Worst Cases for the Exponential Function in the IEEE 754r decimal64 Format

Vincent Lefèvre, Damien Stehlé, Paul Zimmermann

LORIA / INRIA Lorraine

JNAO 2006

31 May – 2 June 2006

Introduction / Outline

- The search for worst cases has already been done for some functions in the IEEE-754 binary double precision (64 bits).
- The goal: to extend the work to the decimal64 format.

Outline:

- The decimal formats.
- Searching for worst cases in decimal.
- The application to the exponential function.

The Decimal Formats

IEEE 854, applied to radix 10 and precision n. Non-special number:

$$x = (-1)^s \cdot 10^E \cdot d_0.d_1d_2 \ldots d_{n-1}$$

where:

- sign: $s \in \{0, 1\}$,
- exponent: $E \in [\![E_{\min}, E_{\max}]\!]$,
- digits: $0 \leq d_i \leq 9$.

We require that if $E \neq E_{\min}$, then $d_0 \neq 0$. $\rightarrow$ Unique representation.

- $d_0 \neq 0$: normal numbers.
- $d_0 = 0$: subnormal numbers.

The Three Standard Decimal Formats

The current IEEE 754r working draft defines three decimal formats: decimal32, decimal64 and decimal128.

Format	decimal32	decimal64	decimal128
Precision n	7	16	34
E_{min}	-95	-383	-6143
E_{max}	96	384	6144

More details on:

`http://www2.hursley.ibm.com/decimal/decbits.html`

The Table Maker's Dilemma

Like in radix 2…

Breakpoint: a value where the rounding changes in one of the rounding modes, i.e., the discontinuity points of the rounding functions.

- D: a machine number (for the directed rounding modes), or
- N: the middle of two consecutive machine numbers (for the rounding-to-nearest mode).

The **Table Maker's Dilemma**: Function f, machine number x, $f(x)$ evaluated with an error ε. If the distance between the result and the breakpoints is less than ε, one cannot decide the correctly-rounded value in the corresponding rounding mode.

The Worst Cases

Worst cases: the arguments x for which the values $f(x)$ are the "hardest to round". Four possible forms, like in radix 2.

For instance, for $f = \exp$ in the decimal64 format:

$$\exp(0.\underbrace{5091077534282133}_{\text{16 digits}}) = \underbrace{1.663806007261509}_{\text{16 digits}}\,\underbrace{5000\ldots00}_{\text{16 digits}}\,49\ldots \quad \text{(N)}$$

$$\exp(0.\underbrace{7906867968553504}_{\text{16 digits}}) = \underbrace{2.204910231771509}_{\text{16 digits}}\,\underbrace{4999\ldots99}_{\text{16 digits}}\,16\ldots \quad \text{(N)}$$

$$\exp(0.00\underbrace{1548443067391468}_{\text{16 digits}}) = \underbrace{1.001549642524374}_{\text{16 digits}}\,\underbrace{9999\ldots99}_{\text{16 digits}}\,26\ldots \quad \text{(D)}$$

$$\exp(0.\underbrace{2953379504777270}_{\text{16 digits}}) = \underbrace{1.343580345589067}_{\text{16 digits}}\,\underbrace{0000\ldots00}_{\text{16 digits}}\,86\ldots \quad \text{(D)}$$

Exhaustive Search for Worst Cases

Similar to radix 2.

- Arguments that give an underflow or an overflow are not tested.
- Probabilistic hypotheses $\rightarrow$ the smallest distance between $f(x)$ and the breakpoints is of the order of 10^{-n} ulp (divided by the number of exponents E), possibly except in some particular domains.
- In decimal32: a few billions of args $\rightarrow$ naive algorithms suffice. See also `http://www.loria.fr/~zimmerma/wc/decimal32.html`
- In decimal64: $\sim 10^{18}$ arguments $\rightarrow$ needs specific algorithms.
- In decimal128: 10^{34} to 10^{38} arguments $\rightarrow$ currently out of reach.

Exhaustive Search for Worst Cases in decimal64

In each domain where the exponent of $f(x)$ does not change, search for the solutions of:

$$|f(x) \bmod u/2| < \varepsilon u\,,$$

where the modulo is centered, $u = \mathrm{ulp}(f(x))$, which is a constant in the considered domain, and $\varepsilon \sim 10^{-n}$.

