# Tolerance Calculation for ApproxEquals in Tests

This document explains how to calculate the tolerance needed to achieve a desired flakiness when approxEquals is used in Privacy on Beam tests.

## Floats

### Laplace Noise

CDF of a Laplace Distribution with mean $0$, epsilon $\varepsilon$ and $l_1$ sensitivity $s_1$ is given by:

$$P(X \leq x) = 1 - \frac{\exp\left(\frac{-\varepsilon x}{s_1}\right)}{2}$$

For $k \geq 0$, in order to achieve a flakiness of $10^{-k}$, i.e. tests pass with probability $1 - 10^{-k}$, we need to have the following:

$$1 - \frac{10^{-k}}{2} = 1 - \frac{\exp\left(\frac{-\varepsilon x}{s_1}\right)}{2}$$

where $\frac{10^{-k}}{2}$ comes from the fact that we need to use a two-sided confidence interval and $x$ is the tolerance. Solving for $x$, we get:

$$\exp\left(\frac{\varepsilon x}{s_1}\right) = 10^k$$

$$\frac{\varepsilon x}{s_1} = k * \ln 10$$

$$x = \frac{s_1 * k * \ln 10}{\varepsilon}$$

For example, in order to achieve a flakiness of $10^{-23}$, i.e. $k = 23$, with $\varepsilon = 50$ and $s_1 = 1$, we need a tolerance of $x = 1.05919$.

### Gaussian Noise

CDF of a Gaussian Distribution with mean $0$, epsilon $\varepsilon$, delta $\delta$ and $l_2$ sensitivity $s_2$ is given by:

$$P(X \leq x) = \frac{1}{2}\left(1 + erf\left(\frac{x}{\sigma\sqrt{2}}\right)\right)$$

where $erf$ is the error function. For $k \geq 0$, in order to achieve a flakiness of $10^{-k}$, i.e. tests pass with probability $1 - 10^{-k}$, we need to have the following:

$$1 - \frac{10^{-k}}{2} = \frac{1}{2}\left(1 + erf\left(\frac{x}{\sigma\sqrt{2}}\right)\right)$$

where $\frac{10^{-k}}{2}$ comes from the fact that we need to use a two-sided confidence interval and $x$ is the tolerance. Solving for $x$, we get:

$$1 - \frac{10^{-k}}{2} = \frac{1}{2} + \frac{1}{2} erf\left(\frac{x}{\sigma\sqrt{2}}\right)$$

$$\frac{1}{2} - \frac{10^{-k}}{2} = \frac{1}{2} erf\left(\frac{x}{\sigma\sqrt{2}}\right)$$

$$1 - 10^{-k} = erf\left(\frac{x}{\sigma\sqrt{2}}\right)$$

$$erfinv\left(1 - 10^{-k}\right) = \frac{x}{\sigma\sqrt{2}}$$

$$x = erfinv\left(1 - 10^{-k}\right)\sigma\sqrt{2}$$

where $erfinv$ is the inverse error function.

We use math package function Erfinv to approximate $erfinv$ and the Go DP Library's sigmaForGaussian function to compute $\sigma$ with $s_2$, $\varepsilon$ and $\delta$.

# Integers

The logic for floats applies to integers as well. However, the caveat is that the noise is rounded to the nearest integer for aggregations on integers.

This is not a problem when the decimal part of the tolerance needed for a given flakiness is less than $0.5$ since all the noise values up to $0.5$ are going to be rounded down to $0$.

When the decimal part of the tolerance is greater than $0.5$, however, we need to round tolerance up to the next integer since all the noise values from $0.5$ up to tolerance would be rounded up.

For example, for a given flakiness, if we need a tolerance of $1.1$, we can use a tolerance of $1.1$ (or any value$\geq 1$). On the other hand, if we need a tolerance of $2.6$, we would need to use a tolerance of $3.0$.

That is why, we round the tolerance to the nearest integer in our tests for integers.

# Multiple Partitions

When there are multiple partitions in a test, noise is applied to each partition independently. If there are $p$ partitions and each has a flakiness of $10^{-k}$, then the overall test has a probability $(1 - 10^{-k})^p$ of passing.

In the tests we use, we always have $p \leq 10$ and want $k \geq 20$, so we can approximate $(1 - 10^{-k})^p \simeq 1 - p * 10^{-k} < 1 - 10^{-k+2}$.

Therefore, in order to get an overall flakiness of $10^{-k+2}$, we need to use the tolerance for partition-level flakiness of $10^{-k}$.

For example, if we want to achieve an overall flakiness of $10^{-23}$, we need to use the tolerance for $10^{-25}$.

# Mean

For mean we can not just take CDF of a Laplace distribution, because bounded mean algorithm is using noisy sum and noisy count in order to calculate mean.

