Snapping Mechanism Notes

Christian Covington

August 31, 2019

1. Introduction

The implementation of the snapping mechanism utilizes a number of ideas described in Mironov (2012)[1] that, as far as I can tell, are not commonly seen in implementations of other DP algorithms. This document provides an overview of these ideas.

2. Mechanism Definitions

Below we present the mechanisms as if they are operating on the set 64-bit floating point numbers, $\mathbb{D}$, rather than $\mathbb{R}$. We will use $\oplus$ and $\otimes$ to represent the floating-point implementations of addition and multiplication, respectively.

2.1. Laplace Mechanism

Let f be a function computed on a data set D with sensitivity Δ_f and ϵ be the desired privacy parameter. Further, let $\lambda = \frac{\Delta_f}{\epsilon}$. Then the Laplace Mechanism is defined as:

$$M_L(D, f(\cdot), \lambda) = f(D) \oplus Y$$

where $Y \sim Laplace(\lambda)$.

2.2. Snapping Mechanism

Let B be a user-chosen quantity that reflects beliefs about reasonable bounds on $f(D)$ and Λ be the smallest power of two at least as large as λ. Using the same notation as above, the snapping mechanism is defined as:

$$M_S(D, f(\cdot), \lambda, B) = clamp_B\left(\lfloor clamp_B\left(f(D)\right) \oplus Y \rceil_\Lambda\right).$$

where $clamp_B(\cdot)$ restricts output to the interval $[-B, B]$ and $\lfloor \cdot \rceil_\Lambda$ rounds to the nearest multiple of Λ, with ties resolved toward $+\infty$.

[1] specifically in section 5.2

3. Sampling from the Laplace

Mironov presents the Laplace noise as

$$S \otimes \lambda \otimes LN(U^*)$$

where S is a uniform draw over $\{\pm 1\}$, $\lambda = \frac{\Delta_f}{\epsilon}$, $LN(\cdot)$ is the floating-point implementation of the natural logarithm with exact rounding, and U^* is the uniform distribution over $\mathbb{D} \cap (0, 1)$ such that each double is output with proportion relative to its unit of least precision. $\oplus$ and $\otimes$ are the floating-point implementations of addition and multiplication, respectively.

Drawing S should be done with whichever source of randomness we end up wanting to use. The current Python implementation uses system-level randomness by utilizing an instance of the **random.SystemRandom** class.

We get U^* as Mironov suggests, by sampling our exponent from a geometric distribution with parameter $p = 0.5$ and a mantissa uniformly from $\{0, 1\}^{52}$. Let e be a draw from $Geom(0.5)$ and $m_1, m_2 \ldots, m_{52}$ be the bits of our mantissa. Then we have

$$U^* = (1.m_1 m_2 \ldots m_{52})_2 * 2^{-e}.$$

The unit of least precision is completely decided by the value of e, so the floating-point values within each unit of least precision are uniformly distributed. Drawing e from the geometric ensures that the probability of drawing a value from a given band is equal to the size of the band.

$LN(\cdot)$ must be implemented with exact rounding, which we define below.[2] Consider that for an arbitrary $x \in \mathbb{D}$ the natural log of x is not necessarily $\in \mathbb{D}$. Let $a < ln(x) < b$ where $a, b \in \mathbb{D}$ and $\nexists c \in \mathbb{D} : a < c < b$. Without loss of generality, assume that $|a - x| < |b - x|$, so that if we had infinite precision in calculating $ln(x)$ (but still had to output an element $\in \mathbb{D}$), we would output a. Many mathematical libraries do what is called *accurate-faithful* rounding, which means that in the scenario above our algorithm would output a with high probability. In an *exact rounding* paradigm, the algorithm outputs a with probability 1. You can read more about exact rounding in section 1.1 here. Section paper 2.1 of the paper just linked appeals to proofs from a set of papers that say you need 118 bits of precision, in the worst case, to calculate the logarithm with exact rounding. So, the current implementation calculates $LN(U^*)$ with at least 118 bits of precision.[3]

Finally, we can choose to perform $\oplus$ and $\otimes$ with greater than normal precision if we so choose. For now, we are using the same 118 bits of precision for all the basic floating-point operations, using the assumption that exact rounding of basic arithmetic operations should require less precision than calculating the log. This should probably be made more rigorous at some point.

