Algorithms for binary-to-decimal representation for floating-point numbers

1 Goal

The following method presupposes that the reader has some understanding of the IEEE-754 norm. Our objective is to convert a floating point number from its binary representation (according to the specifications of the IEEE-754 norm) into a decimal representation so that we can easily output its value in human-readable form (such as the `%e`, `%f` or `%g` `printf` formats).

Let x be a floating point number, $x \neq 0$ as if $x = 0$ we can easily detect this in its binary representation. We also suppose that $x > 0$ because is the only variation in representation is the sign bit. Such a floating point number may be represented as $x = m \times 2^E$ where m in an unsigned integer and E is a signed integer. The usual representation of normalized floating point numbers is in the format $x = m' \times 2^{E'}$, where$1 \leq m < 2$. It is quite simple to see that we can obtain the first representation from this usual one, for example by repeatedly multiplying m' by 2 and decrementing E until we obtain $m' = m$ an unsigned integer. We can do the same for denormalized floating-point numbers.

Given this representation $x = m \times 2^E$, we want to convert x to a decimal representation, in particular we want to decompose x in the form $x = n \times 10^F$ where n is an (unsigned) integer, because once we have this, we will be able to easily obtain its decimal representation using integer operations.

The first problem is of course that a floating point number may not have a finite decimal representation, a representation of this sort may carry an error. The second is that we start with a binary representation of the number x, and in this context we cannot use floating point computations in order to compute the closest decimal representation.

The following pages describe an algorithm enabling us to obtain an approximate decimal representation for x with a good relative error and using only integer arithmetic and precomputed values.

2 Bounds for the binary exponent E

We suppose that x may be a `float`, `double` or a `long double`, with a machine representation $x = m' \times 10^{F'}$, $1 \leqslant m' \times 2^{E'} < 2$. First, we wish to extract from this a representation in the form $x = m \times 2^E$, but in particular we want m such that $2^{63} \leqslant m < 2^{64}$. As we said, this implies decrementing the exponent E' of the machine representation of x in order to maintain the value of x.

To obtain the minimum and maximum bounds for E, we have to study the values that can arise when x is a `long double` variable (all `float` and `double` values can be represented in `long double`). According to the IEEE-754 norm, the machine representation of `long double` types has the following bounds :

— $E'_{min} = 1 - (2^{14} - 1) = -2^{14} + 2 = -16382$
— $E'_{max} = 2^{15} - 2 - (2^{14} - 1) = 2^{14} - 1 = 16383$

However, to obtain m such that $2^{63} \leqslant m < 2^{64}$, we have to decrement these exponents by 63 because $m = 2^{63}m'$. Furthermore, in the worst case, for denormalized numbers, we may have $m' = 2^{-63}$. Therefore, in order to obtain $2^{63} \leqslant m < 2^{64}$ in this worst case, we have to multiply m' by 2 an additional 63 times. As such, the real bounds on our binary exponent are now :

— $E_{min} = E'_{min} - 63 - 63 = -16508$
— $E_{max} = E'_{max} - 63 = 16320$

3 Approximating the decimal exponent F

We now have m and E such that $x = m \times 2^E$ and $2^{63} \leqslant m < 2^{64}$, as well as our binary exponent bounds E_{min} and E_{max}. The object of the following sections will be to try to obtain an approximate (and sufficiently precise) representation of x in decimal notation.

We know, for instance that it is possible to write $x = n \times 10^F$ with $10^{18} \leqslant n < 10^{19}$. The idea is to compute an integer $\tilde{F}$ and an unsigned integer $\tilde{n}$ such that $x \approx \tilde{n} \times 10^{\tilde{F}}$. If we can achieve this, using integer arithmetic we can easily compute a human-readable output in our printf function.

Mathematically, we know that x can be represented in decimal as follows : $x = n' \times 10^{F'}$ where $1 \leqslant n' < 10$ (in other words, in scientific notation). We would like to compute a representation as follows : $x = n \times 10^F$ where $10^{18} \leqslant n < 10^{19}$. Of course, we have no guarantee that n is an integer. Our algorithm will therefore return an approximation of n.

