Dragonbox: A New Floating-Point Binary-to-Decimal Conversion Algorithm

Junekey Jeon

The Department of Mathematics
University of California, San Diego
USA
j6jeon@ucsd.edu

Abstract

We present a new algorithm for efficiently converting a binary floating-point number into the shortest and correctly rounded decimal representation. The algorithm is based on *Schubfach* algorithm [1] introduced in around 2017-2018, and is also inspired from Grisu [2] and Grisu-Exact [3]. In addition to the core idea of Schubfach, Dragonbox utilizes some Grisu-like ideas to minimize the number of expensive 128-bit $\times$ 64-bit multiplications, at the cost of having more branches and divisions-by-constants. According to our benchmarks, Dragonbox performs better than Ryū, Grisu-Exact, and Schubfach for both IEEE-754 binary32 and binary64 formats.

0. Disclaimer

This paper is not a completely formal writing, and is not intended for publications into peer-reviewed conferences or journals (because I'm not a fan of sacrificing clarity to fit in an artifical page limit). Hence, the paper might contain some alleged claims and/or lack of references.

1. Introduction

Due to recent popularity of JavaScript and JSON, interest on fast and correct algorithm for converting between binary and decimal representations of floating-point numbers has been continuously increasing. As a consequence, many new algorithms have been proposed recently, in spite of the long history of the subject.

[Copyright notice will appear here once 'preprint' option is removed.]

We will assume all floating-point numbers are in either IEEE-754 binary32 or binary64 formats, as these are the most common formats used today.[1][2] We will also focus on the binary-to-decimal conversion in this paper and will not discuss how to do decimal-to-binary conversion. Contrary to one might think, in fact decimal-to-binary conversion and binary-to-decimal conversion are largely asymmetric, because of the asymmetric nature of input and output. In general, for the input side, one needs to deal with wide variety of possible input data, but the form of output is usually definitive. On the other hand, for the output side, the input data has a strict format but one needs to choose between various possibilities of outputs. Floating-point I/O is not an exception. When it comes to decimal-to-binary conversion, which corresponds to the input side, the input data can be usually arbitrarily long so we have to somehow deal with that, but any input data can, if not malformed, usually represent a unique floating-point number. On the other hand, in binary-to-decimal conversion, which corresponds to the output side, the input is a single binary floating-point number but the output can be all decimal numbers which any correct parser will read as the original binary floating-point number. To resolve this ambiguity, Steele and White proposed the following criteria in [4]:[3]

1. **Information preservation**: a correct decimal-to-binary converter must return the original binary floating-point number,
2. **Minimum-length output**: the output decimal significand should be as short as possible, and
3. **Correct rounding**: among all possible shortest outputs, the one that is closest to the true value of the given floating-point number should be chosen.

[1] Details of these formats will be reviewed in Section 2

[2] It should be not so difficult to generalize Dragonbox to similar formats, such as IEEE-754 binary16 or binary128.

[3] To be precise, the criteria given by Steele and White were in terms of the character string generated from the decimal representation. However, we can write those criteria in terms of the decimal representation itself as well.

Notable examples of recently proposed binary-to-decimal conversion algorithms include but not limited to Grisu [2], Errol [5], Ryū [6], and Grisu-Exact [3]. Among these, Errol, Ryū, and Grisu-Exact satisfy all of the above criteria. Grisu does not satisfy all of the criteria, but Grisu3, which can detect its failure to satisfy the criteria, with the fallback into Dragon4 [4], proposed by Steele and White and satisfies all the criteria, is still popular.

Schubfach [1] is another example of those algorithms, developed in around 2017-2018, but it seems that, compared to Ryū, it did not get much attention from the public probably because at that time there was no document explaining details of the algorithm. Nevertheless, the underlying idea of Schubfach is theoretically very appealing and its implementation [7] also seems to outperform that of the other algorithms.

Although Schubfach is already a very tight algorithm, there can be ways to improve its performance further. One possible way might be to eliminate the necessity to perform three 128-bit $\times$ 64-bit multiplications all the time. The core idea of Dragonbox is to achieve this by applying some Grisu-like ideas to Schubfach.

2. IEEE-754 Specifications[4]

Before diving into the details of Dragonbox, let us review IEEE-754 and fix some related notations. For a real number w, by *(binary) floating-point representation* we mean the representation

$$w = (-1)^{\sigma_w} \cdot F_w \cdot 2^{E_w}$$

where $\sigma_w = 0, 1$, $0 \le F_w < 2$, and E_w is an integer. We say the above representation is *normal* if $1 \le F_w < 2$. Of course, there is no normal floating-point representation of 0, while any other real number has a unique normal floating-point representation. If the representation is not normal, we say it is *subnormal*.

IEEE-754 specifications consist of the following rules that define a mapping from the set of fixed-length bit patterns $b_{q-1}b_{n-2} \cdots b_0$ for some q into the real line augmented with some special values:

1. The most-significant bit b_{q-1} is the sign σ_w.
2. The least-significant p-bits $b_{p-1} \cdots b_0$ are for storing the significand F_w, while the remaining $(q-p-1)$-bits are for storing the exponent E_w. We call p the *precision* of the representation.[5]
3. If $q-p-1$ exponent bits are not all-zero nor all-one, the representation is normal. In this case, we compute F_w as

$$F_w = 1 + 2^{-p} \cdot \sum_{k=0}^{p-1} b_k \cdot 2^k$$

and E_w as

$$E_w = -(2^{q-p-2} - 1) + \sum_{k=0}^{q-p-2} b_{p+k} \cdot 2^k.$$

The constant term $2^{q-p-2} - 1$ is called the *bias*, and we denote this value as $E_{\max} := 2^{q-p-2} - 1$.

4. If $q-p-1$ exponent bits are all-zero, the representation is subnormal. In this case, we compute F_w as

$$F_w = 2^{-p} \cdot \sum_{k=0}^{p-1} b_k \cdot 2^k$$

and let $E_w = -(2^{q-p-2} - 2)$. Let us denote this value of E_w as $E_{\min} := -(2^{q-p-2} - 2)$.

5. If $q-p-1$ exponent bits are all-one, the pattern represents either $\pm\infty$ when all of p significand bits are zero, or NaN's (Not-a-Number) otherwise.

When $(q, p) = (32, 23)$, the resulting encoding format is called *binary32*, and when $(q, p) = (64, 52)$, the resulting encoding format is called *binary64*.

For simplicity, let us only consider bit patterns corresponding to positive real numbers from now on. Zeros, infinities, and NaN's should be treated specially, and for negative numbers, we can simply ignore the sign until the final output string is generated. Hence, for example, we do not think of all-zero nor all-one patterns, and especially exponent bits are never all-one. Also, we always assume that the sign bit is 0. With these assumptions, the mapping defined above is one-to-one: each bit pattern corresponds to a unique real number, and no different bit patterns correspond to a same real number.

From now on, by saying $w = F_w \cdot 2^{E_w}$ a *floating-point number* we implicitly assumes that

(1) w is a positive number representable within an IEEE-754 binary format with some q and p, and

(2) F_w and E_w are those obtained from the rules above.

In particular, the representation should be normal ($1 \le F_w < 2$) if $E_w \ne E_{\min}$, and it can be subnormal ($0 \le F_w < 1$) only when $E_w = E_{\min}$. If the representation is normal, we call w a *normal number*, and otherwise, we call w a *subnormal number*.

For a floating-point number $w = F_w \cdot 2^{E_w}$, we define w^- as the greatest floating-point number smaller than w. When w is the minimum possible positive floating-number representable within the specified encoding format, that is, $w = 2^{-p} \cdot 2^{E_{\min}}$, then we define $w^- = 0$. Similarly, we define w^+ as the smallest floating-point number greater than w. Again, if w is the largest possible finite number representable within the format, that is, $w = (2 - 2^{-p})2^{E_{\max}}$, then we define $w^+ := 2^{E_{\max}+1}$.

[4] This section is mostly copied from [3].

[5] Usually, it is actually $p+1$ that is called the precision of the format in other literatures. However, we call p the precision in this paper for simplicity.

In general, it can be shown that

$$w^- = \begin{cases} (F_w - 2^{-p-1})2^{E_w} & \text{if } F_w = 1 \text{ and } E_w \neq E_{\min} \\ (F_w - 2^{-p})2^{E_w} & \text{otherwise} \end{cases}$$

and

$$w^+ = (F_w + 2^{-p})2^{E_w}.$$

We will also use the notations

$$m_w^- := \frac{w^- + w}{2} = \begin{cases} (F_w - 2^{-p-2})2^{E_w} & \text{if } F_w = 1 \text{ and} \\ & E_w \neq E_{\min} \\ (F_w - 2^{-p-1})2^{E_w} & \text{otherwise} \end{cases},$$

$$m_w^+ := \frac{w + w^+}{2} = (F_w + 2^{-p-1})2^{E_w}$$

to denote the midpoints of the intervals $[w^-, w]$, $[w, w^+]$, respectively.

2.1 Rounding Modes

Floating-point calculations are inherently imprecise as the available precision is limited. Hence, it is necessary to round calculational results to make them fit into the precision limit. Specifying how any rounding should be performed means to define for each real number a corresponding floating-point number in a consistent way. IEEE-754 currently defines five rounding modes. We can describe those rounding modes by specifying the inverse image in the real line of each floating-point number w:

1. *Round to nearest, ties to even*: If the LSB (Least Significant Bit) of the significand bits of w is 0, then the inverse image is the closed interval $[m_w^-, m_w^+]$. Otherwise, it is the open interval (m_w^-, m_w^+). This is the default rounding mode in most of the platforms. In fact, IEEE-754 mandates it to be the default mode for binary encodings.
2. *Round to nearest, ties away from zero*: The inverse image of w is the half-open interval $[m_w^-, m_w^+)$. This mode is introduced in the 2008 revision of the IEEE-754 standard. Some platforms and languages, such as the recent standards of the C and C++ languages, do not have the corresponding way of representing this rounding mode.
3. *Round toward* 0: The inverse image of w is the half-open interval $[w, w^+)$.
4. *Round toward* $+\infty$: The inverse image of w is the half-open intervals $(w^-, w]$ if w is positive, and $[w, w^+)$ if w is negative.[6]
5. *Round toward* $-\infty$: The inverse image of w is the half-open intervals $[w, w^+)$ if w is positive, and $(w^-, w]$ is w is negative.

[6] We supposed to deal only with positive numbers, so w here is actually a positive number. The phrases "if w is positive" or "if w is negative" simply mean that the original input is positive or negative, respectively.

Though not included in the IEEE-754 standard, we can think of the following additional rounding modes with their obvious meanings:

- *Round to nearest, ties to odd*
- *Round to nearest, ties toward zero*
- *Round to nearest, ties toward* $+\infty$
- *Round to nearest, ties toward* $-\infty$
- *Round away from* 0

Note that if I is the interval given as the inverse image of w according to a given rounding mode, then a correct decimal-to-binary converter must output w from any numbers in I. Therefore, in order to produce a shortest possible decimal representation of w, we need to search for a number inside I that has the least number of decimal significand digits.

2.2 Notations

From now on, we will assume that a floating-point number w and a specific rounding mode is given so the interval I is defined accordingly. Note that for all cases I is an interval contained in the positive real axis and it avoids 0. We will denote the left and the right endpoints of I as w_L and w_R, respectively. For example, when one of the round-to-nearest rounding mode is specified, $w_L = m_w^-$ and $w_R = m_w^+$. We will also denote the length of I as $\Delta := w_R - w_L$. Note that there are only three possible values of Δ:

1. $\Delta = 2^{E_w - p - 1}$, if $w_L = w^-$, $w_R = w$, $F_w = 1$, and $E_w \neq E_{\min}$,
2. $\Delta = 3 \cdot 2^{E_w - p - 2}$ if $w_L = m_w^-$, $w_R = m_w^+$, $F_w = 1$, and $E_w \neq E_{\min}$, and
3. $\Delta = 2^{E_w - p}$ for all other cases.

We also denote

$$e := E_w - p, \quad f_c := F_w 2^p$$

so that f_c is an integer and

$$\begin{aligned} w &= f_c \cdot 2^e, \\ w^- &= \begin{cases} \left(f_c - \frac{1}{2}\right) \cdot 2^e & \text{if } F_w = 1 \text{ and } E_w \neq E_{\min} \\ (f_c - 1) \cdot 2^e & \text{otherwise} \end{cases}, \\ w^+ &= (f_c + 1) \cdot 2^e, \\ m_w^- &= \begin{cases} \left(f_c - \frac{1}{4}\right) \cdot 2^e & \text{if } F_w = 1 \text{ and } E_w \neq E_{\min} \\ \left(f_c - \frac{1}{2}\right) \cdot 2^e & \text{otherwise} \end{cases}, \\ m_w^+ &= \left(f_c + \frac{1}{2}\right) \cdot 2^e. \end{aligned}$$

With this notation, Δ is one of 2^{e-1}, $3 \cdot 2^{e-2}$, or 2^e.

3. Review of Schubfach

In this section, we will briefly review how Schubfach works. Most of the results are from [1], but we changed the nota-

tions and formulations, and also rewrote the proofs to help understanding the rest of our paper.

The beauty of Schubfach is that, not like Ryū or Grisu-Exact, it does not perform an iterative search to find the shortest decimal representation. Rather, Schubfach finds it with just one trial using the following simple fact:[7]

Proposition 3.1.
Let $k_0 \coloneqq -\lfloor \log_{10} \Delta \rfloor$. *Then*

1. $\left|I \cap 10^{-k_0+1}\mathbb{Z}\right| \le 1$ *and*
2. $\left|I \cap 10^{-k_0}\mathbb{Z}\right| \ge 1$.[8]

Here, $|\cdot|$ *denotes the cardinality of the set and for any* $a \in \mathbb{R}$ *and* $A \subseteq \mathbb{R}$, aA *denotes the set* $\{av \colon v \in A\}$.

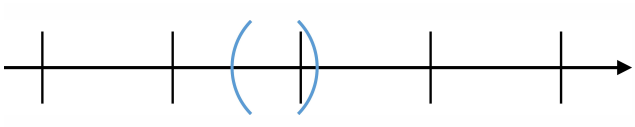

Figure 1. If I is shorter than the unit, then it contains at most one lattice point

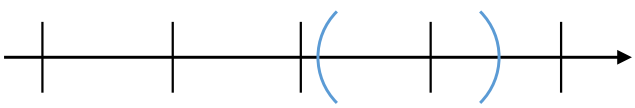

Figure 2. If I is longer than the unit, then it contains at least one lattice point

Proof. By definition of k_0, we have

$$-k_0 \le \log_{10} \Delta < -k_0 + 1,$$

or equivalently,

$$10^{-k_0} \le \Delta < 10^{-k_0+1}. \tag{1}$$

If $\left|I \cap 10^{-k_0+1}\mathbb{Z}\right| > 1$, then it means there are at least two distinct points in I whose distance from each other is 10^{-k_0+1}. Hence, the length of I should be at least 10^{-k_0+1}, or equivalently,

$$\Delta \ge 10^{-k_0+1},$$

which is a contradiction. This shows the first claim.

On the other hand, pick any point $v \in I$, then we know

$$\lfloor 10^{k_0} v \rfloor \le 10^{k_0} v < \lfloor 10^{k_0} v \rfloor + 1.$$

We claim that at least one of $\lfloor 10^{k_0} v \rfloor$ and $\lfloor 10^{k_0} v \rfloor + 1$ is in $10^{k_0} I$. Suppose not, then the left endpoint of $10^{k_0} I$ should lie inside $\left[\lfloor 10^{k_0} v \rfloor, 10^{k_0} v\right]$ and the right endpoint of $10^{k_0} I$ should lie inside $\left[10^{k_0} v, \lfloor 10^{k_0} v \rfloor + 1\right]$. This implies that the length of $10^{k_0} I$ is at most 1, but since $10^{-k_0} \le \Delta$, it follows that $\Delta = 10^{-k_0}$ and $10^{k_0} I = \left(\lfloor 10^{k_0} v \rfloor, \lfloor 10^{k_0} v \rfloor + 1\right)$.

Note that $\Delta = 10^{-k_0}$ is only possible for very rare cases; indeed, since 5 does not appear as a prime factor of Δ (as a rational number), the equality $\Delta = 10^{-k_0}$ can hold only when $k_0 = 0$. Hence, we have $\Delta = 1$, which can hold only when $e = 1$ or $e = 0$ because Δ is one of 2^{e-1}, $3 \cdot 2^{e-2}$, or 2^e, depending on how I is given.[9] However, this implies that $w = f_c \cdot 2^e$ is an integer, but since $w \in I$, we get that $I \cap \mathbb{Z} \ne \emptyset$. This is absurd, because I is an open interval between two consecutive integers. □

It should be noted that the shortest decimal numbers in I are the elements of the intersection $I \cap 10^{-k}\mathbb{Z}$ where k is the smallest integer making the intersection nonempty. Although this sounds somewhat obvious, let us formally prove it. First, we define the number of decimal significand digits of a positive real number v as $\lfloor \log_{10}(10^k v) \rfloor + 1$ where k is the smallest integer such that $10^k v \in \mathbb{Z}$. For example,

- If $v = 1.23$, then $k = 2$ and $\lfloor \log_{10}(10^k v) \rfloor + 1 = 3$,
- If $v = 0.01234$, then $k = 5$ and $\lfloor \log_{10}(10^k v) \rfloor + 1 = 5$, and
- If $v = 1200$, then $k = -2$ and $\lfloor \log_{10}(10^k v) \rfloor + 1 = 2$.

Proposition 3.2.
The set $I \cap 10^{-k}\mathbb{Z}$, *where* k *is the smallest integer making the intersection nonempty, is precisely the set of elements in* I *with the smallest number of decimal significand digits..*

Proof. By the assumption on k, we know that $I \cap 10^{-k}\mathbb{Z}$ is not empty while $I \cap 10^{-k+1}\mathbb{Z}$ is empty. Equivalently, $10^k I \cap \mathbb{Z}$ is not empty while $10^{k-1} I \cap \mathbb{Z}$ is empty. Since I is an interval, $10^k I \cap \mathbb{Z} = \{m, m+1, \cdots, M-1, M\}$ for some integers $m, M \in \mathbb{Z}$. Since $10^{k-1} I \cap \mathbb{Z}$ is empty, there is no multiple of 10 among $m, \cdots, M$. Hence, we get $\lfloor \log_{10} m \rfloor = \lfloor \log_{10} M \rfloor$; otherwise, we have

$$\begin{aligned} \log_{10} m &< \lfloor \log_{10} m \rfloor + 1 \\ &\le \lfloor \log_{10} M \rfloor \le \log_{10} M, \end{aligned}$$

thus

$$m < 10^{\lfloor \log_{10} m \rfloor + 1} \le M,$$

which contradicts to that there is no multiple of 10 among $m, \cdots, M$. Note that for any v in the set

$$I \cap 10^{-k}\mathbb{Z} = \left\{10^{-k} m, \cdots, 10^{-k} M\right\},$$

k is the smallest integer such that $10^k v$ is an integer, thus all such v have $\lfloor \log_{10} m \rfloor + 1$ decimal significand digits.

[7] One might regard this proposition as a form of the pigeonhole principle. In fact, the name *Schubfach* is coming from the German name of the pigeonhole principle, *Schubfachprinzip*, meaning "drawer principle".

[8] In fact, we show in the proof that for any $v \in I$, at least one of $\lfloor 10^{k_0} v \rfloor$ and $\lfloor 10^{k_0} v \rfloor + 1$ should be in $10^{k_0} I$.

[9] In fact, since I is an open interval, the first case is impossible, so we have $e = 0$.

Now, let us show that $\lfloor \log_{10} m \rfloor + 1$ is the minimum possible number of decimal significand digits. We first claim that

$$\lfloor \log_{10}(m-1) \rfloor = \lfloor \log_{10} m \rfloor$$

if $m \neq 1$. Indeed, if not, then we have

$$\begin{aligned} \log_{10}(m-1) &< \lfloor \log_{10}(m-1) \rfloor + 1 \\ &\leq \lfloor \log_{10} m \rfloor \leq \log_{10} m, \end{aligned}$$

thus

$$m - 1 < 10^{\lfloor \log_{10}(m-1) \rfloor + 1} \leq m.$$

Since $10^{\lfloor \log_{10}(m-1) \rfloor + 1}$ is an integer, we must have $m = 10^{\lfloor \log_{10}(m-1) \rfloor + 1}$, which contradicts to that m is not a multiple of 10. This shows the claim.

Next, note that for any $v \in I$ such that there exists $l \in \mathbb{Z}$ with $10^l v \in \mathbb{Z}$, we have $l \geq k$ because of how we chose k. If $l = k$, then $10^l v$ is one of $m, \cdots, M$, so we may assume $l > k$. Note also that we may assume $m \neq 1$, because if $m = 1$ then the number of decimal significand digits of elements in $I \cap 10^{-k}\mathbb{Z}$ is 1, which is of course the smallest possible number of decimal significand digits. Now, since we have

$$\begin{aligned} \left\lfloor \log_{10}(10^l v) \right\rfloor &= \left\lfloor \log_{10}(10^k v) \right\rfloor + (l-k) \\ &\geq \left\lfloor \log_{10}(10^k v) \right\rfloor + 1, \end{aligned}$$

it suffices to show that $\left\lfloor \log_{10}(10^k v) \right\rfloor \geq \lfloor \log_{10} m \rfloor$. This inequality actually follows directly from our previous claim $\lfloor \log_{10}(m-1) \rfloor = \lfloor \log_{10} m \rfloor$; indeed, as $10^{-k}(m-1)$ is not an element of I, we should have $v > 10^{-k}(m-1)$, or equivalently, $10^k v > m - 1$, which implies

$$\left\lfloor \log_{10}(10^k v) \right\rfloor \geq \lfloor \log_{10}(m-1) \rfloor = \lfloor \log_{10} m \rfloor .$$

□

Since we have the following *chain property*

$$I \cap 10^{-k+1}\mathbb{Z} \subseteq I \cap 10^{-k}\mathbb{Z}$$

for all $k \in \mathbb{Z}$, we get the following:

Corollary 3.3.
Let $k_0 := -\lfloor \log_{10} \Delta \rfloor$. Then:

1. *If $I \cap 10^{-k_0+1}\mathbb{Z}$ is not empty, then the unique element in it has the smallest number of decimal significand digits in I.*
2. *Otherwise, elements in $I \cap 10^{-k_0}\mathbb{Z}$ have the smallest number of decimal significand digits.*

Proof. Suppose first that $I \cap 10^{-k_0+1}\mathbb{Z}$ is not empty. Let $l \in \mathbb{Z}$ be the smallest integer such that $I \cap 10^{-l}\mathbb{Z}$ is not empty, then by the chain property, we know

$$\emptyset \neq I \cap 10^{-l}\mathbb{Z} \subseteq I \cap 10^{-k_0+1}\mathbb{Z},$$

but since $I \cap 10^{-k_0+1}\mathbb{Z}$ can have at most 1 element by Proposition 3.1, it follows that the unique element of $I \cap 10^{-k_0+1}\mathbb{Z}$ is the unique element of $I \cap 10^{-l}\mathbb{Z}$. Hence, that unique element has the smallest number of decimal significand digits in I by Proposition 3.2.

Next, suppose that $I \cap 10^{-k_0+1}\mathbb{Z} = \emptyset$. Then again by the chain property, k_0 must be the smallest integer such that $I \cap 10^{-k_0}\mathbb{Z}$ is not empty, so the result follows from Proposition 3.2. □

Note that, since we always have $w \in I$, so given that $I \cap 10^{-k}\mathbb{Z}$ is nonempty for some $k \in \mathbb{Z}$, then at least one of $\left\lfloor 10^k w \right\rfloor 10^{-k}$ and $\left(\left\lfloor 10^k w \right\rfloor + 1\right) 10^{-k}$ must be in $I \cap 10^{-k}\mathbb{Z}$. More precisely, pick any $v \in I \cap 10^{-k}\mathbb{Z}$, then if $v \leq w$, then $\left\lfloor 10^k w \right\rfloor 10^{-k}$ is in $I \cap 10^{-k}\mathbb{Z}$ since $\left\lfloor 10^k w \right\rfloor$ is the largest integer smaller than or equal to $10^k w$, so it should lie in between $10^k v$ and $10^k w$. Similarly, if $v > w$, then $\left(\left\lfloor 10^k w \right\rfloor + 1\right) 10^{-k}$ is in $I \cap 10^{-k}\mathbb{Z}$ since $\left\lfloor 10^k w \right\rfloor + 1$ is the smallest integer strictly greater than $10^k w$, so it should lie in between $10^k w$ and $10^k v$. This leads us to the following strategy of finding the shortest decimal representation of w assuming round-to-nearest, which is the basic skeleton of Schubfach:

Algorithm 3.4 (Skeleton of Schubfach)**.**

1. Compute $k_0 := -\lfloor \log_{10} \Delta \rfloor$.
2. Compute $\left\lfloor 10^{k_0-1} w \right\rfloor$ and $\left\lfloor 10^{k_0-1} w \right\rfloor + 1$. If one of them (and only one of them) belongs to $10^{k_0-1} I$, then call that number s. In this case, $10^{-k_0+1} s$ is the unique number in I with the smallest number of decimal significand digits. However, s might contain trailing decimal zeros; that is, it might be a multiple of a power of 10 as $I \cap 10^{-l}\mathbb{Z}$ might be nonempty for some $l < k_0 - 1$. Thus, let d be the greatest integer such that 10^d divides s, then $\frac{s}{10^d} \times 10^{d-k_0+1}$ is the unique shortest decimal representation of w.
3. Otherwise, we compute $\left\lfloor 10^{k_0} w \right\rfloor$ and $\left\lfloor 10^{k_0} w \right\rfloor + 1$. Then at least one of them must be in $10^{k_0} I$, and if only one of them is inside I, call that number s. In this case, $10^{-k_0} s$ is the number closest to w in I with the smallest number of decimal significand digits. Since we assumed that $I \cap 10^{-k_0+1}\mathbb{Z}$ is empty, s is never divisible by 10 so there is no trailing decimal zeros and $s \times 10^{-k_0}$ is the correctly rounded shortest decimal representation of w.
4. If both $\left\lfloor 10^{k_0} w \right\rfloor$ and $\left\lfloor 10^{k_0} w \right\rfloor + 1$ are inside $10^{k_0} I$, choose the one that is closer to $10^{k_0} w$. When the distances from $10^{k_0} w$ to those numbers are the same, break the tie according to a given rule.[10] Call the chosen number s, then again s cannot have any trailing decimal zeros and $s \times 10^{-k_0}$ is the correctly rounded shortest decimal representation of w.

[10] The most common rule is to choose the even one, but we can consider other rules as well.

Based on the above strategy, the details of Schubfach include the following:

- How to efficiently compute $\lfloor \log_{10} \Delta \rfloor$?
- How to efficiently compute $\lfloor 10^{k_0-1} w \rfloor$, $\lfloor 10^{k_0-1} w \rfloor + 1$, $\lfloor 10^{k_0} w \rfloor$ and $\lfloor 10^{k_0} w \rfloor + 1$?
- How to efficiently compare these numbers to the endpoints of $10^{k_0-1} I$ or $10^{k_0} I$?

Similar to Ryū and Grisu-Exact, Schubfach uses a table of precomputed binary digits of powers of 10 in order to accomplish the second item. In addition to that, it uses an ingenious rounding trick which makes the third item trivial.[11] More precisely, after computing k_0, Schubfach computes approximations of $10^{k_0} w_L$ and $10^{k_0} w_R$ along with that of $10^{k_0} w$, with the aforementioned rounding rule applied, and the construction of the rounding rule ensures that we can just compare with the computed approximations of $10^{k_0} w_L$ and $10^{k_0} w_R$ in order to deduce if a given number is in the interval or not.

However, even with the precomputed cache, computing the approximate multiplications $w_L \times 10^{k_0}$, $w_R \times 10^{k_0}$, and $w \times 10^{k_0}$, is not cheap, because it requires several 64-bit multiplications, which, for typical modern x86 machines, are a lot slower than many other instructions. (We will review how these approximate multiplications can be done in Section 5.1.5.) The core idea of Dragonbox is, thus, on how we can avoid some of these multiplications.

4. Computation of $\lfloor nx \rfloor$ and Related Tricks

In this section, we will state and prove some statements regarding the computation of $\lfloor nx \rfloor$ for a positive real number x and an integer n, and also other related things. The correctness analysis of Dragonbox is crucially dependent on results from this section, but since this section has little to do with the actual implementation, readers who are only interested in the implementation of the algorithm might skip it entirely and go directly to Section 5. However, the results in this section can be interesting in their own because they are very closely related to (and generalize and unify some of) many previously known optimization techniques for integer division pioneered by Granlund and Montgomery [8]. Also, the author believes that these results might be very handy to those who are interested in developing new algorithms related to not only floating-point formatting/parsing but also integer formatting/parsing and division.

