The Table Maker's Dilemma

Results and Applications

Vincent Lefèvre

November 16, 2000

1. Introduction.
2. Exhaustive tests.
3. Timings and results.
4. Application to the implementation of 2^x.
5. Conclusion.

Exact Rounding

IEEE-754 Standard (1985):

- *Active* rounding mode: $\diamond$
- x and y: machine numbers.

When one computes $x \star y$ ($\star$ being $+$, $-$, $\times$ or $\div$), the obtained result must *always* be $\diamond(x \star y)$, i.e. *the rounding of the exact result*.

Same requirement for $\sqrt{x}$.

Unfortunately, not yet specifications for the elementary functions (exp, log, sin, cos…).

The Table Maker's Dilemma

- a floating-point system in base 2, n-bit mantissa;
- an elementary function f (exp, log, sin, cos...);
- a machine number x;
- for $m > n$, one can compute an approximation y' to $y = f(x)$ with an error on its mantissa less than 2^{-m}.

Problem: *Does one obtain the rounding of $f(x)$ by rounding y'?*

Not always possible if y has the form:

- in rounding to the nearest mode,

$$\overbrace{\underbrace{1.xx\ldots xx}_{n\text{ bits}}\,1000\ldots00}^{m\text{ bits}}\,xx\ldots \quad\text{or}\quad \overbrace{\underbrace{1.xx\ldots xx}_{n\text{ bits}}\,0111\ldots11}^{m\text{ bits}}\,xx\ldots$$

- in rounding towards 0, $+\infty$ or $-\infty$ modes,

$$\overbrace{\underbrace{1.xx\ldots xx}_{n\text{ bits}}\,0000\ldots00}^{m\text{ bits}}\,xx\ldots \quad\text{or}\quad \overbrace{\underbrace{1.xx\ldots xx}_{n\text{ bits}}\,1111\ldots11}^{m\text{ bits}}\,xx\ldots$$

This problem is called the *Table Maker's Dilemma* (TMD).

Examples in Double Precision

For

$$\begin{aligned} x &= 0.0111111110011101100111011100111001110100001111101101101 \\ &= \frac{8980155785351021}{18014398509481984}, \end{aligned}$$

$\sin x$ is equal to:

$$0.0111101001100101010000011100110000110001000110100101 01\ 1\,1^{65}\ 0000\ldots$$

Considering the reciprocal, we have: for

$$\begin{aligned} x &= 0.01111010011001010100000111001100001100010001101001 0110 \\ &= \frac{4306410053968715}{9007199254740992}, \end{aligned}$$

$\arcsin x$ is equal to:

$$0.0111111110011101100111011100111001110100001111101101101\ 0\,0^{64}\ 1000\ldots$$

Solving the TMD

- Lindemann, 1882: the exponential of an algebraic number $\neq 0$ is not algebraic;
- the floating-point numbers are algebraic;

$\Rightarrow \exp(x), \sin(x), \cos(x), \arctan(x)$ for $x \neq 0$, and $\log x$ for $x \neq 1$ cannot have infinitely many consecutive 0's or 1's in their binary expansion.

$\Rightarrow$ For all x, there exists m such that the TMD does not occur.

The number of machine numbers is finite $\Rightarrow$ there exists m such that for all x the TMD does not occur.

Problem: *to find this m* (intermediate precision).

Some Estimates

Experiments $\rightarrow$ it seems that $m \approx 2n$.

Warning! This approach is *not* rigorous. We seek to intuitively understand where the relation $m \approx 2n$ comes from. We suppose:

- rounding to the nearest;
- when x is a machine number, the bits of $f(x)$ after the n-th position can be seen as *random sequences of 0's and 1's*, with equal probabilities;
- these sequences can be regarded as *independent* for two different machine numbers.

The mantissa of $y = f(x)$ has the form:

$$y_0.y_1y_2\ldots y_{n-1}\overbrace{01111\ldots11}^{k\text{ bits}}\ldots \quad\text{or}\quad y_0.y_1y_2\ldots y_{n-1}\overbrace{10000\ldots00}^{k\text{ bits}}\ldots$$

with $k \geq 1$. Largest value of k?

Our hypotheses $\rightarrow$ the "probability" to have $k \geq k_0$ is 2^{1-k_0}.

n mantissa bits and n_e exponents: $N = n_e \cdot 2^{n-1}$ machine numbers
$\Rightarrow m_{\max} = n + k_{\max} \approx n + \log_2(N) = 2n + \log_2(n_e) - 1$.