According to the above notations, let $F = \lfloor \log_{10} 2^E m \rfloor - 18$. Therefore, from $m \times 2^E = n \times 10^F$, we deduce that :

$$n = \frac{2^E m}{10^F} = \frac{2^E m}{10^{\lfloor log_{10} 2^E m \rfloor - 18}} = \frac{2^E m}{10^{log_{10} 2^E m - 18 + \delta_1}} = \frac{2^E m}{2^E m \times 10^{-18+\delta_1}} = 10^{18-\delta_1}, -1 < \delta_1 \leqslant 0$$

Therefore, we can conclude that $10^{18} \leqslant n < 10^{19}$ given that $18 \leqslant 10^{18} - \delta_1 < 19$.
Therefore $F = \lfloor \log_{10} 2^E m \rfloor - 18$.
To compute F, we can consider the following :

$$\begin{aligned}
F &= \lfloor \log_{10} 2^E m \rfloor - 18 \\
&= \lfloor E \log_{10}(2) + \log_{10} m - \lfloor \log_{10}(2^{63}) \rfloor + \lfloor \log_{10}(2^{63}) \rfloor \rfloor - 18 \\
&= \lfloor E \log_{10}(2) + \log_{10} m - \lfloor \log_{10}(2^{63}) \rfloor \rfloor + \lfloor \log_{10}(2^{63}) \rfloor - 18 \\
&= \lfloor E \log_{10}(2) + \lfloor E \log_{10}(2) \rfloor - \lfloor E \log_{10}(2) \rfloor + \log_{10} m - \lfloor \log_{10}(2^{63}) \rfloor \rfloor + \lfloor \log_{10}(2^{63}) \rfloor - 18 \\
&= \lfloor E \log_{10}(2) \rfloor + \lfloor E \log_{10}(2) - \lfloor E \log_{10}(2) \rfloor + \log_{10} m - \lfloor \log_{10}(2^{63}) \rfloor \rfloor + \lfloor \log_{10}(2^{63}) \rfloor - 18
\end{aligned}$$

We have $\lfloor \log_{10}(2^{63}) \rfloor = 18$, which leads to :

$$\begin{aligned}
F &= \lfloor E \log_{10}(2) \rfloor + \lfloor E \log_{10}(2) - \lfloor E \log_{10}(2) \rfloor + \log_{10} m - \lfloor \log_{10}(2^{63}) \rfloor \rfloor \\
&= \lfloor E \log_{10}(2) \rfloor + \lfloor \alpha \rfloor
\end{aligned}$$

First of all, we need to establish some bounds on the values of $\lfloor \alpha \rfloor$. We have :

$$\begin{aligned}
\alpha &= E \log_{10}(2) - \lfloor E \log_{10}(2) \rfloor + \log_{10} m - \lfloor \log_{10}(2^{63}) \rfloor && \text{with } -1 < \delta_2 \leqslant 0 \\
&= E \log_{10}(2) - E \log_{10}(2) - \delta_2 + \log_{10} m - \lfloor \log_{10}(2^{63}) \rfloor \\
&= -\delta_2 + \log_{10} m - \lfloor \log_{10}(2^{63}) \rfloor \\
&= -\delta_2 + \log_{10} \left(\frac{m}{2^{63}} \right) + \log_{10} 2^{63} - \lfloor \log_{10}(2^{63}) \rfloor
\end{aligned}$$

We have $\log_{10} 2^{63} - \lfloor \log_{10} 2^{63} \rfloor \approx 0,96$.
We also have $1 \leqslant \frac{m}{2^{63}} < 2$, therefore $0 \leqslant \log_{10}(\frac{m}{2^{63}}) < \log_{10} 2 \approx 0,3$.
Therefore $0,96 \leqslant \alpha \leqslant 2,27$, meaning that $\lfloor \alpha \rfloor \in \{0, 1, 2\}$.
To have a good approximation of F, let $\hat{F} = \lfloor E \log_{10}(2) \rfloor + 1$.