[11] To be honest, I did not look at this rounding trick carefully, and do not fully understand how it works. Dragonbox does not rely on this trick, so it should be irrelevant for the rest of the paper. However, it might be that we can still possibly apply the trick also to Dragonbox so that we can make it even faster.

4.1 Computation of $\lfloor nx \rfloor$

In this subsection, we will obtain a necessary and sufficient condition for ensuring

$$\lfloor nx \rfloor = \lfloor n\xi \rfloor$$

for all n in a prescribed range where x and ξ are given positive real numbers. In practice, x will be a number given by the problem, so that the left-hand side $\lfloor nx \rfloor$ is what we want to compute, while we choose an appropriate ξ depending on x and the range of n (but not on individual n's) making the computation of the right-hand side $\lfloor n\xi \rfloor$ cheap.

For example, we may want to choose $\xi = \frac{m}{2^k}$ for some positive integers k and m. In this case, the computation of $\lfloor n\xi \rfloor$ is simply one integer multiplication followed by one right-shift. In fact, $\xi = \frac{m}{2^k}$ is virtually the only possibility that we might care in practice, but allowing ξ to be any positive real number is convenient for the purpose of proving some other results.

A key concept we will make use of is the following.

Definition 4.1.
Given a real number $x \in \mathbb{R}$, we say a rational number $\frac{p}{q}$ [12] is a *best rational approximation of x from below*, if $\frac{p}{q} \le x$ and for any rational number $\frac{a}{b} \le x$ with $b \le q$, we have

$$\frac{a}{b} \le \frac{p}{q} \le x.$$

Similarly, we say $\frac{p}{q}$ is a *best rational approximation of x from above* if $\frac{p}{q} \ge x$ and for any rational number $\frac{a}{b} \ge x$ with $b \le q$, we have

$$\frac{a}{b} \ge \frac{p}{q} \ge x.$$

For example, given $x = \frac{1}{3}$, 0 is a best rational approximation from below and $\frac{1}{2}$ is a best rational approximation from above. As another example, given $x = \frac{5}{17}$, $\frac{1}{4}$ is a best rational approximation from below and $\frac{1}{3}$ is a best rational approximation from above. Note that in this case $\frac{1}{4}$ is *not* a best rational approximation if we consider both directions of approach at the same time, because $\frac{1}{3}$ is a better rational approximation.

Whenever $\frac{p}{q}$ is a best rational approximation of x from below, we must have $p = \lfloor qx \rfloor$, since if $p > \lfloor qx \rfloor$ then $\frac{p}{q} \le x$ is violated, and if $p < \lfloor qx \rfloor$ then $\frac{p+1}{q}$ is a strictly better approximation of x that is still below x. Similarly, if $\frac{p}{q}$ is a best rational approximation of x from above, then we must have $p = \lceil qx \rceil$.

The concept of best rational approximations from below/above directly leads us to the following proposition, which is a generalization of the classical result of Granlund-Montgomery [8] (Theorem 4.2), and also a generalization of a result by Warren [9] and Lemire et al. [10] (Theorem 1):

[12] Whenever we say "a rational number $\frac{p}{q}$", we always assume that $\frac{p}{q}$ is in its reduced form, that is, $q \in \mathbb{Z}_{>0}$ and $p \in \mathbb{Z}$ with $\gcd(p, q) = 1$.

Theorem 4.2.
Let x be a positive real number and $n_{\max}$ a positive integer. Then for a positive real number ξ, we have the followings.

1. *If $x = \frac{p}{q}$ is a rational number with $q \le n_{\max}$, then we have*
$$\lfloor nx \rfloor = \lfloor n\xi \rfloor$$
for all $n = 1, \cdots, n_{\max}$ if and only if
$$x \le \xi < x + \frac{1}{vq}$$
holds, where v is the greatest integer such that $vp \equiv -1 \pmod q$ and $v \le n_{\max}$.
2. *If x is either an irrational number or a rational number with the denominator strictly greater than $n_{\max}$, then we have*
$$\lfloor nx \rfloor = \lfloor n\xi \rfloor$$
for all $n = 1, \cdots, n_{\max}$ if and only if
$$\frac{p_*}{q_*} \le \xi < \frac{p^*}{q^*}$$
holds, where $\frac{p_*}{q_*}, \frac{p^*}{q^*}$ are the best rational approximations of x from below and above, respectively, with the largest denominators $q_*, q^* \le n_{\max}$.

Proof. Note that
$$\lfloor nx \rfloor = \lfloor n\xi \rfloor$$
is equivalent to
$$\frac{\lfloor nx \rfloor}{n} \le \xi < \frac{\lfloor nx \rfloor + 1}{n},$$
so having $\lfloor nx \rfloor = \lfloor n\xi \rfloor$ for all $n = 1, \cdots, n_{\max}$ is equivalent to
$$\max_{n=1,\cdots,n_{\max}} \frac{\lfloor nx \rfloor}{n} \le \xi < \min_{n=1,\cdots,n_{\max}} \frac{\lfloor nx \rfloor + 1}{n}. \tag{2}$$
Clearly, the left-hand side is precisely $\frac{p_*}{q_*}$, because if not, then pick the smallest n achieving the maximum, then $\frac{\lfloor nx \rfloor}{n}$ must be a best rational approximation of x from below, thus we get $n \le q_*$, contradicting to that $\frac{p_*}{q_*}$ is a best rational approximation from below. Of course, if $x = \frac{p}{q}$ with $q \le n_{\max}$, then clearly $\frac{p_*}{q_*} = \frac{p^*}{q^*} = x$, so this proves the left-half of both *1* and *2*.

When x is irrational or is rational whose denominator is strictly bigger than $n_{\max}$, then the right-hand side of (2) is equal to $\frac{p^*}{q^*}$ by the same reason, which shows the right-half of *2*.

Now it remains to show that, assuming that $x = \frac{p}{q}$ with $q \le n_{\max}$,
$$\min_{n=1,\cdots,n_{\max}} \frac{\lfloor nx \rfloor + 1}{n} = \frac{p}{q} + \frac{1}{vq}$$
where v is the greatest integer such that $vp \equiv -1 \pmod q$ and $v \le n_{\max}$. First, note that such v must exist because in the range $1, \cdots, q-1$ there must be a modular inverse of $-p$, say b so that $bp \equiv -1 \pmod q$ holds, which means that v is given by $v = b + \left\lfloor \frac{n_{\max} - b}{q} \right\rfloor q$.

Next, observe that, let $r = np - \left\lfloor \frac{np}{q} \right\rfloor q$ be the remainder of np divided by q, then
$$\frac{\lfloor nx \rfloor + 1}{n} = \frac{(np - r)/q + 1}{n} = \frac{p}{q} + \frac{q-r}{nq},$$
so it suffices to show that $n = v$ is a minimizer of $\frac{q-r}{n}$.

Suppose not, so that for some $n \le n_{\max}$ we have
$$\frac{q-r}{n} < \frac{1}{v},$$
or equivalently,
$$n > v(q-r).$$
Clearly, to have this inequality r must be strictly smaller than $q-1$, because v is obviously the minimizer of $\frac{q-r}{n}$ among n's such that $r = q-1$.

Note that $vp(q-r) \equiv -(q-r) \equiv r \equiv np \pmod q$, and since p and q are coprime, we have $n \equiv v(q-r) \pmod q$. Therefore, there exists a positive integer e such that
$$n = v(q-r) + eq.$$
However, this implies
$$n_{\max} \ge n = v(q-r) + eq \ge v + q$$
which is absurd because $v + q$ is strictly bigger than v and still satisfies $(v+q)p \equiv -1 \pmod q$. □

With an almost same proof, we can prove the following as well:

Theorem 4.3.
With the settings of Theorem 4.2, we have the followings.

1. *If $x = \frac{p}{q}$ is a rational number with $q \le n_{\max}$, then we have*
$$\lceil nx \rceil = \lceil n\xi \rceil$$
for all $n = 1, \cdots, n_{\max}$ if and only if
$$x - \frac{1}{vq} < \xi \le x$$
holds, where v is the greatest integer such that $vp \equiv -1 \pmod q$ and $v \le n_{\max}$.
2. *If x is either an irrational number or a rational number with the denominator strictly greater than $n_{\max}$, then we have*
$$\lceil nx \rceil = \lceil n\xi \rceil$$
for all $n = 1, \cdots, n_{\max}$ if and only if
$$\frac{p_*}{q_*} < \xi \le \frac{p^*}{q^*}$$

holds, where $\frac{p_*}{q_*}, \frac{p^*}{q^*}$ are the best rational approximations of x from below and above, respectively, with the largest denominators $q_*, q^* \le n_{\max}$.

Proof. Omit. □

Note that finding an admissible ξ boils down to finding $\frac{p_*}{q_*}$ and $\frac{p^*}{q^*}$, or v. In fact, finding v can be also formulated in terms of finding a best rational approximation from above. Indeed, for the modular inverse $b \in \{1, \cdots, q-1\}$ of $-p$ with respect to q, $\frac{\lfloor bp/q \rfloor + 1}{b}$ is the best rational approximation of $\frac{p}{q}$ from above among all rational numbers with the denominator strictly less than q. To see why, proceed just like we did for the second half of the first part of the above theorem, then one can see that if $\frac{\lceil tp/q \rceil}{t} = \frac{\lfloor tp/q \rfloor + 1}{t}$ for some $t \le q-1$ were a strictly better approximation than $\frac{\lfloor bp/q \rfloor + 1}{b}$, then we should have $t \ge b+q$ which contradicts to $t \le q-1$. Then we get v by computing

$$v = b + \left\lfloor \frac{n_{\max} - b}{q} \right\rfloor q.$$

What makes Theorem 4.2 and Theorem 4.3 so useful in practice, especially the second parts of them, is that there indeed exists a very efficient algorithm of finding the best rational approximations from below and above (Algorithm C.9), which is based on the concept of *continued fractions*. Although the theory of continued fractions seems to be an extremely useful tool for all of these things and related topics, we tried to minimize its exposure in the main part of this paper to lessen the divergence from the main algorithm that will be described in Section 5. Instead, relevant backgrounds from the theory of continued fractions are collected in Appendix C.

It is worth mentioning that when x is rational, the referred algorithm of finding best rational approximations is essentially equivalent to the Euclid algorithm applied to the numerator and the denominator of x. In fact, it turns out that even the improved min-max Euclid algorithm from [3], which is an improved and corrected version of a key theorem from [6], follows directly from the theory of continued fractions. See the remark after Algorithm C.13 for further explanation.

4.2 Correspondence of ordering of the fractional parts

In this subsection, we will prove a lemma comparing the ordering of the fractional parts $nx - \lfloor nx \rfloor$ with that of $n\xi - \lfloor n\xi \rfloor$, given that ξ is a good enough approximation of x in the sense of Theorem 4.2. Roughly speaking, the correspondence $(nx - \lfloor nx \rfloor) \leftrightarrow (n\xi - \lfloor n\xi \rfloor)$ indeed preserves the order, modulo the fact that different n's with the same value of $nx - \lfloor nx \rfloor$ might have different values of $n\xi - \lfloor n\xi \rfloor$, and vice versa. The precise statement is the following.

Theorem 4.4.
Let x, ξ be positive real numbers and $n_{\max}$ be a positive integer such that

$$\lfloor nx \rfloor = \lfloor n\xi \rfloor$$

holds for all $n = 1, \cdots, n_{\max}$. Then we have the followings.

1. *If $x = \frac{p}{q}$ is a rational number with $q \le n_{\max}$, then for any $n_1, n_2 = 1, \cdots, n_{\max}$,*

$$n_1 x - \lfloor n_1 x \rfloor < n_2 x - \lfloor n_2 x \rfloor$$

or

$$n_1 x - \lfloor n_1 x \rfloor = n_2 x - \lfloor n_2 x \rfloor \quad \text{and} \quad n_1 < n_2$$

implies

$$n_1 \xi - \lfloor n_1 \xi \rfloor \le n_2 \xi - \lfloor n_2 \xi \rfloor,$$

with possibly the equality only when $\xi = x$.

2. *If x is either an irrational number or a rational number with the denominator strictly greater than $n_{\max}$, then for any $n_1, n_2 = 1, \cdots, n_{\max}$,*

$$n_1 x - \lfloor n_1 x \rfloor < n_2 x - \lfloor n_2 x \rfloor$$

implies

$$n_1 \xi - \lfloor n_1 \xi \rfloor \le n_2 \xi - \lfloor n_2 \xi \rfloor,$$

with possibly the equality only when $n_1 < n_2$ and $\xi = \frac{p_*}{q_*}$ is the best rational approximation of x from below with the largest denominator $q_* \le n_{\max}$.

Note that when x is either irrational or rational with the denominator strictly greater than $n_{\max}$, then the mapping $n \mapsto nx - \lfloor nx \rfloor$ is one-to-one. Indeed, if $n_1 x - \lfloor n_1 x \rfloor = n_2 x - \lfloor n_2 x \rfloor$ holds with $n_1 \ne n_2$, then

$$x = \frac{\lfloor n_2 x \rfloor - \lfloor n_1 x \rfloor}{n_2 - n_1},$$

which is absurd.

Therefore, the theorem establishes the equivalence of two orderings on $\{1, \cdots, n_{\max}\}$ induced from the embeddings

$$n \mapsto (nx - \lfloor nx \rfloor, n) \quad \text{and}$$
$$n \mapsto (n\xi - \lfloor n\xi \rfloor, n)$$

into $[0,1) \times \{1, \cdots, n_{\max}\}$ endowed with the lexicographic ordering.

Proof. 1. We want to show the inequality

$$(n_1 - n_2)\xi \le \lfloor n_1 \xi \rfloor - \lfloor n_2 \xi \rfloor = \lfloor n_1 x \rfloor - \lfloor n_2 x \rfloor,$$

which can be rewritten as

$$(n_1 - n_2)(\xi - x) \le (n_2 x - \lfloor n_2 x \rfloor) - (n_1 x - \lfloor n_1 x \rfloor).$$

Since $x = \frac{p}{q}$ and $q \le n_{\max}$, Theorem 4.2 implies

$$x \le \xi < x + \frac{1}{vq}$$

where v is the largest integer such that $vp \equiv -1 \pmod{q}$ and $v \leq n_{\max}$. In particular, the inequality we want to show is trivial if $n_1 < n_2$ or $x = \xi$, so assume $n_1 > n_2$ and $x \neq \xi$, then we have

$$(n_1 - n_2)(\xi - x) < \frac{n_1 - n_2}{vq}.$$

Let r_1, r_2 be the remainders of $n_1 p$ divided by q and $n_2 p$ divided by q, respectively, then it suffices to show

$$\begin{aligned}(n_2 x - \lfloor n_2 x \rfloor) - (n_1 x - \lfloor n_1 x \rfloor) \\ = \frac{r_2 - r_1}{q} \geq \frac{n_1 - n_2}{vq},\end{aligned}$$

or equivalently,

$$n_1 - n_2 \leq v(r_2 - r_1),$$

assuming $r_2 > r_1$. Suppose not, so assume $n_1 - n_2 > v(r_2 - r_1)$. Note that

$$v(r_2 - r_1)p \equiv r_1 - r_2 \equiv (n_1 - n_2)p \pmod{q},$$

and since p and q are coprime, we get $v(r_2 - r_1) \equiv n_1 - n_2 \pmod{q}$, thus there exists a positive integer e such that

$$n_1 - n_2 = v(r_2 - r_1) + eq.$$

Since $r_2 > r_1$, this implies

$$n_{\max} \geq n_1 = v(r_2 - r_1) + eq + n_2 > v + q$$

which contradicts to the definition of v. Therefore, we get the desired inequality.
The equality can hold only when

$$(n_1 - n_2)(\xi - x) = \frac{n_1 - n_2}{vq},$$

which can happen only when $\xi = x$ as $n_1 \neq n_2$.

2. Let $\frac{p_*}{q_*}$ be the best rational approximation of x from below with the largest denominator $q_* \leq n_{\max}$. Then for any $n = 1, \cdots, n_{\max}$, we know

$$\frac{\lfloor nx \rfloor}{n} \leq \frac{p_*}{q_*} < x,$$

thus

$$\lfloor nx \rfloor \leq \frac{np_*}{q_*} < nx,$$

which implies $\left\lfloor \frac{np_*}{q_*} \right\rfloor = \lfloor nx \rfloor = \lfloor n\xi \rfloor$. Note that from $n_1 x - \lfloor n_1 x \rfloor < n_2 x - \lfloor n_2 x \rfloor$, we get

$$\begin{aligned}n_1 \left(x - \frac{p_*}{q_*} \right) + \frac{n_1 p_*}{q_*} - \left\lfloor \frac{n_1 p_*}{q_*} \right\rfloor \\ < n_2 \left(x - \frac{p_*}{q_*} \right) + \frac{n_2 p_*}{q_*} - \left\lfloor \frac{n_2 p_*}{q_*} \right\rfloor.\end{aligned}$$

We claim that

$$\frac{n_1 p_*}{q_*} - \left\lfloor \frac{n_1 p_*}{q_*} \right\rfloor \leq \frac{n_2 p_*}{q_*} - \left\lfloor \frac{n_2 p_*}{q_*} \right\rfloor.$$

This actually follows directly from the first part of the theorem, because if this is not the case, then the first part applied to $x \leftarrow \frac{p_*}{q_*}$ and $\xi \leftarrow x$ implies

$$n_1 x - \lfloor n_1 x \rfloor \geq n_2 x - \lfloor n_2 x \rfloor,$$

directly contradicting to the assumption.
Now, if

$$\frac{n_1 p_*}{q_*} - \left\lfloor \frac{n_1 p_*}{q_*} \right\rfloor < \frac{n_2 p_*}{q_*} - \left\lfloor \frac{n_2 p_*}{q_*} \right\rfloor$$

holds, then again we can apply the first part with $x \leftarrow \frac{p_*}{q_*}$ and $\xi \leftarrow \xi$ to conclude

$$n_1 \xi - \lfloor n_1 \xi \rfloor < n_2 \xi - \lfloor n_2 \xi \rfloor;$$

note that the equality cannot hold in this case. Therefore, we only need to consider the case

$$\frac{n_1 p_*}{q_*} - \left\lfloor \frac{n_1 p_*}{q_*} \right\rfloor = \frac{n_2 p_*}{q_*} - \left\lfloor \frac{n_2 p_*}{q_*} \right\rfloor.$$

Clearly, in this case we should have $n_1 < n_2$, thus we can again apply the first part to conclude

$$n_1 \xi - \lfloor n_1 \xi \rfloor \leq n_2 \xi - \lfloor n_2 \xi \rfloor,$$

with possibly the equality when $\xi = \frac{p_*}{q_*}$.

□

An almost same proof shows the following:

Theorem 4.5.
Let x, ξ be positive real numbers and $n_{\max}$ be a positive integer such that

$$\lceil nx \rceil = \lceil n\xi \rceil$$

holds for all $n = 1, \cdots, n_{\max}$. Then we have the followings.

1. *If $x = \frac{p}{q}$ is a rational number with $q \leq n_{\max}$, then for any $n_1, n_2 = 1, \cdots, n_{\max}$,*

$$\lceil n_1 x \rceil - n_1 x < \lceil n_2 x \rceil - n_2 x$$

or

$$\lceil n_1 x \rceil - n_1 x = \lceil n_2 x \rceil - n_2 x \quad \textit{and} \quad n_1 < n_2$$

implies

$$\lceil n_1 \xi \rceil - n_1 \xi \leq \lceil n_2 \xi \rceil - n_2 \xi,$$

with possibly the equality only when $\xi = x$.

2. *If x is either an irrational number or a rational number with the denominator strictly greater than $n_{\max}$, then for any $n_1, n_2 = 1, \cdots, n_{\max}$,*

$$\lceil n_1 x \rceil - n_1 x < \lceil n_2 x \rceil - n_2 x$$

implies

$$\lceil n_1 \xi \rceil - n_1 \xi \leq \lceil n_2 \xi \rceil - n_2 \xi,$$

with possibly the equality only when $n_1 < n_2$ and $\xi = \frac{p^*}{q^*}$ is the best rational approximation of x from above with the largest denominator $q^* \leq n_{\max}$.

Proof. Omit. □

4.3 Checking if nx is an integer

As a byproduct of the computationo of $\lfloor nx \rfloor$, we can check if nx is an integer or not. This is a useful ingredients for our main algorithm that will be described in Section 5.

Specifically, what we want to do in this section is to find a criterion on positive integers m, k, and h such that nx is an integer if and only if

$$(nm \bmod 2^k) < h,$$

where we choose m and k in a way so that $\lfloor nx \rfloor = \left\lfloor \frac{nm}{2^k} \right\rfloor$ holds. Hence, the quotient and the remainder of nm divided by 2^k can be respectively utilized for the computation of $\lfloor nx \rfloor$ and the integer check of nx.

Theorem 4.6.
Let x, ξ be positive real numbers and $n_{\max}$ be a positive integer such that

$$\lfloor nx \rfloor = \lfloor n\xi \rfloor$$

holds for all $n = 1, \cdots, n_{\max}$. Then, for a positive real number η, we have the followings.

1. *If $x = \frac{p}{q}$ is a rational number with $2 \leq q \leq n_{\max}$, then we have*

$$\begin{aligned}&\{n \in \{1, \cdots, n_{\max}\} : nx \in \mathbb{Z}\}\\ &\quad = \{n \in \{1, \cdots, n_{\max}\} : n\xi - \lfloor n\xi \rfloor < \eta\}\end{aligned}$$

if and only if

$$\left\lfloor \frac{n_{\max}}{q} \right\rfloor q(\xi - x) < \eta \leq u(\xi - x) + \frac{1}{q}$$

holds, where u is the smallest positive integer such that $up \equiv 1 \pmod{q}$.

2. *If x is either an irrational number or a rational number with the denominator strictly greater than $n_{\max}$, then we have*

$$\begin{aligned}&\{n \in \{1, \cdots, n_{\max}\} : nx \in \mathbb{Z}\} = \emptyset\\ &\quad = \{n \in \{1, \cdots, n_{\max}\} : n\xi - \lfloor n\xi \rfloor < \eta\}\end{aligned}$$

if and only if

$$\eta \leq q_* \xi - p_*$$

holds, where $\frac{p_*}{q_*}$ is the best rational approximations of x from below with the largest denominator $q_* \leq n_{\max}$.

Proof. 1. By the condition on q, the set

$$\{n \in \{1, \cdots, n_{\max}\} : nx \in \mathbb{Z}\}$$

and its complement are both nonempty. Hence by Theorem 4.4, for given $k = 1, \cdots, n_{\max}$, kx is an integer if and only if

$$k\xi - \lfloor k\xi \rfloor < \min_{n; nx \notin \mathbb{Z}} (n\xi - \lfloor n\xi \rfloor)$$

and kx is not an integer if and only if

$$k\xi - \lfloor k\xi \rfloor > \max_{n; nx \in \mathbb{Z}} (n\xi - \lfloor n\xi \rfloor)$$

thus we have

$$\begin{aligned}&\{n \in \{1, \cdots, n_{\max}\} : nx \in \mathbb{Z}\} = \emptyset\\ &\quad = \{n \in \{1, \cdots, n_{\max}\} : n\xi - \lfloor n\xi \rfloor < \eta\}\end{aligned}$$

if and only if

$$\max_{n; nx \in \mathbb{Z}} (n\xi - \lfloor n\xi \rfloor) < \eta \leq \min_{n; nx \notin \mathbb{Z}} (n\xi - \lfloor n\xi \rfloor).$$

Note that the left-hand side is nothing but

$$\begin{aligned}\max_{n; nx \in \mathbb{Z}} (n\xi - \lfloor nx \rfloor) &= \max_{n; nx \in \mathbb{Z}} n(\xi - x)\\ &= \left\lfloor \frac{n_{\max}}{q} \right\rfloor q(\xi - x).\end{aligned}$$

For the right-hand side, Theorem 4.4 shows that

$$\begin{aligned}&\min_{n; nx \notin \mathbb{Z}} (n\xi - \lfloor n\xi \rfloor)\\ &\quad = \min_{n; np \equiv 1 \pmod{q}} (n\xi - \lfloor nx \rfloor)\\ &\quad = \min_{n; np \equiv 1 \pmod{q}} \left(n\xi - \frac{np - 1}{q}\right)\\ &\quad = \min_{n; np \equiv 1 \pmod{q}} n(\xi - x) + \frac{1}{q}\\ &\quad = u(\xi - x) + \frac{1}{q}.\end{aligned}$$

Therefore, we get the desired equivalence.

2. In this case, there is no $n \in \{1, \cdots, n_{\max}\}$ such that nx is an integer. Therefore, we have

$$\begin{aligned}&\{n \in \{1, \cdots, n_{\max}\} : nx \in \mathbb{Z}\} = \emptyset\\ &\quad = \{n \in \{1, \cdots, n_{\max}\} : n\xi - \lfloor n\xi \rfloor < \eta\}\end{aligned}$$

if and only if

$$\eta \leq \min_{n=1, \cdots, n_{\max}} (n\xi - \lfloor n\xi \rfloor).$$

By Theorem 4.4, the minimum is achieved when n minimizes $nx - \lfloor nx \rfloor$, which happens when $n = q_*$ because of Corollary C.12, so the condition is

$$\eta \leq q_* \xi - \lfloor q_* x \rfloor = q_* \xi - p_*,$$

as claimed.

□

Remark 1. We can rewrite the conditions in terms of ξ rather than η:

1. When $x = \frac{p}{q}$ is a rational number with $2 \le q \le n_{\max}$,
$$x + \frac{q\eta - 1}{uq} \le \xi < x + \frac{\eta}{\lfloor n_{\max}/q \rfloor q}.$$
In this case, a useful sufficient condition for having the left-hand side is $\eta \le \frac{1}{q}$.
2. When x is either an irrational number or a rational number with the denominator strictly greater than $n_{\max}$,
$$\frac{p_* + \eta}{q_*} \le \xi.$$
In this case, a useful sufficient condition for having this inequality is
$$\eta \le q_* x - p_* \quad \text{and} \quad x \le \xi.$$

Remark 2. When $x = \frac{1}{q}$, practically the most useful choice for η is $\eta = \xi$. This specific case is addressed by Lemire et al. in [10] (Proposition 1).[13] To see why this is a valid choice, note that the condition given by Theorem 4.6 on η can be written as
$$\left\lfloor \frac{n_{\max}}{q} \right\rfloor q \left(\xi - \frac{1}{q}\right) < \eta \le \left(\xi - \frac{1}{q}\right) + \frac{1}{q} = \xi$$
since $u = 1$. Then with $\eta = \xi$, the left-hand side is satisfied if and only if
$$\left(\left\lfloor \frac{n_{\max}}{q} \right\rfloor q - 1\right) \xi < \left\lfloor \frac{n_{\max}}{q} \right\rfloor,$$
if and only if
$$\xi < \frac{\lfloor n_{\max}/q \rfloor}{\lfloor n_{\max}/q \rfloor q - 1}.$$
By the condition on ξ that $\lfloor nx \rfloor = \lfloor n\xi \rfloor$ holds, Theorem 4.2 gives us
$$\xi < \frac{1}{q}\left(1 + \frac{1}{v}\right)$$
where $v = \left\lfloor \frac{n_{\max}+1}{q} \right\rfloor q - 1$. Thus, the choice $\eta = \xi$ is always valid if
$$\frac{1}{q}\left(1 + \frac{1}{v}\right) \le \frac{\lfloor n_{\max}/q \rfloor}{\lfloor n_{\max}/q \rfloor q - 1}.$$
We can rewrite the above inequality as
$$\left(\left\lfloor \frac{n_{\max}}{q} \right\rfloor q - 1\right)(v+1) \le \left\lfloor \frac{n_{\max}}{q} \right\rfloor qv,$$
which is equivalent to
$$\left\lfloor \frac{n_{\max}}{q} \right\rfloor q \le v + 1 = \left\lfloor \frac{n_{\max}+1}{q} \right\rfloor q,$$
which is indeed always true. Therefore, whenever
$$\frac{1}{q} \le \xi < \frac{1}{q} + \frac{1}{q(\lfloor (n_{\max}+1)/q \rfloor q - 1)},$$
or equivalently, whenever
$$\frac{1}{q} \le \xi < \frac{\lfloor (n_{\max}+1)/q \rfloor}{\lfloor (n_{\max}+1)/q \rfloor q - 1},$$
we have that n is divisible by q if and only if
$$n\xi - \lfloor n\xi \rfloor < \xi$$
for all $n = 1, \cdots, n_{\max}$.

Remark 3. It also follows that the divisibility test algorithm suggested in Section 9 of [8], which is based on the modular inverse of the divisor with respect to 2^k, is also a special case of Theorem 4.6.

