Assessing the Quality of Binomial Samplers: A Statistical Distance Framework

Sourav Chakraborty[1], Kuldeep S. Meel[2], and Uddalok Sarkar[1]

[1] Indian Statistical Institute, India
[2] Georgia Institute of Technology, USA and University of Toronto, Canada

Abstract. Randomized algorithms depend on accurate sampling from probability distributions, as their correctness and performance hinge on the quality of the generated samples. However, even for common distributions like Binomial, exact sampling is computationally challenging, leading standard library implementations to rely on heuristics. These heuristics, while efficient, suffer from approximation and system representation errors, causing deviations from the ideal distribution. Although seemingly minor, such deviations can accumulate in downstream applications requiring large-scale sampling, potentially undermining algorithmic guarantees. In this work, we propose statistical distance as a robust metric for analyzing the quality of Binomial samplers, quantifying deviations from the ideal distribution. We derive rigorous bounds on the statistical distance for standard implementations and demonstrate the practical utility of our framework by enhancing APSEst, a DNF model counter, with improved reliability and error guarantees. To support practical adoption, we propose an interface extension that allows users to control and monitor statistical distance via explicit input/output parameters. Our findings emphasize the critical need for thorough and systematic error analysis in sampler design. As the first work to focus exclusively on Binomial samplers, our approach lays the groundwork for extending rigorous analysis to other common distributions, opening avenues for more robust and reliable randomized algorithms.

1 Introduction

Randomization stands as a cornerstone of computer science, permeating algorithm design from the field's earliest days to its cutting-edge developments. From Quicksort [20], one of the most widely used algorithms, to modern cryptographic protocols, randomization has proven indispensable in achieving efficiency and functionality that deterministic approaches struggle to match. While the fundamental question of whether randomization offers additional computational power over determinism remains open, randomized algorithms have established themselves as the preferred choice in numerous domains, including data structures [42], hash functions [11], and probabilistic data structures [6].

At the heart of every randomized algorithm lies its ability to sample from probability distributions. The algorithm's correctness and performance guarantees intrinsically depend on the quality of these samples. For instance, a hash

table's performance relies on the uniformity of its hash function's output distribution, while a Monte Carlo algorithm's accuracy depends on the fidelity of its random sampling process. This fundamental reliance on sampling has led to the development of sophisticated sampling algorithms implemented as standard library functions across programming languages.

While specialized techniques exist for generating high-quality samples from certain distributions [29], these approaches typically circumvent direct probability mass computation through transformations of basic random processes. However, such techniques remain constrained to specific distributions exhibiting particular mathematical properties. In practice, standard library implementations predominantly rely on transformed rejection sampling [22,24], which necessitates explicit probability mass computation. These computations entail multiple arithmetic operations and specialized function evaluations, including factorial and logarithm computations, thereby introducing approximation errors at each step. The accumulation of these errors can significantly impact the statistical properties of the generated samples, potentially compromising the theoretical guarantees of algorithms that depend on them.

In this work, we focus on analyzing standard library implementations of Binomial samplers, which are largely based on transformed rejection sampling techniques [22,23,24]. These implementations require computation of Binomial distribution probability mass, denoted by $\mathsf{b}_{n,p}(k)$, necessitating approximations of factorial terms [33], logarithmic computations [9], and various arithmetic operations. While such approximations enable efficient sampling, they introduce systematic deviations from the ideal Binomial distribution that current implementations neither quantify nor report to users. These deviations can accumulate and potentially trigger cascading failures in downstream applications [4,49]. Despite the widespread adoption of these libraries, there exists no documentation providing precise analysis of accumulated errors.

The primary research problem we address is: *how to develop a rigorous methodology to analyze the errors in existing samplers to provide meaningful measurement of their impact on downstream applications?* This question is particularly pertinent given the increasing reliance on randomized algorithms in critical applications, where understanding and quantifying sampling errors becomes crucial for ensuring system reliability and correctness.

Our first contribution is a rigorous framework for analyzing the quality of existing samplers through the lens of statistical distance. We advocate statistical distance as a theoretically sound metric for quantifying sampler quality, owing to its fundamental property of indistinguishability. Let p and q be two probability distributions with statistical distance at most η, i.e., $\mathsf{d}_{\mathsf{TV}}(\mathsf{p},\mathsf{q}) \leq \eta$. Then, for any statistical test T (even computationally unbounded), the probability of distinguishing between samples from p and q is bounded by η. This fundamental property has profound implications for sampler quality analysis: if a sampler's output distribution has a statistical distance η from the ideal distribution, then the downstream application cannot experience an error greater than η, regardless of its computational sophistication. Building on this theoretical

foundation, we present a detailed analysis of state-of-the-art implementations, deriving concrete bounds on their deviation from the ideal distributions through careful decomposition of numerical approximation errors. We propose an extension to sampler interfaces that exposes statistical distance as an input/output parameter, enabling users to control and monitor the sampling accuracy.

To demonstrate the practical utility of our theoretical framework, we present a comprehensive case study in the context of DNF model counting. We show how our quality measures can be integrated into `APSEst`, a state-of-the-art DNF counter that relies heavily on Binomial sampling. By incorporating our error bounds into the `APSEst`'s analysis framework, we provide the first implementation that offers both scalability and rigorous error guarantees. This integration not only enhances the reliability of the counter but also establishes a blueprint for how sampler quality analysis can be systematically incorporated into broader algorithmic frameworks. Our empirical evaluation demonstrates that this enhanced implementation maintains the efficiency of existing approaches while providing substantially stronger theoretical guarantees.

We believe our work highlights a fundamental challenge in randomized computation: the need for rigorous analysis of sampler implementations to establish precise error bounds and enhance trust in randomized algorithms. While we have focused on Binomial samplers as a crucial first step, the theoretical framework we develop for analyzing sampling error propagation, combined with our practical demonstration in DNF counting, establishes a foundation for future research. A natural direction for future investigation would be the analysis of other standard distributions such as Poisson, Normal, and Beta distributions, each presenting its own unique challenges in implementation and error analysis. Our approach of integrating error analysis into algorithmic frameworks opens new avenues for developing robust randomized algorithms that maintain both theoretical guarantees and practical efficiency. This work will likely motivate the broader community to examine and enhance the reliability of randomized computation implementations, particularly in the context of standard library functions that serve as building blocks for numerous algorithms.

Organisation In section 2, we present the necessary preliminaries and an overview of related concepts that lay the foundation for the rest of the paper. In section 3, we explore related work on error analysis in computational programs and the evaluation of sampler quality. Section 4 outlines our proposal for using statistical distance as a quality metric for samplers, along with the motivation behind this approach. Section 5 offers a detailed discussion on standard Binomial samplers, including our theoretical results, correctness proofs, and error analysis. In section 6, we include a case study on using our sampler to count the number of solutions of a Boolean formula in the Disjunctive Normal Form. Finally, in section 7, we discuss the limitations of our work and future directions.