4. Implementation of $\lfloor \cdot \rceil_\Lambda$

The $\lfloor \cdot \rceil_\Lambda$ function takes an input and rounds it to the nearest multiple of Λ, where Λ is the smallest power of two greater than or equal to λ. There are multiple steps to this implementation that are worth explaining.

[2]For all rounding, we assume that our goal is to round to the nearest number we are able to represent.
[3]We potentially use > 118 because of the ϵ redefinition described later.

4.1. Finding Λ

The algorithm receives λ as input, but must find Λ itself. First, represent λ in its IEEE-754 64-bit floating-point format:

$$\lambda = (-1)^S (1.m_1 \dots m_{52})_2 * 2^{(e_1 \dots e_{11})_2 - 1023}.$$

We know $\lambda > 0$, so we know $S = 0$. Now, note that powers of two correspond exactly to the IEEE representations with $m_1 = \dots = m_{52} = 0$. If λ is already a power of two, then we simply return λ. Otherwise, we get the smallest power of two greater than λ by changing to mantissa to $(0 \dots 0)_2$ and increasing the exponent by 1. So, we have

$$\Lambda = \begin{cases} \lambda, & \text{if } m_1 = \dots = m_{52} = 0 \\ (1.0 \dots 0)_2 * 2^{(e_1 \dots e_{11})_2 - 1022}, & \text{if } \exists i : m_i \neq 0 \end{cases} \tag{4.1}$$

4.2. Rounding to nearest multiple of Λ

We now want to round our input x to the nearest multiple of Λ. We do so via a three-step process:

1. $x' = \frac{x}{\Lambda}$
2. Round x' to nearest integer, yielding x''
3. $\lfloor x \rceil_\Lambda = \Lambda x''$

We split the process into three steps because each step can be performed exactly (with no introduction of floating-point error) via manipulation of the IEEE floating-point representation.

4.2.1. Calculate $x' = \frac{x}{\Lambda}$

We can perform this division exactly because Λ is a power of two. Let $\Lambda = 2^m$ for some $m \in \mathbb{Z}$ and let x have the IEEE representation

$$x = (-1)^S (1.m_1 \dots m_{52})_2 * 2^{(e_1 \dots e_{11})_2 - 1023}.$$

Then we know that

$$x' = (-1)^S (1.m_1 \dots m_{52})_2 * 2^{(e_1 \dots e_{11})_2 - 1023 - m}.$$

We can rewrite $(e_1 \dots e_{11})_2 - m$ as $(f_1 \dots f_{11})_2$ and represent x' directly as its IEEE implementation:

$$x' = (-1)^S (1.m_1 \dots m_{52})_2 * 2^{(f_1 \dots f_{11})_2 - 1023}.$$

4.2.2. Round x' to nearest integer

Let $y = (f_1 \dots f_{11})_2 - 1023$. We present slightly different rounding algorithms based on the value of y. Note that we abuse notation a bit below; the repeating element notation $\bar{0}$ means to repeat the element (in this case, 0) until the rest of the larger section has been filled. For example, if the mantissa is 52 bits, then $101\bar{0}$ represents 101 followed by 49 trailing zeros.

Case 1: $y \geq 52$
If $y \geq 52$, then we know that only integers are able to be represented at this scale, so there

is no need to round.[4]

Case 2: $y \in \{0, 1, \ldots, 51\}$
We can think of multiplying by 2^y as shifting the radix point to the right y times. We can then write

$$x' = (-1)^S (1m_1 \ldots m_y.m_{y+1} \ldots m_{52})_2.$$

We know $m_1 \ldots m_y$ represent powers of two $\in \mathbb{Z}$ and $m_{y+1} \ldots m_{52}$ are powers of two $\notin \mathbb{Z}$. Specifically, m_{y+1} corresponds to $\frac{1}{2}$. So, we know to round up if $m_{y+1} = 1$ and down if $m_{y+1} = 0$.