To avoid floating point computations, we will compute $\hat{\hat{F}} = \lfloor E \cdot \lfloor 2^{40} \log_{10} 2 \rfloor \cdot 2^{-40} \rfloor + 1$, which we can do using only integer arithmetic, bit-shifts and the precomputed value $\lfloor 2^{40} \log_{10} 2 \rfloor$. We contend that for all possible values of E, $\hat{F} = \hat{\hat{F}}$. In other words : $\forall E, E_{min} \leqslant E \leqslant E_{max}$,

$$\lfloor E \log_{10}(2) \rfloor + 1 = \lfloor E \cdot \lfloor 2^{40} \log_{10} 2 \rfloor \cdot 2^{-40} \rfloor + 1$$

The proof of these equations is done using the "Sollya" tool allowing extended floating point arithmetic. Our script `proof_for_F.sollya` does a case by case check of these roughly 32000 equations, one per possible value of E. Overall, we know that the integer $\hat{F}$ is at worst off of F by 1, this off-by-one error coming from our swapping in 1 for $\lfloor \alpha \rfloor \in \{0, 1, 2\}$.

4 Approximating the decimal mantissa n

We will now use our approximated decimal exponent $\hat{F}$ to compute a decimal mantissa. Mathematically, we can write $x = m \times 2^E = n \times 10^F = \hat{n} \times 10^{\hat{F}}$ with $10^{18} \leqslant n < 10^{19}$. We therefore have :

$$\hat{n} = \frac{2^E m}{10^{\hat{F}}} = \frac{2^E m}{10^{\lfloor E \log_{10} 2 \rfloor + 1}} = \frac{2^E m}{10^{E \log_{10} 2 + \delta_3 + 1}} = \frac{2^E m}{10^{\log_{10}(2^E) + \delta_3 + 1}} = \frac{2^E m}{2^E 10^{\delta_3 + 1}} = \frac{m}{10^{\delta_3 + 1}} = m \cdot 10^{-\delta_3 - 1}$$

In the above equation, $-1 < \delta_3 \leqslant 0$. We can see this if we verify the bounds of $\hat{n}$:

$$10^{17} < 2^{63} \cdot 10^{-1} \leqslant m \cdot 10^{-\delta_3 - 1} < 2^{64} < 10^{20}$$

As we said, our decimal exponent $\hat{F}$ may be off by one compared to the exponent F. These mathematical bounds are therefore coherent. We will now compute an approximation of $\hat{n}$, but how could we do this ?

First of all, we notice that $\hat{n} = m \cdot 2^E \cdot 10^{\hat{F}} = m \times 2^{E-\hat{F}} \times 5^{-\hat{F}}$.

Secondly, we have $\hat{F}_{min} \leqslant \hat{F} \leqslant \hat{F}_{max}$, with :

— $\hat{F}_{min} = \lfloor E_{min} \log_{10} 2 \rfloor + 1 = -4950$

— $\hat{F}_{max} = \lfloor E_{max} \log_{10} 2 \rfloor + 1 = 4913$

We define the following variables :

— $t = 5^{-\hat{F}} \cdot 2^{126-\lfloor \log_2(5^{-\hat{F}}) \rfloor} = 5^{-\hat{F}} \cdot 2^{126-\log_2(5^{-\hat{F}})-\delta_4}$ with $-1 < \delta_4 \leqslant 0$

— $\hat{t} = \lfloor t \rceil = t + \delta_t$, with $-\frac{1}{2} < \delta_t \leqslant \frac{1}{2}$

We have $t = 2^{126-\delta_4}$, so $2^{126} \leqslant t < 2^{127}$ and $2^{126} \leqslant \hat{t} \leqslant 2^{127} < 2^{128}$. Therefore $\hat{t}$ can be represented as a 128 bit unsigned integer.

We have $\hat{t} = t(1 - \frac{\delta_t}{t}) = t(1+\varepsilon_t)$, with $|\varepsilon_t| \leqslant \frac{1/2}{2^{126}} = 2^{-127}$.

We now write $\hat{n} = 2^{E-\hat{F}-126+\lfloor \log_2(5^{-\hat{F}}) \rfloor} \cdot m \cdot t$.