2 Preliminaries

In this work, we are interested in probability distributions over discrete sets and their samplers. A probability distribution, or simply, a distribution (denoted by p) on a discrete set Ω is a mapping $\mathsf{p} : \Omega \to [0,1]$ such that $\sum_{x\in\Omega} \mathsf{p}(x) = 1$. We define $\mathsf{p}(A) = \sum_{x\in A} \mathsf{p}(x)$ for any $A \subseteq \Omega$. A uniform distribution, or uniform randomness over a set Ω is defined as $\mathsf{p}(x) = \frac{1}{|\Omega|}$ for all $x \in \Omega$. We use the notation $\mathsf{b}_{n,p}$ to denote the Binomial distribution with parameters n and p, which is given by $\mathsf{b}_{n,p}(k) = \binom{n}{k} p^k (1-p)^{n-k}$ for $k \in [0,n]$.

Recall that a Turing Machine (TM) is a theoretical model of computation defined as a tuple $(Q, \Sigma, \Gamma, \sqcup, \Delta, s_0, F)$, where Q is a finite set of states, $\Sigma \subseteq \Gamma \setminus \{\sqcup\}$ is the input alphabet, Γ is the tape alphabet, $\sqcup \in \Gamma$ is the blank symbol, $\Delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function, $s_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of final states [21]. A natural extension of a Turing Machine is a Turing Transducer [34], which, in addition to the input and work tapes, has a separate write-only output tape. A Transducer computes a function $f : \{0,1\}^* \to \{0,1\}^*$, and the output is the content of the output tape when the machine halts. A Probabilistic Turing Machine (PTM) is a Turing Machine that, in addition to the input tape, has access to a read-only random tape filled with an infinite sequence of random bits [1]. On a given input $x \in \{0,1\}^*$ and for each fixed random string $u \in \{0,1\}^\infty$, the machine's behavior is deterministic. A probabilistic Transducer is defined analogously as a PTM equipped with an output tape. It computes an output string $M(x;u)$ for each fixed u, and writes it on the output tape.

A randomized algorithm $\mathcal{A}$ is modeled as a probabilistic Transducer. On input x and a source of randomness, the output of the algorithm $\mathcal{A}(x;u)$ is written on the output tape of the corresponding Transducer. Consequently, $\mathcal{A}$ defines a distribution over outputs depending on the randomness. While this definition assumes that the random bits are drawn from the uniform distribution, we allow randomized algorithms to access randomness from arbitrary distributions. The ability to sample from arbitrary distributions is without loss of generality, since any distribution can be simulated using uniformly random bits. An example of a randomized algoithm is a sampler $\mathsf{Samp}_{\mathsf{p}}$ for a distribution p. Given a source of uniform randomness u, the sampler outputs a sample from p, that is, for all $x \in \Omega$, $\Pr_u(\mathsf{Samp}_{\mathsf{p}} \text{ outputs } x) = \mathsf{p}(x)$. Conversely, a sampler induces an associated probability distribution p from which it draws samples.

2.1 Approximating Factorials

Lanczos Approximation [33] is a widely used method to approximate the factorials with remarkable accuracy. For a fixed value of t, g, and a positive integer $n \geq 1$, the Lanczos approximation of $n!$, denoted by $\mathsf{Fact}^{\mathsf{Lancz}}(n)$, is defined as, $\mathsf{Fact}^{\mathsf{Lancz}}(n) = \sqrt{2\pi}\left(n + g + \frac{1}{2}\right)^{n+\frac{1}{2}} e^{-(n+g+\frac{1}{2})} A_{t,g}(n)$. The polynomial $A_{t,g}(n)$ contains t terms. The accuracy of the approximation depends on the number of terms t in its expansion, as well as on the constant g. Here g is any real constant

such that $g+\frac{1}{2} > 0$. The parameters g and t affect the accuracy and convergence rate of the Lanczos approximation, where larger t improves accuracy at the cost of higher computational resources.

In the Lanczos approximation, a uniform error bound [41] can be established, which provides a measure of how closely the Lanczos approximation approximates the factorial function for all relevant inputs. Given t, g the uniform error bound $\zeta_{t,g}$ of the approximation is defined by, $\zeta_{t,g} = \sup_{n\in\mathbb{N}} \left| n! - \mathsf{Fact}^{\mathsf{Lancz}}(n) \right|$. Let $\zeta = \sqrt{\frac{\pi}{e}} \cdot |\zeta_{t,g}|$. The relative error can be bounded as follows [41],

$$\frac{|n! - \mathsf{Fact}^{\mathsf{Lancz}}(n)|}{n!} \leq \zeta \tag{1}$$

2.2 Multiple-precision Arithmetic

Given a working precision $\beta > 0$, the set of all definable numbers in this context is expressed as $\mathbb{F} = \left\{ w \cdot 2^e \ : \ \frac{1}{2} \leq |w| \leq 1 \text{ and } e \in \mathbb{Z} \right\}$ [48,38,27]. Here e is an integer denoting the exponent, and the $\mathrm{ulp}(x) = 2^{e-\beta}$, where ulp denotes the unit in the last place [48]. Let $\mathtt{rnd} : \mathbb{R} \to \mathbb{F}$ be the rounding function that rounds a real number to the nearest definable number. The corresponding relative errors are bounded by $\frac{|\mathtt{rnd}(x)-x|}{|x|} \leq \varepsilon$, for $x \neq 0$, where $\varepsilon = \frac{1}{2^\beta}$, is referred to as the *unit round-off* [48,38,27].

We define a set of operations by *basic operations* for which it is possible to directly compute the correct rounding of the result [17]. These operations are $\{+, -, \times, /, \sqrt{}\}$. For any two numbers $x, y \in \mathbb{R}$, the following bound holds:

$$|(\mathtt{rnd}(x) \circledast \mathtt{rnd}(y)) - (x * y)| \leq \varepsilon \cdot |x * y| \tag{2}$$

where $* \in \{+, -, \times, /\}$, and, $\circledast$ is the corresponding operation in $\mathbb{F}$. Same bound holds for $\sqrt{x}$ [27]. For n real numbers $x_1, x_2, \ldots, x_n$, the computed sum $\hat{s} := \mathtt{rnd}(x_1) \oplus \mathtt{rnd}(x_2) \oplus \ldots \oplus \mathtt{rnd}(x_n)$, regardless of the order of computation, deviates from the exact sum $s = \sum_{i=1}^{n} x_i$ by at most following bound [27],

$$|\hat{s} - s| \leq n\varepsilon \sum_{i=1}^{n} |x_i| \tag{3}$$

The basic operations are the building blocks for other advanced operations, such as logarithms, exponentials, and trigonometric functions.