Best theorems (Nesterenko and Waldschmidt, 1995)
$\rightarrow m_{\max} \leq$ several millions or billions for the functions related to the complex exponential ($\exp$, $\log$, trigonometric and hyperbolic functions).

$\rightarrow$ *Exhaustive tests*

Exhaustive Tests

Problem: consider a floating-point system, a function f on an interval I, and an integer m. What are the machine numbers $x \in I$ such that the mantissa of $f(x)$ has the following form?

$$\overbrace{\underbrace{1.xx\ldots xx}_{n\text{ bits}}\,rbbb\ldots bb}^{m\text{ bits}}\,xx\ldots$$

where all the bits b have the same value.

Estimate of the computation time for an elementary function f, $n = 53$ (double precision), $m \approx 90$, 500 MHz machine, a conventional algorithm (200 cycles): 2^{52} mantissas $\rightarrow$ *57 years* for each exponent!

$\rightarrow$ *We need very fast algorithms.*

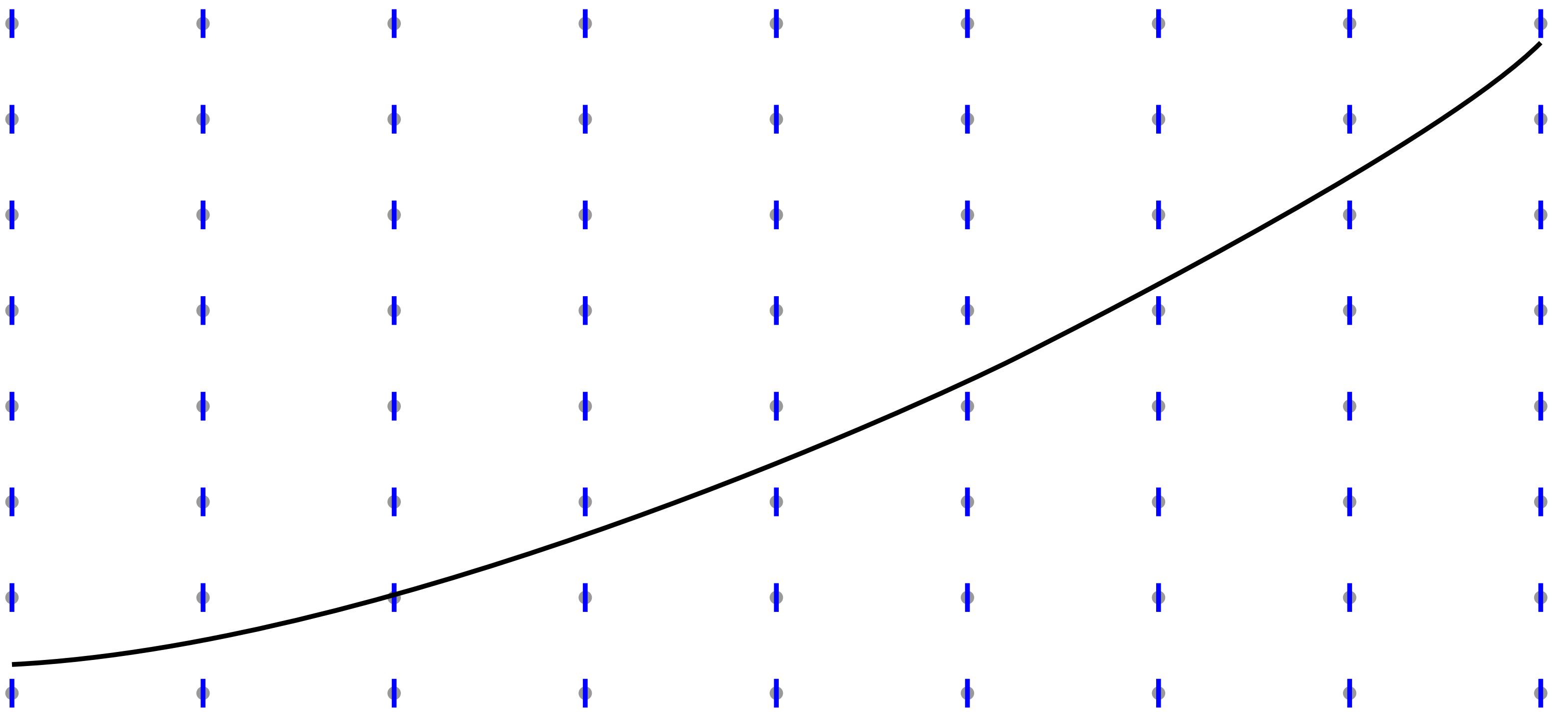
Filters

1. Filter: very fast algorithm (low precision) to select a superset S of all the "worst cases" (arguments such that $m \geq m_0$).
2. Test each machine number in S with a more accurate algorithm, that can be much slower.