Rounding up requires incrementing up by 1 the integral part of the mantissa $(m_1 \ldots m_y)$, and changing the fractional part $(m_{y+1}, \ldots, m_{52})$ to zeros. Note that there is an edge case here where $m_i = 1$ for all i in which the mantissa becomes $(\bar{0})_2$ and we instead increment the exponent. Rounding down requires maintaining the integral part of the mantissa and changing the fractional part to zeros.

Let $(m'_1 \ldots m'_y)_2 = (m_1 \ldots m_y)_2 + 1$ and $(f'_1 \ldots f'_{11})_2 = (f_1 \ldots f_{11})_2 + 1$. Then we get:

$$x'' = \begin{cases} (-1)^S (1.m'_1 \ldots m'_y\bar{0})_2 * 2^{(f_1 \cdots f_{11})_2 - 1023}, & \text{if } m_{y+1} = 1 \text{ and } \exists i : m_i = 0 \\ (-1)^S (1.\bar{0})_2 * 2^{(f'_1 \cdots f'_{11})_2 - 1023}, & \text{if } m_{y+1} = 1 \text{ and } \forall i : m_i = 1 \\ (-1)^S (1.m_1 \ldots m_y\bar{0})_2 * 2^{(f_1 \cdots f_{11})_2 - 1023}, & \text{if } m_{y+1} = 0 \end{cases} \tag{4.2}$$

Case 3: $y = -1$
We think of this case similarly to the beginning of Case 2 and shift the radix point to the left:

$$x'' = (-1)^S (0.1m_1m_2 \ldots)_2.$$

W know the bit directly to the right of the radix point is the implicit 1 in the IEEE representation, so we know we will round up. Rounding up always rounds to 1, so we know

$$x'' = (-1)^S (1.\bar{0})_2 * 2^{(0\bar{1})_2 - 1023}.$$

Case 4: $y < -1$
This is exactly the same as Case 3, except we know there are some number of leading zeros between the radix point and the implicit 1. Therefore, we always round down to 0 and get:

$$x'' = (-1)^0 (1.\bar{0})_2 * 2^{(\bar{0})_2 - 1023}.$$

The notation directly above should not be taken literally, as 0 has a special IEEE representation (all 0s) that does not quite follow the standard formula. We present the standard formally for the sake of consistency and comparison with the earlier cases.

4.2.3. Multiply x'' by Λ

Finally, we multiply x'' by Λ to get our desired result. This is effectively the opposite of Step 1, in that we add m to the exponent of x''. The only difference is that we now have to deal with the case in which $x'' = 0$, because $0 * 2^m \neq 0$ if you implement multiplication by 2^m through exponent addition.[5]

[4]See here for an explanation.
[5]This is due to the special representation of 0.

5. Redefinition of ϵ

Mironov states in the last line of the privacy proof that the snapping mechanism respects

$$\left(\frac{1+12B\eta}{\lambda}+2\eta\right)\text{-DP}$$

where B is the clamping bound and η is the machine-ϵ.[6] We will take η to be our floating-point precision, rather than a value that is machine-imposed, which allows us to use high-precision libraries to get smaller values of η. Because we have rescaled our function to $\Delta f = 1$, we have $\delta = \frac{1}{\epsilon'}$ and this is equivalent to

$$\big(\epsilon'(1+12B\eta)+2\eta\big)\text{-DP}.$$

For ease of notation, we will call this ϵ-DP and it can be thought of as the privacy level relative to the Laplace. That is, if you were to get ϵ'-DP from the Laplace Mechanism, you would get ϵ-DP from the Snapping Mechanism. Put another way, you get ϵ-DP from the Snapping Mechanism when you use ϵ' to parameterize the Laplace noise that is embedded within the Snapping Mechanism.