2.3 Approximate Computation of Logarithm

Logarithm computation is generally approximated using the Taylor series. However, for high precision, *arithmetic-geometric-mean (AGM)* [9] is used. Let us consider two sequences $\{w_n\}, \{z_n\}$ of positive real numbers such that, $w_{n+1} = \frac{w_n + z_n}{2}$, $z_{n+1} = \sqrt{w_n \cdot z_n}$. These two sequences converge to the common limit and are denoted by $AGM(w_0, z_0)$.

For $x \in \mathbb{F}$ represented as $x = w \cdot 2^e$, we define the function exponent$(x) = e$. To compute $\log(x)$ using the AGM method, an integer m is computed such that $x2^m > 2^{\beta/2}$. The algorithm then evaluates $AGM(1, 4/s)$ and computes the logarithm as

$$\mathsf{Log}(x) = \frac{\pi}{2AGM(1, 4/s)} - m \log 2$$

In the MPFR library [48], the value of m is fixed as $m = \left\lceil \frac{\beta+3}{2} \right\rceil - \text{exponent}(x)$. This choice of m ensures that $s = x2^m$ lies within the range $[2^{\beta/2}, 2^{\beta}]$. The following lemma provides an error bound for the AGM method.

Lemma 1 (Prop. 2 of [9]). *For the function AGM, the following holds for any $s \geq 4$: $\left| \frac{\pi}{2AGM(1,4/s)} - \log(s) \right| \leq \frac{64}{s^2}(10 + |\log s|)$.*

Let Log be the function that computes the logarithm using the AGM method. Since, for $\beta > 8$, we have $3\log(s) > 10$ and $2^{\beta/2} \leq s \leq 2^{\beta}$, we make the following conclusion:

$$|\mathsf{Log}(x) - \log(x)| \leq \frac{178\beta}{2^{\beta}} \tag{4}$$

We will use the notation $\tau = \frac{178\beta}{2^{\beta}}$ as the additive error bound for the logarithm approximation in the rest of the paper. Note that, $\tau = \mathcal{O}(\varepsilon)$. A similar error bound can be derived for the Taylor series method as well [7,8].

Evaluating the logarithm of the factorial, rather than the factorial itself, is the standard technique. The function $\mathsf{LogFactorial}$ computes the logarithm of the factorial using the Lanczos approximation with fixed parameters t and g.

$$\begin{aligned}\mathsf{LogFactorial}(k) = \frac{1}{2}\mathsf{Log}(2\pi) + \left(k + \frac{1}{2}\right)\mathsf{Log}\left(k + g + \frac{1}{2}\right) \\ - \left(k + g + \frac{1}{2}\right) + \mathsf{Log}\left(A_{t,g}(k)\right)\end{aligned} \tag{5}$$

3 Related Work

The impact of computational approximations has been a longstanding concern in the literature. Considerable effort has been devoted to designing samplers that generate samples with arbitrary precision and provably no deviation from the original distribution, a concept referred to as *exact sampling*. This line of work dates back to Von Neumann and has been further developed in studies such as [29], which propose arbitrarily precise algorithms for sampling from distributions like the normal and exponential. The core idea involves employing a random process that efficiently generates a sample x with probability e^{-x}. Remarkably, this algorithm achieves an expected runtime of $\mathcal{O}(1)$.

Similarly, significant attention has been given to designing exact Binomial samplers [14,16] as well. The approach in [14] employs the geometric distribution to generate Binomial samples but requires $\mathcal{O}(np)$ time. More recent advances by

[16] achieve $\mathcal{O}(\sqrt{n})$ time complexity. Their approach involves efficiently accessing $\mathsf{b}_{n,1/2}$ and leveraging its samples, combined with the binary representation of p, to generate samples from $\mathsf{b}_{n,p}$.

Constant time sampling algorithms for binomial distributions are categorized under the framework of *transformed rejection sampling* [15,44,22,28]. These algorithms achieve a sampling time complexity of $\mathcal{O}(1)$, but at the cost of approximations. This is because the framework needs to evaluate the probability mass function, which is computationally expensive unless approximated.

The impact of numerical accuracy on computational programs has been extensively studied. Significant research has been conducted to analyze errors in arithmetic operations [10,26,25,27,43]. Recently, [5] and [8] have explored how these errors affect the performance of functions such as log-sum-exp and softmax. These studies underscore the critical need to account for the inherent numerical errors when designing algorithms and assessing their practical performance.

Finally, statistical distance has been widely recognized as a key measure of sampler quality. For instance, a series of works [12,35,40,39,2,32,3] focus on designing tests to determine the quality of samplers in terms of the statistical distance between the sampler and the target distribution.

4 Statistical Distance as Quality Metric

Since exact sampling from distributions such as Binomial is computationally expensive for most parameters of interest, the standard libraries rely on approximations to achieve practical efficiency. While these approximations significantly reduce time complexity, they introduce deviation from the actual distribution, effectively causing the samples to come from a distribution different from the intended one. Therefore, we need to focus on a fundamental question: *how do we make systems that rely on samplers trustworthy?*

Simply ignoring these deviations is not advisable, as they can have cascading effects that compromise the correctness of the entire system. Often, a user designs a randomized algorithm $\mathcal{A}$ to solve a particular problem, with an upper bound δ on its failure probability. If $\mathcal{A}$ relies on a standard Binomial sampler without knowledge of the sampler's quality, the program may experience a higher failure rate due to approximations in the underlying samplers.

Our proposal immediately raises the question: how should one measure the quality of the sampler? To this end, we first focus on the fact that the objective of the measurement of quality is to allow the end user to quantify the impact of the usage of the sampler. There are several metrics, such as KL-divergence, statistical distance, and Hellinger distance, that have been proposed in the literature focused on probability distributions that seek to quantify the distance between two probability distributions. In this regard, a natural question is to ask what distance metric we should choose. To this end, we propose the usage of statistical distance (Definition 1) as the metric to report the quality.

Definition 1 (Statistical Distance). *Suppose two distributions* p, q *are defined over the set* Ω. *The Statistical Distance (denoted by* $\mathsf{d_{TV}}$*) between* p *and* q *is defined by,* $\mathsf{d_{TV}}(\mathsf{p},\mathsf{q}) = \frac{1}{2}\sum_{x\in\Omega}|\mathsf{p}(x) - \mathsf{q}(x)| = \sup_{A\subseteq\Omega}\mathsf{p}(A) - \mathsf{q}(A)$.

Our proposal for statistical distance stems from its ability to allow end users to derive the worst-case bounds on the behavior of the system in a *black-box* manner. Formally, this follows from the folklore lemma below, for which we provide a proof for completeness.