Note:

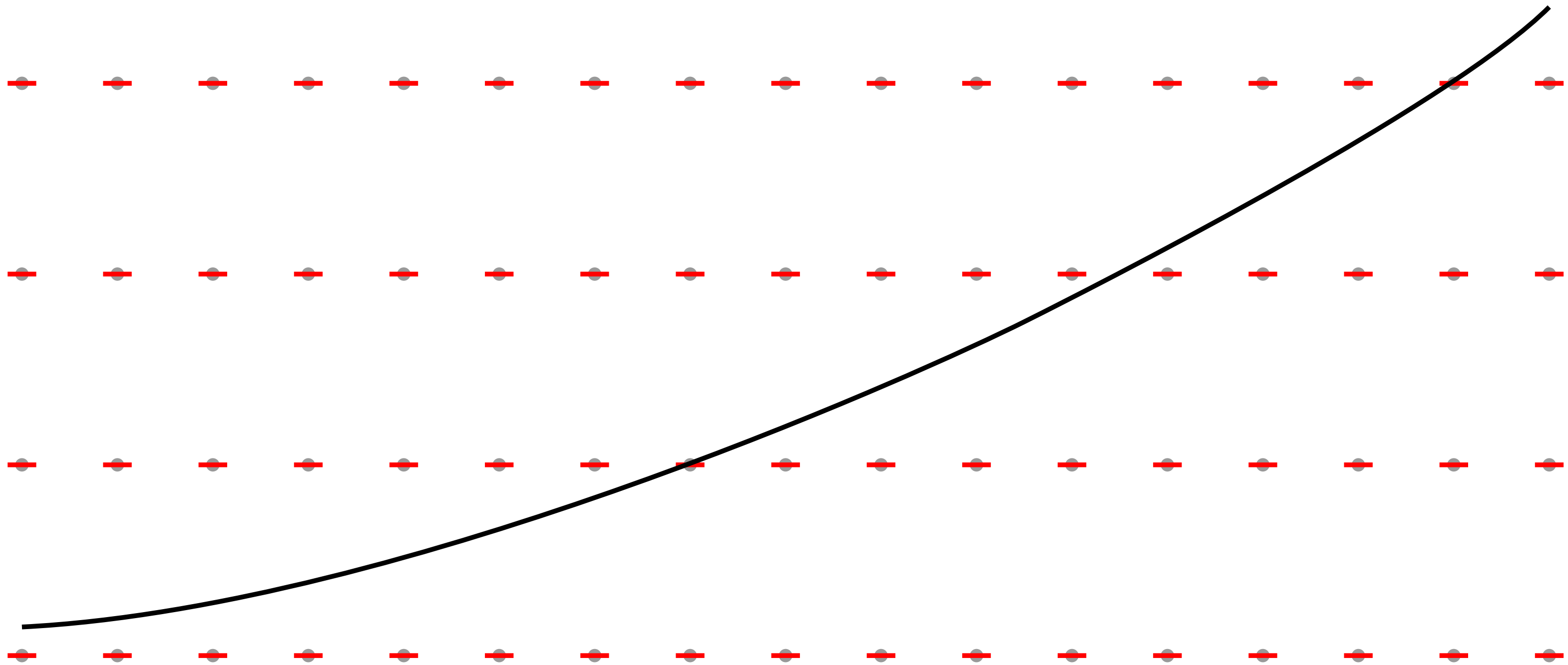
- we may use several filters;
- filters are chosen using the probabilistic hypotheses.