We define $\hat{\hat{n}} = 2^{E-\hat{F}-126+\lfloor \log_2(5^{-\hat{F}}) \rfloor} \cdot m \cdot \hat{t} = \hat{n}(1+\varepsilon_t)$.

We define the function $\tilde{t}(\hat{F}) = t = 5^{-\hat{F}} \cdot 2^{126-\lfloor \log_2(5^{-\hat{F}}) \rfloor}$

We will use this to precompute a table of containing the values of $\hat{t}$ for $\hat{F} \in [\![\hat{F}_{min}, \hat{F}_{max}]\!]$. First of all, let $\Delta = \lfloor \log_2(5^{-\hat{F}}) \rfloor$. We want to be able to compute this at runtime without floating point operations. We therefore write :

$$\Delta = \lfloor \log_2(5^{-\hat{F}}) \rfloor = \lfloor (-\hat{F}) \cdot \log_2 5 \rfloor = \lfloor (-\hat{F}) \cdot \lfloor 2^{40} \cdot \log_2 5 \rfloor \cdot 2^{-40} \rfloor$$

The integer $\lfloor 2^{40} \cdot \log_2 5 \rfloor$ will be a compile time constant. In order to prove the final segment of the equation, we will use the Sollya tool to prove that :

$$\forall \hat{F} \in [\![\hat{F}_{min}, \hat{F}_{max}]\!], \lfloor (-\hat{F}) \cdot \log_2 5 \rfloor = \lfloor (-\hat{F}) \cdot \lfloor 2^{40} \cdot \log_2 5 \rfloor \cdot 2^{-40} \rfloor$$

This proof is executed by the script `proof_for_delta.sollya`. We therefore have :

$$\hat{\hat{n}} = 2^{E-\hat{F}-126+\Delta} \cdot m \cdot \hat{t} = 2^{\sigma} \cdot m \cdot \hat{t}, \text{ where } \sigma = (E - \hat{F} - 126 + \Delta) \in [\![-130, -127]\!]$$

The proof for the bounds on σ is executed by the script `proof_for_sigma.sollya`.

We know that $\forall r \in \mathbb{R}, \lfloor r \rceil = \lfloor r + 0,5 \rfloor$, so for $k \in \mathbb{Z}$,

$$\lfloor r \cdot 2^{-k} \rceil = \lfloor r \cdot 2^{-k} + 0,5 \rfloor = \lfloor (r + 2^{k-1}) \cdot 2^{-k} \rfloor$$

We define $\hat{\hat{\hat{n}}} = \lfloor \hat{\hat{n}} \rceil = \lfloor 2^{\sigma} \cdot m \cdot \hat{t} \rceil = \lfloor 2^{\sigma} \cdot m \cdot \hat{t} + 0,5 \rfloor = \lfloor 2^{\sigma}(m \cdot \hat{t} + 2^{-\sigma-1}) \rfloor$. This computation can be done using only integer arithmetic and bit-shifts. We have $\hat{\hat{n}} = 2^{\sigma} \cdot m \cdot \hat{t} \leqslant 2^{-127} \cdot 2^{127} \cdot m = m \leqslant 2^{64} - 1$. Therefore, $\hat{\hat{\hat{n}}} = \lfloor \hat{\hat{n}} \rceil \leqslant 2^{64} - 1$, so we know that $\hat{\hat{\hat{n}}}$ can be represented as an unsigned 64 bit integer.

5 Quality of the approximation

Mathematically, we have $x = \hat{n} \times 10^{\hat{F}}$, where $10^{\hat{F}}$ is the decimal exponent that we previously computed, and $\hat{n}$ the decimal mantissa that we were approximating in the previous section. There, we defined two successive approximations : $\hat{\hat{n}} = \hat{n}(1+\varepsilon_t)$, and $\hat{\hat{\hat{n}}} = \lfloor \hat{\hat{n}} \rceil$.

The overall approximation is therefore $x \approx \hat{\hat{\hat{n}}} \times 10^{\hat{F}}$.

We can write $\hat{\hat{\hat{n}}} = \lfloor \hat{\hat{n}} \rceil = \hat{\hat{n}} + \delta_5$, with $-\frac{1}{2} < \delta_5 \leqslant \frac{1}{2}$.