Lemma 2. *Let* $\mathcal{A}$ *be a randomized algorithm that uses randomness from a source distribution* p, *and let* Bad *be an event in the output of* $\mathcal{A}$. *If* p *is replaced by another distribution* q, *then the probability of the event* Bad *is bounded by the statistical distance between:*

$$\left|\Pr_{r\sim\mathsf{p}}(\mathcal{A}(x;r)\in\mathsf{Bad}) - \Pr_{r\sim\mathsf{q}}(\mathcal{A}(x;r)\in\mathsf{Bad})\right| \leq \mathsf{d_{TV}}(\mathsf{p},\mathsf{q})$$

Proof. Let $B \subseteq \Omega$ be the set of random strings (or, numbers) that trigger the event Bad. Then, we have $\left|\Pr_{r\sim\mathsf{p}}(\mathcal{A}(x;r)\in\mathsf{Bad}) - \Pr_{r\sim\mathsf{q}}(\mathcal{A}(x;r)\in\mathsf{Bad})\right| = |\mathsf{p}(B) - \mathsf{q}(B)|$. Using the definition of statistical distance (definition 1), $|\mathsf{p}(B) - \mathsf{q}(B)| \leq \sup_{A\subseteq\Omega}\mathsf{p}(A) - \mathsf{q}(A) = \mathsf{d_{TV}}(\mathsf{p},\mathsf{q})$. □

Note that the lemma above imposes no restrictions on $\mathcal{A}$ or the event Bad, highlighting the power of statistical distance as a metric. In particular, if $\mathsf{d_{TV}}(\mathsf{p},\mathsf{q})$ is small, then the end user can be confident in bounding the overall impact on the program. We give a general recipe of how to incorporate statistical distance in the implementation of randomized algorithms.

4.1 Integrating Statistical Distance Analysis in Applications

The correctness of randomized algorithms typically relies on access to exact samples from a target distribution p. However, in practice, algorithms must use samplers that draw from an approximate distribution q, potentially compromising their theoretical guarantees. We propose a systematic framework for incorporating these approximations while maintaining rigorous error bounds through minimal modifications to existing algorithms and their analyses.

Algorithm Modification Let $\mathcal{A}$ be a randomized algorithm that requires samples from distribution p. We modify $\mathcal{A}$ to explicitly track and bound the accumulated error from using an approximate sampler as follows:

1. Introduce an error budget parameter δ_1 representing the maximum allowable error due to sampling approximations.
2. Initialize an error accumulator δ' to track the statistical distance:

$$\delta' \leftarrow 0.$$

3. For each sampler invocation, update the accumulated error:

$$\delta' \leftarrow \delta' + \mathsf{d}_{\mathsf{TV}}(\mathsf{p}, \mathsf{q})$$

   where $\mathsf{d}_{\mathsf{TV}}(\mathsf{p}, \mathsf{q})$ is the pre-computed statistical distance bound.
4. If δ' exceeds the budget ($\delta' > \delta_1$), abort execution.

Analysis Modification Let δ_2 denote the original error probability of algorithm $\mathcal{A}$ assuming access to exact samples from p. After incorporating the sampling approximation error δ_1, the total error probability δ is bounded by:

$$\delta \leq \delta_1 + \delta_2$$

This framework maintains theoretical guarantees while transparently accounting for sampling approximations. The modifications are minimal and the analysis remains straightforward. We demonstrate an end-to-end integration of this approach through a case study in section 6.

An alternative approach would be to directly analyze algorithm $\mathcal{A}$ with respect to the approximate distribution q. However, this presents several challenges. The target distribution p often possesses mathematically convenient properties that facilitate analysis, while the implementation-specific q may lack such properties, making direct analysis intractable. Furthermore, updates to the underlying sampler implementation would inevitably necessitate re-analysis of every dependent algorithm.

Our framework enables separation of concerns: algorithm designers can conduct analysis assuming access to the idealized distribution p, while library developers focus on bounding the statistical distance between p and their implementation q. The errors can then be composed as shown above, providing rigorous bounds with minimal modification to existing analyses.

4.2 Proposal for Extending Sampler Interfaces

To enable seamless integration of our statistical distance framework, we propose extending the interface of existing samplers by incorporating two new components (see fig. 1): (1) an input parameter δ_{in} that allows users to specify the maximum allowable statistical distance from the ideal distribution, and (2) an output parameter δ_{out} that reports the actual statistical distance achieved during sampling. These additions give users fine-grained control over the sampler. By setting δ_{in}, users can explicitly define their tolerance for the statistical distance, while δ_{out} enables real-time monitoring of the sampler's performance. The fine-grained control offered by our interface allows users to make informed decisions about the trade-off between accuracy and performance.

5 Analysis of Standard Binomial Samplers

This section examines standard Binomial sampling algorithms and their inherent errors. We first present the standard Binomial sampling algorithm and then analyze the bounds on the statistical distance between the actual distribution and the distribution from which the sampler draws the samples.

```python
def BinSamp(n, p):
    """
    Input:     n, p
    Output:    sample
    """
    ...
```

```python
def BinSamp(n, p, delta_in):
    """
    Input:     n, p, delta_in
    Output:    sample, delta_out
    """
    ...
```

Fig. 1: Early (left) and new (right) sampler interfaces. The new version includes statistical distance control via `delta_in` and `delta_out`.

5.1 Standard Binomial Sampling Algorithms

This section describes the standard Binomial sampling algorithms commonly used in practice, with particular attention given to Python's implementation. These samplers rely on the method of *transformed rejection sampling* which combines two well-established sampling techniques: (1) inverse transform sampling and (2) rejection sampling. Rejection sampling requires existence of an, *easy to sample from*, hat distribution h such that for all $k \in [n]$, $\mathsf{b}_{n,p}(k) < \alpha \mathsf{h}(k)$ for some $\alpha > 0$, known as *rejection rate*. Inverse transform sampling generates samples from h. Suppose the cumulative distribution of h is denoted by $\mathcal{H}$. Because $\mathcal{H}$ is a cumulative distribution therefore, its inverse $\mathcal{H}^{-1}$ is well defined. To get a sample from h, a uniform random variable u is generated, and correspondingly the sample $k = \lfloor \mathcal{H}^{-1}(u) \rfloor$ is computed. From the principles of inverse transform sampling, we can show that $k \sim \mathsf{h}$. The next step involves rejection sampling. After generating k, another uniform random sample v is generated from $[0, 1]$. The sample is rejected if $v > \frac{\mathsf{b}_{n,p}(k)}{\alpha \mathsf{h}(k)}$, else k is returned.