The formula for the noisy mean is given by $noisyMean = \frac{noisyNormalizedSum}{noisyCount} + midPoint$, where noisyNormalizedSum is noisy sum of distances of the input entities from the $midPoint = \frac{upper+lower}{2}$, and $upper$ and $lower$ represent the upper and lower bounds for the input elements respectively. In other words, if the element is less than lower bound it will be clamped to the lower bound, and if the element is bigger than upper bound it will be clamped to the upper bound.

So, when calculating the tolerance for mean, we have to take into account that we are using two different confidence intervals (one for sum and count each). In order to do that we should calculate the maximum possible difference between the exact mean and noisy mean. We cannot say for sure which difference is bigger - the difference between the exact mean and the minimum possible noisy mean or the difference between the exact mean and the maximum possible noisy mean.

$$meanTolerance = max(|maxNoisyMean - exactMean|, |minNoisyMean - exactMean|)$$

where $minNoisyMean = \frac{minNoisyNormalizedSum}{maxNoisyCount}$ and $maxNoisyMean = \frac{maxNoisyNormalizedSum}{minNoisyCount}$

We take maximum of absolute values in order to be safe (there is an edge case when $maxNoisyMean$ and $minNoisyMean$ swap places).

We need to calculate the tolerance for sum and count in order to achieve desired flakiness for mean. Suppose we want $10^{-k}$ flakiness then:

- The noisyCount (which is in the denominator) is outside of this confidence interval should have $10^{-k}$ flakiness.
- The noisyNormalizedSum (which is in numerator) is outside of this confidence interval should have $10^{-k}$ flakiness.

Both of these things have to be true at the same time for the confidence interval to be correct. Therefore the overall flakiness would be $10^{-2k}$.

So in order to get overall flakiness $10^{-k}$ for mean we should calculate the tolerance for count and sum for $10^{-k/2}$ flakiness.

# Complementary Tolerance Calculation for checkMetricsAreNoisy

We use checkMetricsAreNoisy to ensure that Privacy on Beam is adding noise. The same logic for approxEquals applies here but the tolerance calculation is different because we need to have a "complementary confidence interval" that includes all values except for a narrow interval around the mean.

Unlike approxEquals, we do not round values to the nearest integer to deal with integer valued aggregations because this is a complementary tolerance.

## Laplace Noise

CDF of a Laplace Distribution with mean $0$, epsilon $\varepsilon$ and $l_1$ sensitivity $s_1$ is given by:

$$P(X \leq x) = 1 - \frac{\exp\left(\frac{-\varepsilon x}{s_1}\right)}{2}$$

For $k \geq 0$, in order to achieve a flakiness of $10^{-k}$, i.e. tests pass with probability $1 - 10^{-k}$, we need to have the following:

$$\frac{1}{2} + \frac{10^{-k}}{2} = 1 - \frac{\exp\left(\frac{-\varepsilon x}{s_1}\right)}{2}$$

where $\frac{10^{-k}}{2}$ comes from the fact that we need to use a two-sided confidence interval and $x$ is the tolerance. Solving for $x$, we get:

$$\exp\left(\frac{-\varepsilon x}{s_1}\right) = 1 - 10^{k}$$

$$\frac{-\varepsilon x}{s_1} = \ln\left(1 - 10^{k}\right)$$

$$x = -\frac{s_1 * \ln\left(1 - 10^{k}\right)}{\varepsilon}$$

## Gaussian Noise

CDF of a Gaussian Distribution with mean $0$, epsilon $\varepsilon$, delta $\delta$ and $l_2$ sensitivity $s_2$ is given by:

$$P(X \leq x) = \frac{1}{2}\left(1 + erf\left(\frac{x}{\sigma\sqrt{2}}\right)\right)$$

where $erf$ is the error function. For $k \geq 0$, in order to achieve a flakiness of $10^{-k}$, i.e. tests pass with probability $1 - 10^{-k}$, we need to have the following:

$$\frac{1}{2} + \frac{10^{-k}}{2} = \frac{1}{2}\left(1 + erf\left(\frac{x}{\sigma\sqrt{2}}\right)\right)$$

where $\frac{10^{-k}}{2}$ comes from the fact that we need to use a two-sided confidence interval and $x$ is the tolerance. Solving for $x$, we get:

$$1 + 10^{-k} = 1 + erf\left(\frac{x}{\sigma\sqrt{2}}\right)$$

$$10^{-k} = erf\left(\frac{x}{\sigma\sqrt{2}}\right)$$

$$erfinv\left(10^{-k}\right) = \frac{x}{\sigma\sqrt{2}}$$

$$x = erfinv\left(10^{-k}\right)\sigma\sqrt{2}$$

where $erfinv$ is the inverse error function.

We use math package function Erfinv to approximate $erfinv$ and the Go DP Library's sigmaForGaussian function to compute $\sigma$ with $s_2$, $\varepsilon$ and $\delta$.

# Mean

Complementary tolerance calculation for Mean is the same as regular tolerance calculation for mean, except we use complementary Laplace/Gaussian tolerance instead of the regular Laplace/Gaussian tolerance when computing maximum and minimum values for count and normalizedSum.