b - c \in \mathbb{R}$ satisfies $d_1 \le d \le d_2$. Note that this need not to be true when replacing $a * b - c$ by $c - a * b$.

We want to stress that availability of rounding modes is a convenient way to formulate the following results, but they are by no means mandatory. If only rounding to nearest is available, we may use

$$c := \Box(a \circ b) \text{ for } \circ \in \{+, -, *, /\} \quad \Rightarrow \quad c - \frac{1}{2}\text{eps}|c| - \eta \le a \circ b \le c + \frac{1}{2}\text{eps}|c| + \eta, \tag{2}$$

where eps denotes the relative rounding error unit and η the smallest nonnegative (unnormalized) floating point number. For $\circ \in \{+,-\}$, η may be omitted in (2). We also use symmetry of $\mathbb{F}$, i.e. $x \in \mathbb{F} \Rightarrow -x \in \mathbb{F}$. The predecessor and successor of a finite floating point number is defined by

$$\mathrm{pred}(x) := \max\{\widetilde{x} \in \mathbb{F} : \widetilde{x} < x\} \qquad \text{and} \qquad \mathrm{succ}(x) := \min\{\widetilde{x} \in \mathbb{F} : \widetilde{x} > x\}.$$

For the notation of the set of test points T_{f} for the standard function f see end of Section 4.

3. Long real reference. For suitable sets $R_{\mathrm{f}} \subseteq \mathbb{F}$ of reference points, depending on the elementary function f, we need high accuracy and rigorous error bounds for the values of f. This is achieved by a long precision interval arithmetic. Fortunately, those error bounds have to be computed only once for every architecture, so any reasonably fast approach suffices.

The implementation of a long precision arithmetic is standard, and a number of packages are available, among them [5, 6, 1]. An augmentation by rigorous error bounds is also standard. We choose to write our own package in (pure) Matlab for easy portability of the routines.

Our reference sets R_{f} are as follows:

$$\begin{aligned}
R_{\mathrm{exp}} &= \{\pm(0,1,2,\ldots,2^{14}-1)\cdot 2^{-14}\} \cup \{(0,1,2,\ldots,2^{-14}-1)\cdot 2^{e-14} \quad \text{for } 1 \le e \le 3\},\\
R_{\mathrm{log}} &= \{(2^{13},2^{13}+1,\ldots,2^{14}-1)\cdot 2^{e-14} \text{ for } 0 \le e \le 1\},\\
R_{\mathrm{f}} &= \{(0,1,2,\ldots,[2^{14}\pi/4])\cdot 2^{-14}\} \quad \text{for } f \in \{\sin,\cos,\tan\},\\
R_{\mathrm{atan}} &= \{(0,1,2,\ldots,2^{15}-1)\cdot 2^{-13}\},\\
R_{\mathrm{sinh}} &= \{(0,1,2,\ldots,2^{14}-1)\cdot 2^{e-14} \quad \text{for } 0 \le e \le 3\}, \quad \text{and}\\
R_{\mathrm{tanh}} &= \{(0,1,2,\ldots,2^{14}-1)\cdot 2^{-14}\}.
\end{aligned}$$

Those reference points suffice to achieve high accuracy for all elementary functions and all arguments. To compute the long precision error bounds, we usually use a Taylor series approach. For the logarithm instead, we use a double precision approximation $\widetilde{y} = \Box(\log(x))$ and one long precision Newton correction $y = \widetilde{y} + (x \cdot e^{-\widetilde{y}} - 1)$ plus error term. In the same way we get bounds for atan(x) by $\widetilde{y} = \Box(\mathrm{atan}(x))$ and $y = \widetilde{y} + (x - \tan(\widetilde{y}))/(1+x^2)$ plus some error term. The functions exp, sin, cos, tan, sinh and tanh are conveniently calculated in one routine. Long precision error bounds are calculated to about 80 bits accuracy.

Final output of the initialization procedure are global constants ε_{f} for $f \in \{\exp,\log,\sin,\cos,\tan,\mathrm{atan},$ $\sinh,\tanh\}$ which are defined as follows. Denote

$\widetilde{y} := \Box(f(x))$	double precision value calculated in rounding to nearest,
$\underline{F}(x), \overline{F}(x)$	lower and upper bounds for $f(x)$ in long precision.

Therefore $\underline{F}(x) \le f(x) \le \overline{F}(x)$ for all reference points $x \in R_{\mathrm{f}}$. Note that $\widetilde{y}$ is the approximation to $f(x)$ computed by some built-in routine. Then

$$\varepsilon_{\mathrm{f}} := \max\Big(\max_{x \in R_{\mathrm{f}}}\{(\widetilde{y} - \underline{F}(x))/|\widetilde{y}|\}, \quad \max_{x \in R_{\mathrm{f}}}\{(\overline{F}(x) - \widetilde{y})/|\widetilde{y}|\} \Big). \tag{3}$$

This implies for all $f \in \{\exp,\log,\sin,\cos,\tan,\mathrm{atan},\sinh,\tanh\}$ and all reference points $x \in R_{\mathrm{f}}$,

$$\widetilde{y} - \varepsilon_f \cdot |\widetilde{y}| \le f(x) \le \widetilde{y} + \varepsilon_f \cdot |\widetilde{y}|,$$

using exact arithmetic. Careful application of (1) yields

$$\nabla(\widetilde{y} + (-\varepsilon_{\mathrm{f}}) \cdot |\widetilde{y}|) \le f(x) \le \Delta(\widetilde{y} + \varepsilon_{\mathrm{f}} \cdot |\widetilde{y}|) \tag{4}$$

for all f and all $x \in R_f$. Note that the lower and upper bounds in (4) are computed in double precision. For a Pentium I MMX 300Mhz PC, Matlab 5.3, the computed quantities are (rounded upwards)

	exp	log	sin	cos	tan	atan	sinh	tanh
ε_{f}	1.26*e*-16	1.42*e*-16	1.17*e*-16	8.89*e*-17	1.25*e*-16	1.38*e*-16	2.39*e*-16	2.87*e*-16

TABLE 3.1. *Global error constants with respect to reference points*

Besides the error bounds ε_{f}, some auxiliary constants like $\log 2$ etc. are computed as well. This will be mentioned in the description of the individual standard function. The total time to compute all constants is about 30 minutes on the above mentioned PC for the pure Matlab long precision code, severely suffering from interpretation overhead. Not much effort was spent to speed up computation because it is performed only once.

We mention that the choosen sets of reference points compromise between computing time for the (long precision) initialization procedure, the complexity of the correction formula, and the final accuracy of the elementary function. A larger reference set needs (once) more computing time for the initialization procedure but allows lower order approximation formulas, thus improving computing time.

4. Rigorous standard functions - test sets. In the following we describe algorithms to compute rigorous and sharp error bounds for all suitable floating point arguments for the following functions: exponential, logarithm, trigonometric functions and their inverses, hyperbolic functions and their inverses. We stress again that the purpose of the billions of test cases is to gain confidence in the mentioned worst case of the relative error of 3eps. This is, of course, no proof (and not meant as such). We *prove* worst case 2.07eps for the exponential. Beside that, the computed results are *rigorously correct* for *all* possible arguments.

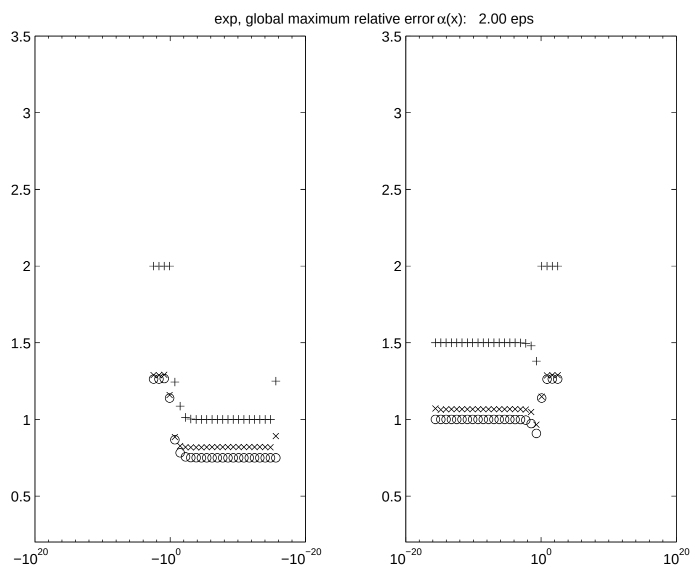
For computed lower and upper bounds $\underline{y}(x), \overline{y}(x)$ by our standard functions for $f(x)$, respectively, we measure

$$\alpha(x) = \frac{\overline{y}(x) - \underline{y}(x)}{|\underline{y}(x)| + |\overline{y}(x)|} \cdot \mathrm{eps}^{-1} \tag{5}$$

for x out of some test set T_{f}. This implies for intervals $Y := [\underline{y}, \overline{y}]$ not containing zero that the relative error of any point out of Y with respect to the midpoint of Y does not exceed $\alpha(x) \cdot \mathrm{eps}$. For intervals containing zero this is also a widely used measure. The test sets T_{f} are chosen carefully to cover a large set of arguments and to cover critical arguments.