Among the various Binomial sampling algorithms, the choice of the (inverse) hat distribution $\mathcal{H}^{-1}(u)$ is a key difference. For example, Hörmann [22] considered the following definitions of hat distribution[3] for $-0.5 \leq u \leq 0.5$, which has high acceptance probabilities for Binomial distributions over varied n, p.

$$\mathcal{H}^{-1}(u) = \left(\frac{2\lambda_{n,p}}{(1/2 - |u|)} + \mu_{n,p} \right) u + \nu_{n,p}, \quad \mathsf{h}^{-1}(u) = \frac{\lambda_{n,p}}{(1/2 - |u|)^2} + \mu_{n,p}$$

The parameters $\lambda_{n,p}, \mu_{n,p}, \nu_{n,p}$ depend on the parameters of Binomial distribution n, p. Specifically, in Hörmann's algorithm, the corresponding parameters were chosen to be: $\lambda_{n,p} = -0.05878 + 0.062744\sqrt{np(1-p)} + 0.01p$, $\mu_{n,p} = 1.15 + 2.53\sqrt{np(1-p)}$, $\nu_{n,p} = np + 0.5$. Importantly, our results are not restricted to any specific choice of hat distribution. Instead, we consider any arbitrary hat distribution h and its inverse $\mathcal{H}^{-1}$ that satisfy the conditions of transformed rejection sampling and involve a constant number of basic arithmetic operations. Therefore, for simplicity and readability, we omit the explicit details of $\mathcal{H}$ and h

[3] Since $(\mathcal{H}^{-1})'(u) = \frac{1}{\mathsf{h}(k)}$, we use the notation $\mathsf{h}^{-1}(u) = \frac{1}{\mathsf{h}(k)}$ directly for simplicity.

in the rest of the paper. We will refer to $\mathcal{H}^{-1}$ as the 'inverse function' and h as the 'hat function' or 'hat distribution'.

The expected runtime of these algorithms is proportional to the rejection rate α, and therefore, the runtime is independent of the parameters of the distribution. These algorithms require computing the rejection ratio $r_k = \frac{\mathsf{b}_{n,p}(k)}{\alpha \mathsf{h}(k)}$. But computing this ratio, especially evaluating $\mathsf{b}_{n,p}(k)$ exactly, can be as expensive as exponential in the number of bits. Therefore, an easily computable approximation $\widetilde{r}_k$ is often obtained to achieve fast scalable practical algorithms. Usually, due to scalability purposes, the logarithm of the rejection ratio $\log r_k$ is computed rather than directly computing r_k, which again suffers from other approximation errors due to log computation. Therefore, these algorithms lack sampling exactly from the distribution.

Python implementation of Binomial Sampler Standard implementations of Binomial samplers, such as those available in libraries like the GNU Scientific Library (GSL) [18], are designed to work with 64-bit floating-point numbers. Similarly, Python's standard libraries and NumPy [19], provide an implementation of Hörmann's algorithm for up to the 64-bit floating-point range. Notably, starting from Python version 3.12, this algorithm has also been integrated into the standard random library of Python, offering support for higher precision computations (though still constrained by the arithmetic computational limits of Python's standard library).

Algorithm 1 presents an abstraction of the standard implementation Binomial Sampler. Consistent with the current Python implementation, we assume that the $\mathsf{BinSamp}$ algorithm uses the Lanczos approximation to compute the logarithm of the factorial function. To employ the Lanczos approximation, the algorithm uses $\mathsf{LogFactorial}$ as described in eq. (5). This is one of the most widely used implementations of the Binomial sampling algorithm in practice. We adopt it as a benchmark for developing our error bounds. For clarity, we denote the distribution generated by Python's standard library implementation of Algorithm 1 as $\mathsf{b}^{\mathsf{BinSamp}}_{n,p}$, as opposed to the notation $\mathsf{b}_{n,p}$, which refers to the Binomial distribution with parameters n, p.

5.2 Our Findings

We begin by stating the main theorem of our paper, followed by the supporting lemmas that are used to establish the theorem.

Theorem 1. *Let the precision of the context be $\beta \geq \max(2\lceil \log_2 n \rceil, \lceil -\log_2 p \rceil)$, and let $\mathsf{b}^{\mathsf{BinSamp}}_{n,p}$ denote the distribution from which $\mathsf{BinSamp}$ samples are drawn. The statistical distance between $\mathsf{b}^{\mathsf{BinSamp}}_{n,p}$ and $\mathsf{b}_{n,p}$ is given by:*

$$\mathsf{d}_{\mathsf{TV}}\left(\mathsf{b}_{n,p}, \mathsf{b}^{\mathsf{BinSamp}}_{n,p}\right) \leq (1110\beta + 3cp + c + \alpha c)n\varepsilon + 15\zeta + o(\varepsilon)$$

Where c is a constant determined by the inverse function pair $(\mathcal{H}, \mathsf{h})$, α is the rejection rate, ζ denotes the uniform error bound due to the Lanczos's approx-

Algorithm 1: $\mathsf{BinSamp}(n, p)$

Input : Parameters n, p
Output: Sample k

1 Initialize inverse-function-pair $(\mathcal{H}, \mathsf{h})$
2 Initialize rejection parameter α
3 $l_n \leftarrow \mathsf{LogFactorial}(n)$
4 **while** *True* **do**
5 $\quad v \leftarrow$ uniform random samples within $[0, 1]$
6 $\quad u \leftarrow$ uniform random samples within $[-0.5, 0.5]$
7 $\quad k \leftarrow \lfloor \mathcal{H}^{-1}(u) \rfloor$
8 $\quad l_k \leftarrow \mathsf{LogFactorial}(k)$, $l_{nk} \leftarrow \mathsf{LogFactorial}(n-k)$
9 $\quad l_v \leftarrow l_n \ominus l_k \ominus l_{nk} \oplus k \otimes \mathsf{Log}(p) \oplus (n-k) \otimes \mathsf{Log}(1-p) \oplus \mathsf{Log}(\mathsf{h}^{-1}(u)) \ominus \mathsf{Log}(\alpha)$
10 $\quad$ **if** $\mathsf{Log}(v) \leq l_v$ **then**
11 $\quad\quad$ **return** k

imation, ε represents the unit round off error $\frac{1}{2^\beta}$, and $o(\varepsilon)$ denotes the higher order terms in ε.

The sources of deviations are categorized into two types of errors: (1) `E1`, which arises from errors in transformed rejection sampling caused by inaccuracies in the computation of $\lfloor \mathcal{H}^{-1}(u) \rfloor$ in line 7 of Algorithm 1; and (2) `E2`, which refers to errors in the computation of the rejection ratio, accumulating throughout lines 3, 8, and 9 of Algorithm 1. We will bound the effects of these two errors independently and then combine them to get our main theorem.

