

# trait impl PSRN for TulapPSRN

Yu-Ju Ku, Jordan Awan, Aishwarya Ramasethu, Michael Shoemate

July 16, 2024

This proof resides in “**contrib**” because it has not completed the vetting process.

Proves soundness of **TulapPSRN**.

**edge** accepts parameter **self**, containing the state of the Tulap sampler and **R** specifying the rounding mode.

This implementation is susceptible to floating-point vulnerabilities.

**Warning 1** (Code is not constant-time). The implementation of **edge** uses procedures that are vulnerable to timing attacks.

## PR History

- [Pull Request #1126](#)

## 1 Hoare Triple

### Preconditions

- Variable **self** is of type **TulapPSRN**.
- Generic **R** denotes the rounding mode, one of "up" or "down".

### Pseudocode

```
1 from math import exp
2
3
4 class TulapPSRN(object):
5     def __init__(self, shift, epsilon, delta) -> None:
6         self.shift = shift
7         self.epsilon = epsilon
8         self.delta = delta
9         self.uniform = UniformPSRN()
10        self.precision = 50
11
12    def q_cnd(self, u, c, R): # CND quantile function for f
13        if u < c:
14            return self.q_cnd(1 - self.f(u, R), self.f(u, R), c) - 1
15        elif c <= u <= 1 - c: # the linear function
16            return (u - 1 / 2) / (1 - 2 * c)
```

```

17         else:
18             return self.q_cnd(self.f(1 - u, R), self.f(u, R), c) + 1
19
20     def f(self, u, _R):
21         epsilon, delta = self.epsilon, self.delta
22         return max(0, 1 - delta - exp(epsilon) * u, exp(-epsilon) * (1 - delta - u))
23
24     def edge(self, R):
25         epsilon, delta = self.epsilon, self.delta
26         unif = self.uniform.edge(R)
27         c = (1 - delta) / (1 + exp(epsilon))
28         if c == 0.5:
29             return None
30
31         return self.q_cnd(unif, c, R)
32
33     def refine(self):
34         self.precision += 1
35         self.uniform.refine()
36
37     def refinements(self):
38         return self.precision

```

## Postcondition

`edge` returns an estimate of the true Tulap sample, a distribution with CDF defined in `make_tulap`. This mechanism is not implemented in a fashion that accurately samples from the Tulap distribution, so it may be subject to artifacts.

## 2 Proof

*Proof.* The cdf of  $\text{Tulap}(0, b, q)$  is

$$F_N(x) = \begin{cases} 0 & F_{N_0}(x) < q/2 \\ \frac{F_{N_0}(x) - q/2}{1 - q} & q/2 \leq F_{N_0}(x) \leq 1 - q/2 \\ 1 & F_{N_0}(x) > 1 - q/2. \end{cases}$$

By inspection, the fixed point of  $f_{\epsilon, \delta}$  is  $c = \frac{1 - \delta}{1 + e^\epsilon}$ . It is easy to verify that  $F_N(x) = c(1/2 - x) + (1 - c)(x + 1/2)$  for  $x \in (-1/2, 1/2)$ .

The function then uses the inverse transform of a sample of a uniform RV to sample a Tulap RV centered at zero. The function then returns the value, shifted by `self.shift`. □