Therefore $\hat{\hat{\hat{n}}} = \hat{\hat{n}}(1 + \frac{\delta_5}{\hat{\hat{n}}}) = \hat{\hat{n}}(1 + \hat{\hat{\varepsilon}})$.

We know that $\hat{\hat{n}} = 2^{\sigma} \cdot m \cdot \hat{t} \geqslant 2^{-130} \cdot 2^{63} \cdot 2^{126} = 2^{59}$.

Therefore $|\hat{\hat{\varepsilon}}| = \frac{|\delta_5|}{|\hat{\hat{n}}|} \leqslant \frac{1/2}{2^{59}} = 2^{-60}$.

We also know that $\hat{\hat{n}} = \hat{n}(1+\varepsilon_t)$, with $|\varepsilon_t| \leqslant 2^{-127}$.

We therefore have a final approximation of :

$$\hat{\hat{\hat{n}}} = \hat{n}(1+\varepsilon_t)(1+\hat{\hat{\varepsilon}}) = \hat{n}(1 + \varepsilon_t + \hat{\hat{\varepsilon}} + \varepsilon_t\hat{\hat{\varepsilon}}) = \hat{n}(1 + \hat{\hat{\hat{\varepsilon}}})$$

The relative error is bounded by $|\hat{\hat{\hat{\varepsilon}}}| \leqslant |\hat{\hat{\varepsilon}}| + |\varepsilon_t| + |\varepsilon_t\hat{\hat{\varepsilon}}| = 2^{-60} + 2^{-127} + 2^{-187} \approx 2^{-60}$.

Our final approximation $\hat{x} = \hat{\hat{\hat{n}}} \times 10^{\hat{F}} \approx x$, which we can output in scientific notation.

Given that $\log_{10}(2^{-60}) \approx -18,06$, we have 17 correct digits after the decimal point.

6 Bipartite table of precomputed values

Our previous algorithm required to compute the integer $\hat{t} = \lfloor t \rceil$, where :

$$t = 5^{-\hat{F}} \cdot 2^{126-\lfloor \log_2(5^{-\hat{F}}) \rfloor} = 5^{-\hat{F}} \cdot 2^{126-\log_2(5^{-\hat{F}})-\delta_4} \text{ with } -1 < \delta_4 \leqslant 0$$

The integer $\hat{t}$ is bounded by $2^{126} \leqslant \hat{t} \leqslant 2^{127} < 2^{128}$, so it fits on a 128 bit unsigned integer. Given that $\hat{F}_{min} = -4950$ and $\hat{F}_{max} = 4913$, this requires a table of 9864 precomputed 128 bit unsigned integers. This means a memory footprint of $9864 \times 16 = 157824$ bytes, or roughly 154 KB. We would like to reduce this footprint.

Our method is the following. First of all, by the Euclidean division of $\hat{F}$ by 2^8, we can write :

$$\hat{F} = 2^8 \cdot \hat{F}_h + \hat{F}_l \text{ such that } \hat{F}_h = \lfloor 2^{-8}\hat{F} \rfloor \in [\![-20, 19]\!] \text{ and } \hat{F}_l \in [\![0, 255]\!]$$

If we break down the formula of $\hat{F}$, we can write it as :