Analysis of `E1`: The error `E1` arises from inaccuracies in the inverse transform sampling, specifically due to deviations in the hat distribution h caused by basic arithmetic operations. The key challenge stems from the inverse sampling procedure evaluating $\mathcal{H}^{-1}(u)$ for a uniform random sample u. Since evaluating $\mathcal{H}^{-1}(u)$ requires basic arithmetic operations, the inverse sampling component may produce an incorrect output k', different from the intended value $k \neq k'$ and thereby deviating h into a modified distribution $\widetilde{\mathsf{h}}$.

For $\beta > 2\log_2\lceil n \rceil$, we show that the possible values of k' are limited to $\{k-1, k, k+1\}$. Through careful analysis of the probabilities of k' being $k-1$ or $k+1$, we show that these probabilities are bounded by $\mathcal{O}(k\varepsilon)$. This leads to the bound on $\widetilde{\mathsf{h}}(k)$ which is stated in the following lemma:

Lemma 3. *Suppose $\widetilde{\mathsf{h}}$ is the deviated version of the hat distribution h due to the error in the computation of $\mathcal{H}^{-1}(u)$. Then, if the error is distributed uniformly over the range $[-\varepsilon_{\mathcal{H}}, \varepsilon_{\mathcal{H}}]$ and $\varepsilon_{\mathcal{H}} \leq \frac{1}{2}$, then for any $k \in [n]$, $(1-\varepsilon_{\mathcal{H}}(3k+1))\mathsf{h}(k) \leq \widetilde{\mathsf{h}}(k) \leq (1 + w_k\varepsilon_{\mathcal{H}}(k+2))\mathsf{h}(k)$ where, $w_k = \max\left(\frac{\mathsf{h}(k-1)}{\mathsf{h}(k)}, \frac{\mathsf{h}(k+1)}{\mathsf{h}(k)}\right)$*

Analysis of `E2`: The error `E2` stems from inaccuracies in the rejection ratio computation, which are influenced by factorial approximations, basic arithmetic operations, and log computation approximations. Specifically, the rejection ratio

error originates from three primary sources: (1) errors introduced by factorial approximations, (2) errors and approximations in basic arithmetic operations and log computation during the rejection ratio computation. Both of these errors contribute to the error in the rejection ratio computation. The computed rejection ratio in line 9 of Algorithm 1 is denoted as $\widetilde{r}_k$, specifically, $\widetilde{r}_k = \frac{\mathsf{b}^{\mathsf{BinSamp}}_{n,p}(k)}{\alpha \mathsf{h}(k)}$.

Since Python's standard library uses the Lanczos approximation for log computations, we bound the relative error of factorial approximation using the corresponding error bound of the Lanczos approximation (see section 2.1 for details). To bound E2, we also need to understand the relative error in basic arithmetic operations and additive error bounds for logarithm approximations (see section 2.2 for details). By combining these error bounds, we establish a relative error bound for the rejection ratio, formalized in the following lemma.

Lemma 4. *Let r_k denote the actual rejection ratio, defined as $r_k = \frac{\mathsf{b}_{n,p}(k)}{\alpha \mathsf{h}(k)}$, and let $\widetilde{r}_k$ denote the computed rejection ratio, defined as $\widetilde{r}_k = \frac{\mathsf{b}^{\mathsf{BinSamp}}_{n,p}(k)}{\alpha \mathsf{h}(k)}$, for any k in $[n]$. Then, for any $k \in [0, n]$, we have:*

$$\left| \frac{\widetilde{r}_k}{r_k} - 1 \right| \leq (1110n + 2540)\beta\varepsilon + 14\varepsilon \log\left(\mathsf{h}(k)\right) + 15\zeta + o(\varepsilon)$$

By combining the error bounds from both the rejection ratio computation and hat distribution deviation, we obtain the final bound on the statistical distance, as stated in Theorem 1.

5.3 Detailed Technical Analysis

We start by proving the main theorem of our paper using Lemma 3 and Lemma 4.

Proof (of theorem 1). Without loss of generality, assume $\mathsf{h}(-1) = \mathsf{h}(n+1) = \mathsf{b}_{n,p}(-1) = \mathsf{b}_{n,p}(n+1) = 0$ and $r_{-1} = r_{n+1} = 1$. Let us define the event accept as the event such that a sample k is sampled by the sampler. In $\mathsf{BinSamp}$ a sample drawn from $\widetilde{\mathsf{h}}$ is accepted with probability $\widetilde{r}_k$. Therefore, the acceptance probability is given by, $\Pr(\mathsf{accept}) = \sum_{k=0}^{n} \Pr(\mathsf{accept}|k) \Pr(k) = \sum_{k=0}^{n} \widetilde{r}_k \widetilde{\mathsf{h}}(k)$. Substituing the lower bounds for $\widetilde{r}_k$, $\widetilde{\mathsf{h}}(k)$ from Lemma 4 and Lemma 3 we get:

$$\begin{aligned}
\Pr(\mathsf{accept}) &\geq \sum_{k=0}^{n} (1 - (1110n + 2540)\beta\varepsilon - 14\varepsilon \log\left(\mathsf{h}^{-1}(u)\right) - 15\zeta - o(\varepsilon)) \\
&\qquad\qquad (1 - (3k+1)\varepsilon_{\mathcal{H}}) r_k h(k) \\
&\geq \sum_{k=0}^{n} (1 - (1110n + 2540)\beta\varepsilon - 14\varepsilon \log\left(\mathsf{h}^{-1}(u)\right) - 15\zeta - o(\varepsilon)) \\
&\qquad\qquad (1 - (3k+1)\varepsilon_{\mathcal{H}}) \frac{\mathsf{b}_{n,p}(k)}{\alpha}
\end{aligned}$$

To complete the lower bound we first observe that,

$$\sum_{k=0}^{n} \log(\mathsf{h}^{-1}(u))\mathsf{b}_{n,p}(k) = \sum_{k=0}^{n} \log\left(\frac{\mathsf{b}_{n,p}(k)}{\mathsf{h}(k)}\right)\mathsf{b}_{n,p}(k) - \sum_{k=0}^{n} \log(\mathsf{b}_{n,p}(k))\mathsf{b}_{n,p}(k).$$

Given that $\frac{\mathsf{b}_{n,p}(k)}{\mathsf{h}(k)} \leq \alpha$ for all k, and that the entropy of Binomial distribution satisfies $\mathbb{E}[-\log(\mathsf{b}_{n,p}(k))] \leq \frac{1}{2}\log_2(2\pi enp(1-p))$, we conclude that

$$\sum_{k=0}^{n} \log(\mathsf{h}^{-1}(u))\mathsf{b}_{n,p}(k) \leq \alpha + \frac{1}{2}\log_2(2\pi enp(1-p)).$$

