`trait impl InverseCDF for TulapRV`

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

November 20, 2024

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

Proves soundness of `TulapRV`. `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 `TulapRV`.
- Generic `R` denotes the rounding mode, one of "up" or "down".

Pseudocode

```python

class TulapRV(object):
    def __init__(self, shift, epsilon, delta) -> None:
        self.shift = shift
        self.exp_eps = Fraction(epsilon.neg_inf_exp())
        self.exp_neg_eps = Fraction((-epsilon).inf_exp())
        self.c = (1 - delta) / (1 + self.exp_eps)
        self.delta = delta
        self.uniform = UniformPSRN()

        if c >= 0.5:
            raise ValueError("c must be less than 1/2")

    def q_cnd(self, unif) -> Fraction | None:  # CND quantile function for f
        if unif < c:
            return self.q_cnd(1 - self.f(unif)) - 1
        elif unif <= 1 - self.c:  # the linear function
```

```
            num = unif - 1 / 2
            den = 1 - 2 * self.c
            if den.is_zero():
                return
            return num / den
        else:
            return self.q_cnd(self.f(1 - unif)) + 1

    def f(self, unif):
        t1 = 1 - self.delta - self.exp_eps * unif
        t2 = self.exp_neg_eps * (1 - self.delta - unif)
        return max(t1, t2, 0)

    def edge(self, r_unif, _refinements, _R):
        return self.q_cnd(r_unif) + self.shift
```

Postcondition

`edge` returns an upper or lower bound for the true Tulap sample, a distribution with CDF defined in `make_tulap`.

2 Proof

Proof.

Proposition 1. *The quantile function $F_f^{-1} : (0,1) \to \mathbb{R}$ for F_f can be expressed as*

$$F_f^{-1}(u) = \begin{cases} F_f^{-1}(1 - f(u)) - 1 & u < c \\ \frac{u-1/2}{1-2c} & c \le u \le 1-c \\ F_f^{-1}(f(1-u)) + 1 & u > 1-c, \end{cases}$$

where c is the unique fixed point of f. Furthermore, for any $u \in (0,1)$, the expression $Q_f(u)$ takes a finite number of recursive steps to evaluate. Thus, if $U \sim U(0,1)$, then $F_f^{-1}(U) \sim F_f$.

The cdf of Tulap$(0, b, q)$ is

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

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. Arbitrarily precise estimates of the lower and upper bound of the uniform sample can be retrieved. $F_N(x)$ is computed conservatively, where the values of b and q are computed according to privacy parameters that are no greater than ϵ, δ.

The computation of F_f^{-1} is handled exactly via fractional arithmetic, as it involves no transcendental functions.

The function then returns the outcome, shifted by `self.shift`, a sample from Tulap$(shift, b, q)$, where $b = \exp(-\epsilon)$ and $q = \frac{2\delta b}{1-b+2\delta b}$.

□