To be precise, [8] shows that for an odd divisor $q < 2^k$, any integer $n = 1, \cdots, 2^k - 1$ is divisible by q if and only if
$$(nm \bmod 2^k) \le \left\lfloor \frac{2^k - 1}{q} \right\rfloor,$$
where m is the modular inverse of q with respect to 2^k. To derive this result from our theorem, take $n_{\max} := 2^k - 1$, $\eta := \left(\left\lfloor \frac{2^k}{q} \right\rfloor + 1\right) 2^{-k}$, and $x := \frac{p}{q}$ where p is the nonnegative integer satisfying such that $qm = 2^k p + 1$. Clearly, then p and q must be coprime. Finally, take $\xi := \frac{m}{2^k}$. With this choice of the parameters, we claim that the condition of the first part of our theorem is satisfied.

First, we need to check that the condition for Theorem 4.2 is satisfied. This is quite clear; note that
$$\xi = \frac{m}{2^k} = \frac{2^k p + 1}{2^k q} = x + \frac{1}{2^k q},$$
and obviously we should have $\frac{1}{2^k q} < \frac{1}{vq}$ where v is the greatest integer satisfying $vp \equiv 1 \pmod{q}$ and $v \le n_{\max}$, since $n_{\max} < 2^k$.

Next, we need to check the inequality
$$\left\lfloor \frac{n_{\max}}{q} \right\rfloor q(\xi - x) < \eta \le u(\xi - x) + \frac{1}{q},$$
where u is the smallest positive integer satisfying $up \equiv 1 \pmod{q}$. The left-hand side is quite obvious, since
$$\left\lfloor \frac{n_{\max}}{q} \right\rfloor q(\xi - x) = \frac{1}{2^k}\left\lfloor \frac{2^k - 1}{q} \right\rfloor < \eta,$$

[13] In fact, aside from the technicality that we allow ξ and η to be irrational (which is a quite useless generalization in this context), Proposition 1 in [10] is weaker than the specialization $\eta = \xi$ of Theorem 4.6, as it only gives a sufficient condition which is not necessarily a necessary condition. For example, when $x = \frac{1}{7}$ and $n_{\max} = 12$, the choice $\xi = \eta = \frac{13}{84}$ is allowed by Theorem 4.6 but is rejected by Proposition 1 of [10].

and for the right-hand side, note that if we let r be the remainder of 2^k divided by q, then

$$-rp \equiv -2^k p \equiv 1 - qm \equiv 1 \pmod{q},$$

thus u must be in fact precisely $q - r$. Then,

$$u(\xi - x) + \frac{1}{q} = \frac{q-r}{2^k q} + \frac{1}{q} = \frac{1}{2^k}\left(\frac{2^k - r}{q} + 1\right) = \eta,$$

so we indeed get the inequality. Therefore, our theorem tells us that for any $n = 1, \cdots, \left\lfloor \frac{2^k-1}{q} \right\rfloor$, $nx = \frac{np}{q}$ is an integer (or equivalently, q divides n) if and only if

$$(nm \mod 2^k) < 2^k \eta = \left\lfloor \frac{2^k}{q} \right\rfloor + 1.$$

The apparent difference $\left\lfloor \frac{2^k}{q} \right\rfloor$ versus $\left\lfloor \frac{2^k-1}{q} \right\rfloor$ is not genuine, because they must be same anyway since 2^k divided by q should have a nonzero remainder.

4.4 Computation of $\lfloor nx - y \rfloor$

In this subsection, we prove a result which gives a criteria for having the equality

$$\lfloor nx - y \rfloor = \lfloor n\xi - \zeta \rfloor .$$

Theorem 4.7.
Let x, ξ be positive real numbers and $n_{\max}$ be a positive integer such that

$$\lfloor nx \rfloor = \lfloor n\xi \rfloor$$

holds for all $n = 1, \cdots, n_{\max}$, and let $\frac{p_*}{q_*}, \frac{p^*}{q^*}$ be the best rational approximations of x from below and above, respectively, with the largest denominators $q_*, q^* \le n_{\max}$. Given $y \in [0, 1]$, define

$$L \coloneqq \{n \in \{1, \cdots, n_{\max}\} : nx - \lfloor nx \rfloor < y\},$$
$$R \coloneqq \{n \in \{1, \cdots, n_{\max}\} : nx - \lfloor nx \rfloor \ge y\}.$$

Then for a real number ζ, we have the followings.

1. *When $L = \emptyset$,*

$$\lfloor nx - y \rfloor = \lfloor n\xi - \zeta \rfloor$$

holds for all $n = 1, \cdots, n_{\max}$ if and only if

$$(q^*\xi - \lfloor q^*\xi \rfloor) - 1 < \zeta \le q_*\xi - \lfloor q_*\xi \rfloor .$$

2. *When $R = \emptyset$,*

$$\lfloor nx - y \rfloor = \lfloor n\xi - \zeta \rfloor$$

holds for all $n = 1, \cdots, n_{\max}$ if and only if

$$\mu\xi - \lfloor \mu\xi \rfloor < \zeta \le (q_*\xi - \lfloor q_*\xi \rfloor) + 1,$$

where μ is defined as follows.

(1) If $x = \frac{p}{q}$ is a rational number with $q \le n_{\max}$, then μ is the greatest integer such that $\mu \le n_{\max}$ and $\mu p \equiv -1 \pmod{q}$.

(2) If x is either an irrational number or a rational number with the denominator strictly greater than $n_{\max}$, then $\mu = q^*$.

3. *When L and R are both nonempty,*

$$\lfloor nx - y \rfloor = \lfloor n\xi - \zeta \rfloor$$

holds for all $n = 1, \cdots, n_{\max}$ if and only if

$$\mu\xi - \lfloor \mu\xi \rfloor < \zeta \le \nu\xi - \lfloor \nu\xi \rfloor ,$$

where μ and ν are defined as follows.

(1) If $x = \frac{p}{q}$ is a rational number with $q \le n_{\max}$, then μ is the greatest integer such that $\mu \le n_{\max}$ and $\mu p \equiv \lceil qy \rceil - 1 \pmod{q}$, and ν is the smallest positive integer such that $\nu p \equiv \lceil qy \rceil \pmod{q}$.

(2) If x is either an irrational number or a rational number with the denominator strictly greater than $n_{\max}$, then μ is the greatest positive integer such that $\mu \le n_{\max}$,

$$\mu p_* \equiv \lceil q_* y \rceil - l \pmod{q_*} \quad \textit{and}$$
$$\mu < \frac{q_* y + l - \lceil q_* y \rceil}{q_* x - p_*},$$

where l is the smallest positive integer allowing such μ to exist. Similarly, ν is the smallest positive integer such that $\nu \le n_{\max}$,

$$\nu p_* \equiv \lceil q_* y \rceil - l \pmod{q_*} \quad \textit{and}$$
$$\nu \ge \frac{q_* y + l - \lceil q_* y \rceil}{q_* x - p_*},$$

where l is the greatest nonnegative integer allowing such ν to exist.

Proof. Note that

$$\lfloor nx - y \rfloor = \begin{cases} \lfloor nx \rfloor & \text{if } nx - \lfloor nx \rfloor \ge y, \\ \lfloor nx \rfloor - 1 & \text{if } nx - \lfloor nx \rfloor < y, \end{cases}$$

so we have the equality

$$\lfloor nx - y \rfloor = \lfloor n\xi - \zeta \rfloor$$

for all $n = 1, \cdots, n_{\max}$ if and only if

$$L = L' \coloneqq \{n \in \{1, \cdots, n_{\max}\} : \zeta - 1 \le n\xi - \lfloor n\xi \rfloor < \zeta\}$$

and

$$R = R' \coloneqq \{n \in \{1, \cdots, n_{\max}\} : \zeta \le n\xi - \lfloor n\xi \rfloor < \zeta + 1\}.$$

Now, as $L \cup R = \{1, \cdots, n_{\max}\}$ and $L' \cap R' = \emptyset$, we have $L = L'$ and $R = R'$ if and only if we have $L \subseteq L'$ and $R \subseteq R'$. Note that $L \subseteq L'$ holds if and only if either $L = \emptyset$ or

$$\max_{n \in L}(n\xi - \lfloor n\xi \rfloor) < \zeta \leq \min_{n \in L}(n\xi - \lfloor n\xi \rfloor) + 1,$$

and $R \subseteq R'$ holds if and only if either $R = \emptyset$ or

$$\max_{n \in R}(n\xi - \lfloor n\xi \rfloor) - 1 < \zeta \leq \min_{n \in R}(n\xi - \lfloor n\xi \rfloor).$$

1. When $L = \emptyset$, we only need to verify

$$\max_{n=1, \cdots, n_{\max}}(n\xi - \lfloor n\xi \rfloor) = q^*\xi - \lfloor q^*\xi \rfloor$$

and

$$\min_{n=1, \cdots, n_{\max}}(n\xi - \lfloor n\xi \rfloor) = q_*\xi - \lfloor q_*\xi \rfloor.$$

The second one follows directly from Theorem 4.4 and Corollary C.12. For the first one, note that $L = \emptyset$ enforces nx to be never an integer, thus $nx - \lfloor nx \rfloor = 1 - (\lceil nx \rceil - nx)$ always holds. Thus, maximizing $nx - \lfloor nx \rfloor$ is equivalent to minimizing $\lceil nx \rceil - nx$, so we can apply Theorem 4.4 and Corollary C.12 to get the conclusion.

2. When $R = \emptyset$, we only need to verify

$$\max_{n=1, \cdots, n_{\max}}(n\xi - \lfloor n\xi \rfloor) = \mu\xi - \lfloor \mu\xi \rfloor$$

and

$$\min_{n=1, \cdots, n_{\max}}(n\xi - \lfloor n\xi \rfloor) = q_*\xi - \lfloor q_*\xi \rfloor.$$

Again, the second one is a trivial consequence of Theorem 4.4 and Corollary C.12, thus we consider the first one. By Theorem 4.4, it is enough to show that μ is the greatest element in $\{1, \cdots, n_{\max}\}$ maximizing $\mu x - \lfloor \mu x \rfloor$. When $x = \frac{p}{q}$ with $q \leq n_{\max}$, this is clear. When x is irrational or is rational with the denominator strictly bigger than $n_{\max}$, then nx is never an integer, so it is enough to find the minimizer of $\lceil nx \rceil - nx$, which is q^*.

3. Now, suppose L and R are both not empty. Then it suffices to show that

$$\mu\xi - \lfloor \mu\xi \rfloor = \max_{n \in L}(n\xi - \lfloor n\xi \rfloor)$$

and

$$\nu\xi - \lfloor \nu\xi \rfloor = \min_{n \in R}(n\xi - \lfloor n\xi \rfloor).$$

By Theorem 4.4, it suffices to show that μ is the greatest maximizer of $nx - \lfloor nx \rfloor$ for $n \in L$ and ν is the smallest minimizer of $nx - \lfloor nx \rfloor$ for $n \in R$.

When $x = \frac{p}{q}$ with $q \leq n_{\max}$, since $n \in R$ if and only if $(np \mod q) \geq \lceil qy \rceil$, we can apply Theorem 4.4 to get the desired conclusion.

Next, assume x is either irrational or rational with the denominator strictly greater than $n_{\max}$. In this case, we can apply Theorem 4.4 to see that the greatest maximizer of $nx - \lfloor nx \rfloor$ for $n \in L$ is precisely the greatest maximizer of $\frac{np_*}{q_*} - \left\lfloor \frac{np_*}{q_*} \right\rfloor$ for $n \in L$. Since

$$\begin{aligned} \frac{np_*}{q_*} - \left\lfloor \frac{np_*}{q_*} \right\rfloor &= (nx - \lfloor nx \rfloor) - n\left(x - \frac{p_*}{q_*}\right) \\ &< nx - \lfloor nx \rfloor, \end{aligned}$$

the maximum possible value of $\frac{np_*}{q_*} - \left\lfloor \frac{np_*}{q_*} \right\rfloor$ for $n \in L$ is at most $\lceil q_*y \rceil - 1$. In general, for $n \in \{1, \cdots, n_{\max}\}$ with $np \equiv \lceil q_*y \rceil - l \pmod{q_*}$ for some positive integer $l \leq \lceil q_*y \rceil$, we have $n \in L$ if and only if

$$n < \frac{q_*y + l - \lceil q_*y \rceil}{q_*x - p_*},$$

thus we get the desired description of μ. We can similarly show that ν is the smallest minimizer of $nx - \lfloor nx \rfloor$ for $n \in R$.

□

In the same way, we can also show the following.

Theorem 4.8.
Let x, ξ be positive real numbers and $n_{\max}$ be a positive integer such that

$$\lceil nx \rceil = \lceil n\xi \rceil$$

holds for all $n = 1, \cdots, n_{\max}$, and let $\frac{p_*}{q_*}, \frac{p^*}{q^*}$ be the best rational approximations of x from below and above, respectively, with the largest denominators $q_*, q^* \leq n_{\max}$. Given $y \in [0, 1]$, define

$$\begin{aligned} L &:= \{n \in \{1, \cdots, n_{\max}\} : \lceil nx \rceil - nx \geq y\}, \\ R &:= \{n \in \{1, \cdots, n_{\max}\} : \lceil nx \rceil - nx < y\}. \end{aligned}$$

Then for a real number ζ, we have the followings.

1. *When $L = \emptyset$,*

$$\lceil nx + y \rceil = \lceil n\xi + \zeta \rceil$$

holds for all $n = 1, \cdots, n_{\max}$ if and only if

$$(\lceil \mu\xi \rceil - \mu\xi) - 1 < \zeta \leq \lceil q^*\xi \rceil - q^*\xi,$$

where μ is defined as follows.

(1) If $x = \frac{p}{q}$ is a rational number with $q \leq n_{\max}$, then μ is the greatest integer such that $\mu \leq n_{\max}$ and $\mu p \equiv 1 \pmod{q}$.

(2) If x is either an irrational number or a rational number with the denominator strictly greater than $n_{\max}$, then $\mu = q_*$.

2. *When $R = \emptyset$,*

$$\lceil nx + y \rceil = \lceil n\xi + \zeta \rceil$$

holds for all $n = 1, \cdots, n_{\max}$ if and only if

$$(\lceil q_*\xi \rceil - q_*\xi) - 1 < \zeta \le \lceil q^*\xi \rceil - q^*\xi.$$

3. *When L and R are both nonempty,*

$$\lceil nx + y \rceil = \lceil n\xi + \zeta \rceil$$

holds for all $n = 1, \cdots, n_{\max}$ if and only if

$$\lceil \mu\xi \rceil - \mu\xi < \zeta \le \lceil \nu\xi \rceil - \nu\xi$$

where μ and ν are defined as follows.

(1) If $x = \frac{p}{q}$ is a rational number with $q \le n_{\max}$, then μ is the greatest integer such that $\mu \le n_{\max}$ and $\mu p \equiv q - \lceil qy \rceil + 1 \pmod{q}$, and ν is the smallest positive integer such that $\nu p \equiv q - \lceil qy \rceil \pmod{q}$.

(2) If x is either an irrational number or a rational number with the denominator strictly greater than $n_{\max}$, then μ is the greatest positive integer such that $\mu \le n_{\max}$,

$$\mu p^* \equiv q^* - \lceil q^* y \rceil + l \pmod{q^*} \quad \textit{and}$$
$$\mu < \frac{q^* y + l - \lceil q^* y \rceil}{p^* - q^* x},$$

where l is the smallest positive integer allowing such μ to exist. Similarly, ν is the smallest positive integer such that $\nu \le n_{\max}$,

$$\nu p^* \equiv q^* - \lceil q^* y \rceil + l \pmod{q^*} \quad \textit{and}$$
$$\nu \ge \frac{q^* y + l - \lceil q^* y \rceil}{p^* - q^* x},$$

where l is the greatest nonnegative integer allowing such ν to exist.

Proof. Omit. □

5. Dragonbox

Now, let us describe our main algorithm, Dragonbox. For this section, we will assume a round-to-nearest rounding rule, because that is the the most relevant and at the same time the most difficult case. Algorithms for other rounding rules can be developed in similar ways, and they will be covered in Appendix A and Appendix B.

5.1 Normal interval case

We will describe Dragonbox for the case when $F_w \ne 1$ or $E_w = E_{\min}$ (we call this *normal interval case*), so that $\Delta = 2^e$. The case $F_w = 1$ and $E_w \ne E_{\min}$ (we call this *shorter interval case*) will be covered in Section 5.2.

5.1.1 Overview

In contrast to Schubfach, consider the following exponent instead of $k_0 := -\lfloor \log_{10} \Delta \rfloor$:

$$k := k_0 + \kappa = -\lfloor \log_{10} \Delta \rfloor + \kappa = \kappa - \lfloor e \log_{10} 2 \rfloor$$

where κ is a positive integer constant in a certain range that we will discuss in Section 5.1.3. Given e, the computation of k can be done with one integer multiplication and one arithmetic right-shift. Details will be explained in Section 6.1.

Similarly to [3], let us use the following notations:

$$\begin{aligned} x &:= 10^k w_L, \\ y &:= 10^k w, \\ z &:= 10^k w_R, \\ \delta &:= z - x = 10^k \Delta, \end{aligned}$$

and for $a \in \mathbb{R}$, we denote $a^{(i)} := \lfloor a \rfloor$, $a^{(f)} := a - \lfloor a \rfloor$. Note that $\Delta < 10^{-k_0+1}$ implies $\delta < 10^{\kappa+1}$.

As observed by the authors of Ryū and Schubfach, we can compute the integer parts of quantities like these only using limited-precision integer arithmetic and a precomputed table of fixed number of bits from 5^k's, and this observation is at the very heart of many of recent floating-point conversion algorithms, in both decimal-to-binary and binary-to-decimal directions. The basis of why this is possible is already all explained in Section 4. We will give more details on how to apply those results to our specific case to enable efficient computation of the integer parts of x, y, z, and δ, in Section 5.1.2–5.1.6.

However, we should keep in mind that the core idea of Dragonbox is on how to avoid actually doing these computations, because they are still relatively heavy computations. The way Dragonbox achieves this goal is to use a Grisu-like idea based on the following simple fact:

Proposition 5.1.
Let s, r be the unique integers satisfying

$$z^{(i)} = 10^{\kappa+1} s + r, \quad 0 \le r < 10^{\kappa+1}.$$

Then, $I \cap 10^{-k_0+1}\mathbb{Z}$ is nonempty if and only if

$$s \in 10^{k_0-1} I,$$

if and only if:

1. *$r + z^{(f)} \le \delta$, when $I = [w_L, w_R]$,*
2. *$r + z^{(f)} < \delta$, when $I = (w_L, w_R]$.*
3. *$r + z^{(f)} \le \delta$ and $r \ne 0$ or $z^{(f)} \ne 0$, when $I = [w_L, w_R)$, and*
4. *$r + z^{(f)} < \delta$ and $r \ne 0$ or $z^{(f)} \ne 0$, when $I = (w_L, w_R)$.*

Proof. We first show that $I \cap 10^{-k_0+1}\mathbb{Z}$ is nonempty if and only if $s \in 10^{k_0-1} I$. Clearly, $10^{-k_0+1} s$ is always an element of $10^{-k_0+1}\mathbb{Z}$, so if it belongs to I, then $I \cap 10^{-k_0+1}\mathbb{Z}$ is nonempty.

Conversely, suppose $I \cap 10^{-k_0+1}\mathbb{Z}$ is nonempty. Let v be any element of it. Then, $v \le w_R$, so

$$10^{k-\kappa-1}v \le \frac{z}{10^{\kappa+1}},$$

but since $10^{k-\kappa-1}v = 10^{k_0-1}v \in \mathbb{Z}$, it follows that

$$10^{k-\kappa-1}v \le \left\lfloor \frac{z}{10^{\kappa+1}} \right\rfloor = s.$$

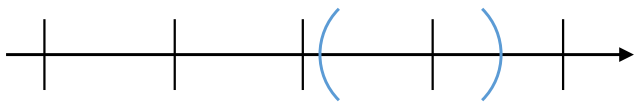

Figure 3. The unique lattice point in I should be the floor of the right endpoint, since I is longer than the unit

Now, since $10^{k_0-1}v$ and s are both integers, if we suppose

$$10^{k_0-1}v \ne s,$$

then

$$10^{k_0-1}v + 1 \le s$$

follows, which implies

$$10^{-k_0+1}s \ge v + 10^{-k_0+1} > v + \Delta \ge w_L + \Delta = w_R$$

by definition of k_0. This is absurd, because

$$10^{-k_0+1}s = 10^{-k} \cdot 10^{\kappa+1}s \le 10^{-k} \cdot z = w_R.$$

Hence, we deduce $s = 10^{k_0-1}v \in 10^{k_0-1}I$, concluding the first "if and only if".

To show the second "if and only if", let us recall that $10^{-k_0+1}s = 10^{-k} \cdot 10^{\kappa+1}s$ is at most w_R. Hence, when $w_R \in I$, $10^{-k_0+1}s$ is in I if and only if its distance from w_L is less than or equal to Δ, or strictly less than Δ, depending on whether or not if w_L is in I, which are precisely the claims *1* and *2*.

On the other hand, if $w_R \notin I$, then we need to rule out the case $w_R = 10^{-k_0+1}s$ in addition, which is precisely the case when $r = 0$ and $z^{(f)} = 0$, thus we have the last two claims as well. □

Note that $r + z^{(f)} \le \delta$ if and only if

1. $r < \delta^{(i)}$, or
2. $r = \delta^{(i)}$ and $z^{(f)} \le \delta^{(f)}$,

and we have a similar equivalence for $r + z^{(f)} < \delta$. As in [3], we can efficiently perform these comparisons. In particular, since

$$x^{(i)} + x^{(f)} = (z^{(i)} - \delta^{(i)}) + (z^{(f)} - \delta^{(f)}),$$

and $-1 < z^{(f)} - \delta^{(f)} < 1$, we conclude

$$x^{(i)} = \begin{cases} z^{(i)} - \delta^{(i)} & \text{if } z^{(f)} \ge \delta^{(f)}, \\ z^{(i)} - \delta^{(i)} - 1 & \text{if } z^{(f)} < \delta^{(f)}, \end{cases}$$

so we just need to compare the parity of $x^{(i)}$ and $z^{(i)} - \delta^{(i)}$ to conclude if the inequality $z^{(f)} \ge \delta^{(f)}$ holds or not. Details of how to compute the parity of $x^{(i)}$ is explained in Section 5.1.6.

Note that we need to compare the fractional parts only when we know $r = \delta^{(i)}$; in this case, note that

$$z^{(i)} - \delta^{(i)} = 10^{\kappa+1}s$$

is always an even number. Thus, we have $z^{(f)} < \delta^{(f)}$ if and only if $x^{(i)}$ is an odd number. When $x^{(i)}$ is an even number, then we have either $z^{(f)} = \delta^{(f)}$ or $z^{(f)} > \delta^{(f)}$. Depending on whether or not w_L is contained in I, we may need to distinguish these two cases. To do that, we check if x is an integer, since $z^{(f)} = \delta^{(f)}$ if and only if $x^{(f)} = 0$ if and only if x is an integer. Details of how to check if x is an integer is explained in Section 5.1.6.

Let us now more precisely describe how to inspect if $I \cap 10^{-k_0+1}\mathbb{Z}$ is empty:

Algorithm 5.2 (Skeleton of Dragonbox, part 1)**.**

1. Compute $k = -\lfloor \log_{10} \Delta \rfloor + \kappa$. Since κ is just a fixed constant, it boils down to calculating $\lfloor \log_{10} \Delta \rfloor$; see Section 6.1 for details.
2. Compute $z^{(i)}$; see Section 5.1.5 for details.
3. Compute s, r by dividing $z^{(i)}$ by $10^{\kappa+1}$. Given that κ is a known constant, this can be done efficiently without actually issuing the notoriously slow integer division instruction, as described in [8]. Compilers these days usually automatically perform this optimization pretty well, but we can sometimes do better than them because of some additional constraints they are usually not aware of. See Section 5.1.7 for details.
4. Compute $\delta^{(i)}$; see Section 5.1.4 for details.
5. Check if the inequality $r > \delta^{(i)}$ holds. If that is the case, then we conclude that $I \cap 10^{-k_0+1}\mathbb{Z}$ is empty.
6. Otherwise, check if the inequality $r < \delta^{(i)}$ holds. If that is the case, we need to check if $r = z^{(f)} = 0$ in addition when $w_R \notin I$. We can inspect the equality $z^{(f)} = 0$ by checking if z is an integer; see Section 5.1.5 for details.
   - If $w_R \notin I$ and $r = z^{(f)} = 0$, then we conclude that $I \cap 10^{-k_0+1}\mathbb{Z}$ is empty.
   - Otherwise, we conclude that $10^{-k+\kappa+1}s$ is the unique element in $I \cap 10^{-k_0+1}\mathbb{Z}$.
7. Otherwise, we have $r = \delta^{(i)}$. Then, compute the parity of $x^{(i)}$.
   - If $x^{(i)}$ is an odd number, then we have $z^{(f)} < \delta^{(f)}$, so we conclude that $10^{-k+\kappa+1}s$ is the unique element in $I \cap 10^{-k_0+1}\mathbb{Z}$.

- If $x^{(i)}$ is an even number and $w_L \notin I$, then we conclude that $I \cap 10^{-k_0+1}\mathbb{Z}$ is empty.
- If $x^{(i)}$ is an even number and $w_L \in I$, then check if x is an integer. If that is the case, then we conclude that $10^{-k+\kappa+1}s$ is the unique element in $I \cap 10^{-k_0+1}\mathbb{Z}$. Otherwise, we conclude that $I \cap 10^{-k_0+1}\mathbb{Z}$ is empty.

8. When we have concluded that $10^{-k+\kappa+1}s$ is the unique element in $I \cap 10^{-k_0+1}\mathbb{Z}$, then since s might contain trailing decimal zeros, find the greatest integer d such that 10^d divides s. Then we conclude that

$$\frac{s}{10^d} \times 10^{-k+\kappa+1+d}$$

is the answer we are looking for.

Note that in order to compare $z^{(f)}$ and $\delta^{(f)}$, we need to compute the parity of $x^{(i)}$ which involves multiplications we want to avoid. Hence, we want to minimize the chance of having $r = \delta^{(i)}$, so we want to choose κ as large as possible. However, choosing too big κ will prevent $z^{(i)}$ and $\delta^{(i)}$ to fit inside a machine word, so there are in fact not so many choices for κ we can have. See Section 5.1.3 for details.

It is worth mentioning that, since we know

$$\begin{aligned} z &= \left(f_c + \frac{1}{2}\right) \cdot 2^e \cdot 10^k \\ &\le \left(2^{p+1} - \frac{1}{2}\right) \cdot 2^e \cdot 10^{-\lfloor e \log_{10} 2 \rfloor + \kappa} \\ &= \left(2^{p+1} - \frac{1}{2}\right) \cdot 2^e \cdot 10^{\lceil -e \log_{10} 2 \rceil + \kappa} \\ &< 2^{p+1} \cdot 2^e \cdot 10^{-e \log_{10} 2 + \kappa + 1} \\ &= 10^{\kappa+1} \cdot 2^{p+1}, \end{aligned} \tag{3}$$

the integer s must be at most $2^{p+1} - 1$. In particular, for binary32, $s \le 2^{24} - 1$ is of at most 8 digits, so the number of possible trailing zeros is at most 7, and similarly, for binary64, $s \le 2^{53} - 1$ is of at most 16 digits, so the number of possible trailing zeros is at most 15.

Next, let us discuss what we do if $I \cap 10^{-k_0+1}\mathbb{Z}$ turns out to be empty. Our procedure in this case is a bit different from the Schubfach's way. Recall that Corollary 3.3 tells us that in this case,

$$I \cap 10^{-k_0}\mathbb{Z} = 10^{-k}\left(10^k I \cap 10^\kappa \mathbb{Z}\right)$$

is not empty and its elements are precisely the elements with the smallest number of significand digits.

We will now compute

$$\begin{aligned} y^{(ru)} &:= \left\lfloor \frac{y}{10^\kappa} + \frac{1}{2} \right\rfloor 10^\kappa \quad \text{and} \\ y^{(rd)} &:= \left\lceil \frac{y}{10^\kappa} - \frac{1}{2} \right\rceil 10^\kappa, \end{aligned}$$

which are the elements in $10^\kappa\mathbb{Z}$ that are closest to $y \in 10^k I$, using a method similar to that described in [3]. In [3], it is shown that both of $y^{(ru)}$ and $y^{(rd)}$ should be in $10^k I$ as we have assumed $F_w \ne 1$ or $E_w = E_{\min}$. For completeness we reproduce that argument here.