Testing f in a Given Domain



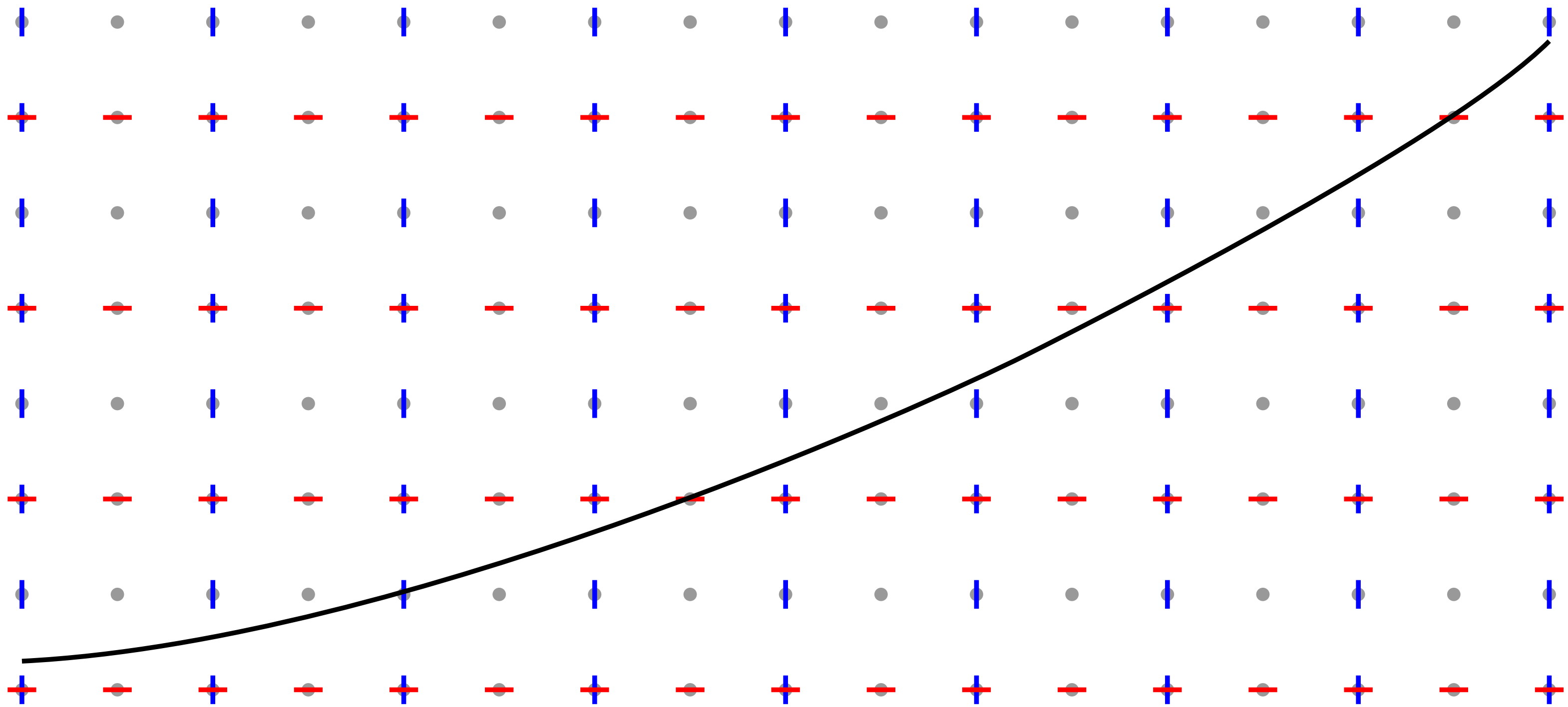
$\rightarrow$ 9 tested arguments.

Testing f^{-1} in the Same Domain



$\longrightarrow$ 4 tested arguments.