We describe the test sets T_{f} by vectors of decimal exponents. For example, $V = 30 : 30 : 300$ denotes the vector $(30, 60, \ldots, 270, 300)$ of length 10 (Matlab notation) and indicates the following test set. For $i = 1 \ldots 9$, denote by $w^{(i)}$ a vector of 10^6 random numbers uniformly distributed within $[V_i, V_{i+1}]$. Then the test set comprises of the $9 \cdot 10^6$ numbers $10^{w_\nu^{(i)}}$ for $1 \le i \le 9, 1 \le \nu \le 10^6$. Furthermore, we use short notation $\pm V$ to indicate $V_1 \cup (-V_2)$ where $V_1 = V$, $V_2 = V$. Note that random numbers in V and $-V$ are generated independently. Finally, we use Matlab notation linspace(a, b, k) to indicate the vector $a, a + \Delta, \ldots, a + (k-1)\Delta$ for $\Delta = (b-a)/(k-1)$.

For example, $T_{\exp} = \pm\mathrm{linspace}(-16, \log_{10} 700, 25)$ represents a vector of some 48 Million test points x with $10^{-16} \le |x| \le 700$. The accuracy graph for the exponential is shown in Graph 4.1, where circles depict the median, crosses the average and plus signs the maximum of $\alpha(x)$ as defined in (5), each over the 10^6 samples in the intervals $[V_i, V_{i+1}]$.



Graph 4.1. *Accuracy exponential*

5. Exponential function. In the initialization procedure we calculate vectors of floating point numbers $E, \underline{E}, \overline{E}$ such that

$$E_i + \underline{E_i} \le \exp(i) \le E_i + \overline{E_i} \quad \text{for} \quad -744 \le i \le 709, \tag{6}$$

and $|\underline{E_i}| \sim |\overline{E_i}| \sim 10^{-16} E_i$. This simulates a "doubled" double precision. For $X \le -745$ or $X \ge 710$, $\exp(X)$ is outside the floating point range, and this case is handled separately.

Let $X \in \mathbb{F}, -745 < X < 709$ be given. For the splitting $X = X_{\text{int}} + x, X_{\text{int}} = \operatorname{sign}(X) \cdot \lfloor |X| \rfloor \in \mathbb{Z}, -1 < x < 1$, define

$$\begin{aligned} \widetilde{x} &= 2^{-14} \cdot \lfloor 2^{14} x \rfloor \\ d &= x - \widetilde{x}. \end{aligned}$$

All those operations are exact without rounding error (for the first, for example, one may use $[f, e] = \log2(x)$ in Matlab). The definition implies that $\widetilde{x}$ has at most 14 nonzero leading bits in its binary representation, and therefore $\widetilde{x} \in R_{\text{exp}}$ (the other elements $\widetilde{x} \in R_{\text{exp}}$ are needed for the hyperbolic sine). Furthermore, $0 \le d < 2^{-14}$. Thus

$$\exp(x) = \exp(\widetilde{x}) \cdot \exp(d) \qquad \text{with} \qquad \sum_{i=0}^{3} \frac{d^i}{i!} \le \exp(d) \le \sum_{i=0}^{3} \frac{d^i}{i!} + \exp(d) \cdot \frac{d^4}{4!}, \tag{7}$$

and

$$\exp(d) \cdot \frac{d^4}{4!} \le 0.2501 d \cdot \frac{d^3}{3!} < 5.8 \cdot 10^{-19}.$$

Thus (7) yields an approximation of good relative accuracy. Define

$$\begin{aligned} \widetilde{y} &= \square(\exp(\widetilde{x})) \\ \underline{c} &= \nabla(((\widetilde{y} d/3 + \widetilde{y}) d/2 + \widetilde{y}) d) \\ \underline{y_1} &= \nabla((\underline{c} + (-\underline{c})\varepsilon_{\text{exp}}) + (-\widetilde{y})\varepsilon_{\text{exp}}) \\ \underline{y} &= \nabla(\widetilde{y} \cdot E(X_{\text{int}}) + ((\widetilde{y} + \underline{y_1}) \cdot \underline{E}(X_{\text{int}}) + \underline{y_1} \cdot E(X_{\text{int}}))) \end{aligned} \tag{8}$$

Since $\widetilde{y}$ and d are nonnegative we obtain

$$\underline{c} + \widetilde{y} \le \widetilde{y} \cdot \sum_{i=1}^{3} \frac{d^i}{i!} + \widetilde{y} \le \widetilde{y} \cdot \exp(d).$$

Remember that floating point operations are used in a mathematical expression iff the latter is preceeded by one of the three rounding symbols. All operations in the previous expression including the exponential are therefore the exact ones. By (3), $\nabla((-c)\varepsilon_{\exp}) \le -c\varepsilon_{\exp}$ and $\nabla((-\widetilde{y})\varepsilon_{\exp}) \le -\widetilde{y}\varepsilon_{\exp}$ we obtain

$$\underline{y_1} + \widetilde{y} \le (c + \widetilde{y})(1 - \varepsilon_{\exp}) \le \widetilde{y}(1 - \varepsilon_{\exp})\exp(d) \le \exp(\widetilde{x} + d),$$

and by (6),

$$\underline{y} \le (E(X_{\text{int}}) + \underline{E}(X_{\text{int}}))(\underline{y_1} + \widetilde{y}) \le \exp(X_{\text{int}}) \cdot \exp(\widetilde{x} + d) = \exp(X).$$

Note that the order of execution, forced by the parantheses in (8), is such that roundoff errors are small. We prove that in the appendix. Using the same $\widetilde{y}$ as above and setting

$$\begin{aligned} \overline{c} &= \Delta((((1 + 0.2501d)\widetilde{y} \cdot d/3 + \widetilde{y})d/2 + \widetilde{y})d) \\ \overline{y_1} &= \Delta((\overline{c} + \overline{c} \cdot \varepsilon_{\exp}) + \widetilde{y} \cdot \varepsilon_{\exp}) \\ \overline{y} &= \Delta(\widetilde{y} \cdot E(X_{\text{int}}) + ((\widetilde{y} + \overline{y_1}) \cdot \overline{E}(X_{\text{int}}) + \overline{y_1} \cdot E(X_{\text{int}}))) \end{aligned} \tag{9}$$

we conclude in the same way as before

$$\underline{y} \le \exp(X) \le \overline{y}.$$

Note that $\underline{y}, \overline{y}$ are computed in double precision. Due to the correct rounding modes the above approach is also suitable for arguments near -744 and 709, where $\exp(X)$ is near under- and overflow, respectively. Of course, the value of $\alpha(X)$ increases for $\underline{y}, \overline{y}$ in the gradual underflow range, that is for $X \lesssim -708$.

The test set for the exponential function consists of $\pm V$ for $V = \text{linspace}(-16, \log_{10}(700), 25)$, i.e. 48 million test points (see Section 4). For arguments x with $|x|$ below 10^{-16}, it is $\exp(x) \in [\text{pred}(1), 1]$ or $\exp(x) \in [1, \text{succ}(1)]$, respectively, and this is computed by (8) and (9). The accuracy graph has already been displayed, as an example, in Section 4.

The maximum error over all measured arguments is below 2eps. This accuracy is achieved by storing $\exp(X_{\text{int}})$ in two parts (see (6)). The median, average and maximum error $\alpha(x)$ over the test intervals is displayed in Table 5.1.

	median $\alpha(x)$	average $\alpha(x)$	maximum $\alpha(x)$
$-700 \le x \le -10^{-16}$	0.75	0.90	2.00
$10^{-16} \le x \le 700$	1.00	1.09	2.00

TABLE 5.1. *Median, average and worst accuracy of exponential*

A proof of the rigorous upper bound 2.07eps for the relative accuracy of the exponential is given in the appendix.