Proposition 5.3.
With the assumption $F_w \ne 1$ or $E_w = E_{\min}$,

$$\begin{aligned} y^{(ru)} &:= \left\lfloor \frac{y}{10^\kappa} + \frac{1}{2} \right\rfloor 10^\kappa \quad \textit{and} \\ y^{(rd)} &:= \left\lceil \frac{y}{10^\kappa} - \frac{1}{2} \right\rceil 10^\kappa \end{aligned}$$

are always inside $10^k I$.

Proof. First, note that $y^{(ru)}$ and $y^{(rd)}$ should be one of $a := \left\lfloor \frac{y}{10^\kappa} \right\rfloor 10^\kappa$ and $b := \left(\left\lfloor \frac{y}{10^\kappa} \right\rfloor + 1\right) 10^\kappa$. More precisely,

1. $y^{(ru)} = y^{(rd)} = a$ if $\left(\frac{y}{10^\kappa}\right)^{(f)} < \frac{1}{2}$,
2. $y^{(ru)} = b$ and $y^{(rd)} = a$ if $\left(\frac{y}{10^\kappa}\right)^{(f)} = \frac{1}{2}$,
3. $y^{(ru)} = y^{(rd)} = b$ if $\left(\frac{y}{10^\kappa}\right)^{(f)} > \frac{1}{2}$.

Note that $\frac{a}{10^\kappa} = \lfloor 10^{k_0} w \rfloor$. As shown in the proof of Proposition 3.1, $w \in I$ implies that at least one of $\frac{a}{10^\kappa} \in 10^{k_0} I$ or $\frac{b}{10^\kappa} \in 10^{k_0} I$ holds, thus we have at least one of $a \in 10^k I$ or $b \in 10^k I$.

Suppose first that $a \notin 10^k I$, so $b \in 10^k I$. We claim that in this case the fractional part of $\frac{y}{10^\kappa}$ should be strictly greater than $\frac{1}{2}$, so $y^{(ru)} = y^{(rd)} = b \in 10^k I$. Since $F_w \ne 1$ or $E_w = E_{\min}$, y must be at the exact center of $10^k I$, so $a \notin 10^k I$ and $b \in 10^k I$ together imply that $b - y \le y - a$. In other words, the fractional part of $\frac{y}{10^\kappa}$ should be at least $\frac{1}{2}$. Now it suffices to show that the fractional part cannot be equal to $\frac{1}{2}$. Suppose on the contrary that $\left(\frac{y}{10^\kappa}\right)^{(f)} = \frac{1}{2}$. Then $b - y = y - a$, but since $a \notin 10^k I$, $b \in 10^k I$, and y is at the center of $10^k I$, it follows that $10^k I = (a, b]$. This implies

$$\begin{aligned} 10^\kappa \mathbb{Z} \ni b = z &= (2f_c + 1) \cdot 2^{e-1} \cdot 10^k \\ &= 2^{e+k-1} \cdot 5^k \cdot (2f_c + 1), \end{aligned}$$

and since $2f_c + 1$ is an odd integer, we must have

$$e + k - 1 = \kappa \quad \text{and} \quad 2f_c + 1 = 5^{e-1}.$$

However, by the same reason, $a = x$ implies

$$e + k - 1 = \kappa \quad \text{and} \quad 2f_c - 1 = 5^{e-1},$$

which is a contradiction. This shows the claim.

Similarly, if we suppose $b \notin 10^k I$, then we must have $\left(\frac{y}{10^\kappa}\right)^{(f)} \le \frac{1}{2}$, and if we further assume $\left(\frac{y}{10^\kappa}\right)^{(f)} = \frac{1}{2}$, then $I = [a, b)$ follows which leads to a contradiction by the same reasoning. Hence, in this case we must have $y^{(ru)} = y^{(rd)} = a \in 10^k I$, so we always have $y^{(ru)}, y^{(rd)} \in 10^k I$. □

As noted in the proof, we have $y^{(ru)} = y^{(rd)} + 1$ if and only if

$$\frac{y}{10^\kappa} - \left\lfloor \frac{y}{10^\kappa} \right\rfloor = \frac{1}{2},$$

and $y^{(ru)} = y^{(rd)}$ otherwise. In other words, $y^{(ru)}$ and $y^{(rd)}$ are same except when there is a tie, so our procedure is to first compute $y^{(ru)}$, detect a tie, and if a tie is detected, then follow the given rule for breaking the tie, and if the conclusion is to prefer $y^{(rd)}$, then decrease the computed value of $y^{(ru)}$ by one. Let $y^{(r)}$ be the output of this procedure, then the correctly rounded decimal representation of w with the shortest number of digits is thus

$$y^{(r)} \times 10^{-k+\kappa}.$$

To compute $y^{(ru)}$, note that

$$\begin{aligned} y^{(ru)} &= \left\lfloor \frac{y + (10^\kappa/2)}{10^\kappa} \right\rfloor \\ &= \left\lfloor \frac{z + (10^\kappa/2) - (z-y)}{10^\kappa} \right\rfloor \\ &= 10s + \left\lfloor \frac{r + (10^\kappa/2) - \epsilon^{(i)} + (z^{(f)} - \epsilon^{(f)})}{10^\kappa} \right\rfloor \end{aligned}$$

where we define

$$\epsilon := z - y.$$

Since we have assumed $F_w \neq 1$ or $E_w = E_{\min}$, w should lie at the exact center of I. Hence in particular, $\epsilon = \frac{\delta}{2}$, so $\epsilon^{(i)} = \left\lfloor \frac{\delta^{(i)}}{2} \right\rfloor$. Also, since κ is a positive integer, $10^\kappa/2$ is an integer. Recall that we already have assumed that $I \cap 10^{-k_0+1}\mathbb{Z}$ is empty; hence, by Proposition 5.1, either $r \geq \delta^{(i)}$ or $r = 0$. Since $\epsilon < \delta$, for the first case we know

$$r + \frac{10^\kappa}{2} - \epsilon^{(i)} > 0.$$

For the case $r = 0$, let us replace r by $10^{\kappa+1}$ and s by $s-1$ so that we still have the inequality above. To be precise, let us define

$$\tilde{s} := \begin{cases} s & \text{if } r \neq 0 \\ s-1 & \text{if } r = 0 \end{cases}, \quad \tilde{r} := \begin{cases} r & \text{if } r \neq 0 \\ 10^{\kappa+1} & \text{if } r = 0 \end{cases},$$

so that we have

$$z^{(i)} = 10^{\kappa+1}\tilde{s} + \tilde{r}$$

and

$$y^{(ru)} = 10\tilde{s} + \left\lfloor \frac{\tilde{r} + (10^\kappa/2) - \epsilon^{(i)} + (z^{(f)} - \epsilon^{(f)})}{10^\kappa} \right\rfloor.$$

Define

$$D := \tilde{r} + (10^\kappa/2) - \epsilon^{(i)},$$

then as $\delta < 10^{\kappa+1}$ we clearly have $D \geq 0$ for all cases.

Next, let t, ρ be the unique integers satisfying

$$D = 10^\kappa t + \rho, \quad 0 \leq \rho < 10^\kappa.$$

Then,

$$y^{(ru)} = (10\tilde{s} + t) + \left\lfloor \frac{\rho + (z^{(f)} - \epsilon^{(f)})}{10^\kappa} \right\rfloor.$$

Note that the residue term

$$\left\lfloor \frac{\rho + (z^{(f)} - \epsilon^{(f)})}{10^\kappa} \right\rfloor$$

is always 0 except when $\rho = 0$ and $z^{(f)} < \epsilon^{(f)}$, and for that case it is equal to -1. As the probability of D being divisibly by 10^κ will be low (and it will be even lower if we choose larger κ), we can mostly ignore the residue term and conclude $y^{(ru)} = 10\tilde{s} + t$.

If D is unfortunately divisible by 10^κ, then we need to compare $z^{(f)}$ and $\epsilon^{(f)}$. Similarly to the comparison of $z^{(f)}$ and $\delta^{(f)}$, this can be done by computing the parity of $y^{(i)}$. Indeed, note that

$$y^{(i)} + y^{(f)} = (z^{(i)} - \epsilon^{(i)}) + (z^{(f)} - \epsilon^{(f)}),$$

and since $-1 < z^{(f)} - \epsilon^{(f)} < 1$, we conclude

$$y^{(i)} = \begin{cases} z^{(i)} - \epsilon^{(i)} & \text{if } z^{(f)} \geq \epsilon^{(f)}, \\ z^{(i)} - \epsilon^{(i)} - 1 & \text{if } z^{(f)} < \epsilon^{(f)}. \end{cases}$$

Therefore, $z^{(f)} \geq \epsilon^{(f)}$ holds if and only if the parity of $y^{(i)}$ and that of $z^{(i)} - \epsilon^{(i)}$ are equal to each other. In fact, since $10^{\kappa+1}$ is even, the parity of $z^{(i)}$ and that of r is same, so we can compare the parity of $y^{(i)}$ with that of $D - (10^\kappa/2)$. If the parities are the same, then we conclude $z^{(f)} \geq \epsilon^{(f)}$ so $y^{(ru)} = 10\tilde{s} + t$, and otherwise, we conclude $z^{(f)} < \epsilon^{(f)}$ so $y^{(ru)} = 10\tilde{s} + t - 1$. Details of how to compute the parity of $y^{(i)}$ will be explained in Section 5.1.6.

Note that a tie happens exactly when $\rho = z^{(f)} - \epsilon^{(f)} = 0$; indeed, it happens when the fractional part of $\frac{y}{10^\kappa}$ is exactly $1/2$, or equivalently,

$$\frac{y}{10^\kappa} + \frac{1}{2} = (10\tilde{s} + t) + \frac{\rho + (z^{(f)} - \epsilon^{(f)})}{10^\kappa}$$

is an integer. Since

$$-1 < \rho + (z^{(f)} - \epsilon^{(f)}) < 10^\kappa,$$

it follows that $\frac{y}{10^\kappa} + \frac{1}{2}$ is an integer if and only if

$$\rho + (z^{(f)} - \epsilon^{(f)}) = 0,$$

if and only if $\rho = z^{(f)} - \epsilon^{(f)} = 0$. Or equivalently, we have a tie if and only if D is divisible by 10^κ and $y = z - \epsilon$ is an integer. When that is the case, we need to choose between $y^{(ru)} = 10\tilde{s} + t$ and $y^{(rd)} = 10\tilde{s} + t - 1$ according to the given tie-break rule. Details of how to check if y is an integer will be explained in Section 5.1.6.

In summary, when $I \cap 10^{-k_0+1}\mathbb{Z}$ turns out to be empty, then:

Algorithm 5.4 (Skeleton of Dragonbox, part 2)**.**

1. Compute $D = \tilde{r} + (10^\kappa/2) - \lfloor \delta^{(i)}/2 \rfloor$.
2. Compute t, ρ by dividing D by 10^κ. Again, given that κ is a known constant, this can be done efficiently using the method described in [8]. In fact, since we do not care about the actual value of ρ and we only need to know if ρ is zero or not, we can do even better; see Section 5.1.8 for details.
3. If $\rho \neq 0$, then $(10\tilde{s} + t) \times 10^{-k+\kappa}$ is the answer we are looking for.
4. Otherwise, compare the parity of $y^{(i)}$ with that of $D - (10^\kappa/2)$. If they are different, then we have $z^{(f)} < \epsilon^{(f)}$, so $(10\tilde{s} + t - 1) \times 10^{-k+\kappa}$ is the answer we are looking for.
5. Otherwise, check if y is an integer. If that is the case, then we have a tie; break it according to the given rule, so that we choose one of $(10\tilde{s} + t - 1) \times 10^{-k+\kappa}$ and $(10\tilde{s} + t) \times 10^{-k+\kappa}$ as the answer.
6. Otherwise, $(10\tilde{s} + t) \times 10^{-k+\kappa}$ is the answer we are looking for.

Again, we want to avoid computing the parity of $y^{(i)}$, so we prefer to choose κ as large as possible.

5.1.2 Precomputed lookup table for the computation of x, y, z, and δ

For each k, define

$$\varphi_k := 10^k \cdot 2^{-e_k},$$

where e_k is the unique integer such that

$$2^{Q-1} \leq \varphi_k < 2^Q$$

which means

$$e_k = \lfloor k \log_2 10 \rfloor - Q + 1.$$

Using Theorem 4.2, we want to show that we can compute $x^{(i)}$, $y^{(i)}$, $z^{(i)}$, or $\delta^{(i)}$ by multiplying a q-bit integer to the Q-bit integer

$$\tilde{\varphi}_k := \lceil \varphi_k \rceil$$ [14]

[14] Strictly speaking, we need to be aware of the possibility of having $\tilde{\varphi}_k = 2^Q$. However, this happens extremely rarely. One can figure out that the equivalent condition for having that is

$$0 < \lceil k \log_2 5 \rceil - k \log_2 5 \leq \log_2 \frac{2^Q}{2^Q - 1}.$$

To check if this inequality might be true for some $k \neq 0$, we just need to check it for the case when $\lceil k \log_2 5 \rceil - k \log_2 5$ is minimized. Hence, when $k > 0$, we only need to consider the case when $\frac{\lceil k \log_2 5 \rceil}{k}$ is a best rational approximation of $\log_2 5$ from above, and similarly when $k < 0$, we only need to consider the case when $\frac{\lfloor -k \log_2 5 \rfloor}{-k}$ is a best rational approximation of $\log_2 5$ from below. Using the algorithm of finding all best rational approximations from below and from above, one can show that, if we take $Q = 64$, the first positive k satisfying the above inequality is 11199596541212005343, and the first negative k satisfying it is -9870257339578654810. Of course the corresponding numbers for $Q = 128$ will be even more ridiculously large.

and then performing a right-shift. The precision Q will be set to $Q = 2q$ for both binary32 and binary64, but for the sake of showing that $Q = 2q$ is indeed sufficient, let us first not assume this and just let Q be any positive integer bigger than or equal to q. The Q-bit integer $\tilde{\varphi}_k$ for all k can be stored in a precomputed lookup table that will be referred in the runtime.

Define

$$\beta := e + e_k + Q - 1 = e + \lfloor k \log_2 10 \rfloor,$$

then our precise goal is that if we take $Q = 2q$, then the formula

$$\lfloor n \cdot 2^{e-1} \cdot 10^k \rfloor = \left\lfloor \frac{2^\beta n \tilde{\varphi}_k}{2^Q} \right\rfloor \tag{4}$$

is always valid for all $n = 1, \cdots, n_{\max}$ where we take $n_{\max} := 2^{p+2}$. Then, we can compute $z^{(i)}$ as

$$z^{(i)} = \left\lfloor \frac{2^\beta (2f_c + 1) \tilde{\varphi}_k}{2^Q} \right\rfloor,$$

since $2f_c + 1 \leq n_{\max}$ always holds, and we can also similarly compute $x^{(i)}$ and $y^{(i)}$.

Define $\frac{a}{b} := 2^{e-1} \cdot 10^k$, that is,

$$(a, b) = \begin{cases} (2^{e+k-1} \cdot 5^k, 1) & \text{if } e + k - 1 \geq 0 \text{ and } k \geq 0, \\ (2^{e+k-1}, 5^{-k}) & \text{if } e + k - 1 \geq 0 \text{ and } k < 0, \\ (5^k, 2^{-e-k+1}) & \text{if } e + k - 1 < 0 \text{ and } k \geq 0. \end{cases}$$

(The case $e + k - 1 < 0$ and $k < 0$ is impossible by definition of k.) Then by Theorem 4.2, a necessary and sufficient condition for having (4) is

$$\frac{\tilde{\varphi}_k}{2^Q} < \frac{a}{2^\beta b} + \frac{1}{2^\beta v b}$$

when $b \leq n_{\max}$, and

$$\frac{\tilde{\varphi}_k}{2^Q} < \frac{a^*}{2^\beta b^*}$$

when $b > n_{\max}$, where $v \leq n_{\max}$ is the greatest integer such that $va \equiv -1 \pmod{b}$ and $\frac{a^*}{b^*}$ is the best rational approximation of $\frac{a}{b}$ from above with the greatest denominator $b^* \leq n_{\max}$.

In addition to that, we want to know whether or not the number $n \cdot 2^{e-1} \cdot 10^k$ is an integer. We can use Theorem 4.6 for that. Specifically, we want to let the threshold to be $\eta = \frac{1}{2^{Q-q}}$ because that makes the actual implementation for binary32 and binary64 efficient. Then by the first remark after Theorem 4.6, we have that $n \cdot 2^{e-1} \cdot 10^k$ is an integer if and only if

$$\left(2^\beta n \tilde{\varphi}_k \mod 2^Q\right) < 2^q, \tag{5}$$

provided that

$$\begin{aligned} \frac{a}{2^\beta b} + \frac{b - 2^{Q-q}}{2^{Q-q+\beta} u b} &\leq \frac{\tilde{\varphi}_k}{2^Q} \\ &< \frac{a}{2^\beta b} + \frac{1}{2^{Q-q+\beta} \lfloor n_{\max}/b \rfloor b} \end{aligned} \tag{6}$$

when $b \le n_{\max}$, and

$$\frac{2^{Q-q}a_* + 1}{2^{Q-q+\beta}b_*} \le \frac{\tilde{\varphi}_k}{2^Q}$$

when $b > n_{\max}$, where u is the smallest positive integer such that $ua \equiv -1 \pmod{b}$ and $\frac{a_*}{b_*}$ is the best rational approximation of $\frac{a}{b}$ from below with the greatest denominator $b_* \le n_{\max}$.

In fact, since

$$\tilde{\varphi}_k = \lceil 10^k \cdot 2^{-e_k} \rceil = \lceil 2^{Q-\beta} \cdot 2^{e-1} \cdot 10^k \rceil = \left\lceil \frac{2^{Q-\beta}a}{b} \right\rceil,$$

the left-hand side of (6) is automatically satisfied if $b \le 2^{Q-q}$. Also, a sufficient condition for having the right-hand side is

$$2^\beta n_{\max} < 2^q,$$

because if the above inequality holds, then

$$\frac{2^{q-\beta}}{\lfloor n_{\max}/b \rfloor b} \ge \frac{2^{q-\beta}}{n_{\max}} > 1.$$

By choosing an appropriate value of κ, we will make the above inequality to always hold; see Section 5.1.3.

On the other hand, when $b > n_{\max}$, similarly

$$\frac{2^{Q-q}a_* + 1}{2^{Q-q+\beta}b_*} \le \frac{a}{2^\beta b}$$

is a sufficient condition, which can be rewritten as

$$\frac{1}{2^{Q-q}b_*} \le \frac{a}{b} - \frac{a_*}{b_*},$$

or

$$\frac{b}{ab_* - a_*b} \le 2^{Q-q}. \tag{7}$$

In summary, we want

$$\frac{\tilde{\varphi}_k}{2^Q} < \frac{a}{2^\beta b} + \frac{1}{2^\beta vb} \quad \text{and} \quad b \le 2^{Q-q}$$

when $b \le n_{\max}$, and

$$\frac{\tilde{\varphi}_k}{2^Q} < \frac{a^*}{2^\beta b^*} \quad \text{and} \quad \frac{b}{ab_* - a_*b} \le 2^{Q-q} \tag{8}$$

when $b > n_{\max}$, for all $e \in [E_{\min} - p, E_{\max} - p]$.

To check if these are really the case, we wrote a program that verifies the above inequalities; see our reference implementation [11]. Thanks to the algorithm for computing best rational approximations, the verification program runs quite fast.[15] The result of this verification program is shown in Figure 4.

[15] In the author's laptop with Intel(R) Core(TM) i7-7700HQ, it takes less than 5 seconds to run with no optimization enabled, and is instantaneous with optimization enabled.

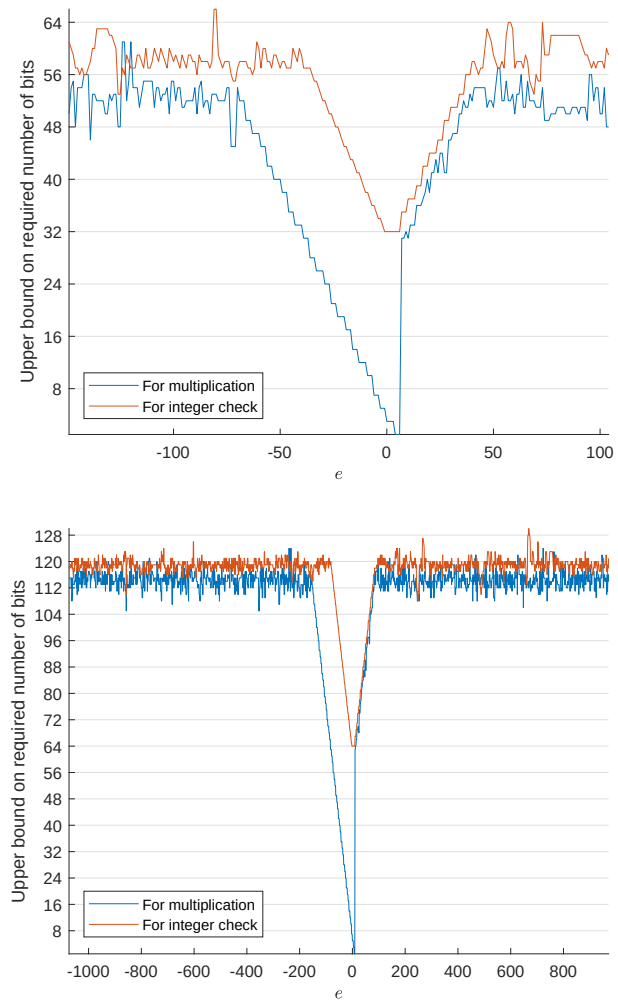

Figure 4. Upper bounds on the minimum admissible Q for each e (top: binary32, bottom: binary64)

Unfortunately, (8) turns out to be not always true. Specifically, the condition for having (4) is satisfied, but (7) is violated for several e's. However, the counterexamples turn out to be very rare.

Note that in order to have (5), we must have

$$\frac{na}{b} - \left\lfloor \frac{na}{b} \right\rfloor < \frac{1}{2^{Q-q}}. \tag{9}$$

Indeed, since (4) holds, (5) implies

$$\begin{aligned}\frac{na}{b} - \left\lfloor \frac{na}{b} \right\rfloor &= \left(\frac{2^\beta n\tilde{\varphi}_k}{2^Q} - \left\lfloor \frac{2^\beta n\tilde{\varphi}_k}{2^Q} \right\rfloor \right) + n\left(\frac{a}{b} - \frac{2^\beta \tilde{\varphi}_k}{2^Q} \right) \\ &\le \frac{2^\beta n\tilde{\varphi}_k}{2^Q} - \left\lfloor \frac{2^\beta n\tilde{\varphi}_k}{2^Q} \right\rfloor < \frac{1}{2^{Q-q}}.\end{aligned}$$

Using the theory of continued fractions, we can develop an algorithm for enumerating all n's with (9); see Algorithm

C.13. With this algorithm, one can figure out that (9) holds only for the following cases:

- For binary32,
  - $e = -81$, $k = 26$, and $n = 29711844$,
  - $e = -80$, $k = 26$, and $n = 14855922$, and
  - $e = -80$, $k = 26$, and $n = 29711844$.
- For binary64,
  - $e = 668$, $k = -199$, $n = 4443527624677894$,
  - $e = 668$, $k = -199$, $n = 8887055249355788$,
  - $e = 669$, $k = -199$, $n = 2221763812338947$,
  - $e = 669$, $k = -199$, $n = 4443527624677894$, and
  - $e = 670$, $k = -199$, $n = 2221763812338947$.

(See [11] for the actual program.)

One can verify that indeed for these cases, (5) holds while $n \cdot 2^{e-1} \cdot 10^k$ is not an integer.

However, except possibly for the first and the third cases for binary32, all other cases should be never actually encountered in Algorithm 5.2 and Algorithm 5.4, because we demand integer checks of $n \cdot 2^{e-1} \cdot 10^k$ only for n's that are at least $2^{p+1} - 1$. (Note that $2^{p+1} - 1 = 2f_c - 1$ when $f_c = 2^p$, which means all significand bits are zero. For binary32, we have $2^{p+1} - 1 = 16777215$ and for binary64, we have $2^{p+1} - 1 = 9007199254740991$.)

Hence, we only need to be careful about the first and the third cases for binary32, and there is no actual counterexample for binary64. Another observation is that the $n = 29711844$ in those counterexamples is even. Since $2f_c \pm 1$ is an odd number, the only possible problematic scenario is when we need to check if y is an integer for this case, and integer checks for x and z should just work fine. Fortunately, with careful examination, one can see that for any rounding modes the inputs corresponding to these possible error cases (which are 29711844×2^{-82} and 29711844×2^{-81}) do not prompt the integer check for y when we follow Algorithm 5.2 and Algorithm 5.4.[16] Therefore, we have no problem.

5.1.3 Choice of κ

In Section 5.1.1, we have seen why we would prefer to choose κ as large as possible. However, in order to simplify the integer part computations and integer checks of x, y, and z, we demand the number $2^\beta n$ from the previous section to fit in q-bits. Hence, we want $p + 2 + \beta < q$ to hold, thus

$$\beta < q - p - 2.$$

Note that this in particular makes the condition for having successful integer checks trivial for the case $b \le n_{\max}$.

[16] For 29711844×2^{-82}, the algorithm marches to the second step in Algorithm 5.4 with $D = 38$. As D is not divisible by 10, we do not need to check if y is an integer. For 29711844×2^{-81}, we have $r = 7$ and $\delta^{(i)} = 82$ in the fifth step of Algorithm 5.2, so we do not need to check if y is an integer.

Since

$$-e \log_{10} 2 \le -\lfloor e \log_{10} 2 \rfloor < -e \log_{10} 2 + 1,$$

we have

$$\begin{aligned} \beta &= e + \lfloor k \log_2 10 \rfloor \\ &= e + \lfloor (\kappa - \lfloor e \log_{10} 2 \rfloor) \log_2 10 \rfloor \\ &\in [\lfloor \kappa \log_2 10 \rfloor, \lfloor (\kappa + 1) \log_2 10 \rfloor], \end{aligned} \tag{10}$$

so a sufficient condition for β to satisfy $\beta < q - p - 2$ is

$$\kappa \le \lfloor (q - p - 2) \log_{10} 2 \rfloor - 1.$$

For binary32 ($p = 23$, $q = 32$), this gives $\kappa \le 1$ and for binary64 ($p = 52$, $q = 64$), this gives $\kappa \le 2$. Therefore, we take $\kappa = 1$ for binary32 and $\kappa = 2$ for binary64.

5.1.4 Computation of $\delta^{(i)}$

Recall that

$$\delta^{(i)} = \lfloor 2^e \cdot 10^k \rfloor = \lfloor 2 \cdot 2^{e-1} \cdot 10^k \rfloor.$$

Obviously, we have $2 \le n_{\max} = 2^{p+2}$, so we get

$$\delta^{(i)} = \left\lfloor \frac{2^{\beta+1} \tilde{\varphi}_k}{2^Q} \right\rfloor = \left\lfloor \frac{\tilde{\varphi}_k}{2^{Q-\beta-1}} \right\rfloor,$$

which means that $\delta^{(i)}$ is nothing but just the upper $(\beta + 1)$-bits of the Q-bit integer $\tilde{\varphi}_k$.

5.1.5 Computation of $z^{(i)}$

To compute

$$z^{(i)} = \left\lfloor \frac{2^\beta (2f_c + 1) \tilde{\varphi}_k}{2^Q} \right\rfloor,$$

we left-shift the q-bit integer $2f_c + 1$ by β-bits (where the result is still a q-bit integer because of the choice of κ), perform the multiplication with the Q-bit integer $\tilde{\varphi}_k$ to obtain a $(q + Q)$-bit integer, and then take the upper q-bits. Also, note that the integer condition (5) is equivalent to that the upper $(Q - q)$-bits of the remaining Q-bits are all zero. Hence, we extract the upper Q-bits from the $(q + Q)$-bit result of the multiplication, and use the upper q-bits for obtaining $z^{(i)}$ and the lower $(Q - q)$-bits for the integer check of z.

The only nontrivial part here is how to actually compute the upper Q-bits of the result of multiplication of a q-bit integer and a Q-bit integer. We will explain the procedure in terms of q-bit full and half multiplications. By *q-bit full multiplication*, we mean computing the $2q$-bit result of multiplication of two q-bit integers, and by *q-bit half multiplication*, we mean computing only the lower q-bits of the $2q$-bit result of multiplication.