$f \leftrightarrow f^{-1}$ Equivalence

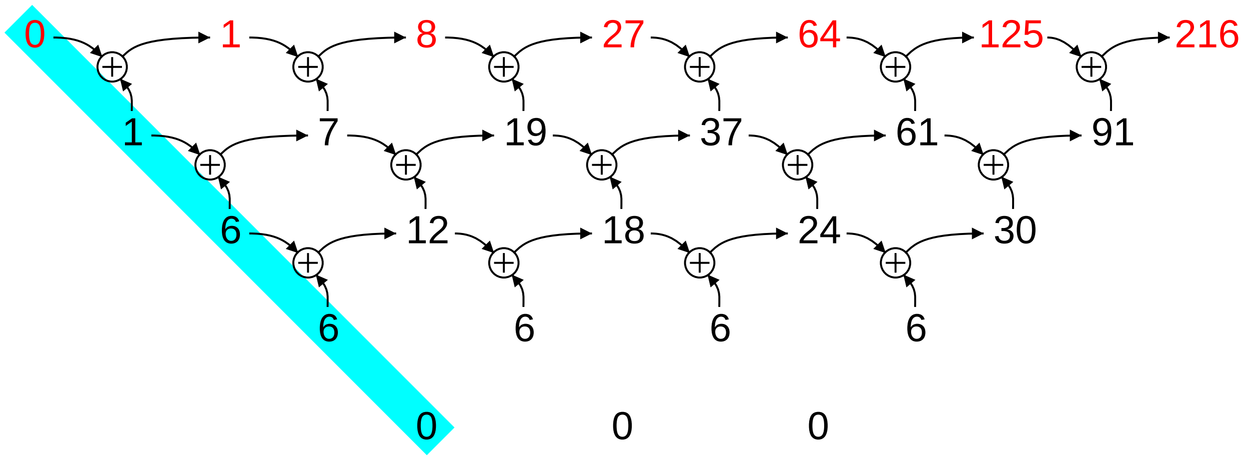


$\rightarrow$ 7 tested arguments (f^{-1}) instead of $9 + 4 = 13$.

Approximating a Function by a Polynomial

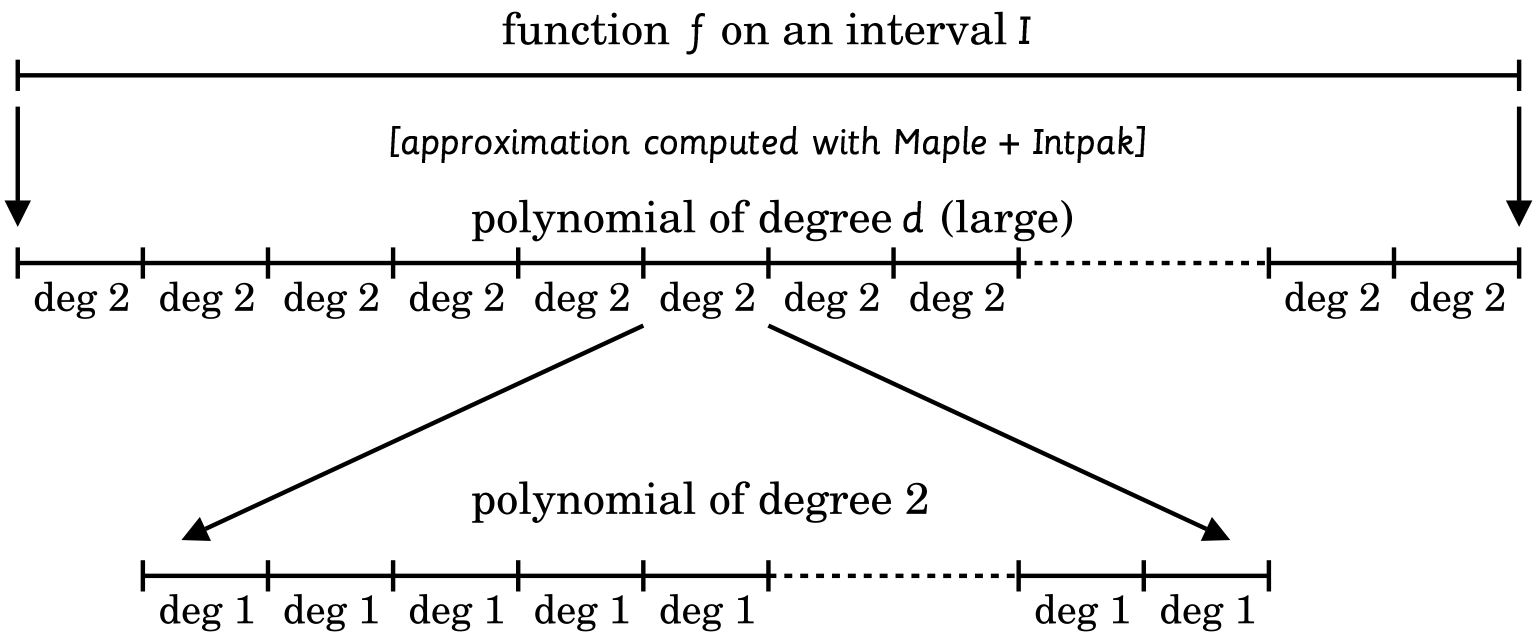
because the machine numbers are regularly spaced and computing the successive values of a polynomial can be performed very quickly.

E.g., polynomial $P(X) = X^3$. Difference table:



Coefficients in the basis $\left\{1, X, \dfrac{X(X-1)}{2}, \dfrac{X(X-1)(X-2)}{3!}, \ldots\right\}$.