6. Logarithm.

With suitable case distinctions we may assume $0 < x \in \mathbb{F}$. Define

$$x = f \cdot 2^e \text{ and } \left\{ \begin{array}{l} \widetilde{x} = 2^{-13} \cdot \langle 2^{13} f \rangle \\ \widetilde{x} = 2^{-14} \cdot \langle 2^{14} f \rangle \end{array} \right\} \text{ with } \left\{ \begin{array}{ll} 1 \le f < 2 & \text{for } x \in [1, 2] \\ 0.5 \le f < 1 & \text{otherwise} \end{array} \right. ,$$

where $\langle \cdot \rangle$ denotes rounding to the nearest integer. This splitting is to ensure numerical stability for x being slightly larger than 1. It is $\widetilde{x} \in R_{\log}$ and

$$\log x = e \log 2 + \log \widetilde{x} + \log(1 + (f - \widetilde{x})/\widetilde{x}). \tag{10}$$

We store the value log 2 in "doubled" double precision by constants $\varphi, \underline{\varphi}, \overline{\varphi} \in \mathbb{F}$ such that $\varphi + \underline{\varphi} < \log 2 < \varphi + \overline{\varphi}$ and $y \approx \log 2$, $\underline{\varphi}, \overline{\varphi} \approx 2^{-53} y$. For $f \ge \widetilde{x}$ and $d := \nabla((f - \widetilde{x})/\widetilde{x})$ follows $d \ge 0$ and

$$y_1 = \nabla(((((-d)/4 + 1/3)d - 0.5)d + 1)d) \le \log(1 + d). \tag{11}$$

The rounding is correct because $d \geq 0$. For $\widetilde{y} = \square(\log(\widetilde{x}))$ we obtain by (4) and (10),

$$\nabla(e\varphi + (\widetilde{y} + (y_1 + ((-\varepsilon_{\log}) \cdot |\widetilde{y}| + e \cdot \underline{\varphi})))) \leq \log x. \tag{12}$$

For $f < \widetilde{x}$ we set $d = \Delta((\widetilde{x} - f)/\widetilde{x})$ and

$$y_1 := -\Delta((((d/4 + 1/3)d + 0.5)d + 1)d).$$

Then $y_1 \leq \log(1 - d)$ and we use again (12). The upper bound is computed similarly using

$$\log(1 + d) \leq \Delta(((((d/5 - 0.25)d + 1/3)d - 0.5)d + 1)d) =: y_2. \tag{13}$$

The test set for the logarithm consists of $V_1 = [-300{:}30{:}-30\ \ -25{:}5{:}-10 -9{:}-1]$ and $V_2 = [1{:}9\ \ 10{:}5{:}25\ \ 30 : 30 : 300]$, covering small and large arguments. For arguments close to 1 we use in addition $V_3 = \text{linspace}(-1, 1, 21)$. The test set is then $T_{\log} = V_1 \cup V_2 \cup V_3$. The average and worst accuracy $\alpha(x)$ over the test intervals is displayed in Table 6.1.

	median $\alpha(x)$	average $\alpha(x)$	maximum $\alpha(x)$
$10^{-300} \leq x \leq 10^{-1}$	0.37	0.38	1.00
$10^{-1} \leq x \leq 10$	0.60	0.67	2.48
$10 \leq x \leq 10^{300}$	0.37	0.38	1.00

TABLE 6.1. *Median, average and worst accuracy of logarithm*

Finally we mention that using $\log_{10}(x) = \log(x) \cdot (1/\log(10))$ we achieve a maximum error $\alpha(x)$ of $3 \cdot \text{eps}$ over the entire floating point range. Constants $\underline{\varphi}, \overline{\varphi} \in \mathbb{F}$ with $\underline{\varphi} < 1/\log(10) < \overline{\varphi}$ are computed in the initialization procedure.

7. Trigonometric functions.

One of the main problems for accurate bounds of the trigonometric functions is the range reduction. The obvious formula $y = x - 2k\pi$ produces large cancellation errors such that for $|x| \gtrsim 10^{16}$ no correct digit can be expected. Only the backward error is of the order eps.

7.1. Range reduction.

To achieve small forward errors we have to solve the problem of argument reduction for large arguments. This is done as follows.[1] For $x \in \mathbb{F}$ let $x = k \cdot \pi/2 + y$ for $k \in \mathbb{Z}$ and $|y| < \pi/4$. With suitable case distinctions it suffices to know bounds for $y \in \mathbb{R}$ to solve the range reduction problem. By $2/\pi \cdot x = k + \widetilde{y}$, $\widetilde{y} = 2/\pi \cdot y$ is the fractional part of $2/\pi \cdot x$. In addition, we need only the value k mod 4. Let

$$x = \pm 1.m_1 m_2 \ldots m_{52} \cdot 2^e = \pm \sum_{i=0}^{52} m_i 2^{e-i} \quad \text{where } m_0 := 1,$$

and let

$$2/\pi = 0.p_1 p_2 \ldots = \sum_{i=1}^{\infty} p_i 2^{-i}, \quad p_i \in \{0, 1\}$$

denote the binary representation of $2/\pi$. We replace $2/\pi$ by $\gamma := \sum_{i=\alpha}^{\beta} p_i 2^{-i}$, where $\alpha := \max(1, e - 53)$ and $\beta := \min(1, e + 76)$. Then

$$c := \left(\sum_{i=1}^{\alpha-1} p_i 2^{-i} \right) \cdot x = q \cdot 2^{e-52-\alpha+1} \text{ for some } q \in \mathbb{Z},$$

[1]The author was told by J.-M. Muller that a similar method was developed by M. Payne and R. Hanek [20], see also [19]. Apparently, an (approximate) accurate range reduction is included in Matlab V 5.3f [18].

and $e - \alpha - 51 \geq 2$ implies $c \in \mathbb{Z}$ and $c \equiv 0 \bmod 4$. That means replacing $2/\pi$ by γ retains the information $k \bmod 4$. Furthermore,

$$d := \left(\sum_{i=\beta+1}^{\infty} p_i 2^{-i} \right) \cdot x < 2^{-\beta} \cdot 2^{e+1} \leq 2^{-75}.$$

Hence, when replacing $2/\pi$ by γ, the relative error in the fractional part $\widetilde{y}$ is less than 2^{-75}.

For the multiplication $\gamma \cdot x$ we split the mantissa of x into a 26-bit and 27-bit part, and the mantissa of γ into five 26-bit parts. Then the product $\gamma \cdot x$ requires some 10 floating-point multiplications, which are added in increasing order of magnitude. The product $\pi/2 \cdot \widetilde{y}$ is computed similarly.

Performing above computations with suitable roundings one obtains $k \bmod 4$ and very accurate bounds for y. The necessary constants are precomoputed once, especially the binary expansion of $2/\pi$ with as many bits as needed corresponding to the exponent range.

Summarizing, the range reduction produces sharp bounds for x^*, $0 \leq x^* < \pi/4$ with $\sin x, \cos x \in \{\pm \sin x^*, \pm \cos x^*\}$. In case x is near a root of $\sin x$, or $\cos x$, the transformation is performed in such way that $\sin x, \cos x \in \{\pm \sin x^*\}$ in order to retain high accuracy.

7.2. Sine and Cosine. For $0 \leq x < \pi/4$ we set $\widetilde{x} = 2^{-14} \cdot \lfloor 2^{14} x \rfloor$ and $d := x - \widetilde{x}$. All those operations are performed without rounding error. Then $\widetilde{x} \in R_{\text{sin}}$ and $\widetilde{x} \in R_{\text{cos}}$, and $0 \leq d < 2^{-14}$. For

$$\widetilde{s} = \square(\sin(\widetilde{x})) \text{ and } \widetilde{c} = \square(\cos(\widetilde{x}))$$

and using (3), $\sin(d) \geq d(1 - d^2/6)$ and $\cos(d) \geq 1 - d^2/2$, we have

$$\begin{aligned} \sin(x) = \sin(\widetilde{x} + d) &\geq \widetilde{s}(1 - \varepsilon_{\text{sin}})(1 - d^2/2) + \widetilde{c}(1 - \varepsilon_{\text{cos}})d(1 - d^2/6) \\ &= \widetilde{s} + ((\widetilde{s}(-d^2/2) + \widetilde{s}(\varepsilon_{\text{sin}}(d^2/2 - 1))) + \widetilde{c}(1 - \varepsilon_{\text{cos}})(d - d^3/6)). \end{aligned}$$

The lower bound for $\sin(x)$ is obtained by computing this expression with suitable setting of rounding modes. The upper bound for $\sin(x)$ is computed along the same lines using $\sin(d) \leq d(1 + (d^2/20 - 1)d^2/6)$ and $\cos(d) \leq 1 + (d^2/12 - 1)d^2/2$.

The cosine is treated much the same way. The point is that the final result is expressed as the sum of the majorizing terms $\widetilde{s}, \widetilde{c}$ plus certain error terms, respectively.

The formulas above imply 1 ulp accuracy for $|x| < 10^{-16}$. Otherwise, we generate test samples through the entire exponent range, that is $V_1 = -16{:}10$, and $V_2 = [10{:}5{:}25 \quad 30{:}30{:}300]$, and $T_{\text{sin}} = T_{\text{cos}} = (\pm V_1) \cup (\pm V_2)$.