$$\begin{aligned}
t &= 5^{-\hat{F}} \cdot 2^{126-\lfloor \log_2(5^{-\hat{F}}) \rfloor} \\
&= 5^{-(2^8 \cdot \hat{F}_h + \hat{F}_l)} \cdot 2^{126-\lfloor \log_2(5^{-\hat{F}}) \rfloor} \\
&= 5^{-2^8 \cdot \hat{F}_h} \cdot 5^{-\hat{F}_l} \cdot 2^{126-\lfloor \log_2(5^{-\hat{F}}) \rfloor - \lfloor \log_2(5^{-2^8 \cdot \hat{F}_h}) \rfloor - \lfloor \log_2(5^{-\hat{F}_l}) \rfloor + \lfloor \log_2(5^{-2^8 \cdot \hat{F}_h}) \rfloor + \lfloor \log_2(5^{-\hat{F}_l}) \rfloor} \\
&= (5^{-2^8 \cdot \hat{F}_h} \cdot 2^{126-\lfloor \log_2(5^{-2^8 \cdot \hat{F}_h}) \rfloor})(5^{-\hat{F}_l} \cdot 2^{126-\lfloor \log_2(5^{-\hat{F}_l}) \rfloor}) \cdot 2^{-126-\lfloor \log_2(5^{-\hat{F}}) \rfloor + \lfloor \log_2(5^{-2^8 \cdot \hat{F}_h}) \rfloor + \lfloor \log_2(5^{-\hat{F}_l}) \rfloor} \\
&= (5^{-2^8 \cdot \hat{F}_h} \cdot 2^{126-\lfloor \log_2(5^{-2^8 \cdot \hat{F}_h}) \rfloor})(5^{-\hat{F}_l} \cdot 2^{126-\lfloor \log_2(5^{-\hat{F}_l}) \rfloor}) \cdot 2^{2-128-\lfloor \log_2(5^{-\hat{F}}) \rfloor + \lfloor \log_2(5^{-2^8 \cdot \hat{F}_h}) \rfloor + \lfloor \log_2(5^{-\hat{F}_l}) \rfloor} \\
&= \tilde{t}_1(\hat{F}_h) \cdot \tilde{t}_2(\hat{F}_l) \cdot 2^{-128} \cdot 2^{\tau(\hat{F}, \hat{F}_h, \hat{F}_l)}
\end{aligned}$$

In the above equations, we have defined the functions :

— $\tilde{t}_1(\hat{F}_h) = (5^{-2^8 \cdot \hat{F}_h} \cdot 2^{126-\lfloor \log_2(5^{-2^8 \cdot \hat{F}_h}) \rfloor})$

— $\tilde{t}_2(\hat{F}_l) = (5^{-\hat{F}_l} \cdot 2^{126-\lfloor \log_2(5^{-\hat{F}_l}) \rfloor})$

— $\tau(\hat{F}, \hat{F}_h, \hat{F}_l) = 2 - \lfloor \log_2(5^{-\hat{F}}) \rfloor + \lfloor \log_2(5^{-2^8 \cdot \hat{F}_h}) \rfloor + \lfloor \log_2(5^{-\hat{F}_l}) \rfloor$

Let $t_1 = \tilde{t}_1(\hat{F}_h)$, $t_2 = \tilde{t}_2(\hat{F}_l)$ and $\tau = \tau(\hat{F}, \hat{F}_h, \hat{F}_l)$. We can precompute the values $\lfloor t_1 \rceil$ and $\lfloor t_2 \rceil$ for all possible values of $\hat{F}_h$ and $\hat{F}_l$. At runtime we can compute $\tau = 2 - \lfloor \log_2(5^{-\hat{F}}) \rfloor + \lfloor \log_2(5^{-2^8 \cdot \hat{F}_h}) \rfloor + \lfloor \log_2(5^{-\hat{F}_l}) \rfloor$ using the same method that we used for δ in the previous sections, only here $2^8 \cdot \hat{F}_h$ can reach values as low as $-20 \times 256 = -5120$, so we extend the range of F from $[\![-4950, 4913]\!]$ to $[\![-5120, 4913]\!]$ in our script `proof_for_delta.sollya`. At runtime we can compute $\hat{\hat{t}} = \lfloor \lfloor t_1 \rceil \cdot \lfloor t_2 \rceil \cdot 2^{-128} \cdot 2^{\tau(\hat{F}, \hat{F}_h, \hat{F}_l)} \rceil$ using only bit-shifts and integer arithmetic.

We can define $\hat{n}' = 2^\sigma \cdot m \cdot \hat{\hat{t}}$ and $\hat{\hat{n}}' = \lfloor \hat{n} \rceil$.

We can compute $\hat{\hat{n}}' = \lfloor 2^\sigma (m \cdot \hat{\hat{t}} + 2^{-\sigma-1}) \rfloor$ with only integer arithmetic and bit-shifts.

Our new approximation is now $x \approx \hat{\hat{n}}' \times 10^{\hat{F}}$.