Hierarchical Approximations



Degree-1 polynomials: fast algorithm that computes a lower bound on the distance between a segment and $\mathbb{Z}^2$ (extension of Euclid's algorithm).

Parallelizing the Computations

Target: a network of workstations (LIP + PSMN + Matra Capitan + student-lab machines).

Server + clients. The clients connect to the server to get an interval number (i) and other parameters → in general, 5 minutes to 2 hours of computations.

We implemented the clients / computation processes so that they

- run with a low priority (*nice*);
- automatically stop after a given time;
- automatically detect when a machine is used (keyboard, mouse…) and stop if this is the case;
- can automatically detect when there is another running process.

Timings

In practice, here at the ENS-Lyon, a few days to a few weeks per exponent (2^{53} mantissas).

Up to 35 arguments tested per cycle in average on a Sun Ultra-5. May be improved in future implementations.

The choice of interval sizes is very important. For instance ($\exp$, $x_0 = 16$, 2^{40} arguments), on a 333 MHz Sun Ultra-5:

$\#K_{i,j}$	$\#L_k$	time
32768	32768	9530 s
4096	4096	930 s
32768	8192	430 s
32768	4096	360 s
32768	2048	500 s

Results: exp and log in Double Precision

- $\exp(x)$ is tested for x between $1/2$ and $\log(2^{1024})$, and for x between $\log(2^{-1074})$ and $-1/2$ (subnormal numbers taken into account).
- $\log(x)$ is tested for x between $1/2$ and 2.

Results for exp, $m \geq 111$

For $|x| \geq 2^{-32}$:

E	mantissa	R	m
5	$-1.0001001011010011000110100010000011111011001110001011$	N	**112**
-13	$-1.1010001011111110111111101111110101011000000011011111$	D	**111**
-27	$-1.1110110100110001100011101111101101100010011111101010$	D	**113**
-29	$-1.0011010001110101101011000000010111001110101011010111$	D	**111**
-32	$1.0111111111111100111111111111110111000000000000100100$	D	**111**
-32	$1.1000000000000001011111111111111011011111111111011100$	D	**111**
-31	$1.1001111010011100101110111111110101100000100000001011$	N	**111**
2	$1.1000001111010100101111001101111010111011001111110100$	D	**111**

Otherwise, the non-trivial worst case is:

E	mantissa	R	m
-53	$1.1111111111111111111111111111111111111111111111111111$	D	**158**

Results for log, $m \geq 115$

E	mantissa	R	m
86	1.10010001111011000100010000001001011000011010001001111	D	**115**
245	1.1100100100001000000100001101001101010100011000011000	D	**117**
656	1.1000011001110000110111100000101101101000110010101101	D	**116**
678	1.0110001010101000100001100001001101100010100110110110	D	**118**
732	1.1111110100010101110110101010011011001110001100110010	N	**115**
772	1.01111011000111011001011111001001000000010100110000101	D	**115**

Worst cases for $x < 1$:

E	mantissa	R	m
-509	1.1110101001110001110110000101110011101110000000100000	D	**114**
-384	1.1001010001110110111000110000010011001101011111000111	N	**114**
-232	1.0010011011101001110001001101001100100111100101100000	D	**114**
$-\ 35$	1.0110000100111001010101011101110010000000001011111000	N	**114**

Results: 2^x and $\log_2(x)$ in Double Precision

Easier than $\exp$ and $\log$, because $\log_2(2^k t) = k + \log_2(t)$.

If k is an integer:

- $2^k t$ and t have the same mantissa;
- $k + \log_2(t) \to$ shift in the mantissa.

$\log_2(x)$ tested in $[1/2, 2)$.
2^x tested in $[1, 2)$ and $[32, 33)$.

Results for 2^x, $m \geq 111$

E	mantissa	R	m
-15	$-1.0010100001100011101010111010111010101111011110110010$	D	**111**
-20	$-1.0100000101101111011011000110010001000101101011001111$	D	**111**
-32	$-1.0000010101010110000000011100100010101011001111110001$	D	**111**
-33	$-1.0001100001011011100011011011011010101100000011101$	D	**111**
-29	$1.0101011010001110100011001110110001001111011001101100$	N	**111**
-27	$1.0001001010110001010010100011000110001111100100000100$	D	**112**
-25	$1.1011111110111011110111100100010011101101111111000101$	D	**113**
-10	$1.1110010001011001011001010010011010111111100101001101$	N	**113**
-10	$1.1110011101100000010010010000011100110000011001111111$	N	**111**
$-\ 8$	$1.1111100110011010111111101111101000110000110101100101$	D	**111**
$-\ 6$	$1.1000111111101010100000001111110110111010100010010110$	N	**111**