This is not really sufficient to be usable in a system that might require budgeting of a privacy budget, as you cannot (as written above) simply set a privacy allocation for the Snapping Mechanism, as it relies on B and η. So rather than returning ϵ as a function of ϵ', we should do the reverse.

$$\begin{aligned} \epsilon &= \epsilon'(1+12B\eta)+2\eta \\ \implies \epsilon' &= \frac{\epsilon-2\eta}{1+12B\eta} \end{aligned}$$

This ϵ' is what will be used in the parameterization in the Laplace within the Snapping Mechanism. Thus, we need $\epsilon' > 0$, or $\epsilon > 2\eta$, where $\eta = 2^{-p}$, and p is our floating point precision. This simplifies to $\epsilon > 2^{-p+1}$. Assuming we want minimum sufficient precision, we can find the smallest power of two $\geq \epsilon$, call it 2^{-m}, and let $p = m + 2$. For setting our mechanism-level precision, we use the larger of the p we just found or the level necessary to perform exact rounding on the logarithm as described earlier (which we currently believe to be 118).

6. Utility

6.1. Error

As originally written, it is difficult to reason about error/utility of the Snapping Mechanism. The output of the Snapping Mechanism relies on the user-provided bound, B, which could potentially cause the output to be arbitrarily far from $f(D)$. The potential for bias of the Snapping Mechanism decreases as B increases in absolute value. Note however that ϵ' decreases as B increases, thus causing the amount of added Laplace noise to increase. Throughout this section, we will assume that $f(D) \geq 0$, though a very similar process can be taken to reason about $f(D) \leq 0$.

[6] This is a relatively minor point, but I have normally seen machine-ϵ defined as the smallest value ϵ such that $1 \oplus \epsilon \neq 1$. Mironov defines it as the largest ϵ such that $1 \oplus \epsilon = 1$.

One possibility is that, instead of the user choosing B, we set it for them within the mechanism. We can use the data bounds the user provides to calculate a bound on $f(D)$.[7] This gives the minimum possible B such that we are sure that $clamp_B\left(f(D)\right)$ is non-binding. This will make it a bit easier to think about the error of the mechanism, as every component of the mechanism is now something with a well-defined (and predictable) value or distribution. For now, we will ignore exactly how this is done and focus on general analysis assuming that such a B can be chosen.

Consider the following quantity:

$$N \sim \left|f(D) - clamp_B\big(\lfloor f(D) + Y' \rceil_\Lambda\big)\right|$$

with $Y' \sim Laplace(\lambda')$ where $\lambda' = \frac{1}{e'} = \frac{1+12B\eta}{\epsilon - 2\eta}$. This is the amount of noise added to $f(D)$ by the snapping mechanism.[8] We have then:

$$N = \begin{cases} B + f(D), & \text{if } clamp_B\left(\lfloor f(D) + Y' \rceil_\Lambda\right) = -B \\ B - f(D), & \text{if } clamp_B\left(\lfloor f(D) + Y' \rceil_\Lambda\right) = B \\ \left|f(D) - \lfloor f(D) + Y' \rceil_\Lambda\right|, & \text{otherwise} \end{cases} \tag{6.1}$$

We can examine these cases further. Let $F_{Y'}$ be the CDF of $Y' \sim Laplace(\lambda')$, which we write:

$$F_{Y'}(x) = \begin{cases} \frac{1}{2}\exp\left(\frac{x}{\lambda'}\right), & \text{if } x < 0 \\ 1 - \frac{1}{2}\exp\left(\frac{-x}{\lambda'}\right), & \text{if } x \geq 0 \end{cases} \tag{6.2}$$