In addition we tested $x = \square(k \cdot \pi/2)$ for $1 \leq k \leq 2 \cdot 10^9$, and, as above, in all cases the accuracy was better than $3 \cdot \text{eps}$. Median, average and worst case accuracy for sine and cosine over the test set is displayed in Table 7.1. The data indicates that the accuracy for generic argument is much better than 3eps.

	median/average/worst accuracy $\alpha(x)$	
	sin	cos
$-10^{300} \leq x \leq -10^{10}$	0.99/1.00/2.50	0.99/1.00/2.49
$-10^{10} \leq x \leq -10^{-16}$	1.15/1.19/2.50	0.75/0.82/2.49
$10^{-16} \leq x \leq 10^{10}$	1.15/1.19/2.50	0.75/0.82/2.50
$10^{10} \leq x \leq 10^{300}$	0.99/1.00/2.49	0.99/1.00/2.50

TABLE 7.1. *Median, average and worst accuracy of sine and cosine*

7.3. Tangent and cotangent. Using the argument reduction described in the Section 7.1 we calculate for given $x \in \mathbb{F}$ sharp bounds for some x^* such that $\tan x, \cot x \in \{\pm \tan x^*, \pm 1/\tan x^*\}$, where $0 \leq x^* <$

$\pi/4$. For $\widetilde{x} = 2^{-14} \cdot \lfloor 2^{14}x^* \rfloor$ and $d = x^* - \widetilde{x}$, all computed without error, it is $\widetilde{x} \in R_{\tan}$ and $0 \le d < 2^{-14}$. Then

$$\tan x^* = \tan(\widetilde{x} + d) = \frac{\tan\widetilde{x} + \tan d}{1 - \tan\widetilde{x}\tan d} = \tan\widetilde{x} + \tan d \frac{1 + \tan^2\widetilde{x}}{1 - \tan\widetilde{x}\tan d}.$$

Furthermore, $d + \frac{1}{3}d^3 \le \tan(d) \le d + \frac{1}{3}d^3 + \zeta$ where $0 \le \zeta \le \frac{\tan^{(IV)}(d)}{4!}d^4 \le 9.3 \cdot 10^{-18} \cdot d$. Putting things together and using (3) produces an accuracy better than $2.5 \cdot \mathrm{eps}$ over all test cases in the entire floating point range. For the same test set as for $\sin, \cos$, we obtained the results given in Table 7.2.

	median/average/worst accuracy $\alpha(x)$	
	tan	cot
$-10^{300} \le x \le -10^{10}$	1.49/1.52/2.98	1.49/1.52/2.90
$-10^{10} \le x \le -10^{-16}$	0.85/0.92/3.01	1.09/1.13/2.92
$10^{-16} \le x \le 10^{10}$	0.85/0.92/2.96	1.09/1.13/2.90
$10^{10} \le x \le 10^{300}$	1.49/1.52/2.84	1.49/1.52/2.92

TABLE 7.2. *Median, average and worst accuracy of tangent and cotangent*

7.4. Secans and Cosecans. For secans and cosecans it suffices to use $\sec(x) = 1/\cos(x)$ and $\csc(x) = 1/\sin(x)$ to achieve high accuracy. For the same test set as for sin and cosine we obtain accuracy results shon in Table 7.3.

	median/average/worst accuracy $\alpha(x)$	
	sec	csc
$-10^{300} \le x \le -10^{10}$	1.43/1.40/2.87	1.43/1.40/2.88
$-10^{10} \le x \le -10^{-16}$	1.00/1.21/2.81	1.54/1.56/2.87
$10^{-16} \le x \le 10^{10}$	1.00/1.21/2.86	1.54/1.56/2.86
$10^{10} \le x \le 10^{300}$	1.43/1.40/2.88	1.43/1.40/2.86

TABLE 7.3. *Median, average and worst accuracy of secans and cosecans*

7.5. Inverse trigonometric functions. The inverse trigonometric functions are essentially based on the inverse tangent. For negative $x \in \mathbb{F}$, we use $\mathrm{atan}(-x) = -\mathrm{atan}(x)$, and for $x \ge 4$,

$$\mathrm{atan}(x) = \pi/2 - \mathrm{atan}(2/(x - 1/x))/2. \tag{14}$$

Suppose $x \in \mathbb{F}, 0 \le x < 4$, and set

$$\widetilde{x} = 2^{-13} \cdot \lfloor 2^{13}x \rfloor, \quad d = x - \widetilde{x}.$$

Then $\widetilde{x} \le x$ and $2^{13}\widetilde{x} \in \mathbb{N}$, hence $0 \le x < 4$ implies $\widetilde{x} \in R_{\mathrm{atan}}$. Furthermore, $0 \le d < 2^{-13}$. For $\widetilde{y} := \mathrm{atan}(\widetilde{x})$ define δ such that $\mathrm{atan}(x) = \widetilde{y} + \delta$. Then

$$x = \widetilde{x} + d = \tan(\widetilde{y} + \delta) = (\tan\widetilde{y} + \tan\delta)/(1 - \tan\widetilde{y}\tan\delta),$$

and a calculation yields

$$\delta = \mathrm{atan}(E) \quad \text{with} \quad E = \frac{d}{1 + x\widetilde{x}}. \tag{15}$$

It is $E^5/5 < 5 \cdot 10^{-17} \cdot d$. Hence,

$$E - E^3/3 \le \mathrm{atan}(E) \le E - E^3/3 + E^5/5$$

is of high relative accuracy, and combined with (15), (14) and (4) using suitable rounding modes it yields sharp bounds over the entire floating point range. The correction formula (15) for $\mathrm{atan}(\widetilde{x})$ establishes the good quality of the result.

The arctangent is accurate enough such that $\mathrm{acot}(x) = \mathrm{atan}(1/x)$ is sufficiently accurate over the entire floating point range.

We use the test set $T_{\mathrm{atan}} = (\pm V_1) \cup (\pm V_2)$ for $V_1 = -16{:}10$ and $V_2 = [10{:}5{:}25\ \ 30{:}30{:}300]$, and $T_{\mathrm{acot}} = T_{\mathrm{atan}}$.

The median, average and worst accuracy for the inverse tangent and cotangent are shown in Table 7.2.

	median/average/worst accuracy $\alpha(x)$	
	atan	acot
$-10^{300} \le x \le -10^{10}$	0.64/0.64/0.64	0.71/0.72/1.00
$-10^{10} \le x \le -10^{-16}$	0.64/0.58/1.67	0.64/0.78/2.00
$10^{-16} \le x \le 10^{10}$	0.64/0.58/1.67	0.64/0.78/2.00
$10^{10} \le x \le 10^{300}$	0.64/0.64/0.64	0.71/0.72/1.00

Table 7.4. *Median, average and worst accuracy atan and acot*

For the arcsine we use

$$\mathrm{asin}(x) = \mathrm{atan}(x/\sqrt{1-x^2}) \quad \text{for } 0 \le x < 0.75, \tag{16}$$

and for $0.75 \le x \le 1$,

$$\mathrm{asin}(x) = (\pi/2 - \mathrm{atan}(\sqrt{e + e \cdot x}/x) \quad \text{where } e = 1 - x. \tag{17}$$

For lower and upper bounds we use suitable rounding modes and double precision floating point bounds for $\pi/2$, the latter being computed in the initialization procedure. For negative arguments, $\mathrm{asin}(-x) = -\mathrm{asin}(x)$ is used.

For the arccosine, formulas (16) and (17) are adapted to $\mathrm{acos}(x) = \pi/2 - \mathrm{asin}(x)$. For the test sets $T_{\mathrm{asin}} = T_{\mathrm{acos}} = \pm\mathrm{linspace}(-16, 0, 25)$ the median, average and worst accuracy of the inverse sine and cosine are listed in Table 7.5.

	median/average/worst accuracy $\alpha(x)$	
	asin	acos
$-10^{25} \le x \le 10^{-16}$	0.83/0.94/2.50	0.64/0.65/1.50
$10^{-16} \le x \le 10^{25}$	0.83/0.94/2.50	0.64/0.67/2.94

Table 7.5. *Median, average and worst accuracy asin and acos*

For the inverse secans we use $\mathrm{asec}(-x) = \pi - \mathrm{asec}(x)$ for negative arguments, and for $x \ge 1$ we use the identities

$$\begin{array}{lcll} \mathrm{asec}(x) & = & \mathrm{atan}\sqrt{(x-1)(x+1)} & \text{for } 1 \le x \le 1.5 \\ \mathrm{asec}(x) & = & \mathrm{atan}\sqrt{x^2-1} & \text{for } 1.5 < x \le 10^{17}. \end{array}$$

Note that the difference $x - 1$ is executable without rounding error for $1 \le x \le 1.5$. For the inverse cosecans we use $\mathrm{acsc}(-x) = -\mathrm{acsc}(x)$ for negative arguments, and for $x \ge 1$ we use the identities

$$\begin{array}{lcll} \mathrm{acsc}(x) & = & \mathrm{atan}(1/\sqrt{(x-1)(x+1)}) & \text{for } 1 \le x \le 1.5 \\ \mathrm{acsc}(x) & = & \mathrm{atan}(1/\sqrt{x^2-1}) & \text{for } 1.5 < x \le 10^{17} \end{array}$$

with suitable setting of the rounding mode.