Thus we can lower bound acceptance probability $\Pr(\mathsf{accept})$ as follows,

$$\Pr(\mathsf{accept}) \geq \big(1 - (1110n+2554)\beta\varepsilon - 14\alpha\varepsilon + \log\left(\mathsf{h}^{-1}(u)\right) - 15\zeta \\ -(3np+1)\varepsilon_{\mathcal{H}} - o(\varepsilon)\big) \cdot \frac{1}{\alpha}$$

Therefore, the probability of observing a point k under the sampler's output distribution is given by $\mathsf{b}^{\mathsf{BinSamp}}_{n,p}(k) = \Pr(k|\mathsf{accept})$. Applying Bayes' rule we get, $\mathsf{b}^{\mathsf{BinSamp}}_{n,p}(k) = \frac{\Pr(\mathsf{accept}|k)\Pr(k)}{\Pr(\mathsf{accept})} = \frac{\widetilde{r_k}\widetilde{\mathsf{h}}(k)}{\Pr(\mathsf{accept})}$. By using the upper bound of $\widetilde{r}_k$, from Lemma 4,

$$\begin{aligned}\mathsf{b}^{\mathsf{BinSamp}}_{n,p}(k) &\leq \frac{1 + (1110n+2540)\beta\varepsilon + 14\varepsilon\log\left(\mathsf{h}^{-1}(u)\right) + 15\zeta + o(\varepsilon)}{1 - (1110n+2554)\beta\varepsilon - 15\zeta - (3np+1)\varepsilon_{\mathcal{H}} - o(\varepsilon)} \cdot \alpha r_k \widetilde{\mathsf{h}}(k) \\ &\leq (1 + (2220n+5080)\beta\varepsilon + 28\varepsilon\log\left(\mathsf{h}^{-1}(u)\right) + 30\zeta \\ &\qquad + (6np+2)\varepsilon_{\mathcal{H}} + o(\varepsilon)) \cdot \frac{\mathsf{b}_{n,p}(k)\widetilde{\mathsf{h}}(k)}{\mathsf{h}(k)}\end{aligned}$$

The last inequality follows from assuming $(1110n+2540)\beta\varepsilon + (3np+1)\varepsilon_{\mathcal{H}} + 14\varepsilon\log\left(\mathsf{h}^{-1}(u)\right) + 15\zeta + o(\varepsilon) \leq \frac{1}{2}$. Next we bound the ratio $\frac{\widetilde{\mathsf{h}}(k)}{\mathsf{h}(k)}$ using Lemma 3,

$$\begin{aligned}\mathsf{b}^{\mathsf{BinSamp}}_{n,p}(k) &\leq \Big(1 + (2220n+5080)\beta\varepsilon + 28\varepsilon\log\left(\mathsf{h}^{-1}(u)\right) + 30\zeta \\ &\qquad + (6np+2)\varepsilon_{\mathcal{H}} + o(\varepsilon)\Big) \cdot (1 + w_k\varepsilon_{\mathcal{H}}(k+2)) \cdot \mathsf{b}_{n,p}(k) \\ &\leq \Big(1 + (2220n+5080)\beta\varepsilon + 28\varepsilon\log\left(\mathsf{h}^{-1}(u)\right) + 30\zeta + (6np+2)\varepsilon_{\mathcal{H}} + \\ &\qquad w_k\varepsilon_{\mathcal{H}}(k+2) + o(\varepsilon)\Big) \cdot \mathsf{b}_{n,p}(k)\end{aligned}$$

Where $w_k = \max\left(\frac{\mathsf{h}(k-1)}{\mathsf{h}(k)}, \frac{\mathsf{h}(k+1)}{\mathsf{h}(k)}\right)$. Since $r_k \leq 1$, it follows that $\mathsf{h}(k) \geq \frac{\mathsf{b}_{n,p}(k)}{\alpha}$ and $\mathsf{h}(k+1) = \frac{\mathsf{b}_{n,p}(k+1)}{\alpha r_{k+1}}$. This implies $\frac{\mathsf{h}(k+1)}{\mathsf{h}(k)} \leq \frac{\mathsf{b}_{n,p}(k+1)}{\mathsf{b}_{n,p}(k)r_{k+1}}$. Similarly, we have $\frac{\mathsf{h}(k-1)}{\mathsf{h}(k)} \leq \frac{\mathsf{b}_{n,p}(k-1)}{\mathsf{b}_{n,p}(k)r_{k-1}}$. Combining these, w_k can be upper bounded as follows,

$$w_k = \max\left(\frac{\mathsf{h}(k-1)}{\mathsf{h}(k)}, \frac{\mathsf{h}(k+1)}{\mathsf{h}(k)}\right) \leq \max\left(\frac{\mathsf{b}_{n,p}(k-1)}{\mathsf{b}_{n,p}(k)r_{k-1}}, \frac{\mathsf{b}_{n,p}(k+1)}{\mathsf{b}_{n,p}(k)r_{k+1}}\right)$$

This yields the following bound: $w_k \mathsf{b}_{n,p}(k) \leq \max\left(\frac{\mathsf{b}_{n,p}(k-1)}{r_{k-1}}, \frac{\mathsf{b}_{n,p}(k+1)}{r_{k+1}}\right)$. Thus, $w_k \mathsf{b}_{n,p}(k) \leq \max\left(\mathsf{h}(k-1), \mathsf{h}(k+1)\right) \leq \mathsf{h}(k-1) + \mathsf{h}(k+1)$. Using this bound, we can upper bound the sum $\sum_{k=0}^{n} w_k \varepsilon_{\mathcal{H}}(k+2)\mathsf{b}_{n,p}(k)$ as follows,

$$\begin{aligned}\sum_{k=0}^{n} w_k \varepsilon_{\mathcal{H}}(k+2)\mathsf{b}_{n,p}(k) &\leq \alpha\varepsilon_{\mathcal{H}} \sum_{k=0}^{n}(k+2)\left(\mathsf{h}(k-1) + \mathsf{h}(k+1)\right) \\ &\leq 2\alpha\varepsilon_{\mathcal{H}}(n+2)\end{aligned}$$