In general multiplication of integers with smaller bitwidth tend to be faster than that of integers with larger bitwidth, so we prefer multiplication of smaller-sized integers

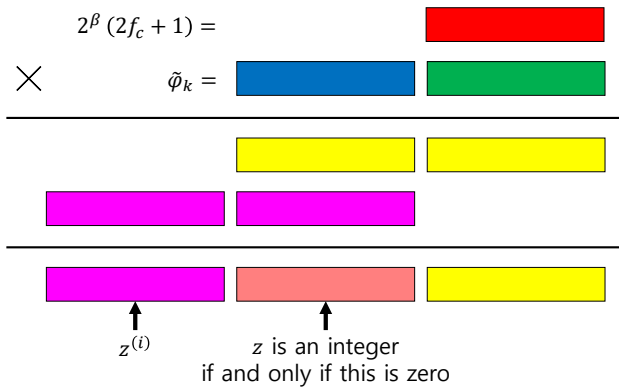

Figure 5. Illustration of the computation of $z^{(i)}$ and integer check of z

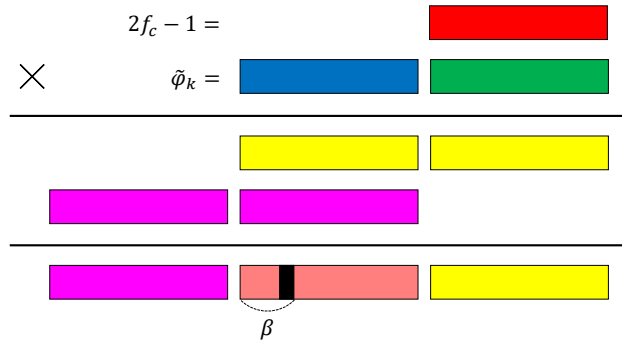

Figure 6. Illustration of the computation of the parity of $x^{(i)}$

if possible. Also, modern platforms may or may not provide a direct instruction for full multiplication, but even if they do, it is often slower than the corresponding instruction for half multiplication, so we prefer half multiplications over full multiplication.

When the machine does not provide a direct instruction for full multiplication, we can emulate it with several half multiplications. See [12], for example.

Now, assuming that the native word size of the platform is 64-bits, for binary32 ($q = 32$, $Q = 64$), getting the upper 64-bits from the 96-bit result of the multiplication of a 32-bit integer and a 64-bit lookup table entry can be done with one 64-bit full multiplication: treat the given 32-bit integer as a 64-bit integer, shift it to left by 32-bits, perform 64-bit full multiplication with the lookup table entry, and then extract the upper 64-bits out of it.

Similarly, for binary64 ($q = 64$, $Q = 128$), getting the upper 128-bits from the 192-bit result of the multiplication of a 64-bit integer and a 128-bit lookup table entry can be done with two 64-bit full multiplications and one 64-bit addition possibly with carry: perform 64-bit full multiplications of the given 64-bit integer with the upper 64-bits and the lower 64-bits of the lookup table entry, respectively, extract the upper 64-bits from the result of the latter and add it to the lower 64-bits of the result of the former, and if the carry is generated, add it back to the upper 64-bits of it.

5.1.6 Computation of $x^{(i)}$ and $y^{(i)}$

Recall that we do not need to fully compute $x^{(i)}$ and $y^{(i)}$; rather, what we only care about is the parity of them and whether or not x and y are integers. This makes the computation a little bit simpler than that of $z^{(i)}$.

We will only consider the case of $x^{(i)}$ since the case of $y^{(i)}$ is identical. Recall we have

$$x^{(i)} = \left\lfloor \frac{2^{\beta}(2f_c - 1)\tilde{\varphi}_k}{2^Q} \right\rfloor$$

as $2f_c - 1 \leq n_{\max}$. We can regard the above as taking the upper q-bits from the $(q+Q)$-bit result of multiplication of the q-bit integer $2^{\beta}(2f_c - 1)$ and the Q-bit integer $\tilde{\varphi}_k$, but we can also regard it as taking the upper $(q+\beta)$-bits from the $(q+Q)$-bit result of multiplication of the q-bit integer $2f_c - 1$ and the Q-bit integer $\tilde{\varphi}_k$. This second interpretation is advantageous in this case because it allows us to ignore the upmost q-bit of the result of multiplication. Hence, the computation can be done with one q-bit full multiplication and one q-bit half multiplication followed by an addition without carry as illustrated in Figure 6, contrast to two q-bit full multiplications and additions with carry handling as in the case of $z^{(i)}$.

We first compute the q-bit full multiplication of $2f_c - 1$ and the lower half of $\tilde{\varphi}_k$ (yellow boxes in Figure 6), and compute the q-bit half multiplication of $2f_c - 1$ and the upper half of $\tilde{\varphi}_k$ (the second purple box in Figure 6). Add the upper half from the former and the latter (the pink box in Figure 6), then the parity of $x^{(i)}$ must be the β-th bit of the result, counting from the MSB (most significant bit).

Integer check for x can be also done within this computation. Note that the integer condition (5) is equivalent to

$$(n\tilde{\varphi}_k \mod 2^{Q-\beta}) < 2^{q-\beta}.$$

Hence, we conclude that x is an integer if and only if only the lower $(q-\beta)$-bits of the pink box in Figure 6 and the upper β-bits of the yellow box in the bottom line of Figure 6 are all zero.

In practice, computing the parity of $x^{(i)}$ is still a heavy operation compared to others. However, one should note that in Algorithm 5.2, we need to compute the parity of $x^{(i)}$ only when r and $\delta^{(i)}$ turn out to be the same, which is quite unlikely in general.

5.1.7 Division by $10^{\kappa+1}$

Recall that modern compilers are well-aware of the technique pioneered by Granlund-Montgomery [8] of turning an integer division by a constant into a multiplication followed

by a shift. However, there are often situations where manually doing so results in a better code, because we can leverage some restrictions on the range of inputs that the compiler might not be aware of. The step in Algorithm 5.2 of obtaining s, r satisfying

$$z^{(i)} = 10^{\kappa+1}s + r$$

falls into that category.

Recall from (3) that we have

$$z < 10^{\kappa+1} \cdot 2^{p+1}$$

By Theorem 4.2, we always have

$$s = \left\lfloor \frac{z^{(i)}m}{2^u} \right\rfloor$$

if u and m satisfy

$$\frac{2^u}{10^{\kappa+1}} \le m < \frac{2^u}{10^{\kappa+1}} + \frac{2^u}{10^{\kappa+1}(2^{p+1} \cdot 10^{\kappa+1} - 1)}.$$

For binary32 ($p = 23$, $\kappa = 1$), the smallest u allowing the magic number m to exist is 37, giving the unique solution $m = 1374389535$, and for binary64 ($p = 52$, $\kappa = 2$), the smallest u is 71, giving the unique solution $m = 2361183241434822607$.[17]

It is worth noting that for binary64, what we get in the above is strictly better than what the compiler would do without knowing the upper bound on $z^{(i)}$. Indeed, if we use the trivial upper bound $2^q - 1$ which the compiler would use instead of $10^{\kappa+1} \cdot 2^{p+1} - 1$, the smallest u is 74. The problem is, not like our case, in this case m must be of at least 65-bits, thus it is not possible to put it inside a single 64-bit word, which hinders the compiler to generate the optimal code.

5.1.8 Division by 10^κ

In Algorithm 5.4, there is a step of computing the division of a number D by 10^κ. A notable feature of this step is that, along with the quotient, we do need to know whether the dividend D is divisible by 10^κ, but we actually do not care what the remainder specifically is, which makes it a perfect example where we can apply Theorem 4.6.

More precisely, recall the definition of D:

$$D := \tilde{r} + \frac{10^\kappa}{2} - \left\lfloor \frac{\delta^{(i)}}{2} \right\rfloor.$$

By definition of δ, we know $10^\kappa \le \delta < 10^{\kappa+1}$, so we have $D \le \tilde{r} \le 10^{\kappa+1}$. Then, to have

$$\left\lfloor \frac{D}{10^\kappa} \right\rfloor = \left\lfloor \frac{Dm}{2^u} \right\rfloor$$

for all $D \in [0, 10^{\kappa+1}]$, it suffices to have

$$\frac{2^u}{10^\kappa} \le m < \frac{2^u}{10^\kappa} + \frac{2^u}{10^\kappa(10^{\kappa+1} - 1)} = \frac{2^u \cdot 10}{10^{\kappa+1} - 1}.$$

Furthermore, by the second remark after Theorem 4.6, with such m, we conclude D is divisible by 10^κ if and only if

$$(Dm \bmod 2^u) < m.$$

For both binary32 and binary64, we can do this entire calculation within 32-bit arithmetic; for binary32, we choose $u = 16$ and $m = 6554$, and for binary64, we choose $u = 16$ and $m = 656$, for example. Note that for popular platforms like x86, taking the modular of a 32-bit integer with respect to 2^{16} often not even cost a single instruction, because we can just take the lower-half of the bits.

5.2 Shorter interval case

So far, we have assumed that either $F_w \ne 1$ or $E_w = E_{\min}$ holds, so that the length of the interval Δ is always equal to 2^e. In this section, we will assume $F_w = 1$ and $E_w \ne E_{\min}$ so that $\Delta = 3 \cdot 2^{e-2}$. Note that presence of this shorter interval case complicates a lot of things we argued in the last section, including but not limited to computation of k and $\delta^{(i)}$, integer checks, and also the claim that $y^{(ru)}$ and $y^{(rd)}$ are always in $10^k I$ is no longer true, etc.. Thus, we will follow a completely separate path for the shorter interval case.

We will in fact more closely mimic the original Schubfach algorithm rather than what is described in Section 5.1 in this case, because of the following reasons:

1. Shorter interval cases are rare, especially extremely rare for the binary64 format. Thus, whatever we do with them will not affect the average performance very much.
2. The algorithm given in Section 5.1 is already more complicated than the original Schubfach algorithm, and shoehorning the shorter interval case into it will likely produce a much more complicated one, because lots of the assumptions we made are simply not true for that case.
3. Because we have $F_w = 1$, computing the approximate multiplication by 10^k is no more a heavy operation; in particular, no actual multiplication is needed. Thus, there is little reason to try hard to avoid it. We will give some detailed explanation on this in Section 5.2.1.

Following Schubfach [1], we will work with $k_0 = -\lfloor \log_{10} \Delta \rfloor$ rather than $k = k_0 + \kappa$. Let us define

$$x := 10^{k_0} w_L,$$
$$y := 10^{k_0} w,$$
$$z := 10^{k_0} w_R$$

as before, where k is replaced by k_0. First, we compute $x^{(i)}$ and $z^{(i)}$; see Section 5.2.1 for details. Next, define

$$\tilde{x}^{(i)} := \min(10^{k_0} I \cap \mathbb{Z}), \quad \tilde{z}^{(i)} := \max(10^{k_0} I \cap \mathbb{Z}).$$

[17] In fact, we can conversely compute using Theorem 4.2 that the maximum $n_{\max}$ such that $\left\lfloor \frac{n}{10^{\kappa+1}} \right\rfloor = \left\lfloor \frac{nm}{2^u} \right\rfloor$ holds for all $n = 1, \cdots, n_{\max}$ with our choice of u and m is $n_{\max} = 4908534052$ for binary32 and $n_{\max} = 15534100272597517998$ for binary64. As we can see here, $n_{\max}$ for binary32 exceeds the maximum possible value of 32-bit integers.

In other words, $\tilde{x}^{(i)}$ is $x^{(i)}$ if x is an integer and is contained in $10^{k_0}I$, or $\tilde{x}^{(i)}$ is $x^{(i)}+1$ otherwise, and similarly, $\tilde{z}^{(i)}$ is $z^{(i)}$ if z is not an integer or is contained in $10^{k_0}I$, or $\tilde{z}^{(i)}$ is $z^{(i)}-1$ otherwise.

Proposition 5.5.
$I \cap 10^{-k_0+1}\mathbb{Z}$ is nonempty if and only if

$$\tilde{x}^{(i)} \le \left\lfloor \frac{\tilde{z}^{(i)}}{10} \right\rfloor \cdot 10.$$

If the above inequality is true, then $\left\lfloor \frac{\tilde{z}^{(i)}}{10} \right\rfloor \cdot 10^{-k_0+1}$ is the unique element in $I \cap 10^{-k_0+1}\mathbb{Z}$.

Proof. By applying Proposition 5.1 with $\kappa = 0$, we conclude that $I \cap 10^{-k_0+1}\mathbb{Z}$ is nonempty if and only if

$$s \in 10^{k_0-1}I$$

where we define s, r to be the unique integer satisfying $z^{(i)} = 10s + r$, $0 \le r < 10$.[18]

Note that we can in fact replace $z^{(i)}$ by $\tilde{z}^{(i)}$ when we compute s. Indeed, suppose that z is an integer and is not contained in $10^{k_0}I$, so that $\tilde{z}^{(i)} = z^{(i)} - 1$. Assume first that $s \in 10^{k_0-1}I$. In this case, we should have $r \ne 0$ since otherwise we have $z^{(i)} \in 10^{k_0}I$. Thus, we get the same quotient when we replace $z^{(i)}$ by $\tilde{z}^{(i)}$.

Next, assume that $s \notin 10^{k_0-1}I$. Again, we are okay if $r \ne 0$, so suppose that $r = 0$, thus

$$\tilde{z}^{(i)} = z^{(i)} - 1 = 10s - 1 = 10(s-1) + 9.$$

We claim that in this case we still have $\left\lfloor \frac{\tilde{z}^{(i)}}{10} \right\rfloor = s - 1 \notin 10^{k_0-1}I$. If not, then we have $s - 1 \in 10^{k_0-1}I$ but $s \notin 10^{k_0-1}I$. Note that $s = \frac{z}{10}$ is the right endpoint of the interval $10^{k_0-1}I$, thus we get that the length of the interval $10^{k_0-1}I$ is at least 1, or equivalenty,

$$\Delta \ge 10^{-k_0+1},$$

which is absurd by the definition of k_0; see (1).

Therefore, $\left\lfloor \frac{z^{(i)}}{10} \right\rfloor$ is in $10^{k_0-1}I$ if and only if $\left\lfloor \frac{\tilde{z}^{(i)}}{10} \right\rfloor$ is in $10^{k_0-1}I$, and if one of them is true, then we should have $\left\lfloor \frac{z^{(i)}}{10} \right\rfloor = \left\lfloor \frac{\tilde{z}^{(i)}}{10} \right\rfloor$.

Now, it remains to show that $\left\lfloor \frac{\tilde{z}^{(i)}}{10} \right\rfloor \in 10^{k_0-1}I$ if and only if

$$\tilde{x}^{(i)} \le \left\lfloor \frac{\tilde{z}^{(i)}}{10} \right\rfloor \cdot 10.$$

This is in fact trivial; note that $\left\lfloor \frac{\tilde{z}^{(i)}}{10} \right\rfloor \in 10^{k_0-1}I$ if and only if

$$\left\lfloor \frac{\tilde{z}^{(i)}}{10} \right\rfloor \cdot 10 \in 10^{k_0}I,$$

if and only if

$$\tilde{x}^{(i)} \le \left\lfloor \frac{\tilde{z}^{(i)}}{10} \right\rfloor \cdot 10 \le \tilde{z}^{(i)}$$

by definition of $\tilde{x}^{(i)}$ and $\tilde{z}^{(i)}$, but the inequality

$$\left\lfloor \frac{\tilde{z}^{(i)}}{10} \right\rfloor \cdot 10 \le \tilde{z}^{(i)}$$

is obvious. This concludes the proof. □

Again, $\left\lfloor \frac{\tilde{z}^{(i)}}{10} \right\rfloor$ might contain trailing zeros, so we need to deal with them.

Next, it remains to discuss what should we do if $I \cap 10^{-k_0+1}\mathbb{Z}$ turns out to be empty. In this case, we first compute

$$y^{(ru)} = \left\lfloor y + \frac{1}{2} \right\rfloor.$$

Again this can be done without actually performing a multiplication; see Section 5.2.2 for details.

And then, detect if we have a tie, that is, there are two elements in $I \cap 10^{-k_0+1}\mathbb{Z}$ that is closest to w. If that is indeed the case, then follow the given rule for breaking the tie, and if the conclusion is to prefer $y^{(rd)}$, then let $y^{(r)} := y^{(rd)}$. Otherwise, let $y^{(r)} := y^{(ru)}$. Details of how to detect a tie is explained in Section 5.2.4.

Note that, not like the normal interval case, $y^{(ru)}$ and $y^{(rd)}$ are both not guaranteed to be inside $10^{k_0}I$. However, recall that it is very simple to check if an integer is in $10^{k_0}I$: we can just compare it with $\tilde{x}^{(i)}$ and $\tilde{z}^{(i)}$.

And another good news here is that $y^{(ru)}$ (and thus $y^{(rd)}$ as well) is guaranteed to be at most $\tilde{z}^{(i)}$, and also whenever $y^{(r)}$ is not in $10^{k_0}I$, $y^{(r)}+1$ is the correctly rounded decimal representation of w with the shortest number of digits. See Section 5.2.5 for details.

In conclusion, we can describe the algorithm for the shorter interval case as:

Algorithm 5.6 (Skeleton of Dragonbox, part 3)**.**

1. Compute k_0 and β, where we define β as

   $$\beta := e + e_{k_0} + Q - 1 = e + \lfloor k_0 \log_2 10 \rfloor$$

   as in the normal interval case, except for that k is replaced by k_0.
2. Compute $x^{(i)}$ and $z^{(i)}$; see Section 5.2.1 for details.
3. Compute $\tilde{x}^{(i)}$ and $\tilde{z}^{(i)}$. This involves how to check if x or z are integers. Details of how to check that will be explained in Section 5.2.3.
4. Compute $\left\lfloor \frac{\tilde{z}^{(i)}}{10} \right\rfloor$ and check if the inequality

   $$\tilde{x}^{(i)} \le \left\lfloor \frac{\tilde{z}^{(i)}}{10} \right\rfloor \cdot 10$$

[18] To be precise, we have assumed $\kappa > 0$ before stating Proposition 5.1, but the proof of Proposition 5.1 does not depend on that assumption and it can be applied for the case $\kappa = 0$ as well.

holds. If it holds, then we conclude that $\left\lfloor \frac{\tilde{z}^{(i)}}{10} \right\rfloor \cdot 10^{-k_0+1}$ is the unique element in $I \cap 10^{-k_0+1}\mathbb{Z}$. In this case, $\left\lfloor \frac{\tilde{z}^{(i)}}{10} \right\rfloor$ might contain trailing decimal zeros, so find the greatest integer d such that 10^d divides $\left\lfloor \frac{\tilde{z}^{(i)}}{10} \right\rfloor$, then

$$\left(\left\lfloor \frac{\tilde{z}^{(i)}}{10} \right\rfloor / 10^d \right) \times 10^{d-k_0+1}$$

is the answer we are looking for.

5. Otherwise, compute $y^{(ru)}$; see Section 5.2.2 for details.
6. Detect tie, as described in Section 5.2.4. If we have tie, then choose between $y^{(ru)}$ and $y^{(rd)} = y^{(ru)} - 1$ according to a given rule. Let $y^{(r)}$ be the chosen one, then $y^{(r)} \times 10^{k_0}$ is the answer we are looking for.
7. Otherwise, check if $y^{(ru)} \geq \tilde{x}^{(i)}$ holds. If that is the case, then $y^{(ru)} \times 10^{k_0}$ is the answer we are looking for.
8. Otherwise, $(y^{(ru)}+1) \times 10^{k_0}$ is the answer we are looking for.

5.2.1 Computing $x^{(i)}$ and $z^{(i)}$

Recall that for shorter interval case, we have

$$w_L = \left(f_c - \frac{1}{4} \right) \cdot 2^e,$$

and since $F_w = 1$, which means $f_c = 2^p$, we get

$$\begin{aligned} x^{(i)} &= \left\lfloor (2^{p+2} - 1) \cdot 2^{e-2} \cdot 10^{k_0} \right\rfloor \\ &= \left\lfloor \frac{2^{\beta-1}(2^{p+2} - 1) \cdot \varphi_{k_0}}{2^Q} \right\rfloor \\ &= \left\lfloor \left(\varphi_{k_0} - \frac{\varphi_{k_0}}{2^{p+2}} \right) \cdot 2^{-(Q-p-\beta-1)} \right\rfloor. \end{aligned}$$

It is reasonable to expect that we can probably replace the number

$$\varphi_{k_0} - \frac{\varphi_{k_0}}{2^{p+2}}$$

by an appropriate integer so that the effect of multiplying $2^{-(Q-p-\beta-1)}$ is just a right-shift.

Indeed, it can be exhaustively verified that for the case of binary32,

$$x^{(i)} = \left\lfloor \left(\tilde{\varphi}_{k_0} - \left\lfloor \frac{\tilde{\varphi}_{k_0}}{2^{p+2}} \right\rfloor \right) \cdot 2^{-(Q-p-\beta-1)} \right\rfloor$$

always holds, and for the case of binary64,

$$x^{(i)} = \left\lfloor \left(\left\lfloor \frac{\tilde{\varphi}_{k_0}}{2^{Q-q}} \right\rfloor - \left\lfloor \frac{\tilde{\varphi}_{k_0}}{2^{Q-q+p+2}} \right\rfloor \right) \cdot 2^{-(q-p-\beta-1)} \right\rfloor$$

always holds. See our reference implementation [11] for the verification program. Note that for both cases, the whole computation can be now done with two right-shifts and two subtractions of integers of at most 64-bits.

Similarly, it can be exhaustively verifed that for the case of binary32,

$$z^{(i)} = \left\lfloor \left(\tilde{\varphi}_{k_0} + \left\lfloor \frac{\tilde{\varphi}_{k_0}}{2^{p+1}} \right\rfloor \right) \cdot 2^{-(Q-p-\beta-1)} \right\rfloor$$

always holds, and for the case of binary64,

$$z^{(i)} = \left\lfloor \left(\left\lfloor \frac{\tilde{\varphi}_{k_0}}{2^{Q-q}} \right\rfloor + \left\lfloor \frac{\tilde{\varphi}_{k_0}}{2^{Q-q+p+1}} \right\rfloor \right) \cdot 2^{-(q-p-\beta-1)} \right\rfloor$$

always holds.

5.2.2 Computing $y^{(ru)}$

Note that

$$y = f_c \cdot 2^e \cdot 10^{k_0} = 2^{p+\beta+1-Q} \varphi_{k_0},$$

thus

$$\begin{aligned} y^{(ru)} &= \left\lfloor y + \frac{1}{2} \right\rfloor = \left\lfloor \frac{2y+1}{2} \right\rfloor \\ &= \left\lfloor \frac{2^{p+\beta+2-Q} \varphi_{k_0} + 1}{2} \right\rfloor \\ &= \left\lfloor \frac{\left\lfloor 2^{p+\beta+2-Q} \varphi_{k_0} \right\rfloor + 1}{2} \right\rfloor. \end{aligned}$$

Applying (10) to $\kappa = 0$, we get $0 \leq \beta \leq 3$, so $p + \beta + 2 \leq p + 5$. Note that for both binary32 and binary64, $Q = 2q$ is strictly bigger than $p + 5$, so $2^{p+\beta+2-Q}$ is a negative power of 2. Hence, we can expect to have

$$\left\lfloor 2^{p+\beta+2-Q} \varphi_{k_0} \right\rfloor = \left\lfloor 2^{p+\beta+2-Q} \tilde{\varphi}_{k_0} \right\rfloor.$$

Indeed, this holds if and only if either $\varphi_{k_0} = \tilde{\varphi}_{k_0}$ or $\tilde{\varphi}_{k_0}$ is not a multiple of $2^{Q-p-\beta-2}$. And it turns out that this is indeed the case; our reference implementation [11] includes a program verifying it.

Therefore, we have

$$y^{(ru)} = \left\lfloor \frac{\left\lfloor 2^{p+\beta+2-Q} \tilde{\varphi}_{k_0} \right\rfloor + 1}{2} \right\rfloor,$$

which means that $y^{(ru)}$ can be computed with one subtraction, one increment, and two shifts.

5.2.3 Integer Checks

It would be possible to apply Theorem 4.6 for the integer checks of x and z, but in this case there is a simpler way.

Recall that

$$x = \left(f_c - \frac{1}{4} \right) \cdot 2^e \cdot 10^{k_0} = (2^{p+2} - 1) \cdot 2^{e+k_0-2} \cdot 5^{k_0}.$$

Suppose that $2^{p+2} - 1$ is d_1 times divisible by 5.[19] Then since $2^{p+2} - 1$ is an odd number, it follows that x is an integer if and only if:

1. $e + k_0 - 2 \geq 0$, and

[19] Note that $2^{p+2} - 1$ is a multiple of 5 if and only if $p \equiv 2 \pmod 4$, which is not the case for both binary32 ($p = 23$) and binary64 ($p = 52$), so in fact $d_1 = 0$ in all cases.

2. $k_0 + d_1 \geq 0$.

Using the definition $k_0 = -\left\lfloor \log_{10}(3 \cdot 2^{e-2}) \right\rfloor$, the first condition is equivalent to

$$e - 2 \geq \left\lfloor \log_{10}(3 \cdot 2^{e-2}) \right\rfloor,$$

which is again equivalent to

$$\log_{10}(3 \cdot 2^{e-2}) < e - 1.$$

Rewriting the above gives

$$(e-2)\log_{10} 2 + \log_{10} 3 < (e-2) + 1,$$

which is equivalent to

$$(e-2)\log_{10} 5 > \log_{10} \frac{3}{10}.$$

Hence, it follows that $e + k_0 - 2 \geq 0$ if and only if

$$e - 2 > \log_5 \frac{3}{10},$$

which is equivalent to $e \geq 2$.

On the other hand, the second condition is equivalent to

$$\left\lfloor \log_{10}(3 \cdot 2^{e-2}) \right\rfloor \leq d_1,$$

so

$$\log_{10}(3 \cdot 2^{e-2}) < d_1 + 1,$$

which can be rewritten as

$$2^{e-2} < \frac{10^{d_1+1}}{3}.$$

Hence, it follows that $k_0 + d_1 \geq 0$ if and only if

$$e < 2 + \log_2 \frac{10^{d_1+1}}{3},$$

or equivalently,

$$e \leq 2 + \left\lfloor \log_2 \frac{10^{d_1+1}}{3} \right\rfloor.$$

Thus, x is an integer if and only if

$$2 \leq e \leq 2 + \left\lfloor \log_2 \frac{10^{d_1+1}}{3} \right\rfloor.$$

Similarly, since

$$z = \left(f_c + \frac{1}{2}\right) \cdot 2^e \cdot 10^{k_0} = (2^{p+1} + 1) \cdot 2^{e+k_0-1} \cdot 5^{k_0},$$

suppose that $2^{p+1} + 1$ is d_2 times divisible by 5[20], then z is an integer if and only if:

1. $e + k_0 - 1 \geq 0$, and
2. $k_0 + d_2 \geq 0$.

Again, the first condition is equivalent to

$$\log_{10}(3 \cdot 2^{e-2}) < e,$$

and by rewriting the above we get

$$(e-2)\log_{10} 2 + \log_{10} 3 < (e-2) + 2,$$

which is equivalent to

$$(e-2)\log_{10} 5 > \log_{10} \frac{3}{100}.$$

Or, equivalently,

$$e > \log_5 \frac{3}{100} + 2 = \log_5 \frac{75}{100} = \log_5 \frac{3}{4},$$

which is equivalent to $e \geq 0$.

For the second condition, since there is nothing different from the case of x other than d_1 is replaced by d_2, we get that z is an integer if and only if

$$0 \leq e \leq 2 + \left\lfloor \log_2 \frac{10^{d_2+1}}{3} \right\rfloor.$$

Therefore, just two comparisons are enough for determining if x or z is an integer.

5.2.4 Detecting Tie

In this section, we will show that when we search the correctly rounded integer in $10^{k_0} I \cap \mathbb{Z}$, we have tie so we need to choose between $y^{(ru)}$ and $y^{(rd)} = y^{(ru)} - 1$ if and only if

$$\begin{aligned} -p - 2 - \lfloor (p+4)\log_5 2 - \log_5 3 \rfloor &\leq e \\ &\leq -p - 2 - \lfloor (p+2)\log_5 2 \rfloor. \end{aligned}$$

Note that tie occurs exactly when $y + \frac{1}{2}$ is an integer, or equivalently,

$$2y + 1 = 2^{p+e+1} \cdot 10^{k_0} + 1 = 2^{p+e+k_0+1} \cdot 5^{k_0} + 1$$

is an even integer. Note that this happens exactly when:

1. $p + e + k_0 + 1 = 0$, and
2. $k_0 \geq 0$.