For large arguments $x > 10^{17}$, for both asec and acsc overflow problems are avoided by observing $\sqrt{x^2-1} \in [\mathrm{pred}(x), x]$.

The test sets for the inverse secans and cosecans are $T_{\mathrm{asec}} = T_{\mathrm{acsc}} = (\pm V_1) \cup (\pm V_2)$ for $V_1 = \mathrm{linspace}(0, 10, 21)$ and $V_2 = [10{:}5{:}25\ 30{:}30{:}300]$. The computational results are displayed in Table 7.6.

	median/average/worst accuracy $\alpha(x)$	
	asec	acsc
$-10^{300} \le x \le -10^{10}$	1.27/1.27/1.27	1.06/1.08/1.98
$-10^{10} \le x \le -1$	1.27/1.27/1.66	1.15/1.19/2.98
$1 \le x \le 10^{10}$	0.64/0.68/2.49	1.15/1.19/2.99
$10^{10} \le x \le 10^{300}$	0.64/0.64/0.64	1.06/1.08/1.98

TABLE 7.6. *Median, average and worst accuracy asec and acsc*

8. Hyperbolic functions. For large arguments $x \ge 8$ we use $\sinh(x) = (e^x - 1/e^x)/2$. To avoid unneccessary overflow, we use for $x \ge 709$

$$\nabla(\underline{e} \cdot \exp(x-1)/2) \le \sinh(x) \le \Delta(\overline{e} \cdot \exp(x-1)/2), \tag{18}$$

where $\underline{e}, \overline{e} \in \mathbb{F}$ are the closest floating point numbers with $\underline{e} < \exp(1) < \overline{e}$. The bounds $\underline{e}, \overline{e}$ are sufficiently far away from exp(1) to cover the additional error caused by e^{-x} in $\sinh(x) = (e^x - e^{-x})/2$. Both formulas produce sufficiently accurate results, and by $\sinh(-x) = -\sinh(x)$ the case $0 \le x < 8$ remains. For $x = f \cdot 2^e$, $0.5 \le f < 1$, $b = 14 + \min(e, 0)$ and $\widetilde{x} = 2^{e-b} \cdot \lfloor 2^b f \rfloor$ we have $\widetilde{x} \le x$, and for $d = x - \widetilde{x}, 0 \le d < 2^{-11}$, where d is computed without rounding error. Moreover,

$$\sinh(x) = \sinh(\widetilde{x} + d) = \sinh(\widetilde{x})\cosh(d) + \cosh(\widetilde{x})\sinh(d).$$

Then

$$\begin{aligned} d + d^3/3! \le \sinh(d) \le d + d^3/3! + \sinh(d) \cdot d^4/4!, \\ 1 + d^2/2 + d^4/4! \le \cosh(d) \le 1 + d^2/2 + d^4/4! + \sinh(d) \cdot d^5/5!. \end{aligned}$$

Furthermore $\widetilde{x} \in R_{\text{sinh}}$, and by (4) and $\widetilde{s} = \square(\sinh(\widetilde{x}))$,

$$\nabla(\widetilde{s}(1 - \varepsilon_{\text{sinh}}) \le \sinh(\widetilde{x}) \le \Delta(\widetilde{s}(1 + \varepsilon_{\text{sinh}}).$$

Also $\widetilde{x} \in R_{\exp}$, and therefore for $\widetilde{e} = \square(\exp(\widetilde{x})), \underline{E} = \nabla(\widetilde{e}(1 - \varepsilon_{\exp})), \overline{E} = \Delta(\widetilde{e}(1 + \varepsilon_{\exp}))$,

$$\underline{c} := \nabla(0.5(\underline{E} + 1/\overline{E})) \le \cosh(\widetilde{x}) \le \Delta(0.5(\overline{E} + 1/\underline{E})) =: \overline{c}.$$

To achieve high accuracy, we combine the formulas in the following way:

$$\begin{aligned} dd &= \nabla(d \cdot d) \\ \underline{s} &= \nabla(\widetilde{s} + ((\widetilde{s}(-\varepsilon_{\text{sinh}} + (1 - \varepsilon_{\text{sinh}}) \cdot dd/2 \cdot (1 + dd/12)) + \underline{c} \cdot d \cdot dd/6) + \underline{c} \cdot d)) \\[1em] dd &= \Delta(d \cdot d) \\ \overline{s} &= \Delta(\widetilde{s} + ((\widetilde{s}(\varepsilon_{\text{sinh}} + (1 + \varepsilon_{\text{sinh}}) \cdot dd/2 \cdot (1 + dd/2 \cdot (1 + 4.8 \cdot 10^{-8}))) \\ &\qquad + \overline{c} \cdot d \cdot dd/6 \cdot (1 + 6 \cdot 10^{-8})) + \overline{c} \cdot d)), \end{aligned}$$

where the error terms follow by $d < 2^{-11}$. It follows $\underline{s} \le \sinh(x) \le \overline{s}$ with bounds $\underline{s}, \overline{s} \in \mathbb{F}$ computed in double precision.

For the hyperbolic cosine we obtain sufficiently accurate bounds by

$$0.5(\underline{E} + 1/\overline{E}) \le \cosh(x) \le 0.5(\overline{E} + 1/\underline{E}) \quad \text{for } x < 709,$$

where $\underline{E} \le \exp(x) \le \overline{E}$ are bounds computed by the algorithm for the exponential function described in Section 5. For very large arguments $x \ge 709$, we use a formula similar (18).

The test sets for the hyperbolic sine and cosine are $T_{\text{sinh}} = T_{\text{cosh}} = \pm\text{linspace}(-16, \log_{10}(700), 25)$. For larger arguments the accuracy is similar to that of the exponential function. The results are listed in Table 8.1.

	median/average/worst accuracy $\alpha(x)$	
	sinh	cosh
$-700 \le x \le -10^{-16}$	0.44/0.81/2.73	1.50/1.54/2.98
$10^{-16} \le x \le 700$	0.44/0.81/2.72	1.50/1.54/2.98

TABLE 8.1. *Median, average and worst accuracy sinh and cosh*

For the hyperbolic tangent we use

$$\tanh(x) = 1 + \frac{-2}{\exp(2x)+1} \qquad \text{for } x \ge 1,$$

where bounds for the exponential are computed according to Section 5. For $0 \le x < 1$, let $\widetilde{x} = 2^{-14} \cdot \lfloor 2^{14}x \rfloor$ and $d = x - \widetilde{x}$, where $0 \le d < 2^{-14}$. The Taylor series of tanh and $d^5 \cdot 2/15 \le 1.910^{-18} \cdot d$ yield

$$\nabla(d \cdot (1 + (-d) \cdot d/3)) \le \tanh(d) \le \Delta(d \cdot (1 + ((-d) \cdot d/3 + 1.9 \cdot 10^{-18}))). \tag{19}$$

Furthermore,

$$\tanh(x) = \tanh(\widetilde{x} + d) = \frac{\tanh \widetilde{x} + \tanh d}{1 + \tanh \widetilde{x} \tanh d} = \tanh \widetilde{x} + \tanh d \cdot \frac{1 - \tanh^2 \widetilde{x}}{1 + \tanh \widetilde{x} \tanh d}. \tag{20}$$

By the definition, $\widetilde{x} \in R_{\text{tanh}}$, so that (3) implies $\widetilde{y}(1 - \varepsilon_{\text{tanh}}) \le \tanh(\widetilde{x}) \le \widetilde{y}(1 + \varepsilon_{\text{tanh}})$ for $\widetilde{y} = \square(\tanh(\widetilde{x}))$. Combining this with (19) and (20) produces accurate bounds for the hyperbolic tangent.

As a test set we use $T_{\text{tanh}} = (\pm V_1) \cup (\pm V_2)$ where $V_1 = -16{:}10$ and $V_2 = [10{:}5{:}25 \;\; 30{:}30{:}300]$.