7 Error analysis for bipartite table algorithm

We prove in `proof_for_T.sollya` that we have $2^{126} \leqslant \hat{\hat{t}} \leqslant 2^{127}$.
We can also prove, in the same way that we did for $\lfloor t \rceil$ that $2^{126} \leqslant \lfloor t_1 \rceil, \lfloor t_2 \rceil \leqslant 2^{127}$.
Let $T = \lfloor t_1 \rceil \cdot \lfloor t_2 \rceil \cdot 2^{-128} \cdot 2^{\tau(\hat{F}, \hat{F}_h, \hat{F}_l)}$ such that $\hat{\hat{t}} = \lfloor T \rceil = T + \delta_T$ with $|\delta_T| \leqslant 1$.
Therefore $\hat{\hat{t}} = T(1 + \frac{\delta_T}{T}) = \varepsilon_T$ with $|\varepsilon_T| \leqslant 2^{126}$, given that $T \geqslant 2^{126}$.
In the same fashion, $\lfloor t_1 \rceil = t1(1 + \varepsilon_{t_1})$, $\lfloor t_2 \rceil = t1(1 + \varepsilon_{t_2})$ with $|\varepsilon_{t_1}|, |\varepsilon_{t_2}| \leqslant 2^{-127}$.
Therefore $T = t_1 \cdot t_2 \cdot 2^{-128+\tau}(1 + \varepsilon_{t_1})(1 + \varepsilon_{t_1}) = t(1 + \varepsilon_{\hat{t}})$.
In the above equation, $(1 + \varepsilon_{\hat{t}}) = (1 + \varepsilon_{t_1})(1 + \varepsilon_{t_1}) = 1 + \varepsilon_{t_1} + \varepsilon_{t_2} + \varepsilon_{t_1}\varepsilon_{t_2}$.
If we distribute the product, we find that$|\varepsilon_{\hat{t}}| \leqslant 2^{-125}$.
Therefore $\hat{\hat{t}} = T(1 + \varepsilon_T) = t(1 + \varepsilon_{\hat{t}})(1 + \varepsilon_T)$.
Again, by distributing, we can find that $\hat{\hat{t}} = t(1 + \tilde{\varepsilon})$ with $|\tilde{\varepsilon}| \leqslant 2^{124}$.
We know that $\hat{\hat{n}}' = 2^\sigma \cdot m \cdot \hat{\hat{t}} \geqslant 2^{130} \cdot 2^{63} \cdot 2^{126} = 2^{59}$.
Therefore, $\hat{\hat{\hat{n}}}' = \lfloor \hat{\hat{n}}' \rceil = \hat{\hat{n}}' + \delta_6$, with $|\delta_6| \leqslant \frac{1}{2}$.

$\hat{\hat{n}}' = \hat{n}'(1 + \frac{1/2}{\hat{\hat{n}}'}) = \hat{n}'(1 + \varepsilon_{\hat{\hat{n}}})$, with $|\varepsilon_{\hat{\hat{n}}}| \leqslant 2^{60}$.

Similarly to $\hat{\hat{n}}$, we compute our final approximation error :

$$\hat{\hat{n}}' = 2^{\sigma} \cdot m \cdot \hat{\hat{t}}(1 + \varepsilon_{\hat{\hat{n}}}) = 2^{\sigma} \cdot m \cdot t(1 + \tilde{\varepsilon})(1 + \varepsilon_{\hat{\hat{n}}}) = \hat{n}(1 + \tilde{\varepsilon} + \varepsilon_{\hat{\hat{n}}} + \varepsilon_{\hat{\hat{n}}}\tilde{\varepsilon})$$

In the above equation, we have $|\tilde{\varepsilon} + \varepsilon_{\hat{\hat{n}}} + \varepsilon_{\hat{\hat{n}}}\tilde{\varepsilon}| \leqslant 2^{-124} + 2^{-60} + 2^{-184} \approx 2^{-60}$.

This is the same error bound as with our previous algorithm. As before, if we write our approximation of the floating point number x in scientific notation, we have 17 correct digits after the decimal point.