Let us first solve the first equation. The equation can be rewritten as

$$p + e + 1 = \left\lfloor \log_{10}(3 \cdot 2^{e-2}) \right\rfloor,$$

which is equivalent to the inequality

$$p + e + 1 \leq \log_{10}(3 \cdot 2^{e-2}) < p + e + 2.$$

We can rewrite this inequality as

$$10^{p+3} \cdot 10^{e-2} \leq 3 \cdot 2^{e-2} < 10^{p+4} \cdot 10^{e-2},$$

[20] Again $d_2 = 0$ for both binary32 ($p = 23$) and binary64 ($p = 52$).

which is equivalent to

$$10^{p+3} \cdot 5^{e-2} \le 3 < 10^{p+4} \cdot 5^{e-2},$$

or,

$$3 \cdot 10^{-p-4} < 5^{e-2} \le 3 \cdot 10^{-p-3}.$$

Taking log, we get

$$\begin{aligned} -p-4-(p+4)\log_5 2+\log_5 3 &< e-2 \\ &\le -p-3-(p+3)\log_5 2+\log_5 3, \end{aligned}$$

or equivalently,

$$\begin{aligned} -p-2-((p+4)\log_5 2-\log_5 3) &< e \\ &\le -p-1-((p+3)\log_5 2-\log_5 3). \end{aligned}$$

On the other hand, the second condition $k_0 \ge 0$ is equivalent to $e \le 3$ (specialize the arguments in Section 5.2.3 with $d_1 = 0$), which is always true if

$$e \le -p-1-((p+3)\log_5 2-\log_5 3)$$

whenever $p \ge 0$, hence, $y + \frac{1}{2}$ is an integer if and only if

$$\begin{aligned} -p-2-\lfloor (p+4)\log_5 2-\log_5 3\rfloor &\le e \\ &\le -p-2-\lfloor (p+3)\log_5 2-\log_5 3\rfloor. \end{aligned}$$

We will show in Section 5.2.5 that $y^{(ru)}$ is always bounded above by $\tilde{z}^{(i)}$. Note that if we have $y^{(rd)} \notin 10^{k_0} I$, then it is wiser to consider the case not as a tie because $y^{(rd)}$ is no longer a valid choice. Thus, we will now derive an equivalent condition for having $y^{(rd)} \ge \tilde{x}^{(i)}$, which then automatically implies $y^{(ru)}, y^{(rd)} \in 10^{k_0} I$ as $y^{(rd)} \le y^{(ru)} \le \tilde{z}^{(i)}$.

Assuming we have tie so that $y - \frac{1}{2}$ is an integer, we have $y^{(rd)} < \tilde{x}^{(i)}$ if and only if

$$y - \frac{1}{2} < x \quad \text{or} \quad y - \frac{1}{2} \le x,$$

depending on the rounding rule. In fact, since $y - \frac{1}{2}$ is assumed to be an integer, we have $p + e + k_0 + 1 = 0$, and as explained in Section 5.2.3, x is an integer only if $e + k_0 - 2 \ge 0$, which is not the case because

$$e + k_0 - 2 = -p - 3 < 0.$$

Hence, since $y - \frac{1}{2}$ is an integer and x is not an integer, above two inequalities have no difference, so let us work with

$$y - \frac{1}{2} < x$$

for simplicity. Using the definitions of x and y, the above inequality can be written as

$$2^{p+e} \cdot 10^{k_0} - \frac{1}{2} < \left(2^p - \frac{1}{4}\right) \cdot 2^e \cdot 10^{k_0}.$$

Rewriting the above, we get

$$\frac{1}{4} \cdot 2^e \cdot 10^{k_0} < \frac{1}{2}.$$

Since we have assumed $p + e + k_0 + 1 = 0$, we have

$$k_0 = -p - e - 1,$$

so the inequality can be rewritten as

$$2^e \cdot 10^{-p-e-1} < 2,$$

or equivalently,

$$5^{p+e+1} > 2^{-p-2}.$$

Taking log, we get

$$e + p + 1 > -(p+2)\log_5 2,$$

thus

$$e > -p - 1 - (p+2)\log_5 2.$$

Note that the above bound

$$-p - 1 - (p+2)\log_5 2$$

is strictly less than the bound

$$-p - 1 - ((p+3)\log_5 2 - \log_5 3).$$

Hence, more strict equivalent condition for having tie is

$$\begin{aligned} -p-2-\lfloor (p+4)\log_5 2-\log_5 3\rfloor &\le e \\ &\le -p-2-\lfloor (p+2)\log_5 2\rfloor, \end{aligned}$$

and when this is the case, we do not need to worry about the possibility of having $y^{(rd)} \notin 10^{k_0} I$.

5.2.5 Some Facts about Correct Rounding

In this section, we will show the following things:

1. We always have $y^{(ru)} \le \tilde{z}^{(i)}$.
2. Whenever $y^{(ru)} \notin 10^{k_0} I$, the unique integer in $10^{k_0} I$ that is closest to y is $y^{(ru)} + 1$.

The consequence is that, we can check if $y^{(ru)} \notin 10^{k_0} I$ only by checking if $y^{(ru)} < \tilde{x}^{(i)}$, and if that happens, we just need to increase $y^{(ru)}$ by one.

To show the first claim, note that

$$y^{(ru)} \le y + \frac{1}{2} = z - (z - y) + \frac{1}{2},$$

and

$$z - y = \frac{2\delta}{3}.$$

Recall from (1) that

$$10^{-k_0} \le \Delta < 10^{-k_0+1},$$

so $\delta := \Delta \cdot 10^{k_0}$ satisfies

$$1 \le \delta < 10.$$

Hence,

$$y^{(ru)} \le z + \frac{1}{2} - \frac{2\delta}{3} \le z + \frac{1}{2} - \frac{2}{3} = z - \frac{1}{6}.$$

Therefore, $y^{(ru)}$ must be at most $\tilde{z}^{(i)}$.

To show the second claim, suppose $y^{(ru)} \notin 10^{k_0} I$. Then by the first claim, we must have $y^{(ru)} \le x$. Then,

$$y^{(ru)} + 1 \le x + 1 = z + 1 - \delta,$$

and again since $\delta \ge 1$, we get

$$y^{(ru)} + 1 \le z.$$

In fact, the inequality should be strict; otherwise, we should have $\delta = 1$, which is impossible since

$$\delta = \Delta \cdot 10^{k_0} = 3 \cdot 2^{e-2} \cdot 10^{k_0}$$

and there is no way to cancel out the factor 3. On the other hand, note that

$$y^{(ru)} = \left\lfloor y + \frac{1}{2} \right\rfloor > y - \frac{1}{2},$$

so

$$y^{(ru)} + 1 > y + \frac{1}{2} > x.$$

Therefore, we always have

$$x < y^{(ru)} + 1 < z$$

if $y^{(ru)} \notin 10^{k_0} I$. Note that in this case, since we have $y^{(ru)} \le x < y$ and $y^{(ru)}$ is equal to either $\lfloor y \rfloor$ or $\lfloor y \rfloor + 1$, it follows that $y^{(ru)} = \lfloor y \rfloor$. Hence, we conclude that $\lfloor y \rfloor$ is not in $10^{k_0} I$ while $\lfloor y \rfloor + 1 = y^{(ru)} + 1$ is in $10^{k_0} I$, thus $y^{(ru)} + 1$ must be the unique integer inside $10^{k_0} I$ that is closest to y. Therefore, the second claim is also proven.

6. Efficient Log Computations

In this section, we describe how to apply results from Section 4 into efficient computation of $\lfloor n \log_{10} 2 \rfloor$, $\lfloor n \log_2 10 \rfloor$, and $\left\lfloor n \log_{10} 2 - \log_{10} \frac{4}{3} \right\rfloor$.

6.1 Computation of $\lfloor n \log_{10} 2 \rfloor$

From the continued fraction expansion

$$\log_{10} 2 = [0; 3, 3, 9, 2, 2, 4, 6, 2, 1, 1, 3, 1, 18, 1, 6, \cdots],$$

we can find out that the pair $\left(\frac{p_*}{q_*}, \frac{p^*}{q^*}\right)$ of the best rational approximations from below and from above with the largest denominators change as follows, as we vary $n_{\max}$:

- ...
- $485 \le n_{\max} < 681$: $\frac{p_*}{q_*} = \frac{59}{196}$, $\frac{p^*}{q^*} = \frac{146}{485}$,
- $681 \le n_{\max} < 1166$: $\frac{p_*}{q_*} = \frac{205}{681}$, $\frac{p^*}{q^*} = \frac{146}{485}$,
- $1166 \le n_{\max} < 1651$: $\frac{p_*}{q_*} = \frac{351}{1166}$, $\frac{p^*}{q^*} = \frac{146}{485}$,
- $1651 \le n_{\max} < 2136$: $\frac{p_*}{q_*} = \frac{497}{1651}$, $\frac{p^*}{q^*} = \frac{146}{485}$,
- $2136 \le n_{\max} < 2621$: $\frac{p_*}{q_*} = \frac{643}{2136}$, $\frac{p^*}{q^*} = \frac{146}{485}$,
- $2621 \le n_{\max} < 4757$: $\frac{p_*}{q_*} = \frac{643}{2136}$, $\frac{p^*}{q^*} = \frac{789}{2621}$,
- $4757 \le n_{\max} < 6893$: $\frac{p_*}{q_*} = \frac{643}{2136}$, $\frac{p^*}{q^*} = \frac{1432}{4757}$,
- ...

Recall that by Theorem 4.2 and Theorem 4.3, the formula

$$\lfloor nx \rfloor = -\lceil -nx \rceil = \left\lfloor \frac{nm}{2^k} \right\rfloor = -\left\lceil \frac{-nm}{2^k} \right\rceil$$

holds for all $|n| \le n_{\max}$ if and only if

$$\frac{2^k p_*}{q_*} < m < \frac{2^k p^*}{q^*}.$$

(Note that even for negative n, the division by a poewr of 2 followed by the floor function corresponds to the signed arithmetic shift.)

Since we want the multiplication nm to be done inside a fixed-width word without overflow, we prefer m to be as small as possible, so we want to choose k as small as possible. Specifically, for each range of $n_{\max}$, the minimum k that allows existence of an integer m satisfying the above inequality and the corresponding smallest m (in fact, unique m) is as follows:

- ...
- $485 \le n_{\max} < 681$: $k = 12$, $m = 1233$,
- $681 \le n_{\max} < 1166$: $k = 18$, $m = 78913$,
- $1166 \le n_{\max} < 1651$: $k = 18$, $m = 78913$,
- $1651 \le n_{\max} < 2136$: $k = 20$, $m = 315653$,
- $2136 \le n_{\max} < 2621$: $k = 20$, $m = 315653$,
- $2621 \le n_{\max} < 4757$: $k = 23$, $m = 2525223$,
- $4757 \le n_{\max} < 6893$: $k = 23$, $m = 2525223$,
- ...

If we want to restrict the computation to be done entirely within (signed) 32-bit arithmetics, the maximum range of n that avoids overflow for each of the choices of m above is:

- $|n| \le \left\lfloor \frac{2^{31}-1}{m} \right\rfloor = 1741673$, for $m = 1233$,
- $|n| \le \left\lfloor \frac{2^{31}-1}{m} \right\rfloor = 27213$, for $m = 78913$,
- $|n| \le \left\lfloor \frac{2^{31}-1}{m} \right\rfloor = 6803$, for $m = 315653$, and
- $|n| \le \left\lfloor \frac{2^{31}-1}{m} \right\rfloor = 850$, for $m = 2525223$.

As a conclusion, the choice of k and m that allows the maximum range of n is $k = 20$ and $m = 315653$, which yields the formula

$$\lfloor n \log_{10} 2 \rfloor = \left\lfloor \frac{315653 \cdot n}{2^{20}} \right\rfloor,$$

which holds for all $|n| \leq 2620$. Indeed, this formula produces wrong results for $n = \pm 2621$. Note that the magic number 315653 is in fact equal to $\lceil 2^{20} \cdot \log_{10} 2 \rceil$.

This range is enough for our use-case; we need to evaluate $\lfloor e \log_{10} 2 \rfloor$ for $e \in [E_{\min} - p, E_{\max} - p]$, so $e \in [-149, 104]$ for binary32 and $e \in [-1074, 971]$ for binary64.

6.2 Computation of $\lfloor n \log_2 10 \rfloor$

We do the same thing for $\log_2 10$. The continued fraction expansion is

$$\log_2 10 = [3; 3, 9, 2, 2, 4, 6, 2, 1, 1, 3, 1, 18, 1, 6, 1, \cdots],$$

and the pair $\left(\frac{p_*}{q_*}, \frac{p^*}{q^*}\right)$ can be found accordingly:

- $\ldots$
- $643 \leq n_{\max} < 789$: $\frac{p_*}{q_*} = \frac{485}{146}$, $\frac{p^*}{q^*} = \frac{2136}{643}$,
- $789 \leq n_{\max} < 1432$: $\frac{p_*}{q_*} = \frac{2621}{789}$, $\frac{p^*}{q^*} = \frac{2136}{643}$,
- $1432 \leq n_{\max} < 2075$: $\frac{p_*}{q_*} = \frac{4757}{1432}$, $\frac{p^*}{q^*} = \frac{2136}{643}$,
- $2075 \leq n_{\max} < 2718$: $\frac{p_*}{q_*} = \frac{6893}{2075}$, $\frac{p^*}{q^*} = \frac{2136}{643}$,
- $2718 \leq n_{\max} < 3361$: $\frac{p_*}{q_*} = \frac{9029}{2718}$, $\frac{p^*}{q^*} = \frac{2136}{643}$,
- $3361 \leq n_{\max} < 4004$: $\frac{p_*}{q_*} = \frac{11165}{3361}$, $\frac{p^*}{q^*} = \frac{2136}{643}$,
- $4004 \leq n_{\max} < 4647$: $\frac{p_*}{q_*} = \frac{13301}{4004}$, $\frac{p^*}{q^*} = \frac{2136}{643}$,
- $\ldots$

Then the minimum k that allows existence of an integer m satisfying

$$\frac{2^k p_*}{q_*} < m < \frac{2^k p^*}{q^*},$$

and the smallest (in fact unique) such m for each range of $n_{\max}$ is:

- $\ldots$
- $643 \leq n_{\max} < 789$: $k = 17$, $m = 435411$,
- $789 \leq n_{\max} < 1432$: $k = 19$, $m = 1741647$,
- $1432 \leq n_{\max} < 2075$: $k = 19$, $m = 1741647$,
- $2075 \leq n_{\max} < 2718$: $k = 19$, $m = 1741647$,
- $2718 \leq n_{\max} < 3361$: $k = 19$, $m = 1741647$,
- $3361 \leq n_{\max} < 4004$: $k = 19$, $m = 1741647$,
- $4004 \leq n_{\max} < 4647$: $k = 22$, $m = 13933177$,
- $\ldots$

And the maximum range of n to avoid overflow for each of the choices of m above is:

- $|n| \leq \left\lfloor \frac{2^{31}-1}{m} \right\rfloor = 4932$, for $m = 435411$,
- $|n| \leq \left\lfloor \frac{2^{31}-1}{m} \right\rfloor = 1233$, for $m = 1741647$, and
- $|n| \leq \left\lfloor \frac{2^{31}-1}{m} \right\rfloor = 154$, for $m = 13933177$.

Therefore, the choice of k and m that allows the maximum range of n is $k = 19$ and $m = 1741647$. This yields the formula

$$\lfloor n \log_2 10 \rfloor = \left\lfloor \frac{1741647 \cdot n}{2^{19}} \right\rfloor$$

which holds for all $|n| \leq 4003$, but only for $|n| \leq 1233$ we do not have overflow over 32-bits. In this case, the magic number 1741647 is equal to $\lfloor 2^{19} \cdot \log_2 10 \rfloor$.

This range is enough for our use-case; we need to evaluate $\lfloor k \log_2 10 \rfloor$ for $k \in [k_{\min}, k_{\max}]$ where

$$k_{\min} := \min\Bigg(-\lfloor (E_{\max} - p) \log_{10} 2 \rfloor + \kappa, \\ - \left\lfloor (E_{\max} - p) \log_{10} 2 - \log_{10} \frac{4}{3} \right\rfloor \Bigg)$$

and

$$k_{\max} := \max\Bigg(-\lfloor (E_{\min} - p) \log_{10} 2 \rfloor + \kappa, \\ - \left\lfloor (E_{\min} - p) \log_{10} 2 - \log_{10} \frac{4}{3} \right\rfloor \Bigg)^{21},$$

so $k \in [-31, 46]$ for binary32 and $k \in [-292, 326]$ for binary64.

6.3 Computation of $\lfloor n \log_{10} 2 - \log_{10} \frac{4}{3} \rfloor$

We try to find positive integers k, m, and f such that

$$\left\lfloor n \log_{10} 2 - \log_{10} \frac{4}{3} \right\rfloor = \left\lfloor \frac{nm - f}{2^k} \right\rfloor$$

holds for all $|n| \leq n_{\max}$ for large enough $n_{\max}$. It is not easy to find the choice of parameters that allows the maximum possible value of $n_{\max}$, but Theorem 4.7 and Theorem 4.8 with $x \leftarrow \log_{10} 2$ and $y \leftarrow \log_{10} \frac{4}{3}$ allow us to find a good enough choice for them.

Note that the optimizers μ and ν from the mentioned theorems quite sensitively depend on $\frac{p_*}{q_*}$ and $\frac{p^*}{q^*}$, and different values for those two give radically different values for μ and ν. Then, the parameter $\zeta := \frac{f}{2^k}$ is sensitively depending on both the resulting values of μ and ν and the choice of m and k (because the bounds on ζ is given by the remainder of μm and νm divided by 2^k). Thus, simply choosing the minimum possible m and k is not the best strategy.

The author is currently not aware of any elegant starategy for finding the best choice which can be fully automated, and also did not bother to write a program doing an exhaustive search, but it seems that the following yields a good enough range of n.

As the first step, fix $\frac{p_*}{q_*} = \frac{497}{1651}$ and $\frac{p^*}{q^*} = \frac{146}{485}$, which is valid for $1651 \le n_{\max} < 2136$, so let $n_{\max} = 2135$. Hence, if we can find $\xi := \frac{m}{2^k}$ and $\zeta := \frac{f}{2^k}$ satisfying the conditions in Theorem 4.7 and Theorem 4.8, then the formula

$$\left\lfloor n \log_{10} 2 - \log_{10} \frac{4}{3} \right\rfloor = \left\lfloor \frac{nm - f}{2^k} \right\rfloor$$

must be valid for at least all $|n| \le 2135$. Note that this predicted range of n may not be the optimal range for the given specific choice of k, m, and f, and the actual range might be bigger.

We first find μ and ν given by Theorem 4.7. Note that they do not depend on ξ as long as the identity $\lfloor nx \rfloor = \lfloor n\xi \rfloor$ holds, so we can adjust ξ later on to allow valid range for ζ. Note that with out choice of $\frac{p_*}{q_*}$ and $\frac{p^*}{q^*}$, we have $\lceil q_* y \rceil = 207$ and $\lceil q^* y \rceil = 61$.

For μ, we first try $l = 1$. The modular inverse of $p_* = 497$ with respect to $q_* = 1651$ is 1166, thus we consider

$$\begin{aligned} n &\equiv 1166 \cdot (\lceil q_* y \rceil - l) \\ &\equiv 1166 \cdot (207 - 1) \equiv 801 \pmod{1651}. \end{aligned}$$

Since

$$\frac{q_* y + 1 - \lceil q_* y \rceil}{q_* x - p_*} = 523.78 \cdots$$

is less than 801, $l = 1$ does not yield any possible value for μ. Then we try $l = 2$, so

$$\begin{aligned} n &\equiv 1166 \cdot (\lceil q_* y \rceil - l) \\ &\equiv 1166 \cdot (207 - 2) \equiv 1286 \pmod{1651}. \end{aligned}$$

Since

$$\frac{q_* y + 2 - \lceil q_* y \rceil}{q_* x - p_*} = 2436.40 \cdots$$

is greater than 1286, $l = 2$ is the right answer for μ. Since 1286 is the only positive integer that is equivalent to 1286 modular 1651 and less than $2436.40 \cdots$, we get $\mu = 1286$.

For ν, we already have seen in the above that the maximum l such that there exists a solution for

$$\begin{aligned} np_* &\equiv \lceil q_* y \rceil - l \pmod{q_*} \quad \text{and} \\ n &\ge \frac{q_* y + l - \lceil q_* y \rceil}{q_* x - p_*}, \end{aligned}$$

exists is $l = 1$ (because $\frac{q_* y + l - \lceil q_* y \rceil}{q_* x - p_*}$ is already bigger than $n_{\max}$ for $l = 2$), which gives $\nu = 801$.

Next, we find μ and ν given by Theorem 4.8. The modular inverse of $p^* = 146$ with respect to $q^* = 485$ is 196, so we try $l = 1$ which yields

$$\begin{aligned} n &\equiv 196 \cdot (q^* - \lceil q^* y \rceil + l) \\ &\equiv 196 \cdot (485 - 61 + 1) \equiv 365 \pmod{1651}. \end{aligned}$$

Note that

$$\frac{q^* y + 1 - \lceil q^* y \rceil}{p^* - q^* x} = 1316.70 \cdots$$

is greater than 365, so we should have $\mu \equiv 365 \pmod{1651}$. The greatest possible choice of μ is then

$$\mu = 365 + 485 = 850.$$

For ν, note that

$$\frac{q_* y + 2 - \lceil q_* y \rceil}{q_* x - p_*} = 3528.59 \cdots$$

is bigger than $n_{\max}$, thus we need to take $l \le 1$. When $l = 1$, the equation

$$\begin{aligned} n &\equiv 365 \pmod{1651}, \\ n &\ge \frac{q^* y + 1 - \lceil q^* y \rceil}{p^* - q^* x} = 1316.70 \cdots \end{aligned}$$

has a solution, and the smallest solution is $\nu = 1335$.

Now, let us take $k = 21$ and

$$m = \left\lceil \frac{2^k p_*}{q_*} \right\rceil = 631305.$$

Then $\xi = \frac{m}{2^k}$ satisfies $\lfloor nx \rfloor = \lfloor n\xi \rfloor$ for all $|n| \le n_{\max}$, since

$$\left\lfloor \frac{2^k p^*}{q^*} \right\rfloor = 631307.$$

With these, the bound on $f = 2^k \zeta$ given by Theorem 4.7 is

$$\begin{aligned} (\mu m \bmod 2^k) = 260406 &< f \\ &\le (\nu m \bmod 2^k) = 261673, \end{aligned}$$

and the bound given by Theorem 4.8 is

$$\begin{aligned} 2^k - (\mu m \bmod 2^k) = 261662 &< f \\ &\le 2^k - (\nu m \bmod 2^k) = 262929. \end{aligned}$$

Therefore, the resulting range of f is

$$261663 \le f \le 261673.$$

Therefore, we can choose whatever value of f from this range to ensure that the formula

$$\left\lfloor n \log_{10} 2 - \log_{10} \frac{4}{3} \right\rfloor = \left\lfloor \frac{631305 \cdot n - f}{2^{21}} \right\rfloor$$

is valid for all $|n| \le n_{\max} = 2135$ which is enough for our use, so we just choose $f = 261663$. Actually, a mechanical verification shows that every choice of f in the above range makes the above formula valid for all $n \in [-2985, 2936]$.

Note that our first magic number 631305 is precisely $\lfloor 2^{21} \log_{10} 2 \rfloor$, but the second magic number 261663 is quite different from

$$2^{21} \log_{10} \frac{4}{3} = 262015.52 \cdots .$$

In fact, the maximum range of n we can get by just naively assigning the truncated binary expansions of $\log_{10} 2$ and $\log_{10} \frac{4}{3}$ into m and f, constrained by the possible presence of overflow over 32-bit signed arithmetic, is $|n| \leq 1700$, which is achieved by the choice $k = 22$, $m = \lfloor 2^{22} \log_{10} 2 \rfloor = 1262611$, and $f = \lfloor 2^{22} \log_{10} \frac{4}{3} \rfloor = 524031$. Hence, our analysis based on Theorem 4.7 and Theorem 4.8 gives a strictly better result, although $|n| \leq 1700$ is already enough for our use.

7. Performance

We compared the performance of Dragonbox with the reference implementations of Ryū [13], Grisu-Exact [14], and a C++ implementation of Schubfach [7], for the task of producing a decimal string representation of a given floating-point number. The source code for the benchmark is available in [11].

We did two set of benchmarks. The first set is testing floating-point numbers with the given number of decimal digits. (See Figure 7.) Since it is not easy to uniformly randomly generate such floating-point numbers, we first uniformly randomly generated an integer with the given number of digits, combined it with a uniformly randomly generated exponent in the valid decimal exponent range and a uniformly randomly generated sign, converted the result into a string, and then converted it back to a floating-point number. If the resulting string does not fall in the valid range or if there exists a shorter representation of the same floating-point number, then we discarded he number and repeated the procedure. Although this will not give us the uniform distribution as the probability of collision will not be uniform, one may nonethelessly claim that this will give a reasonable approximation. We generated $100,000$ samples per each number of digits, and measured the time elapsed for repeating the string generation $1,000$ times for each sample.

The second set is testing uniformly randomly generated floating-point numbers. (See Figure 8.) For this benchmark, we generated $1,000,000$ samples and measured the time elapsed for repeating the task $1,000$ times for each sample. Since $1,000,000$ samples are too many to make a visible plot, we randomly sampled $10,000$ among them for the plot shown in Figure 8. The statistics attached on the plot is drawn from all of $1,000,000$ samples.

The benchmark data is obtained on a machine with Intel (R) Core™ i7-7700HQ CPU @2.80GHz, and the benchmark code is compiled with Clang-cl compiler shipped with Visual Studio 2022 17.0.4.

In our benchmarks, Dragonbox performed better than the competitors for all number of digits and also for the uniformly random data.

(Note: just like every other benchmark, this benchmark is not perfectly fair. Especially, the benchmark measures the total time consumed for generating strings rather than the decimal significand and exponents, and all of these implementations use different algorithm for string generation. In [7], however, one can find more fair comparison of these algorithms.)