Computational results are listed in Table 8.2.

	median/average/worst accuracy $\alpha(x)$
$-10^{300} \le x \le -10^{10}$	0.25/0.25/0.25
$-10^{10} \le x \le -10^{-16}$	0.41/0.64/2.39
$10^{-16} \le x \le 10^{10}$	0.41/0.64/2.36
$10^{10} \le x \le 10^{300}$	0.25/0.25/0.25

TABLE 8.2. *Median, average and worst accuracy tanh*

The bounds for the hyperbolic tangent are accurate enough that $\coth(x) = 1/\tanh(x)$ produces bounds of sufficient accuracy for the hyperbolic cotangent. The test set is $T_{\text{coth}} = (\pm V_1) \cup (\pm V_2)$ for $V_1 = [-300{:}30{:}-30 \;\; -25{:}5{:}-10]$ and $V_2 = -10{:}16$, testing especially for small arguments. Results are listed in Table 8.3.

	median/average/worst accuracy $\alpha(x)$
$-10^{16} \le x \le -10^{-10}$	0.50/0.83/2.92
$-10^{-10} \le x \le -10^{-300}$	1.08/1.08/1.50
$10^{-300} \le x \le 10^{-10}$	1.08/1.08/1.50
$10^{-10} \le x \le 10^{16}$	0.50/0.83/2.94

TABLE 8.3. *Median, average and worst accuracy coth*

9. Inverse hyperbolic functions.

Similar to the inverse trigonometric functions, the inverse hyperbolic functions are based on atanh, where special care is necessary for the inverse hyperbolic sine. We need bounds for the function $\text{log1}(x) := \log(1+x)$ for $0 \le x \le 1$, which are realized by

$$\sum_{i=1}^{4} \frac{(-1)^{i+1}x^i}{i} \le \log(1+x) \le \sum_{i=1}^{5} \frac{(-1)^{i+1}x^i}{i} \tag{21}$$

Then

$$\text{atanh}(x) = \log\left(1 + \frac{2x}{1-x}\right)/2$$

is sufficiently accurate, where for $x < 0.33$ definition (21) is used, and otherwise the general logarithm as described in Section 6. For the test set $T_{\text{atanh}} = \pm(-16 : 0)$ the accuracy results are listed in Table 9.1.

	median/average/worst accuracy $\alpha(x)$
$-1 \le x \le -10^{-16}$	1.00/1.11/2.52
$10^{-16} \le x \le 1$	1.00/1.11/2.52

TABLE 9.1. *Median, average and worst accuracy atanh*

The inverse hyperbolic cotangent is realized by $\coth(-x) = -\coth(x)$ and

$$\operatorname{acoth}(x) = \begin{cases} \log\left(1 + \frac{2}{x-1}\right)/2 & \text{for } x < 4 \\ \operatorname{atanh}(1/x) & \text{otherwise} \end{cases}$$

with suitable roundings. For the logarithm and inverse hyperbolic tangent the functions as described in Section 6 and the previous paragraph are used, respectively.

The test set $T_{\text{acoth}} = (\pm V_1) \cup (\pm V_2)$ with $V_1 = 0{:}10$ and $V_2 = [10{:}5{:}25\ \ 30{:}30{:}300]$ yields the accuracy results listed in Table 9.2.

	median/average/worst accuracy $\alpha(x)$
$-10^{300} \le x \le -10^{10}$	1.08/1.10/2.00
$-10^{10} \le x \le -1$	1.50/1.54/3.00
$1 \le x \le 10^{10}$	1.50/1.54/3.00
$10^{10} \le x \le 10^{300}$	1.08/1.10/2.00

TABLE 9.2. *Median, average and worst accuracy acoth*

For the inverse hyperbolic sine we use $\operatorname{asinh}(-x) = -\operatorname{asinh}(x)$ for negative arguments, and for $x \ge 0$ we use the following expansions:

$$\operatorname{asinh}(x) = \begin{cases} \log(2x + \zeta_1) & \text{for } x > 10^{10} \\ \log(1 + (2x\sqrt{x^2+1} + 2x^2))/2 & \text{for } 0.35 < x \le 10^{10} \\ \log(1 + 2\sqrt{x^4 + x^2} + 2x^2)/2 & \text{for } 10^{-9} < x \le 0.35 \\ x - \zeta_2 \cdot x & \text{for } 0 \le x \le 10^{-9} \end{cases} \tag{22}$$

where $\zeta_1 = 2 \cdot \text{eps}$, $\zeta_2 = 0$ for an upper bound, and $\zeta_1 = 0$, $\zeta_2 = 1.7 \cdot 10^{-9}$ for a lower bound, respectively. For the third case in (22) we use again the bounds for $\log(1 + x)$ as defined in (21).

For arguments less than 10^{-9} in absolute value, the bounds are adjacent floating point numbers and of least significant bit accuracy. Otherwise we use the test set $T_{\text{asinh}} = (\pm V_1) \cup (\pm V_2)$, where $V_1 = -9{:}10$ and $V_2 = [10{:}5{:}25\ \ 30{:}30{:}300]$. Results are given in Table 9.3.

	median/average/worst accuracy $\alpha(x)$
$-10^{300} \le x \le -10^{10}$	0.37/0.37/1.00
$-10^{10} \le x \le -10^{-9}$	0.57/0.76/2.52
$10^{-9} \le x \le 10^{10}$	0.57/0.76/2.52
$10^{10} \le x \le 10^{300}$	0.37/0.37/1.00

TABLE 9.3. *Median, average and worst accuracy asinh*

Finally, the inverse hyperbolic cosine is realized by:

$$\operatorname{acosh}(x) = \begin{cases} \log(2x + \zeta_1) & \text{for } x > 10^{10} \\ \log(x + \sqrt{x^2 - 1}) & \text{for } 1.25 < x \le 10^{10} \\ \log(1 + E + \sqrt{E(2 + E)}) & \text{for } 1 \le x \le 1.25, \end{cases}$$

where $\zeta_1 = 2 \cdot \text{eps}$ for an upper bound and $\zeta_1 = 0$ otherwise. Furthermore, $E = x - 1$ in the third case, where again (21) is used. Note that E is calculated without rounding error.

For the test set $T_{\text{acosh}} = (0 : 10) \cup [10{:}5{:}25\ \ 30{:}30{:}300]$ the median, average and worst case accuracy are listed in Table 9.4.

	median/average/worst accuracy $\alpha(x)$
$1 \le x \le 10^{10}$	0.40/0.45/2.52
$10^{10} \le x \le 10^{300}$	0.37/0.37/1.00

Table 9.4. *Median, average and worst accuracy acosh*

10. Rigorous standard functions for complex arguments. For $x \in \mathbf{C}$, denote $X := <x, r> := \{z \in \mathbf{C} : |z - x| \le r\}$. Rigorous inclusions of a disc $Y = <y, s>$ in the complex plane containing $\{f(x) : x \in X\}$ for an elementary function f are obtained by the following results by Börsken [3]:

$$\begin{array}{rll} \exp: & y = \exp(x), & s = |y| \cdot (\exp(r) - 1) \\ \log: & y = \log(x), & s = -\log(1 - r/|x|) \\ \mathrm{sqrt}: & y = \mathrm{sqrt}(x), & s = |\mathrm{sqrt}(|x| - r) - \mathrm{sqrt}(|x|)|. \end{array}$$

All other elementary functions can be expressed by those three. This reduces the problem of rigorous bounds for the complex interval Y to the problem of rigorous bounds for $f(x)$ where $x \in \mathbf{C}$. This is straightforward using the algorithms described in the previous sections.

We note that this is one approach for rigorous bounds. The radii of the bounds for exp, log and sqrt are sharp. Overestimation (for example in terms of area of the resulting disc) is mainly due to the fact that the image of the midpoint of the input disc X is used as the midpoint of the resulting disc Y. For the other elementary functions the bounds are sometimes crude due to dependencies.

Obtaining sharp bounds for all standard functions in the complex case is quite involved. It has been solved for input rectangles in the complex plane in [4] and [15], see also [16].

11. Conclusion. The presented approach for calculation of rigorous bounds for the elementary functions is based on the result of built-in functions at a specific set of arguments. The bounds are very accurate over the entire range of definition, they are fast to compute and the algorithms are suitable for vector input. Moreover, all implementations can easily be adapted to other formats and/or other number systems.

The presented approach can be applied to non-elementary transcendental functions as well. For example, rigorous bounds for the Gamma function can be obtained by the same approach of precomputed bounds for $\Gamma(x)$ and some $\Psi^{(i)}(x)$ and x out of a reference set combined with well known recurrence relations [2].

A sample implementation of the presented library in Matlab [18] is with INTLAB [22] freely available from our homepage www.ti3.tu-harburg.de/rump/intlab/index.html for non-profit use. The execution time for the pure Matlab implementation, however, suffers from interpretation overhead. Together with the newly developed very fast interval arithmetic [21] and rigorous input/output [22], INTLAB is an interactive fast interval toolbox entirely written in Matlab.