Then:

$$\begin{aligned} \mathbb{P}\big(clamp_B\left(\lfloor f(D) + Y' \rceil_\Lambda\right) = -B\big) &= \mathbb{P}\big(\lfloor f(D) + Y' \rceil_\Lambda \leq -B\big) \\ &= \mathbb{P}\big(f(D) + Y' < -B + \frac{\Lambda}{2}\big) \\ &= \mathbb{P}\big(Y' < -B - f(D) + \frac{\Lambda}{2}\big) \\ &= F_{Y'}\big(-B - f(D) + \frac{\Lambda}{2}\big) \end{aligned}$$

$$\begin{aligned} \mathbb{P}\big(clamp_B\left(\lfloor f(D) + Y' \rceil_\Lambda\right) = B\big) &= \mathbb{P}\big(\lfloor f(D) + Y' \rceil_\Lambda \geq B\big) \\ &= \mathbb{P}\big(f(D) + Y' \geq B - \frac{\Lambda}{2}\big) \\ &= \mathbb{P}\big(Y' \geq B - f(D) - \frac{\Lambda}{2}\big) \\ &= 1 - \mathbb{P}\big(Y' < B - f(D) - \frac{\Lambda}{2}\big) \\ &= 1 - F_{Y'}\big(B - f(D) - \frac{\Lambda}{2}\big) \end{aligned}$$

[7] This may not be true in full generality, but should be for the statistics for which the Laplace is currently used in PSI.

[8] Note that we ignore the inner $clamp_B$ inside the Snapping Mechanism because we set $B \geq f(D)$.

Recall from the triangle inequality that $|x-z| \leq |x-y| + |y-z|$. Then we have

$$\begin{aligned}
&\big|f(D) - \lfloor f(D) + Y' \rceil_\Lambda\big| \\
&\leq \big|f(D) - (f(D) + Y')\big| + \big|f(D) + Y' - \lfloor f(D) + Y' \rceil_\Lambda\big| \\
&= |-Y'| + \frac{\Lambda}{2} \\
&= |Y'| + \frac{\Lambda}{2}
\end{aligned}$$

It is important to note that this Y' is associated with the Laplace noise inside the snapping mechanism which uses the redefined ϵ', rather than the user's desired ϵ. So, we need to do a bit more work in order to compare the error of the Snapping Mechanism to that of the Laplace Mechanism.

Let $Y \sim Laplace(\lambda)$, where $\lambda = \frac{1}{\epsilon}$, be the distribution of noise generated by the Laplace Mechanism. We showed in subsection 2.2 that we can represent the distribution of Laplace noise as $S \otimes \lambda \otimes LN(U^*)$. So, we have

$$Y \sim S \otimes \frac{1}{\epsilon} \otimes LN(U^*)$$

$$Y' \sim S \otimes \frac{1 + 12B\eta}{\epsilon - 2\eta} \otimes LN(U^*)$$

and thus,

$$Y' = \frac{\epsilon(1 + 12B\eta)}{\epsilon - 2\eta} Y$$

So we can rewrite (6.1) as

$$N \begin{cases} = B + f(D), & \text{with probability } F_{Y'}\big(-B - f(D) + \frac{\Lambda}{2}\big) \\ = B - f(D), & \text{with probability } 1 - F_{Y'}\big(B - f(D) - \frac{\Lambda}{2}\big) \\ \leq |Y'| + \frac{\Lambda}{2}, & \text{otherwise} \end{cases} \tag{6.3}$$

Note that N is always $\leq B + f(D)$.

6.2. Accuracy

We would like to convert the error into an accuracy estimate. Keeping in line with existing standards in PSI-Library, we define the accuracy a for a given α as the a such that $\alpha = \mathbb{P}(N > a)$, where N is (as above) the error (relative to $f(D)$) introduced by the Snapping Mechanism. It is difficult to get an exact accuracy guarantee because of our inability to model Λ exactly, so we will instead refer to a as the smallest x such that $\mathbb{P}(N > x) \leq \alpha$.

Recall that $N \leq B + f(D)$. If we can get an upper bound on $P_L = \mathbb{P}(N = B + f(D))$, then we can start concerning ourselves only with the second two lines of (6.3), which we will call N_{sub}.