A. Right-Closed Directed Rounding Case

In this section, we describe the algorithm for the case when the interval I is given as

$$I = (w^-, w].$$

In this case, there are not so much differences between the normal interval case and the shorter interval case, so we will not treat them differently. One thing to note for this case is that when we know that $I \cap 10^{-k_0+1}\mathbb{Z}$ is empty, we just need to find the greatest integer from $10^{k_0} I$, which can be done directly by just adding the quotient of r divided by 10^κ to $10s$. Besides those, there are not so much differences from the nearest rounding case. Here is the skeleton:

Algorithm A.1 (Skeleton of Dragonbox, Right-Closed Directed Rounding Case)**.**

1. Compute $k = -\lfloor \log_{10} \Delta \rfloor + \kappa$ as described in Section 6.1. But in this case, we need to be careful that $\Delta = 2^{e-1}$ if $F_w = 1$ and $E \neq E_{\min}$, and $\Delta = 2^e$ otherwise.
2. Compute $z^{(i)}$, as described in Section 5.1.5
3. Compute s, r by dividing $z^{(i)}$ by $10^{\kappa+1}$ with the optimization described in Section 5.1.7.
4. Compute $\delta^{(i)}$ as described in Section 5.1.4. But in this case, again we need to take care of the presence of the shorter interval case. The only difference is, however, that we need to shift by one less amount of bits compared to the normal interval case.
5. Check if the inequality $r > \delta^{(i)}$ holds. If that is the case, then we conclude that $I \cap 10^{-k_0+1}\mathbb{Z}$ is empty.
6. Otherwise, check if the inequality $r < \delta^{(i)}$ holds. If that is the case, then we conclude that $10^{-k+\kappa+1}s$ is the unique element in $I \cap 10^{-k_0+1}\mathbb{Z}$.
7. Otherwise, we have $r = \delta^{(i)}$. Then, compute the parity of $x^{(i)}$, as described in Section 5.1.6. Again, we need to take care of the presence of the shorter interval case, since we have

   $$x = (f_c - 1) \cdot 2^e \cdot 10^k = (2f_c - 2) \cdot 2^{e-1} \cdot 10^k$$

   for the normal interval case but we have

   $$x = \left(f_c - \frac{1}{2}\right) \cdot 2^e \cdot 10^k = (2f_c - 1) \cdot 2^{e-1} \cdot 10^k$$

   for the shorter interval case.
   - If $x^{(i)}$ is an odd number, then we have $z^{(f)} < \delta^{(f)}$, so we conclude that $10^{-k+\kappa+1}s$ is the unique element in $I \cap 10^{-k_0+1}\mathbb{Z}$.
   - Otherwise, we conclude that $I \cap 10^{-k_0+1}\mathbb{Z}$ is empty.
8. When we have concluded that $10^{-k+\kappa+1}s$ is the unique element in $I \cap 10^{-k_0+1}\mathbb{Z}$, then we might need to remove

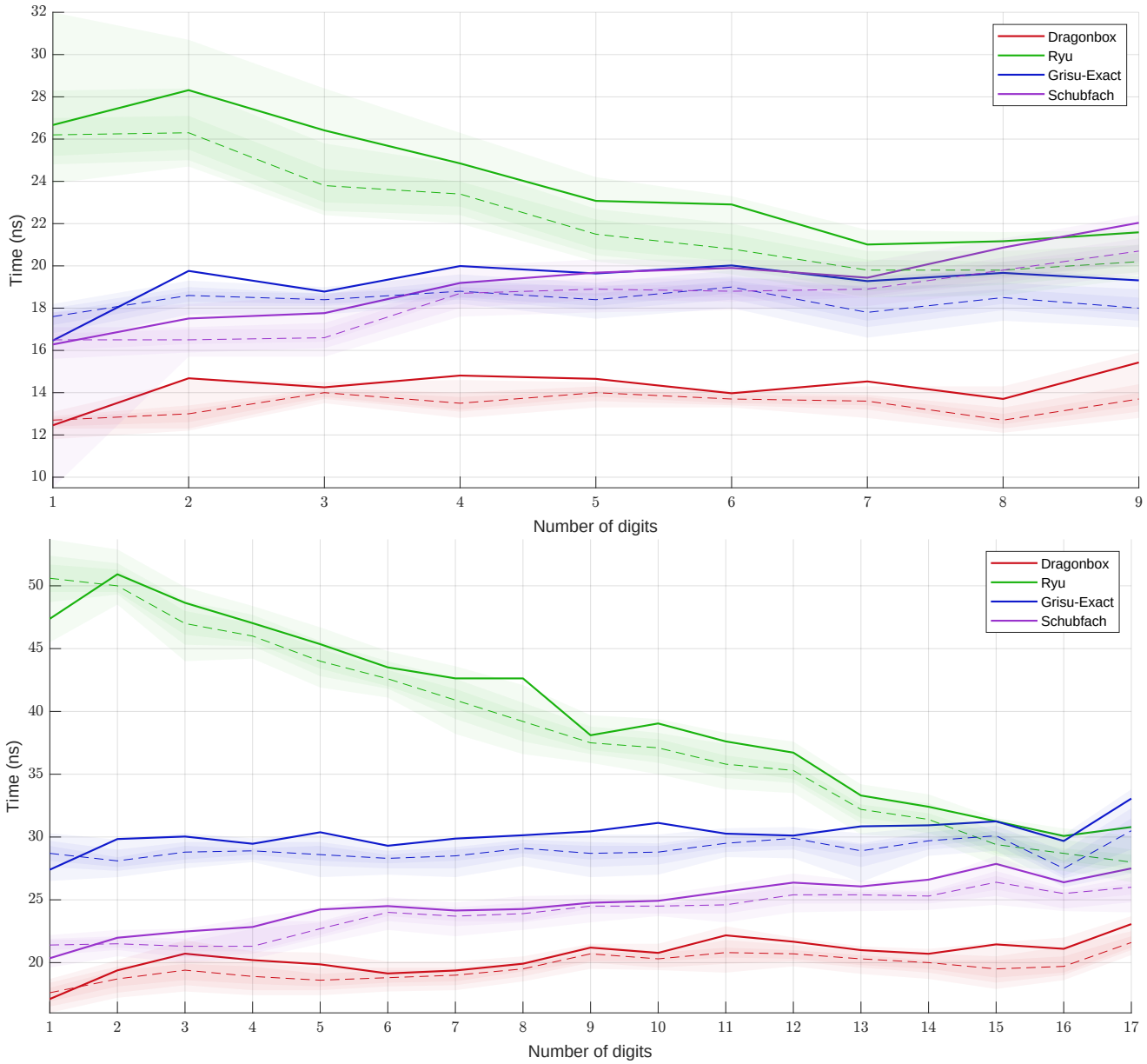

Figure 7. Performances of Dragonbox, Ryū, Grisu-Exact, and Schubfach for random floating-point numbers with given number of digits; solid lines are averages, dashed lines are medians, and shaded regions show 30%, 50%, and 70% percentiles. (top: binary32, bottom: binary64)

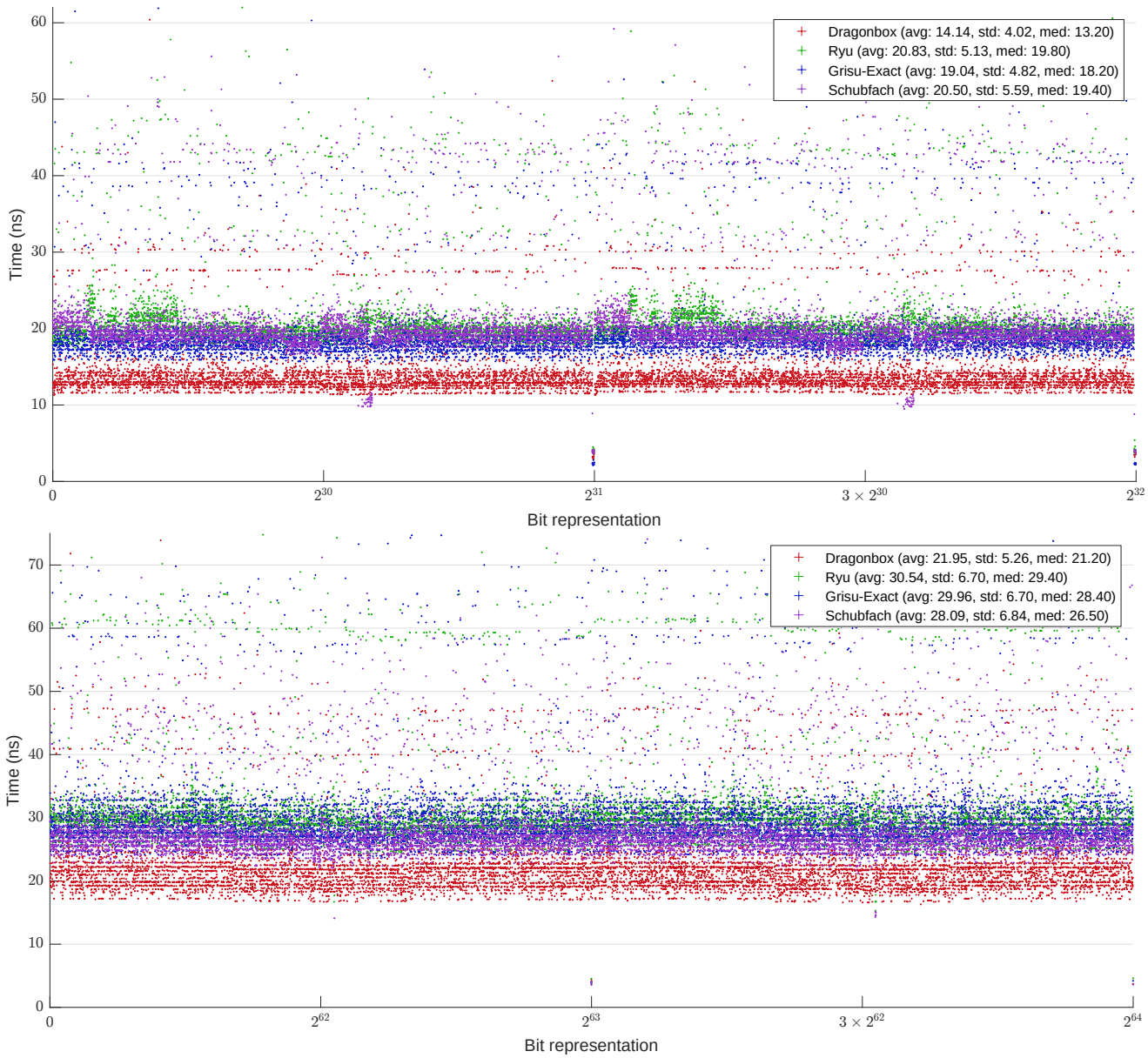

Figure 8. Performances of Dragonbox, Ryū, Grisu-Exact, and Schubfach for uniform random floating-point numbers (top: binary32, bottom: binary64)

trailing zeros from s. Find the greatest integer d such that 10^d divides s. Then we conclude that

$$\frac{s}{10^d} \times 10^{-k+\kappa+1+d}$$

is the answer we are looking for.

9. When we have concluded that $I \cap 10^{-k_0+1}\mathbb{Z}$ is empty, then $(10s+t) \times 10^{-k+\kappa}$ is the answer we are looking for, where $t := \left\lfloor \frac{r}{10^\kappa} \right\rfloor$.

B. Left-Closed Directed Rounding Case

In this section, we describe the algorithm for the case when the interval I is given as

$$I = [w, w^+).$$

In this case, the strategy is to take the mirror image of the algorithm explained in Section A. This is a little bit more complex than the right-closed directed rounding case, but a good thing is that we do not have the shorter interval case; we always have $\Delta = 2^e$ and $w^+ = (f_c + 1) \cdot 2^e$. Here is the skeleton:

Algorithm B.1 (Skeleton of Dragonbox, Left-Closed Directed Rounding Case)**.**

1. Compute $k = -\lfloor \log_{10} \Delta \rfloor + \kappa$ as described in Section 6.1.
2. Compute $x^{(i)}$, as described in Section 5.1.5. Note that we can still apply the completely same routine to x rather than z.
3. Check if x is an integer; define
$$\tilde{x}^{(i)} = \begin{cases} x^{(i)} & \text{if } x \text{ is an integer} \\ x^{(i)} + 1 & \text{if } x \text{ is not an integer} \end{cases}.$$
Note that $\tilde{x}^{(i)}$ is nothing but the ceiling of x. To check if x is an integer, we can apply the method described in Section 5.1.5. However, the exceptional cases for binary32 mentioned in Section 5.1.2 actually matters here, so we need to add a branch for them. Since x cannot be an integer if e is small enough, checking if $e \le -80$ is enough.
4. Compute the unique integers $\tilde{s}, \tilde{r}$ satisfying
$$\tilde{x}^{(i)} = 10^{\kappa+1}\tilde{s} - \tilde{r}, \quad 0 \le \tilde{r} < 10^{\kappa+1}.$$
This requires a little modification to the plain division:
$$\tilde{s} = \begin{cases} \left\lfloor \frac{\tilde{x}^{(i)}}{10^{\kappa+1}} \right\rfloor & \text{if } 10^{\kappa+1} \text{ divides } \tilde{x}^{(i)} \\ \left\lfloor \frac{\tilde{x}^{(i)}}{10^{\kappa+1}} \right\rfloor + 1 & \text{otherwise} \end{cases}.$$
(This is again nothing but the ceiling.) The optimization described in Section 5.1.7 still applies.
5. Compute $\delta^{(i)}$ as described in Section 5.1.4.
6. Check if the inequality $r > \delta^{(i)}$ holds. If that is the case, then we conclude that $I \cap 10^{-k_0+1}\mathbb{Z}$ is empty.
7. Otherwise, check if the inequality $r < \delta^{(i)}$ holds. If that is the case, then we conclude that $10^{-k+\kappa+1}\tilde{s}$ is the unique element in $I \cap 10^{-k_0+1}\mathbb{Z}$.
8. Otherwise, we have $r = \delta^{(i)}$. Then, compute the parity of $z^{(i)}$, as described in Section 5.1.6. Again, no further modification is needed and we can just apply what is described in Section 5.1.6 to $z^{(i)}$ as well.
   - If $z^{(i)}$ is an odd number, then $I \cap 10^{-k_0+1}\mathbb{Z}$ is empty.
   - If $z^{(i)}$ is an even number, then check if z is an integer. In this case, we do not need a special treatment for the exceptional cases mentioned in Section 5.1.2, because for the problematic inputs we do not have $r = \delta^{(i)}$. If z is an integer, then we conclude that $I \cap 10^{-k_0+1}\mathbb{Z}$ is empty. Otherwise, we conclude that $10^{-k+\kappa+1}\tilde{s}$ is the unique element in $I \cap 10^{-k_0+1}\mathbb{Z}$.
9. When we have concluded that $10^{-k+\kappa+1}\tilde{s}$ is the unique element in $I \cap 10^{-k_0+1}\mathbb{Z}$, then we might need to remove trailing zeros from $\tilde{s}$. Find the greatest integer d such that 10^d divides $\tilde{s}$. Then we conclude that
$$\frac{\tilde{s}}{10^d} \times 10^{-k+\kappa+1+d}$$
is the answer we are looking for.
10. When we have concluded that $I \cap 10^{-k_0+1}\mathbb{Z}$ is empty, then $(10\tilde{s} - t) \times 10^{-k+\kappa}$ is the answer we are looking for, where $t := \left\lfloor \frac{\tilde{r}}{10^\kappa} \right\rfloor$.

To elaborate more on the step 8, let us define
$$\tilde{x}^{(f)} := \tilde{x}^{(i)} - x,$$
then $0 \le \tilde{x}^{(f)} < 1$. Then
$$z^{(i)} + z^{(f)} = x + \delta = \tilde{x}^{(i)} + \delta^{(i)} + (\delta^{(f)} - \tilde{x}^{(f)}).$$
Now, if $\delta^{(i)} = \tilde{r}$, then
$$z^{(i)} + z^{(f)} = 10^{\kappa+1}\tilde{s} + (\delta^{(f)} - \tilde{x}^{(f)}),$$
thus $z^{(i)}$ is an odd number if and only if $\tilde{x}^{(f)} > \delta^{(f)}$. In this case, we have that $\tilde{s}$ is not in $10^{k_0-1}I$, so $I \cap 10^{-k_0+1}\mathbb{Z}$ is empty.

When $z^{(i)}$ is an even number, then $\tilde{x}^{(f)} \le \delta^{(f)}$. In this case, we have $\tilde{s} \in 10^{k_0-1}I$ if and only if $\tilde{x}^{(f)} > \delta^{(f)}$, so we need to check if $\tilde{x}^{(f)} = \delta^{(f)}$, which is the case if and only if z is an integer.

C. Continued fractions

In this section, we collect some relevant results from the theory of continued fractions. Many materials here are compiled from several references ([15] for example). We did not try very hard to find references for all possibly known results, so lack of reference does not mean novelty. It can be possible that some results here might be not very widely known, but most of the things here are probably standard.

A *continued fraction* means either a finite or infinite sequence of the form
$$a_0,\ a_0 + \frac{b_1}{a_1},\ a_0 + \cfrac{b_1}{a_1 + \cfrac{b_2}{a_2}},\ a_0 + \cfrac{b_1}{a_1 + \cfrac{b_2}{a_2 + \cfrac{b_3}{a_3}}},\ \cdots.$$

In this paper, we will always assume:

1. All of b_i's are equal to 1,
2. a_0 is an integer, and
3. All of other a_i's are positive integers,

unless stated otherwise. With this convention, there is a unique continued fraction determined by a finite or infinite sequence $a_0, a_1, a_2, \cdots$. We formally denote
$$[a_0; a_1, a_2, \cdots] := a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\cdots}}}.$$

When the sequence is finite, the right-hand side evaluates to a rational number. Even when the sequence is infinite, it can be shown that the right-hand side, understood as the limit of

the sequence $([a_0; a_1, \cdots, a_i])_{i=0}^{\infty}$ of rational numbers, always converges to a real number (see below). Thus, the formal expression $[a_0; a_1, a_2, \cdots]$ actually refers to a unique real number, and we will not make any distinction between these two.

For each i such that $a_0, \cdots, a_i$ are well-defined, the *ith convergent* of the continued fraction is the rational number $[a_0; a_1, \cdots, a_i]$ obtained by cutting the sequence at the ith term. For example, the 5th convergent of $[0; 1, 2, 3, 4, \cdots]$ is

$$0 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{3 + \cfrac{1}{4 + \cfrac{1}{5}}}}} = \frac{157}{225}.$$

We can uniquely express the ith convergent as $\frac{p_i}{q_i}$, where q_i is a positive integer and p_i is an integer coprime to q_i. Then we get the following very well-known result:

Lemma C.1.
For each i, we have

$$\begin{pmatrix} p_i & p_{i-1} \\ q_i & q_{i-1} \end{pmatrix} = \begin{pmatrix} a_0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a_1 & 1 \\ 1 & 0 \end{pmatrix} \cdots \begin{pmatrix} a_i & 1 \\ 1 & 0 \end{pmatrix}$$

where we set $p_{-1} := 1$ and $q_{-1} := 0$.

Proof. Use induction; see [15] for details. □

Looking at the formula above, it is natural to define $p_{-2} := 0$ and $q_{-2} := 1$, then we get

$$\begin{pmatrix} p_i & p_{i-1} \\ q_i & q_{i-1} \end{pmatrix} = \begin{pmatrix} p_{i-1} & p_{i-2} \\ q_{i-1} & q_{i-2} \end{pmatrix} \begin{pmatrix} a_i & 1 \\ 1 & 0 \end{pmatrix},$$

so

$$\begin{cases} p_i = p_{i-2} + a_i p_{i-1}, \\ q_i = q_{i-2} + a_i q_{i-1}. \end{cases}$$

Also, since the determinant of the matrix $\begin{pmatrix} a_i & 1 \\ 1 & 0 \end{pmatrix}$ is -1, we get

$$p_i q_{i-1} - p_{i-1} q_i = (-1)^{i+1}. \tag{11}$$

The recurrence relation shows that q_i strictly increases exponentially. Also, (11) shows

$$\frac{p_i}{q_i} - \frac{p_{i-1}}{q_{i-1}} = \frac{(-1)^{i+1}}{q_i q_{i-1}},$$

so we inductively get

$$\frac{p_i}{q_i} = a_0 + \sum_{j=0}^{i-1} \frac{(-1)^j}{q_j q_{j+1}}.$$

In particular, by the alternating series test, we conclude:

Corollary C.2.
We have

$$\frac{p_0}{q_0} < \frac{p_2}{q_2} < \cdots < \frac{p_3}{q_3} < \frac{p_1}{q_1}.$$

Also, if the sequence $(a_i)_i$ is infinite, then the sequence $\left(\frac{p_i}{q_i}\right)_{i=0}^{\infty}$ of convergents converges to a real number.

Conversely, for a given real number x, there is a well-known algorithm for obtaining a continued fraction converging to x. First, take the integer part $a_0 := \lfloor x \rfloor$. Then $x - a_0 \in [0, 1)$. If it is zero, then that means x is an integer, and we terminate the procedure and returns the sequence (a_0) consisting of the single term a_0. If not, then consider the reciprocal $x_1 := \frac{1}{x - a_0} \in (1, \infty)$. Take the integer part $a_1 := \lfloor x_1 \rfloor$, then again we have $x_1 - a_1 \in [0, 1)$. If it is zero, then stop and returns the sequence $(a_0, [a_0; a_1])$ consisting of two terms. Otherwise, take the reciprocal and let $x_2 := \frac{1}{x_1 - a_1} \in (1, \infty)$. Again, take the integer part $a_2 := \lfloor x_2 \rfloor$, and if $x_2 - a_2 = 0$, then stop and returns the sequence $(a_0, [a_0; a_1], [a_0; a_1, a_2])$ consisting of three terms. Otherwise, take the reciprocal $x_3 := \frac{1}{x_2 - a_2}$ and continue.

To see why the resulting continued fractions converge to x, it is sufficient to see $\frac{p_{2i}}{q_{2i}} \le x$ and $x \le \frac{p_{2i+1}}{q_{2i+1}}$ always holds, because of Corollary C.2. Note that by induction one can easily see

$$x = [a_0; a_1, \cdots, a_i, x_{i+1}]$$

always holds (here we are allowing non-integers to appear in continued fractions), so it is enough to show that for given positive real numbers $a < b$,

$$[a_0; a_1, \cdots, a_{2i}, a] < [a_0; a_1, \cdots, a_{2i}, b]$$

and

$$[a_0; a_1, \cdots, a_{2i+1}, a] > [a_0; a_1, \cdots, a_{2i+1}, b]$$

holds. These can be easily shown using induction.

Note that if this algorithm terminates in a finite step, then that means the final term in the resulting sequence is precisely the number x we are given with. In particular, x must be a rational number in this case, so if x is irrational then the algorithm produces an infinite sequence.

On the other hand, if $x = \frac{p}{q}$ is rational, then what the algorithm does is basically what the Euclid algorithm does to p and q to eventually produce $\gcd(p, q) = 1$. With this correspondence, one can see that whenever x is rational the algorithm must terminate in a finite step. Therefore, the algorithm terminates in a finite step if and only if x is rational.

Now, suppose we have $x = [a_0; a_1, \cdots]$. Then, we must have

$$a_0 \le x < a_0 + 1$$

unless either a_0 is the last term or a_1 is the last term and $a_1 = 1$. Since $[a_0; a_1, \cdots] = [a_0; b_1, \cdots]$ implies $[a_1; a_2, \cdots] = [b_1; b_2, \cdots]$, we get that two continued

fractions $[a_0; a_1, \cdots]$ and $[b_0; b_1, \cdots]$ yield the same real number if and only if:

1. They are identical sequences, or
2. Both are finite continued fractions with the lengths differ by 1, and assuming $[a_0; a_1, \cdots] = [a_0; a_1, \cdots, a_i]$ is the shorter one, $a_j = b_j$ holds for all $j < i$ and $a_i = b_i + 1$, $b_{i+1} = 1$ hold.

In other words,

1. When x is irrational, the continued fraction generated by the algorithm described above is the unique one convergent to x, and
2. When x is rational, there is another one which is obtained by replacing the last coefficient a_i with $[a_i - 1; 1]$. Note that whenever the algorithm terminates in a finite step, the last coefficient must be at least 2.

From now on, we will call the continued fraction $[a_0; a_1, \cdots]$ obtained from the algorithm as *the continued fraction expansion of* x, and the ith convergent of it as *the ith convergent of* x.

Given a continued fraction $[a_0; a_1, \cdots]$, whenever we say "the ith coefficient" of "the ith convergent" or anything like that, we are always assuming implicitly that those are well-defined. In particular, the index i will be always assumed to be at least 0. However, sometimes it is convenient to talk about "the -1st convergent" which is conventionally considered as ∞. In a similar spirit, we often view "the coefficient next to the last coefficient" as being ∞, and the one next to it being 0, and the one next to it being ∞, and so on. Then it is natural to conceptually think that, if $\frac{p_i}{q_i}$ is the last convergent, then p_{i+1} and q_{i+1} or both infinite, but if we take the quotient than it becomes equal to $\frac{p_i}{q_i}$. Then, we have $p_{i+2} = p_i$ and $q_{i+2} = q_i$, and then $p_{i+3} = p_{i+1} = \infty$ and $q_{i+3} = q_{i+1} = \infty$ and $\frac{p_{i+3}}{q_{i+3}} = \frac{p_i}{q_i}$, and then again $p_{i+4} = p_i$ and $q_{i+4} = q_i$, and so on, although precisely speaking these all do not make sense.

C.1 Best rational approximations from below/above

We characterize best rational approximations from below/above (Definition 4.1) defined in Section 4.

Definition C.3 (Semiconvergents)**.**
Let $\frac{p_i}{q_i}$ denote the ith convergent of a given continued fraction $x = [a_0; a_1, \cdots]$. Then for given two consecutive convergents $\frac{p_{i-1}}{q_{i-1}}$, $\frac{p_i}{q_i}$ (allowing $i = 0$), a rational number of the form

$$\frac{p_{i-1} + sp_i}{q_{i-1} + sq_i}$$

for an integer $s \in [0, a_{i+1}]$ with the convention $a_{i+1} = \infty$ if a_{i+1} is not defined (in which case $s = a_{i+1}$ is impossible), is called a *semiconvergent*. When $a_{i+1} < \infty$, or when $s = 0$ and $q_{i-1} \neq 0$ (which means $i \neq 0$), we call $\frac{p_{i-1}+sp_i}{q_{i-1}+sq_i}$ a *proper semiconvergent*.[22]

When i is even, we call $\frac{p_i}{q_i}$ an *even convergent*, and when i is odd, we call it an *odd convergent*. Similarly, any semiconvergent of the form

$$\frac{p_{2i} + sp_{2i+1}}{q_{2i} + sq_{2i+1}}$$

is called an *even semiconvergent*, and any semiconvergent of the form

$$\frac{p_{2i-1} + sp_{2i}}{q_{2i-1} + sq_{2i}}$$

is called an *odd semiconvergent*.

The following lemma is well-known.

Lemma C.4.
Let $\frac{p_i}{q_i}$ denote the ith convergent of a given continued fraction $x = [a_0; a_1, \cdots]$. Then for a given two consecutive convergents $\frac{p_{i-1}}{q_{i-1}}$, $\frac{p_i}{q_i}$, as s varies from 0 to a_{i+1}, the semiconvergent

$$\frac{p_{i-1} + sp_i}{q_{i-1} + sq_i}$$

strictly increases from $\frac{p_{i-1}}{q_{i-1}}$ to $\frac{p_{i+1}}{q_{i+1}}$ when i is odd, and it strictly decreases from $\frac{p_{i-1}}{q_{i-1}}$ to $\frac{p_{i+1}}{q_{i+1}}$ when i is even, where we think $\frac{p_{i+1}}{q_{i+1}}$ as being equal to x if $\frac{p_{i+1}}{q_{i+1}}$ is not defined.

Proof. Consider the function

$$f(s) := \frac{p_{i-1} + sp_i}{q_{i-1} + sq_i}$$

of real variable s, then from the identity

$$p_i q_{i-1} - p_{i-1} q_i = (-1)^{i+1}$$

it can be easily verified that $f'(s) > 0$ when i is odd and $f'(s) < 0$ when i is even, on $s > 0$. Hence, the claim follows. □

In fact, $s = a_{i+1}$ is the last s such that the order between x and $\frac{p_{i-1}+sp_i}{q_{i-1}+sq_i}$ is retained; that is, we have

$$x \leq \frac{p_{i-1} + (a_{i+1}+1)p_i}{q_{i-1} + (a_{i+1}+1)q_i} = \frac{p_i + p_{i+1}}{q_i + q_{i+1}}$$

when i is odd, and

$$x \geq \frac{p_{i-1} + (a_{i+1}+1)p_i}{q_{i-1} + (a_{i+1}+1)q_i} = \frac{p_i + p_{i+1}}{q_i + q_{i+1}}$$

when i is even. To see why, note that we are implicitly assuming $a_{i+1} < \infty$ here, since otherwise the above is a nonsense. If $a_{i+2} = \infty$, then $x = \frac{p_{i+1}}{q_{i+1}} = \frac{p_{i-1}+a_{i+1}p_i}{q_{i-1}+a_{i+1}q_i}$ so the

[22] The term *proper semiconvergent* is not a standard terminology.

desired statement obviously holds with strict inequalities. If $a_{i+2} < \infty$, then

$$\begin{aligned}
&\frac{p_i + p_{i+1}}{q_i + q_{i+1}} - \frac{p_{i+2}}{q_{i+2}} \\
&\quad = \frac{(p_i q_{i+2} - p_{i+2} q_i) + (p_{i+1} q_{i+2} - p_{i+1} q_{i+2})}{(q_i + q_{i+1}) q_{i+2}} \\
&\quad = \frac{a_{i+2}(p_i q_{i+1} - p_{i+1} q_i) + (-1)^i}{(q_i + q_{i+1}) q_{i+2}} \\
&\quad = \frac{(-1)^i (a_{i+2} - 1)}{(q_i + q_{i+1}) q_{i+2}} \geq 0.
\end{aligned}$$

Since $x \leq \frac{p_{i+2}}{q_{i+2}}$ when i is odd and $x \geq \frac{p_{i+2}}{q_{i+2}}$ when i is even, we get

$$x \leq \frac{p_{i-1} + (a_{i+1} + 1) p_i}{q_{i-1} + (a_{i+1} + 1) q_i} = \frac{p_i + p_{i+1}}{q_i + q_{i+1}}$$

when i is odd, and

$$x \geq \frac{p_{i-1} + (a_{i+1} + 1) p_i}{q_{i-1} + (a_{i+1} + 1) q_i} = \frac{p_i + p_{i+1}}{q_i + q_{i+1}}$$

when i is even, with the equalities if and only if $x = \frac{p_{i+2}}{q_{i+2}}$ and $a_{i+2} = 1$.

Note that this equality condition means that x is rational and the continued fraction $[a_0; a_1, \cdots]$ ended with 1, which can never happen if a_i's are generated from the algorithm we described. Hence, we get:

Lemma C.5.
Let $\frac{p_i}{q_i}$ denote the ith convergent of a given continued fraction $x = [a_0; a_1, \cdots]$. Then

$$a_{i+1} = \left\lfloor \frac{q_{i-1} x - p_{i-1}}{p_i - q_i x} \right\rfloor$$

holds for each i.