12. Appendix. Following we will give a rigorous proof that the relative accuracy of the exponential function as described in Section 5 is bounded by 2.07eps. The proof uses standard techniques from floating point error analysis. To make it not more technical we restrict the floating point input argument x to $x \ge -666$. This avoids impleasant discussions of underflow topics. In fact, we will see that with this minor restriction (note that $e^{-666} \approx 10^{-290}$) we can leave out underflow from our negotiations.

We use standard notation from floating point error analysis [10, Chapter 3], especially

$$\gamma_n := n \cdot \mathrm{eps}/(1 - n \cdot \mathrm{eps}) \text{ and } (1 + \Theta_k)(1 + \Theta_j) = 1 + \Theta_{k+j} \tag{23}$$

for quantities $|\Theta_\nu| \le \gamma_\nu$. Furthermore, we will see that for $x \ge -666$ all operations will remain outside the underflow range. Therefore, all operations satisfy for $a, b \in \mathbb{F}$,

$$\bigcirc(a \circ b) = (a \circ b)(1 + \Theta) \qquad \text{for } \circ \in \{+, -, *, /\}, \quad \bigcirc \in \{\nabla, \Delta\} \text{ and } |\Theta| \le \mathrm{eps}. \tag{24}$$

Note that (cf.(1)) $\bigcirc(a \circ b)$ is the result of the floating point operation in use. For IEEE 754 double precision, eps $\approx 2.22 \cdot 10^{-16}$. Next we apply (23) and (24) and traditional techniques of floating point error analysis to the exponential function as described in Section 5. Quantities Θ_ν, Θ'_ν and so forth will always satisfy $|\Theta_\nu| \le \gamma_\nu$.

Observing $\widetilde{y} \ge 0$ and $d \ge 0$ yields

$$\underline{c} = \widetilde{y}\left(\sum_{i=1}^{3} \frac{d^i}{i!}\right)(1+\Theta_6),$$

where the index in Θ_6 results from a mere counting of operations in the Horner scheme evaluation of $\underline{c}$ as defined in (7) [notice that division by 2 is exact]. In the same way we obtain

$$\overline{c} = \widetilde{y}\left(\sum_{i=1}^{3} \frac{d^i}{i!} + 0.2501\frac{d^4}{4!}\right)(1+\Theta_{10}),$$

where the index 10 comes from the fact that $0.2501 \notin \mathbb{F}$. Putting things together and using $0 \le d < 2^{-14}$ we obtain

$$0 \le \underline{c} \le \overline{c} \le \varphi_1 \cdot \widetilde{y} \qquad \text{with } \varphi_1 < 6.104 \cdot 10^{-5}. \tag{25}$$

Furthermore

$$\begin{aligned} \overline{c} - \underline{c} &\le \widetilde{y}\left(\sum_{i=1}^{3} \frac{d^i}{i!} + 0.2501\frac{d^4}{4!}\right)(1+\gamma_{10}) - \widetilde{y}\left(\sum_{i=1}^{3} \frac{d^i}{i!}\right)(1-\gamma_6) \\ &= \widetilde{y}[(\gamma_6+\gamma_{10})\sum_{i=1}^{3} \frac{d^i}{i!} + 0.2501\frac{d^4}{4!}(1+\gamma_{10})] \end{aligned}$$

yields

$$\overline{c} - \underline{c} \le \varphi_2 \cdot \widetilde{y} \qquad \text{with } \varphi_2 < 6.11 \cdot 10^{-19}. \tag{26}$$

By (7) we have

$$\begin{aligned} \underline{y_1} &= ((\underline{c} - \underline{c} \cdot \varepsilon_{\exp}(1+\Theta))(1+\Theta') - \widetilde{y} \cdot \varepsilon_{\exp}(1+\Theta''))(1+\Theta''') \\ &= \underline{c}(1+\Theta_2) - \underline{c} \cdot \varepsilon_{\exp}(1+\Theta_3) - \widetilde{y} \cdot \varepsilon_{\exp}(1+\Theta'_2) \end{aligned} \tag{27}$$

by using the fact that the negation is exact. In the same way we treat $\overline{y_1}$ and obtain

$$\max(|\underline{y_1}|, |\overline{y_1}|) \le (\overline{c}(1+\varepsilon_{\exp}) + \widetilde{y} \cdot \varepsilon_{\exp})(1+\gamma_3),$$

and together with (25) and $\varepsilon_{\exp}$ from Table 3.1,

$$\max(|\underline{y_1}|, |\overline{y_1}|) \le \varphi_3 \cdot \widetilde{y} \qquad \text{with } \varphi_3 < 6.105 \cdot 10^{-5}. \tag{28}$$

Exploring (27) and the similar formula for $\overline{y_1}$, and using (25), (26),

$$\begin{aligned} \overline{y_1} - \underline{y_1} &\le \overline{c} - \underline{c} + \varepsilon_{\exp}(\underline{c} + \overline{c}) + 2\widetilde{y} \cdot \varepsilon_{\exp} + \gamma_3((\underline{c} + \overline{c})(1+\varepsilon_{\exp}) + 2\widetilde{y} \cdot \varepsilon_{\exp}) \\ &\le \widetilde{y}(\varphi_2 + 2\varphi_1\varepsilon_{\exp} + 2\varepsilon_{\exp} + \gamma_3(2\varphi_1(1+\varepsilon_{\exp}) + 2\varepsilon_{\exp})) \\ &\le \varphi_4 \cdot \widetilde{y} \quad \text{with} \quad \varphi_4 < 2.524 \cdot 10^{-16}. \end{aligned} \tag{29}$$

For the further analysis we split the computation of $\underline{y}, \overline{y}$ into three parts, according to the parentheses:

$$\begin{aligned} \underline{s_1} &= \nabla((\widetilde{y} + \underline{y_1}) \cdot \underline{E}) & \overline{s_1} &= \Delta((\widetilde{y} + \overline{y_1}) \cdot \overline{E}) \\ \underline{s_2} &= \nabla(\underline{s_1} + \underline{y_1} \cdot E) & \overline{s_2} &= \Delta(\overline{s_1} + \overline{y_1} \cdot E) \\ \underline{y} &= \nabla(\widetilde{y}E + \underline{s_2}) & \overline{y} &= \nabla(\widetilde{y}E + \overline{s_2}), \end{aligned} \tag{30}$$

where indices (X_{int}) are omitted for better readability. Next we observe for $i \geq -666$,

$$\begin{aligned} &\max(|\overline{E_i}|, |\underline{E_i}|) \leq \alpha \cdot E_i && \text{with } \alpha < 1.06 \cdot 10^{-16}, \\ &\overline{E_i} - \underline{E_i} \leq \beta \cdot E_i && \text{with } \beta < 1.24 \cdot 10^{-32}, \end{aligned} \tag{31}$$

which is taken from the precomputed data. By (30),

$$\underline{s_1} = (\widetilde{y} + \underline{y_1})\underline{E} + \Theta_2(\widetilde{y} + |\underline{y_1}|)\underline{E}, \qquad \overline{s_1} = (\widetilde{y} + \overline{y_1})\overline{E} + \Theta_2(\widetilde{y} + |\overline{y_1}|)\overline{E},$$

such that by (28) and (31),

$$\begin{aligned} \overline{s_1} - \underline{s_1} &\leq \widetilde{y}(\overline{E} - \underline{E}) + |\underline{y_1}\underline{E}| + |\overline{y_1}\overline{E}| + 2\gamma_2(\widetilde{y} + \max(|\underline{y_1}|, |\overline{y_1}|)) \cdot \max(|\underline{E}|, |\overline{E}|) \\ &\leq \widetilde{y}(\beta E + 2\varphi_3 \alpha E + 2\gamma_2(1 + \varphi_3) \cdot \alpha E) \\ &\leq \varphi_5 \widetilde{y} E \quad \text{with} \quad \varphi_5 < 1.30 \cdot 10^{-20}. \end{aligned} \tag{32}$$

Furthermore,

$$\begin{aligned} \max(|\underline{s_1}|, |\overline{s_1}|) &\leq (\widetilde{y} + \max(|\underline{y_1}|, |\overline{y_1}|)) \cdot \max(|\underline{E}|, |\overline{E}|)(1 + \gamma_2) \\ &\leq \widetilde{y}(1 + \varphi_3) \cdot \alpha E \cdot (1 + \gamma_2) \\ &\leq \varphi_6 \widetilde{y} E \quad \text{with} \quad \varphi_6 < 1.07 \cdot 10^{-16}. \end{aligned}$$