- Split the tested domain into subintervals. In each subinterval:
- Approximate the function f by a polynomial of small degree.
- Scale/translate the input and output values to reduce the problem to the following…

The Real Small Value Problem

Real Small Value Problem (Real SValP). Given positive integers M and T, and a polynomial P with real coefficients, find all integers $|t| < T$ such that:

$$|P(t) \bmod 1| < \frac{1}{M}\,.$$

Efficient algorithms:

- Lefèvre's algorithm, requiring degree-1 polynomials.
- The Stehlé-Lefèvre-Zimmermann (SLZ) algorithm, based on Coppersmith's technique and the LLL algorithm.
  3 variants (Stehlé's PhD thesis): "fast", "heuristic" and "proved".
  The first two are faster but may fail (infinite loop or FAIL result).

Correctly Rounding the Exponential Function

Non-special decimal64 number: $x = (-1)^s \cdot 10^E \cdot d_0.d_1 d_2 \dots d_{15}$ where $s \in \{0, 1\}$ and $-383 \le E \le 384$.

- Largest finite machine number: $10^{385} - 10^{369}$.
- Smallest positive normal machine number: 10^{-383}.
- Smallest positive machine number: 10^{-398}.

First, eliminate the special cases…

Four couples of consecutive machine numbers (a^-, a^+), (b^-, b^+), (c^-, c^+) and (d^-, d^+) that determine the following intervals:

$$\underbrace{-\infty \ \ldots \ a^-}_{+0} \ \underbrace{a^+ \ \ldots \ b^-}_{\text{search}} \ \underbrace{b^+ \ \ldots \ c^-}_{1} \ \underbrace{c^+ \ \ldots \ d^-}_{\text{search}} \ \underbrace{d^+ \ \ldots \ +\infty}_{+\infty}$$

- In intervals 1, 3 and 5, the rounded values in rounding-to-nearest are respectively $+0$, 1 and $+\infty$ (results for the directed rounding modes can be deduced, with the *same* interval bounds).
- In intervals 2 and 4, a search for worst cases is needed:
  $[a^+, b^-] = [-917.1220141921901, -5.000000000000001 \cdot 10^{-17}]$,
  $[c^+, d^-] = [4.999999999999999 \cdot 10^{-16}, 886.4952608027075]$.

Searching for Worst Cases of exp

Current implementation:

- Split the tested domain into intervals in which both the argument x and the result $\exp(x)$ have a constant (possibly different) exponent. Done with a small Maple program.
- Test each interval (in parallel on various machines). Done with BaCSeL (`http://www.loria.fr/~stehle/`), which uses SLZ. We chose degree-3 polynomials; their coefficients are computed with MPFR, to obtain guaranteed error bounds. Note: For values of x close enough to 0, the fast LLL variant fails, so that the proved variant is used in this domain.

Some Results

Domains currently tested:

- $[10^{-9}, 25.21686424838368]$;
- some subintervals of $[25.21686424838369, 886.4952608027075]$.

x	$\exp x$
$6.581539478341669 \cdot 10^{-9}$	$1.000000006581539\,5\,0^{15}\,177\ldots$
$2.662858264545929 \cdot 10^{-8}$	$1.000000026628583\,0\,0^{15}\,318\ldots$
$3.639588333766983 \cdot 10^{-8}$	$1.000000036395884\,0\,0^{15}\,240\ldots$
$6.036998017773271 \cdot 10^{-8}$	$1.000000060369982\,0\,0^{15}\,379\ldots$
$6.638670361402304 \cdot 10^{-7}$	$1.000000663867256\,4\,9^{15}\,569\ldots$
$9.366572213364879 \cdot 10^{-7}$	$1.000000936657659\,9\,9^{15}\,883\ldots$
$7.970613003079781 \cdot 10^{-6}$	$1.000007970644768\,5\,0^{15}\,362\ldots$