Let's attempt to get bounds on $\mathbb{P}(N = B + f(D))$ and $\mathbb{P}(N = B - f(D))$ in terms of quantities we know. Remember that $\frac{\lambda'}{2} \leq \frac{\Lambda}{2} < \lambda'$.

$$\begin{aligned}
P_L &= \mathbb{P}(N = B + f(D)) \\
&= F_{Y'}(-B - f(D) + \frac{\Lambda}{2}) \\
&< F_{Y'}(-B - f(D) + \lambda') \\
&= P_L^+
\end{aligned}$$

$$
\begin{aligned}
P_U &= \mathbb{P}(N = B - f(D)) \\
&= 1 - F_{Y'}(B - f(D) - \frac{\Lambda}{2}) \\
&\geq 1 - F_{Y'}(B - f(D) - \lambda') \\
&= P_U^-
\end{aligned}
$$

Because we know that $N \leq P_L < P_L^+$, we can reframe our original goal. We wanted to find the smallest x such that $\alpha \geq \mathbb{P}(N > x)$, but this is equivalent to finding the smallest x such that

$$\alpha - P_L^+ \geq \mathbb{P}(N_{sub} > x)$$

which we further reinterpret as

$$\alpha - P_L^+ \geq 1 - \mathbb{P}(N_{sub} \leq x) = 1 - F_{N_{sub}}(x).$$

Consider N_{sub}; we know that

$$Y' \sim Laplace(\lambda')$$

with $\lambda' = \frac{1}{\epsilon'} = \frac{1+12B\eta}{\epsilon - 2\eta}$, which implies that

$$|Y'| \sim Exponential(\epsilon')$$

We now construct upper and lower bounds on the CDF of $Z = |Y'| + \frac{\Lambda}{2}$, which we will call F_Z:

$$
\begin{aligned}
F_Z(z) &= \mathbb{P}(Z \leq z) \\
&= \mathbb{P}\left(|Y'| + \frac{\Lambda}{2} \leq z\right) \\
&= \mathbb{P}\left(|Y'| \leq z - \frac{\Lambda}{2}\right) \\
&\leq \mathbb{P}\left(|Y'| \leq z - \frac{\lambda'}{2}\right) \\
&= 1 - \exp\left(-\epsilon'(z - \frac{\lambda'}{2})\right) \\
&= 1 - \exp\left(-\epsilon' z + \frac{1}{2}\right) \\
&= F_Z^+(z)
\end{aligned}
$$

$$
\begin{aligned}
F_Z(z) &= \mathbb{P}(Z \leq z) \\
&= \mathbb{P}\left(|Y'| + \frac{\Lambda}{2} \leq z\right) \\
&= \mathbb{P}\left(|Y'| \leq z - \frac{\Lambda}{2}\right) \\
&> \mathbb{P}\left(|Y'| \leq z - \lambda'\right) \\
&= 1 - \exp\left(-\epsilon'(z - \lambda')\right) \\
&= 1 - \exp\left(-\epsilon' z + 1\right) \\
&= F_Z^-(z)
\end{aligned}
$$

Let f_Z be the PDF of Z, $m = \min(B - f(D), x)$, and $n = \min(B + f(D), x)$. Then we have:

$$\begin{aligned}
F_{N_{sub}}(x) &= \int_0^m f_Z(z)dz + \left(P_U + \int_{B-f(D)}^{n} f_Z(z)dz\right) \cdot \mathbb{1}\left(x \geq B - f(D)\right) \\
&= F_Z(m) + (P_U + F_Z(n) - F_Z(B - f(D))) \cdot \mathbb{1}\left(x \geq B - f(D)\right) \\
&\geq F_Z^-(m) + \left(P_U^- + F_Z^-(n) - F_Z^+(B - f(D)\right) \cdot \mathbb{1}\left(x \geq B - f(D)\right) \\
&= F_{N_{sub}}^-(x)
\end{aligned}$$

Now that we have $F_{N_{sub}}^-(x)$ as a lower bound on $F_{N_{sub}}(x)$, we know that $1 - F_{N_{sub}}^-(x)$ is an upper bound on $1 - F_{N_{sub}}(x)$. So, our objective is now to find the minimum x such that $1 - F_{N_{sub}}^-(x) = \alpha - P_L^+$. We rewrite this condition as $F_{N_{sub}}^-(x) = 1 + P_L^+ - \alpha$.