Results for $\log_2(x)$, $m \geq 106$

E	mantissa	R	m
0	1.10110100111010111110010000001100100101011010000000001	N	**107**
1	1.0001101110100011100111111111001010001110001111101010	D	**106**
2	1.0001101110100011100111111111001010001110001111101010	D	**107**
2	1.1000100111011001010010001010100101001111111000010111	N	**106**
4	1.0001101110100011100111111111001010001110001111101010	N	**108**
16	1.1001010101101011011011110011010000011111010111010000	N	**106**
64	1.0110000101010101010111110111010110001000010110110100	D	**106**
128	1.0110000101010101010111110111010110001000010110110100	D	**107**
128	1.1101001100001010010000110111011100111101110100011011	D	**106**
256	1.0110000101010101010111110111010110001000010110110100	D	**108**
256	1.1101001100001010010000110111011100111101110100011011	N	**107**
512	1.0110000101010101010111110111010110001000010110110100	D	**109**

(E: -1, 0 or power of 2 only; use $\log_2(2^k t) = k + \log_2(t)$ for the other exp.)

Results: Other Functions in Double Precision

Values of $m_{\max}$ for f and f^{-1}, in N and D rounding modes:

f	domain	f, N	f, D	f^{-1}, N	f^{-1}, D
sin	2^{-5} to 2	110	119	108	118
cos	2^{-6} to $12867/8192$	108	109	111	116
tan	2^{-5} to $\arctan(2)$	111	109	108	109
sh	1 to 2^4	107	107	112	109
ch	2^{-1} to 2^5	111	109	115	111

Results: exp in Single Precision, $m \geq 51$

For $|x| \geq 2^{-15}$:

E	mantissa	R	m
5	$-1.01101101011110110001100$	D	**51**
3	$-1.11010010001001011001101$	N	**52**
-2	$-1.10101100111111110010101$	D	**51**
-8	$-1.11100001110110111110001$	N	**51**
-9	$-1.01100101100111101100100$	D	**52**
-10	$-1.11000001110001001011100$	N	**51**
-10	$1.01100010011110101001111$	D	**52**

Otherwise, the non-trivial worst case is:

E	mantissa	R	m
-24	$1.11111111111111111111111$	D	**71**

Results: log in Single Precision, $m \geq 55$

E	mantissa	R	m
-66	1.00010000100010100101101	D	**57**
-65	1.00100010110101010111000	N	**55**
3	1.00101111000111111101011	N	**55**
25	1.10111010110010110100101	N	**57**
27	1.11000000100111010111110	N	**56**
76	1.10110001001000011010011	N	**58**
78	1.01010001100100001100000	N	**55**
117	1.00101111111001100001010	D	**56**

Results: $f(x) = 1/x^2$ and $g(x) = 1/\sqrt{x}$

Tests for various precisions: $19 \leq n \leq 58$. Worst cases:

$n = 21$:
$f(0.110100100010001011100) = 1.01111011111100010100\,1\,1^{29}\,0110\ldots$

$n = 21$:
$g(1.01111011111100010101) = 0.110100100010001011011\,1\,1^{30}\,0101\ldots$

$n = 20$:
$f(0.11010010001000101110) = 1.0111101111110001010\,0\,1^{30}\,0110\ldots$