By (30),

$$\underline{s_2} = \underline{s_1} + \underline{y_1}E + \Theta_2(|\underline{s_1}| + |\underline{y_1}|E), \qquad \overline{s_2} = \overline{s_1} + \overline{y_1}E + \Theta_2(|\overline{s_1}| + |\overline{y_1}|E),$$

and as before we deduce

$$\begin{aligned} \max(|\underline{s_2}|, |\overline{s_2}|) &\leq (\max(|\underline{s_1}|, |\overline{s_1}|) + \max(|\underline{y_1}|, |\overline{y_1}|) \cdot E)(1 + \gamma_2) \\ &\leq \widetilde{y}E(\varphi_6 + \varphi_3)(1 + \gamma_2) \\ &\leq \varphi_7 \cdot \widetilde{y}E \quad \text{with} \quad \varphi_7 < 6.11 \cdot 10^{-5}, \end{aligned} \tag{33}$$

$$\begin{aligned} \overline{s_2} - \underline{s_2} &\leq \overline{s_1} - \underline{s_1} + (\overline{y_1} - \underline{y_1})E + \gamma_2(\max(|\underline{s_1}|, |\overline{s_1}|) + \max(|\underline{y_1}|, |\overline{y_1}|) \cdot E) \\ &\leq \widetilde{y}E(\varphi_5 + \varphi_4 + \gamma_2(\varphi_6 + \varphi_3)) \\ &\leq \varphi_8 \cdot \widetilde{y}E \quad \text{with} \quad \varphi_8 < 2.525 \cdot 10^{-16}. \end{aligned} \tag{34}$$

For an upper bound of $\alpha(x)$ as defined in (4) we need lower bounds for $\underline{y}$ and $\overline{y}$. By (24), (30) and $\widetilde{y} \geq 0, E \geq 0$,

$$\begin{aligned} \min(|\underline{y}|, |\overline{y}|) &\geq \widetilde{y}E(1 - \gamma_2) - (1 + \text{eps}) \cdot \max(|\underline{s_2}|, |\overline{s_2}|) \\ &\geq \widetilde{y}E(1 - \gamma_2 - (1 + \text{eps})\varphi_7) \\ &\geq \varphi_9 \cdot \widetilde{y}E \quad \text{with} \quad \varphi_9 > 1 - 6.12 \cdot 10^{-5} \end{aligned} \tag{35}$$

For the upper bound of $\overline{y} - \underline{y}$,

$$\begin{aligned} \overline{y} - \underline{y} &\leq \Delta(\widetilde{y}E) - \nabla(\widetilde{y}E) + \overline{s_2} - \underline{s_2} + \text{eps} \cdot (\nabla(\widetilde{y}E) + \Delta(\widetilde{y}E) + 2 \cdot \max(|\underline{s_2}|, |\overline{s_2}|)) \\ &\leq \widetilde{y}E(\text{eps} + \varphi_8 + 2\text{eps}(1 + \text{eps} + \varphi_7)) \\ &\leq \varphi_{10} \widetilde{y}E \quad \text{with} \quad \varphi_{10} < 9.187 \cdot 10^{-16}. \end{aligned} \tag{36}$$

Finally, inserting the results into the definition (5) of $\alpha(x)$ yields

$$\alpha(x) = \frac{\overline{y} - \underline{y}}{|\underline{y}| + |\overline{y}|} \cdot \text{eps}^{-1} \leq \frac{\varphi_{10}\widetilde{y}E}{2\varphi_9 \cdot \widetilde{y}E} \cdot \text{eps}^{-1} < 4.594 \cdot 10^{-16} \cdot \text{eps}^{-1} < 2.069.$$

As mentioned at the beginning, these estimations are valid ignoring underflow. However, carefully checking the operations, especially computation of $\underline{s_1}$ and $\overline{s_2}$ in (30), and observing the size of the involved quantities for floating point $x \geq -666$ shows that no underflow is possible. Henceforth, (24) is valid and we have the following result.

THEOREM 12.1. *Let $x \in \mathbb{F}, x \geq -666$, and let $\underline{y}, \overline{y} \in \mathbb{F}$ denote the inclusion of e^x computed by (8) and (9). Then the relative accuracy of the inclusion as defined in (5) satisfies*

$$\alpha(x) = \frac{\overline{y} - \underline{y}}{|\underline{y}| + |\overline{y}|} \cdot eps^{-1} < 2.07.$$

The smallest representable positive normalized floating point number in IEEE 754 double precision is $realmin \approx 2.22 \cdot 10^{-308}$. For $y = e^x$ below that number, that is for $x \lesssim -708$, the relative accuracy $\alpha(x)$ increases with the ratio $realmin/y$. The question remains what happens for $-708 \leq x \leq -666$. To adapt the proof of the preceding theorem to take care of underflow is quite technical, especially for x near -708. Instead, we performed extensive numerical tests in the range $-708 \leq x \leq -666$. It seems that for $x \geq -707$, the maximum value for $\alpha(x)$ is still around 2, whereas for $-708 \leq x \leq 707$ it increases to 2.5.

Acknowledgement. The author wishes to thank the referees for detailed comments and stimulating remarks.

REFERENCES

[1] O. Aberth and M. Schaefer. Precise Computation Using Range Arithmetic, via C++. *ACM Trans. Math. Softw.*, 18, No. 4:481–491, 1992.

[2] M. Abramowitz and I. Stegun. *Handbook of Mathematical Functions.* Dover publications, New York, 1968.

[3] N.C. Börsken. Komplexe Kreis-Standardfunktionen (Diplomarbeit). Freiburger Intervall-Ber. 78/2, Inst. f. Angewandte Mathematik, Universität Freiburg, 1978.

[4] K.D. Braune. *Hochgenaue Standardfunktionen für reelle und komplexe Punkte und Intervalle in beliebigen Gleitpunktrastern.* Dissertation, Universität Karlsruhe, 1987.

[5] R.P. Brent. A Fortran Multiple-Precision Arithmetic Package. Technical report, Dept. of Computer Science, Australian National University, Canberra, 1975.

[6] R.P. Brent, J.A. Hooper, and J.M. Yohe. An AUGMENT Interface for Brent's Multiple Precision Arithmetic Package. *ACM Trans. Math. Software*, 6, No. 2:204–217, 1980.

[7] B.M. Brown, D.K.R. McCormack, and A. Zettl. On the existence of an eigenvalue below the essential spectrum. *Proc. R. Soc. Lond.*, 455:2229–2234, 1999.

[8] J.-P. Eckmann and P. Wittwer. *Computer Methods and Borel Summability Applied to Feigenbaum's Equation*, volume 227 of *Lecture Notes in Physics.* Springer Verlag, Berlin Heidelberg New York Tokyo, 1985.

[9] T.C. Hales. Cannonballs and Honeycombs. *Notices of the AMS*, 47(4):440–449, 2000.

[10] N.J. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM Publications, Philadelphia, 1996.

[11] O. Holzmann, B. Lang, and H. Schütt. Newton's constant of gravitation and verified numerical quadrature. *Reliable Computing*, 2(3):229–239, 1996.

[12] *ANSI/IEEE 754-1985, Standard for Binary Floating-Point Arithmetic*, 1985.

[13] R. Klatte, U. Kulisch, M. Neaga, D. Ratz, and Ch. Ullrich. *PASCAL-XSC — Sprachbeschreibung mit Beispielen.* Springer, 1991.

[14] R. Klatte, U. Kulisch, A. Wiethoff, C. Lawo, and M. Rauch. *C-XSC A C++ Class Library for Extended Scientific Computing.* Springer, Berlin, 1993.

[15] W. Krämer. *Inverse Standardfunktionen für reelle und komplexe Intervallargumente mit a priori Fehlerabschätzung für beliebige Datenformate.* Dissertation, Universität Karlsruhe, 1987.

[16] W. Krämer. Die Berechnung von Standardfunktionen in Rechenanlagen. *Jahrb. Überbl. Math. 1992*, pages 97–115, 1992.

[17] B. Lang. Verified Quadrature in Determining Newton's Constant of Gravitation. *J. Universal Computer Science*, 4(1):16–24, 1998.

[18] MATLAB User's Guide, Version 5. The MathWorks Inc., 1997.

[19] J.M. Muller. *Elementary Functions, Algorithms and Implementation.* Birkhauser, Boston, 1997.

[20] M. Payne and R. Hanek. Radian Reduction for Trigonometric Functions. *SIGNUM Newsletter*, 18:19–24, 1983.

[21] S.M. Rump. Fast and parallel interval arithmetic. *BIT*, 39(3):539–560, 1999.

[22] S.M. Rump. INTLAB - INTerval LABoratory. In Tibor Csendes, editor, *Developments in Reliable Computing*, pages 77–104. Kluwer Academic Publishers, Dordrecht, 1999.

[23] P.T.P. Tang. Table-lookup algorithms for elementary functions and their error analysis. Technical Report MCS-P194-1190, Aragonne National Lab., January 1991.

[24] W.F. Wong and E. Goto. Fast hardware-based algorithms for elementary function computations. In *Proc. International Symposium on Supercomputing*, pages 56–65, Fukuoka, Japan, 1991. Kyushu University Press.