Observe that if for a given x' we have $F_Z^-(x') \geq 1 + P_L^+ - \alpha$, we know that

$$\exists x \leq x' : x = 1 + P_L^+ - \alpha$$

We can find this x by performing algebra on $F_Z^-(x)$, which yields the following algorithm:

Algorithm 1 Finding Accuracy

function GetAccuracy$(f(D), B, P_L^+, P_U^-, \epsilon', \alpha)$
  if $P_L^+ \geq \alpha$ **then**
    return B + f(D)
  else if $F_Z^-(B - f(D)) \geq 1 + P_L^+ - \alpha$ **then**
    return $\frac{1-\ln(\alpha - P_L^+)}{\epsilon'}$
  else if $F_Z^-(B - f(D)) + P_U^- \geq 1 + P_L^+ - \alpha$ **then**
    return B - f(D)
  else
    return $\frac{1-\ln(\alpha - P_L^+ + P_U^-)}{\epsilon'}$
  end if
end function

Calculations/explanations of the return statements above can be found in Appendix A.

Appendices

A. Accuracy Calculations

We iterate over the possible cases for where our optimal a could lie. We consider the sections of the PDF of N representing where the bounds bind (Cases 1 and 3) as special cases, and use our bounds on F_Z for the sections between the bounds.

Case 1: $P_L^+ \geq \alpha$

$P_L^+ \geq \alpha$ means that the largest bound $(B + f(D))$ on the noise binds with probability $\geq \alpha$. So, we can make no tighter accuracy guarantee at the α level than bounding by the greatest possible noise $B + f(D)$.

Case 2: Case 1 does not hold and $F_Z^-(B - f(D)) \geq 1 + P_L^+ - \alpha$

We know $a = x \leq B - f(D) : F_Z^-(x) = 1 + P_L^+ - \alpha$.

$$\begin{aligned}
F_Z^-(a) &= 1 + P_L^+ - \alpha \\
1 - \exp(-\epsilon' a + 1) &= 1 + P_L^+ - \alpha \\
\epsilon' a + 1 &= \ln(\alpha - P_L^+) \\
a &= \frac{1 - \ln(\alpha - P_L^+)}{\epsilon'}
\end{aligned}$$

Case 3: Case 1,2 do not hold and $F_Z^-(B - f(D)) + P_U^- \geq 1 + P_L^+ - \alpha$
This is conceptually similar to Case 1, in the sense that the probability of a bound binding (in this case, the bound if $B - f(D)$) pushes us past our α level. Similar to before, we can make an accuracy guarantee that corresponds to the binding bound, $B - f(D)$.

Case 4: Case 1,2,3 do not hold
We know $a = x \in (B - f(D), B + f(D))$ such that

$$F_Z^-(x) - F_Z^-(B - f(D)) = 1 + P_L^+ - \alpha - F_Z^-(B - f(D)) - P_U^-$$

$$\begin{aligned}
F_Z^-(a) - F_Z^-(B - f(D)) &= 1 + P_L^+ - \alpha - F_Z^-(B - f(D)) - P_U^- \\
F_Z^-(a) &= 1 + P_L^+ - \alpha - P_U^- \\
1 - \exp(-\epsilon' a + 1) &= 1 + P_L^+ - \alpha - P_U^- \\
-\epsilon' a + 1 &= \ln(\alpha - P_L^+ + P_U^-) \\
a &= \frac{1 - \ln(\alpha - P_L^+ + P_U^-)}{\epsilon'}
\end{aligned}$$