1/2

x	$\exp x$
$3.089765552852523 \cdot 10^{-5}$	$1.000030898132866\,0\,0^{15}\,241\ldots$
$1.302531956641873 \cdot 10^{-4}$	$1.000130261678980\,0\,0^{16}\,798\ldots$
$2.241856702421245 \cdot 10^{-4}$	$1.000224210801727\,5\,0^{15}\,118\ldots$
$7.230293679121590 \cdot 10^{-4}$	$1.000723290816653\,4\,9^{16}\,127\ldots$
$5.259640428979129 \cdot 10^{-3}$	$1.005273496619909\,4\,9^{15}\,739\ldots$
$9.407822313572878 \cdot 10^{-2}$	$1.098645682066338\,5\,0^{16}\,278\ldots$
$1.267914924960933 \cdot 10^{-1}$	$1.135180299492843\,0\,0^{16}\,706\ldots$
$5.091077534282133 \cdot 10^{-1}$	$1.663806007261509\,5\,0^{15}\,492\ldots$
3.359104074009002	$28.76340944572687\,5\,0^{16}\,904\ldots$
19.10511686234796	$1.982653538414981\,9\,9^{15}\,735\ldots\mathrm{E}8$
294.9551257293143	$1.251363586659789\,5\,0^{15}\,108\ldots\mathrm{E}128$
587.9131381356093	$2.125356221825522\,4\,9^{15}\,594\ldots\mathrm{E}255$

2/2

For $c^+ \leq x < 10^{-8}$, many bad cases have some patterns in their mantissa. For instance:

$$3.897940\mathbf{992}403028 \cdot 10^{-9}\,,$$
$$4.230932\mathbf{991}049603 \cdot 10^{-9}\,,$$
$$4.291382\mathbf{990}792016 \cdot 10^{-9}\,,$$
$$4.581289\mathbf{989}505891 \cdot 10^{-9}\,.$$

The reason: $\exp(x)$ can be approximated by $1 + x + x^2/2 + x^3/6$ in this domain, and even by $1 + x + x^2/2$ for smaller values of x.

$$\boxed{1.000000000000000}$$
$$x\ \boxed{\ldots d_i\, 999 d_j \ldots}$$
$$\boxed{x^2/2\ \text{(positive)}}$$

Some bad cases for $x < 10^{-9}$:

x	$\exp x$
$5.999879998200072 \cdot 10^{-10}$	$1.000000000599988\,0\,0^{16}\,431\ldots$
$6.000119998199928 \cdot 10^{-10}$	$1.000000000600011\,9\,9^{16}\,567\ldots$
$1.019999999994798 \cdot 10^{-11}$	$1.000000000010199\,9\,9^{17}\,646\ldots$
$1.039999999994592 \cdot 10^{-11}$	$1.000000000010399\,9\,9^{17}\,625\ldots$
$1.099999999999395 \cdot 10^{-12}$	$1.000000000001099\,9\,9^{20}\,556\ldots$
$1.199999999999280 \cdot 10^{-12}$	$1.000000000001199\,9\,9^{20}\,423\ldots$
$1.199999999999928 \cdot 10^{-13}$	$1.000000000000119\,9\,9^{23}\,423\ldots$
$1.399999999999902 \cdot 10^{-13}$	$1.000000000000139\,9\,9^{23}\,085\ldots$
$1.999999999999980 \cdot 10^{-14}$	$1.000000000000019\,9\,9^{25}\,733\ldots$
$2.999999999999955 \cdot 10^{-14}$	$1.000000000000029\,9\,9^{25}\,099\ldots$
$1.999999999999998 \cdot 10^{-15}$	$1.000000000000001\,9\,9^{28}\,733\ldots$
$3.999999999999992 \cdot 10^{-15}$	$1.000000000000003\,9\,9^{27}\,786\ldots$
$9.999999999999995 \cdot 10^{-16}$	$1.000000000000000\,9\,9^{29}\,666\ldots$

Conclusion

- Very similar to binary arithmetic.
- For the 754r decimal64 format: specific algorithms (same as those for the binary double-precision format, i.e., 64 bits).
- Partial results for exp. Current worst case:

$$\begin{aligned}&\exp(9.407822313572878 \cdot 10^{-2})\\ &= \underbrace{1.098645682066338}_{\text{16 digits}}\,\underbrace{50000000000000000}_{\text{17 digits}}\,278\ldots\end{aligned}$$

- Complete results in a few weeks or months. Then, negative arguments will be tested. Then, other functions.
- Standardization?