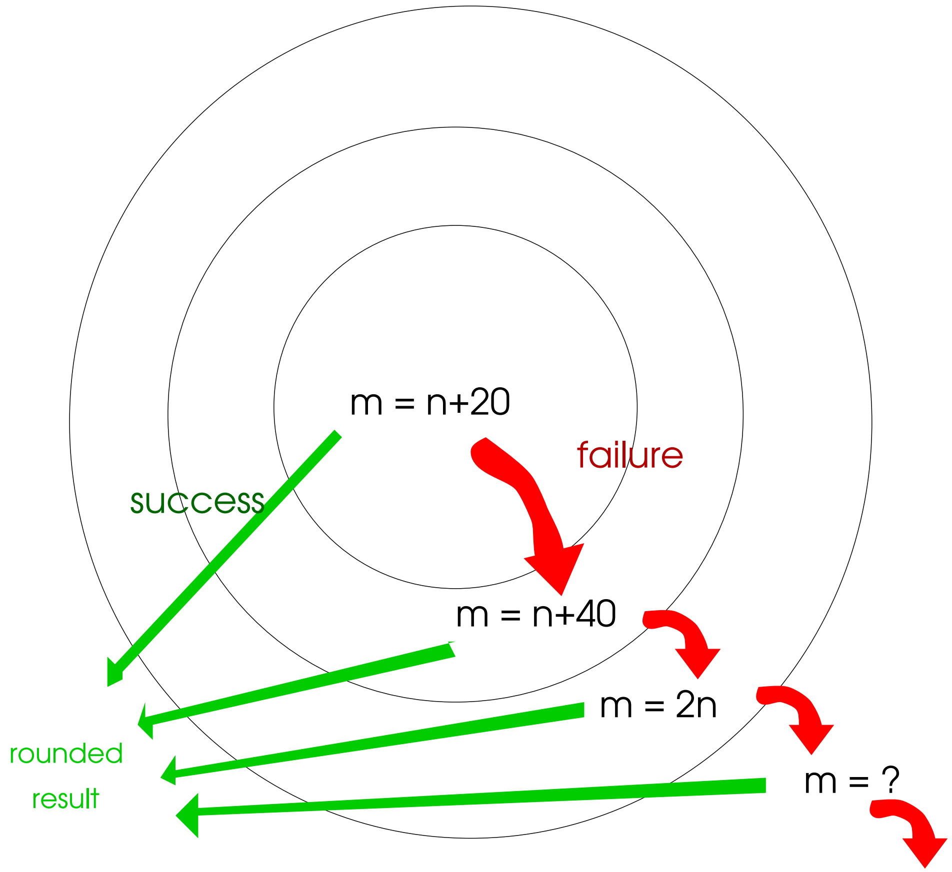
Correspond to: $1556245 \times 430359^2 = 2^{58} + 101$

Our Results and the Probabilistic Hypotheses

Average number of worst cases per exponent such that $k \geq k_0$:

k_0	estimate	$\exp(x)$	$\exp(-x)$	$\sin(x)$	$\cos(x)$
45	768	788.7	761.0	774.8	762.5
46	384	393.4	377.9	398.0	396.5
47	192	200.3	190.1	198.7	190.0
48	96	98.2	95.9	104.5	94.5
49	48	52.9	46.8	52.8	41.5
50	24	27.4	23.5	26.2	18.0
51	12	14.0	11.4	12.2	6.5
52	6	7.0	5.2	6.3	3.0
53	3	2.6	2.4	3.3	2.0
exponents		-1 to 8	0 to 7	-5 to 0	-2 to -1

Ziv's Strategy



Using the Results of the Tests

1. Build an algorithm that computes f, using Ziv's strategy, until m_0-bit precision.
2. Test the algorithm with all the worst cases satisfying $m \geq m_0$.
3. Keep the worst cases for which the TMD occurs (will be stored in a table) and complete the implementation.

Small $m_0 \rightarrow$ fast implementation, but large table.

- The algorithm determines if x is a real worst case or not.
- The algorithm computes an approximation to $f(x)$.

$\rightarrow$ For a real worst case x, we do not need to store the whole number x (but a hash code), nor the rounded result (1 bit is sufficient).

Implementation of 2^x

We aim at showing that exact rounding is possible at low cost.
$\rightarrow$ *We focused on worst cases.* Ziv's strategy is *not* used (yet).

Range reduction $\rightarrow |x| \leq \frac{1}{2}$.

If $|x| < 2^{-40}$, special table-based algorithm: $2^x \approx 1 + x.\log(2)$ and the boundary arguments to 2 different rounded values are stored in a table ($\approx 17\,000$ values).

Otherwise, 2^x computed with an error $< 2^{-97} + 2^{-101}$.
All possible worst cases are tested $\rightarrow$ real worst cases.

Rounding mode	Worst cases	Exact	Inexact
towards $-\infty$	52 231	51 148	1 083
towards $+\infty$	52 231	51 028	1 203
to the nearest	52 224	26 174	26 050

$\rightarrow$ only the worst cases that correspond to an inexact rounding are stored. In the algorithm: first hash-code (12 bits) to reduce the set; second hash-code (2 bytes) for the comparisons (at most 4 for the directed rounding modes, and 17 for the rounding to the nearest mode).

Timing: up to $4.10\,\mu s$ on a Sun Ultra-5 at 333 MHz.

(IBM's ml4j: up to $4.4\,ms$ for $\exp$.)

Conclusion