Proof. Follows directly from the discussion above together with that

$$\frac{p_{2i} + s p_{2i+1}}{q_{2i} + s q_{2i+1}} \leq x$$

is equivalent to

$$s \leq \frac{q_{2i} x - p_{2i}}{p_{2i+1} - q_{2i+1} x}$$

and

$$\frac{p_{2i-1} + s p_{2i}}{q_{2i-1} + s q_{2i}} \geq x$$

is equivalent to

$$s \leq \frac{p_{2i-1} - q_{2i-1} x}{q_{2i} x - p_{2i}}.$$

□

The following simple computational result is useful to keep in mind:

Lemma C.6.
Let $\frac{p_{i-1}}{q_{i-1}}, \frac{p_i}{q_i}$ be two consecutive convergents of a continued fraction $[a_0; a_1, \cdots]$. Then for any two consecutive semiconvergents

$$\frac{p_{i-1} + s p_i}{q_{i-1} + s q_i} \quad \textit{and} \quad \frac{p_{i-1} + (s+1) p_i}{q_{i-1} + (s+1) q_i},$$

we have

$$\begin{aligned}
&\frac{p_{i-1} + (s+1) p_i}{q_{i-1} + (s+1) q_i} - \frac{p_{i-1} + s p_i}{q_{i-1} + s q_i} \\
&\qquad = \frac{(-1)^{i+1}}{(q_{i-1} + s q_i)(q_{i-1} + (s+1) q_i)}
\end{aligned}$$

and

$$\frac{p_i}{q_i} - \frac{p_{i-1} + s p_i}{q_{i-1} + s q_i} = \frac{(-1)^{i+1}}{q_i (q_{i-1} + s q_i)}.$$

Proof. Follows directly from (11). □

Now we show that proper semiconvergents are precisely the best rational approximations from below/above.

Lemma C.7.
Let $\frac{p_i}{q_i}$ denote the ith convergent of a given continued fraction $x = [a_0; a_1, \cdots]$. Then for any $y \in [\lfloor x \rfloor, x)$, there uniquely exists a pair of two consecutive even semiconvergents $\frac{p_{2i} + s p_{2i+1}}{q_{2i} + s q_{2i+1}}$ and $\frac{p_{2i} + (s+1) p_{2i+1}}{q_{2i} + (s+1) q_{2i+1}}$ such that

$$\frac{p_{2i} + s p_{2i+1}}{q_{2i} + s q_{2i+1}} \leq y < \frac{p_{2i} + (s+1) p_{2i+1}}{q_{2i} + (s+1) q_{2i+1}}.$$

Similarly, for any $z \in (x, \infty)$, there uniquely exists a pair of two consecutive odd semiconvergents $\frac{p_{2i-1} + s p_{2i}}{q_{2i-1} + s q_{2i}}$ and $\frac{p_{2i-1} + (s+1) p_{2i}}{q_{2i-1} + (s+1) q_{2i}}$ such that

$$\frac{p_{2i-1} + (s+1) p_{2i}}{q_{2i-1} + (s+1) q_{2i}} < z \leq \frac{p_{2i-1} + s p_{2i}}{q_{2i-1} + s q_{2i}}.$$

Proof. By Corollary C.2, y must either lie in between two consecutive even convergents $\frac{p_{2i}}{q_{2i}}$ and $\frac{p_{2i+2}}{q_{2i+2}}$, that is,

$$\frac{p_{2i}}{q_{2i}} \leq y < \frac{p_{2i+2}}{q_{2i+2}},$$

or lie in between the last even convergent $\frac{p_{2i}}{q_{2i}}$ and $x = \frac{p_{2i+1}}{q_{2i+1}}$, that is,

$$\frac{p_{2i}}{q_{2i}} \leq y < \frac{p_{2i+1}}{q_{2i+1}} = x.$$

Then for either case, Lemma C.4 shows that there uniquely exists an integer $s \in [0, a_{2i+2})$ such that

$$\frac{p_{2i} + s p_{2i+1}}{q_{2i} + s q_{2i+1}} \leq y < \frac{p_{2i} + (s+1) p_{2i+1}}{q_{2i} + (s+1) q_{2i+1}}$$

holds.

Similarly, with the convention $\frac{p_{-1}}{q_{-1}} = \infty$, z must satisfy either

$$\frac{p_{2i+1}}{q_{2i+1}} < z \leq \frac{p_{2i-1}}{q_{2i-1}}$$

or

$$x = \frac{p_{2i}}{q_{2i}} < z \leq \frac{p_{2i-1}}{q_{2i-1}},$$

and for either case we can find a unique integer $s \in [0, a_{2i+2})$ such that

$$\frac{p_{2i-1} + (s+1)p_{2i}}{q_{2i-1} + (s+1)q_{2i}} < z \leq \frac{p_{2i-1} + sp_{2i}}{q_{2i-1} + sq_{2i}}.$$

□

Proposition C.8.
Let $x = [a_0; a_1, \cdots]$ be the continued fraction expansion of a given real number x. Then every even proper semiconvergent of x is a best rational approximation of x from below, and every best rational approximation of x from below except possibly x itself is an even proper semiconvergent. Similarly, every odd proper semiconvergent of x is a best rational approximation of x from above, and every best rational approximation of x from above except possibly x itself is an odd proper semiconvergent.

Proof. Take any two consecutive even semiconvergents $\frac{p_{2i}+sp_{2i+1}}{q_{2i}+sq_{2i+1}}$ and $\frac{p_{2i}+(s+1)p_{2i+1}}{q_{2i}+(s+1)q_{2i+1}}$ and think of any rational number $\frac{p}{q}$ in between; that is,

$$\frac{p_{2i} + sp_{2i+1}}{q_{2i} + sq_{2i+1}} \leq \frac{p}{q} < \frac{p_{2i} + (s+1)p_{2i+1}}{q_{2i} + (s+1)q_{2i+1}}.$$

Then by Lemma C.6, we have

$$\begin{aligned}
&\frac{p}{q} - \frac{p_{2i} + sp_{2i+1}}{q_{2i} + sq_{2i+1}} \\
&\quad < \frac{p_{2i} + (s+1)p_{2i+1}}{q_{2i} + (s+1)q_{2i+1}} - \frac{p_{2i} + sp_{2i+1}}{q_{2i} + sq_{2i+1}} \\
&\quad = \frac{1}{(q_{2i} + sq_{2i+1})(q_{2i} + (s+1)q_{2i+1})}.
\end{aligned}$$

On the other hand, if $\frac{p}{q} \neq \frac{p_{2i}+sp_{2i+1}}{q_{2i}+sq_{2i+1}}$, we should have

$$\begin{aligned}
&\frac{p}{q} - \frac{p_{2i} + sp_{2i+1}}{q_{2i} + sq_{2i+1}} \\
&\quad = \frac{(p_{2i} + sp_{2i+1})q - p(q_{2i} + sq_{2i+1})}{q(q_{2i} + sq_{2i+1})} \\
&\quad \geq \frac{1}{q(q_{2i} + sq_{2i+1})}
\end{aligned}$$

as well, thus we get $q > q_{2i} + (s+1)q_{2i+1}$. Therefore, $q_{2i} + sq_{2i+1}$ is the smallest denominator of any rational number $\frac{p}{q}$ in the interval $\left[\frac{p_{2i}+sp_{2i+1}}{q_{2i}+sq_{2i+1}}, \frac{p_{2i}+(s+1)p_{2i+1}}{q_{2i}+(s+1)q_{2i+1}}\right)$, and any other rational number should have the denominator strictly larger than $q_{2i} + (s+1)q_{2i+1}$.

We now show that $\frac{p_{2i}+sp_{2i+1}}{q_{2i}+sq_{2i+1}}$ is a best rational approximation of x from below if it is a proper semiconvergent. Indeed, take any rational number $\frac{p}{q}$ in the interval $\left[\frac{p_{2i}+sp_{2i+1}}{q_{2i}+sq_{2i+1}}, x\right]$. Since $\frac{p_{2i}+sp_{2i+1}}{q_{2i}+sq_{2i+1}}$ is proper, $\frac{p_{2i+2}}{q_{2i+2}}$ is well-defined, and this in particular means that if x is rational, then its denominator must be at least q_{2i+2}. As $\frac{p_{2i}+(s+1)p_{2i+1}}{q_{2i}+(s+1)q_{2i+1}}$ is a semiconvergent, we have implicitly assumed that $s+1 \leq a_{2i+2}$, thus we have $q_{2i}+sq_{2i+1} < q_{2i+2}$. Therefore, if $\frac{p}{q} = x$, then we must have $q > q_{2i} + sq_{2i+1}$.

On the other hand, if $\frac{p}{q} < x$, then $\frac{p}{q}$ should be either strictly less than $\lfloor x \rfloor$ or in between two consecutive semiconvergents. The first case cannot happen as $\frac{p_{2i}+sp_{2i+1}}{q_{2i}+sq_{2i+1}} \geq \frac{p_{2i}}{q_{2i}} \geq \frac{p_0}{q_0} = \lfloor x \rfloor$, and for the second case, the denominator of the smaller one of those two semiconvergents must be at least $q_{2i} + sq_{2i+1}$, so in particular q should be at least $q_{2i} + sq_{2i+1}$. This shows that $\frac{p_{2i}+sp_{2i+1}}{q_{2i}+sq_{2i+1}}$ is a best rational approximation of x from below.

Now, the only case of proper semiconvergent that is not covered by the claim above is for x itself, but for that case x is obviously a best rational approximation of itself 'from below.

Next, we show that any best rational approximation $\frac{p}{q}$ of x from below that is strictly less than x must be a proper even semiconvergent. Indeed, since $\frac{p}{q}$ is strictly less than x, we should have either $\frac{p}{q} < \lfloor x \rfloor$ or there exists a unique pair of two consecutive semiconvergents $\frac{p_{2i}+sp_{2i+1}}{q_{2i}+sq_{2i+1}}$ and $\frac{p_{2i}+(s+1)p_{2i+1}}{q_{2i}+(s+1)q_{2i+1}}$ such that

$$\frac{p_{2i} + sp_{2i+1}}{q_{2i} + sq_{2i+1}} \leq \frac{p}{q} < \frac{p_{2i} + (s+1)p_{2i+1}}{q_{2i} + (s+1)q_{2i+1}}.$$

The first case is obviously impossible because $\frac{\lfloor x \rfloor}{1}$ should be a better approximation than $\frac{p}{q}$ in that case, and for the second case, we already have shown that either $\frac{p}{q} = \frac{p_{2i}+sp_{2i+1}}{q_{2i}+sq_{2i+1}}$ or $q > q_{2i} + (s+1)q_{2i+1}$. Since $\frac{p}{q}$ is assumed to be a best rational approximation from below, we must have $\frac{p}{q} = \frac{p_{2i}+sp_{2i}}{q_{2i}+sq_{2i}}$. Then, it is clear that $\frac{p_{2i}+sp_{2i+1}}{q_{2i}+sq_{2i+1}}$ must be a proper semiconvergent; otherwise, we should have $a_{2i+2} = \infty$ and $s \neq 0$, which means that $x = \frac{p_{2i+1}}{q_{2i+1}}$ and $q_{2i} + sq_{2i+1}$ is strictly bigger than q_{2i+1}, so $\frac{p}{q}$ cannot be a best rational approximation from below, as x itself is a strictly better rational approximation.

The claim about best rational approximations from above and odd semiconvergents can be shown similarly; we omit details. □

This characterization of best rational approximations from below/above immediately leads us to the following algorithm, for a given real number x and a positive integer $n_{\max}$, of finding the best rational approximations $\frac{p_*}{q_*}$ from below and $\frac{p^*}{q^*}$ from above of x, with the largest denominators $q_*, q^* \leq n_{\max}$:

Algorithm C.9 (Finding best rational approximations from below and above)**.**

1. Find the last convergent $\frac{p_i}{q_i}$ of x such that $q_i \le n_{\max}$.
2. If $x = \frac{p_i}{q_i}$, then return $\left(\frac{p_*}{q_*}, \frac{p^*}{q^*}\right) = \left(\frac{p_i}{q_i}, \frac{p_i}{q_i}\right)$.
3. Otherwise, find the last semiconvergent $\frac{p_{i-1}+sp_i}{q_{i-1}+sq_i}$ such that $q_{i-1} + sq_i \le n_{\max}$.
4. If i is even, then return $\left(\frac{p_*}{q_*}, \frac{p^*}{q^*}\right) = \left(\frac{p_i}{q_i}, \frac{p_{i=1}+sp_i}{q_{i-1}+sq_i}\right)$.
5. If i is odd, then return $\left(\frac{p_*}{q_*}, \frac{p^*}{q^*}\right) = \left(\frac{p_{i=1}+sp_i}{q_{i-1}+sq_i}, \frac{p_i}{q_i}\right)$.

C.2 Enumerating all good enough approximations

In this section, we will derive an algorithm for enumerating all n's such that $nx - \lfloor nx \rfloor$ is smaller than a certain threshold, where x is any fixed positive real number.

Lemma C.10.
Let $\frac{a}{b} < \frac{c}{d}$ be positive rational numbers. Then any rational number $\frac{x}{y} \in \left(\frac{a}{b}, \frac{c}{d}\right)$ can be uniquely written as a form

$$\frac{x}{y} = \frac{qa + pc}{qb + pd}$$

for some coprime positive integers p, q. If $bc - ad = 1$, then we have

$$\begin{cases} p = bx - ay, \\ q = cy - dx, \end{cases} \quad \text{and} \quad \begin{cases} x = qa + pc, \\ y = qb + pd. \end{cases}$$

Hence, $\frac{qa+pc}{qb+pd}$ is in its reduced form in this case. The exact same result is true even when we allow $\frac{a}{b} = \frac{0}{1} = 0$ or $\frac{c}{d} = \frac{1}{0} = \infty$ or both.

Proof. Consider the function

$$f(t) \coloneqq \frac{a + tc}{b + td}$$

of real variable t, then

$$f'(t) = \frac{bc - ad}{(b + td)^2} > 0,$$

so f is strictly increasing on the domain $t > 0$. Hence, if we have

$$\frac{q_1 a + p_1 c}{q_1 b + p_1 d} = \frac{q_2 a + p_2 c}{q_2 b + p_2 d}$$

for some positive integers p_1, q_1, p_2, q_2, then we must have

$$\frac{p_1}{q_1} = \frac{p_2}{q_2}.$$

This shows the uniquenss part. Also, it can be easily verified that

$$\frac{x}{y} = \frac{(cy - dx)a + (bx - ay)c}{(cy - dx)b + (bx - ay)d}$$

always holds. Indeed, the numerator is equal to $(bc - ad)x$ while the denominator is equal to $(bc - ad)y$. Hence, let $g \coloneqq \gcd(cy - dx, bx - ay)$, then we have $p = (bx - ay)/g$ and $q = (cy - dx)/g$. Then it only remains to show that $g = 1$ if $bc - ad = 1$.

Since x, y are coprime, we can find integers z, w such that $xz - yw = 1$. Then, the determinant of the matrix

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} y & -z \\ -x & w \end{pmatrix} = \begin{pmatrix} ay - bx & bw - az \\ cy - dx & dw - cz \end{pmatrix}$$

must be equal to $bc - ad$. Hence, when $bc - ad = 1$, then $bx - ay$ and $cy - dx$ must be indeed coprime to each other. □

Corollary C.11.
Let $\frac{p_i}{q_i}$ denote the ith convergent of a given continued fraction $[a_0; a_1, \cdots]$. Then any rational number between two consecutive semiconvergents $\frac{p_{i-1}+sp_i}{q_{i-1}+sq_i}$ and $\frac{p_{i-1}+(s+1)p_i}{q_{i-1}+(s+1)q_i}$ is of the form

$$\frac{bp_{i-1} + (a + bs)p_i}{bq_{i-1} + (a + bs)q_i}$$

for some rational number $\frac{a}{b} \in [0, 1]$.

Proof. By replacing p_i by $p_i - a_0 q_i$ if necessary, we can assume without loss of generality that $a_0 = 0$. Then it immediately follows from the previous lemma that any rational number between those two semiconvergents must be of the form

$$\frac{qp_{i-1} + pp_i}{qq_{i-1} + pq_i}$$

for some coprime nonnegative integers p, q, because such a rational number must be in between $\frac{p_{i-1}}{q_{i-1}}$ and $\frac{p_i}{q_i}$. Note that

$$\begin{aligned} &(qp_{i-1} + pp_i)(q_{i-1} + sq_i) - (qq_{i-1} + pq_i)(p_{i-1} + sp_i) \\ &\quad = (p - qs)(p_i q_{i-1} - p_{i-1} q_i) = (-1)^{i-1}(q - ps), \end{aligned}$$

and similarly the same computation with s replaced by $s + 1$ gives $(-1)^{i-1}(q - p(s + 1))$. Therefore, $\frac{qp_{i-1}+pp_i}{qq_{i-1}+pq_i}$ lies in between $\frac{p_{i-1}+sp_i}{q_{i-1}+sq_i}$ and $\frac{p_{i-1}+(s+1)p_i}{q_{i-1}+(s+1)q_i}$ if and only if $\frac{p}{q} \in [s, s+1]$. Hence, let $a = p - qs$ and $b = q$, then we get the desired result. □

Corollary C.12.
Let $\frac{p_i}{q_i}$ denote the ith convergent of a given continued fraction $x = [a_0; a_1, \cdots]$. Then for any rational number $\frac{p}{q}$ between two consecutive semiconvergents, in either $\left[\frac{p_{i-1}+sp_i}{q_{i-1}+sq_i}, \frac{p_{i-1}+(s+1)p_i}{q_{i-1}+(s+1)q_i}\right)$ or $\left(\frac{p_{i-1}+(s+1)p_i}{q_{i-1}+(s+1)q_i}, \frac{p_{i-1}+sp_i}{q_{i-1}+sq_i}\right]$ depending on the parity of i, we always have

$$|qx - p| \ge |(q_{i-1} + sq_i)x - (p_{i-1} + sp_i)|.$$

The equality condition is:

1. *When $s+1 = a_{i+1}$ and $x = \frac{p_{i+1}}{q_{i+1}}$, then the equality holds if and only if*

$$\frac{p}{q} = \frac{bp_{i-1} + (bs + (b - 1))p_i}{bq_{i-1} + (bs + (b - 1))q_i}$$

for a positive integer b.

2. *Otherwise, the equality holds if and only if*

$$\frac{p}{q} = \frac{p_{i-1} + sp_i}{q_{i-1} + sq_i}.$$

Proof. Using the lemma above, let us write

$$\frac{p}{q} = \frac{bp_{i-1} + (a+bs)p_i}{bq_{i-1} + (a+bs)q_i}$$

for some rational number $\frac{a}{b} \in [0,1)$. Then, assuming $p_i - q_i x \neq 0$, we get

$$\begin{aligned} |qx - p| &= |b(q_{i-1}x - p_{i-1}) - (a+bs)(p_i - q_i x)| \\ &= b\,|p_i - q_i x| \left| \frac{q_{i-1}x - p_{i-1}}{p_i - q_i x} - \left(s + \frac{a}{b}\right) \right|, \end{aligned}$$

and similarly

$$\begin{aligned} &|(q_{i-1} + sq_i)x - (p_{i-1} + sp_i)| \\ &\quad = |p_i - q_i x| \left| \frac{q_{i-1}x - p_{i-1}}{p_i - q_i x} - s \right|. \end{aligned}$$

Recall $a_{i+1} = \left\lfloor \frac{q_{i-1}x - p_{i-1}}{p_i - q_i x} \right\rfloor \geq s+1$. Also, note that $\frac{q_{i-1}x - p_{i-1}}{p_i - q_i x}$ is always positive, so

$$\begin{aligned} &\frac{|qx - p| - |(q_{i-1} + sq_i)x - (p_{i-1} + sp_i)|}{|p_i - q_i x|} \\ &\quad = (b-1)\left(\frac{q_{i-1}x - p_{i-1}}{p_i - q_i x} - \left(s + \frac{a}{b}\right) \right) - \frac{a}{b}. \end{aligned}$$

Then since

$$\frac{q_{i-1}x - p_{i-1}}{p_i - q_i x} \geq s + 1 \geq s + \frac{a+1}{b},$$

we get

$$\begin{aligned} &\frac{|qx - p| - |(q_{i-1} + sq_i)x - (p_{i-1} + sp_i)|}{|p_i - q_i x|} \\ &\quad \geq \frac{b-1}{b} - \frac{a}{b} \geq 0, \end{aligned}$$

as desired. Also, the equality holds if and only if either $\frac{a}{b} = 0$ or $a = b - 1$ and

$$\frac{q_{i-1}x - p_{i-1}}{p_i - q_i x} = s + 1,$$

which means $x = \frac{p_{i+1}}{q_{i+1}}$.

The case $p_i - q_i x = 0$ is simpler; in this case, we have

$$|qx - p| = b\,|q_{i-1}x - p_{i-1}|$$

and

$$\begin{aligned} &|(q_{i-1} + sq_i)x - (p_{i-1} + sp_i)| \\ &\quad = |q_{i-1}x - p_{i-1}|, \end{aligned}$$

so we get the desired inequality and the equality condition is $b = 1$. □

Using the above result, we get the following algorithm of finding all $n = 1, \cdots, n_{\max}$ such that $nx - \lfloor nx \rfloor < \epsilon$, for any positive real number x and ϵ:

Algorithm C.13 (Finding all n's with $nx - \lfloor nx \rfloor < \epsilon$)**.**

1. When $\epsilon \geq 1$, $nx - \lfloor nx \rfloor < \epsilon$ always holds, so there is nothing to do. Thus we may assume $\epsilon < 1$.
2. Find the first even semiconvergent $\frac{p_{2i} + sp_{2i+1}}{q_{2i} + sq_{2i+1}}$ such that

   $$(q_{2i} + sq_{2i+1})x - (p_{2i} + sp_{2i+1}) < \epsilon.$$

   Then any nonnegative rational number $\frac{p}{q} \leq x$ with $qx - p < \epsilon$ must be at least $\frac{p_{2i} + sp_{2i+1}}{q_{2i} + sq_{2i+1}}$. Hence, possibly except for n's such that nx is an integer, all n's with $nx - \lfloor nx \rfloor < \epsilon$ must be at least $q_{2i} + sq_{2i+1}$. If $\frac{p_{2i} + sp_{2i+1}}{q_{2i} + sq_{2i+1}} = x$, then go to step 7.
3. Otherwise, starting from $\frac{p_{2i} + sp_{2i+1}}{q_{2i} + sq_{2i+1}}$, we iterate over all even semiconvergents $\frac{p_{2j} + tp_{2j+1}}{q_{2j} + tq_{2j+1}}$ with $q_{2j} + tq_{2j+1} \leq n_{\max}$.
4. Given such a semiconvergent $\frac{p_{2j} + tp_{2j+1}}{q_{2j} + tq_{2j+1}}$, we know that $n = q_{2j} + tq_{2j+1}$ satisfies $nx - \lfloor nx \rfloor < \epsilon$. In addition to that, compute

   $$b_{\max} := \left\lfloor \frac{n_{\max}}{q_{2j} + tq_{2j+1}} \right\rfloor.$$

   Then iterate over all $b = 1, \cdots, b_{\max}$.
5. Given $b = 1, \cdots, b_{\max}$, compute

   $$a_{\max} := \min\left(\left\lfloor \frac{n_{\max} - b(q_{2j} + tq_{2j+1})}{q_{2j+1}} \right\rfloor, b - 1 \right).$$

   When $b = 1$, we let $a = 0$, and otherwise, we iterate over $a = 1, \cdots, a_{\max}$ with $\gcd(a, b) = 1$. In any case,

   $$\frac{p}{q} := \frac{bp_{2j} + (a+bs)p_{2j+1}}{bq_{2j} + (a+bs)q_{2j+1}}$$

   is a candidate for having $qx - p < \epsilon$.
6. Compute

   $$d_{\max} := \min\left(\left\lceil \frac{\epsilon}{qx - p} \right\rceil - 1, \left\lfloor \frac{n_{\max}}{q} \right\rfloor \right).$$

   If $d_{\max} = 0$, then $qx - p \geq \epsilon$. Otherwise, any $n = qd$ with $d = 1, \cdots, d_{\max}$ satisfies $nx - \lfloor nx \rfloor < \epsilon$. Indeed, since $qdx - pd < \epsilon < 1$, we must have $\lfloor nx \rfloor = pd$.
7. All the other counterexamples must be n's such that nx is an integer. If $x = \frac{p}{q}$ is a rational number with $q \leq n_{\max}$, then all such n's are $q, 2q, \cdots, \left\lfloor \frac{n_{\max}}{q} \right\rfloor q$. Otherwise, there is no such n's.

Also, by replacing $\left\lceil \frac{\epsilon}{qx-p} \right\rceil - 1$ by $\left\lfloor \frac{\epsilon}{qx-p} \right\rfloor$, we get the algorithm for finding n's with $nx - \lfloor nx \rfloor \leq \epsilon$.

Remark. Corollary C.12 shows that semiconvergents are precisely the best rational approximations from below/above

in the strong sense: we call a rational number $\frac{p}{q}$ a *best rational approximation of x from below in the strong sense*, if $\frac{p}{q} \leq x$ and for any rational number $\frac{a}{b} \leq x$ with $b \leq q$,

$$qx - p \leq bx - a$$

holds, and similarly, we call $\frac{p}{q}$ a *best rational approximation of x from above in the strong sense*, if $\frac{p}{q} \geq x$ and for any rational number $\frac{a}{b} \geq x$ with $b \leq q$,

$$p - qx \leq a - bx$$

holds. By dividing both sides of the inequalities by q, one can immediately see that best rational approximations from below/above in the strong sense are best rational approximation as we defined in Definition 4.1, justifying the terminology. Then Corollary C.12 shows that in fact converse is also true thus those two concepts are equal things, because both are precisely the semiconvergents.

Therefore, Algorithm C.9 is in fact also the algorithm of finding $n \in \{1, \cdots, n_{\max}\}$ such that $nx - \lfloor nx \rfloor$ and $\lceil nx \rceil - nx$ are minimized. In particular, when $x = \frac{p}{q}$ is rational, it finds n that minimizes

$$nx - \lfloor nx \rfloor = \frac{1}{q}\left(p - \left\lfloor \frac{np}{q} \right\rfloor q\right) = \frac{(np \bmod q)}{q}$$

and

$$\begin{aligned} \lceil nx \rceil - nx &= \frac{1}{q}\left(\left\lceil \frac{np}{q} \right\rceil q - p\right) \\ &= \begin{cases} \frac{q-(np \bmod q)}{q} & \text{if } q \text{ does not divide } n, \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

Hence, it is obvious that what the improved min-max algorithm from [3] is doing is (almost) exactly what Algorithm C.9 is doing.

References

[1] R. Giulietti. The Schubfach Way to Render Doubles. 2020. https://drive.google.com/file/d/1KLtG_LaIbK9ETXI290zqCxvBW94dj058/view (Sep. 2020)

[2] F. Loitsch. Printing Floating-Point Numbers Quickly and Accurately with Integers. In *Proceedings of the ACM SIGPLAN 2010 Conference on Programming Language Design and Implementation, PLDI 2010*. ACM, New York, NY, USA, 233–243. https://doi.org/10.1145/1806596.1806623

[3] J. Jeon. Grisu-Exact: A Fast and Exact Floating-Point Printing Algorithm. 2020. https://github.com/jk-jeon/Grisu-Exact/blob/master/other_files/Grisu-Exact.pdf. (Sep. 2020)

[4] G. L. Steel Jr. and J. L. White. How to Print Floating-Point Numbers Accurately. In *Proceedings of the ACM SIGPLAN 1990 Conference on Programming Language Design and Implementation, PLDI 1990*. ACM, New York, NY, USA, 112–126. https://doi.org/10.1145/93542.93559

[5] M. Andrysco, R. Jhala, and S. Lerner. Printing Floating-Point Numbers: a Faster, Always Correct Method. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016*. ACM, New York, NY, USA, 555–567. https://doi.org/10.1145/2837614.2837654

[6] U. Adams. Ryū: Fast Float-to-String Conversion In *Proceedings of the ACM SIGPLAN 2018 Conference on Programming Language Design and Implementation, PLDI 2018*. ACM, New York, NY, USA, 270–282. https://doi.org/10.1145/3296979.3192369

[7] https://github.com/abolz/Drachennest. (May. 2021)

[8] T. Granlund and P. L. Montgomery. Division by Invariant Integers using Multiplication. In *ACM SIGPLAN Notices, Vol 29, Issue 6, Jun. 1994*. ACM, New York, NY, USA, 61–72. https://doi.org/10.1145/773473.178249

[9] H. S. Warren, Jr. *Hacker's Delight*. Addison-Wesley, Boston, 1st edition, 2002.

[10] D. Lemire, C. Bartlett and O. Kaser. Integer division by constants: optimal bounds. In *Heliyon, Vol 7, Issue 6, E07422, Jun. 2021*. https://doi.org/10.1016/j.heliyon.2021.e07442

[11] https://github.com/jk-jeon/dragonbox. (Sep. 2022)

[12] https://stackoverflow.com/questions/25095741/how-can-i-multiply-64-bit-operands-and-get-128-bit-result-portably. (Jun. 2020)

[13] https://github.com/ulfjack/ryu. (Jun. 2022)

[14] https://github.com/jk-jeon/Grisu-Exact. (Sep. 2022)

[15] M. Einsiedler and T. Ward. *Ergodic Theory with a view towards Number Theory*. Graduate Texts in Mathematics 259. Springer.