The last inequality follows from the fact that $k \leq n$, $\sum_{i=0}^{n-1} \mathsf{h}(i) \leq 1$ and $\sum_{i=1}^{n} \mathsf{h}(i) \leq 1$. Therefore, the statistical distance between the sampler's distribution and the Binomial distribution is given by,

$$\begin{aligned}&\mathsf{d}_{\mathsf{TV}}(\mathsf{b}_{n,p}^{\mathsf{BinSamp}}, \mathsf{b}_{n,p}) = \frac{1}{2}\sum_{k=0}^{n}\left|\mathsf{b}_{n,p}^{\mathsf{BinSamp}}(k) - \mathsf{b}_{n,p}(k)\right| \\ \leq &\frac{1}{2}\sum_{k=0}^{n}\Big|(2220n+5080)\beta\varepsilon + 28\varepsilon\log\left(\mathsf{h}^{-1}(u)\right) + 30\zeta + (6np+2)\varepsilon_{\mathcal{H}} \\ &\qquad\qquad +2\alpha\varepsilon_{\mathcal{H}}(n+2) + o(\varepsilon)\Big| \cdot \mathsf{b}_{n,p}(k) \\ \leq &(1110n+2554)\beta\varepsilon + 14\alpha\varepsilon + 15\zeta + (3np+1)\varepsilon_{\mathcal{H}} + \alpha\varepsilon_{\mathcal{H}}(n+2) + o(\varepsilon) \\ = &(1110\beta + 3cp + c + \alpha c)n\varepsilon + 15\zeta + o(\varepsilon)\end{aligned}$$

This completes the proof. □

Error in Inverse Transform Sampling In this subsection, we analyze the error introduced during the inverse transform sampling process which arises from the arithmetic operations involved in evaluating the inverse hat function $\mathcal{H}^{-1}(u)$. Instead of assuming a specific hat distribution, we argue that for any hat distribution, the multiplicative error introduced during the computation of $\mathcal{H}^{-1}(u)$ is multiplicatively bounded by $\varepsilon_{\mathcal{H}} := c\varepsilon$, where $c > 0$ depends on the number of basic arithmetic operations[4] $*$ used in $\mathcal{H}^{-1}$ with $* \in \{+, -, \times, /, \sqrt{}\}$. Consequently, the computed value of $\mathcal{H}^{-1}(u)$ in line 7 of $\mathsf{BinSamp}$ falls within the range $[(1-\varepsilon_{\mathcal{H}})\mathcal{H}^{-1}(u), (1+\varepsilon_{\mathcal{H}})\mathcal{H}^{-1}(u)]$. To model the impact of the error we assume it to be uniformly distributed over the range $[-\varepsilon_{\mathcal{H}}, \varepsilon_{\mathcal{H}}]$. While a Gaussian distribution might seem like a natural choice for such errors, the uniform distribution offers a more conservative approach. Among all zero-mean Gaussian like distributions with bounded support, the uniform distribution has the heaviest tails within its support. This characteristic makes it suitable for handling errors in the worst-case scenario. In particular, if the mean of the error is centered at $\mathcal{H}^{-1}(u)$, then assuming a uniform distribution for the error allows us to upper bound deviations in $\mathsf{h}(k)$ more conservatively. This ensures that our bounds remain valid even under pessimistic error assumptions. We restate the Lemma 3 for completness and provide its proof in the full version.

[4] For most practical hat functions, we typically have $c \leq 100$, which implies that $\varepsilon_{\mathcal{H}} \leq \frac{1}{2}$ when the precision parameter $\beta > 8$.

Lemma 3. *Suppose* $\widetilde{\mathsf{h}}$ *is the deviated version of the hat distribution* h *due to the error in the computation of* $\mathcal{H}^{-1}(u)$*. Then, if the error is distributed uniformly over the range* $[-\varepsilon_{\mathcal{H}}, \varepsilon_{\mathcal{H}}]$ *and* $\varepsilon_{\mathcal{H}} \leq \frac{1}{2}$*, then for any* $k \in [n]$*,* $(1-\varepsilon_{\mathcal{H}}(3k+1))\mathsf{h}(k) \leq \widetilde{\mathsf{h}}(k) \leq (1 + w_k\varepsilon_{\mathcal{H}}(k+2))\mathsf{h}(k)$ *where,* $w_k = \max\left(\frac{\mathsf{h}(k-1)}{\mathsf{h}(k)}, \frac{\mathsf{h}(k+1)}{\mathsf{h}(k)}\right)$

Error in Rejection Ratio Computation In this subsection, we will analyze the impact of factorial approximations, basic arithmetic operations, and log computation approximations on the rejection sampling process. Before proceeding further, we introduce the notation $\mathsf{b}^{\mathsf{Inter}}_{n,p}$, which we refer to it as the *intermediate* distribution, where only the factorial computations are approximated. The key result of this subsection is Lemma 4, whose proof builds on two auxiliary lemmas: one addressing errors from arithmetic and log approximations, and another handling factorial approximation. Proofs are deferred to the full version due to space constraints.

Lemma 5. *Let* r^{Inter}_k *denotes the ratio* $r^{\mathsf{Inter}}_k = \frac{\mathsf{b}^{\mathsf{Inter}}_{n,p}(k)}{\alpha\mathsf{h}(k)}$*, and let* $\widetilde{r}_k$ *denote the computed rejection ratio, defined as* $\widetilde{r}_k = \frac{\mathsf{b}^{\mathsf{BinSamp}}_{n,p}(k)}{\alpha\mathsf{h}(k)}$*, for any* k *in* $[n]$*. Then, for all* $k \in [n]$*,* $\left|\frac{\widetilde{r}_k}{r^{\mathsf{Inter}}_k} - 1\right| \leq (1110n + 2540)\beta\varepsilon + 14\varepsilon\log(\mathsf{h}(k)) + o(\varepsilon)$

Lemma 6. *For all* $k \in [n]$*,* $(1-15\zeta)\mathsf{b}_{n,p}(k) \leq \mathsf{b}^{\mathsf{Inter}}_{n,p}(k) \leq (1+15\zeta)\mathsf{b}_{n,p}(k)$*, where* ζ *denotes the uniform error bound due to the Lanczos approximation and* $\mathsf{b}^{\mathsf{Inter}}_{n,p}$ *denotes the intermediate distribution.*

For completeness, we restate the Lemma 4 whose proof follows by combining Lemma 5 and Lemma 6.

Lemma 4. *Let* r_k *denote the actual rejection ratio, defined as* $r_k = \frac{\mathsf{b}_{n,p}(k)}{\alpha\mathsf{h}(k)}$*, and let* $\widetilde{r}_k$ *denote the computed rejection ratio, defined as* $\widetilde{r}_k = \frac{\mathsf{b}^{\mathsf{BinSamp}}_{n,p}(k)}{\alpha\mathsf{h}(k)}$*, for any* k *in* $[n]$*. Then, for any* $k \in [0, n]$*, we have:*

$$\left|\frac{\widetilde{r}_k}{r_k} - 1\right| \leq (1110n + 2540)\beta\varepsilon + 14\varepsilon\log(\mathsf{h}(k)) + 15\zeta + o(\varepsilon)$$

6 Case Study: DNF Counting

This case study demonstrates how our proposed bounds on the statistical distance between the sampler and the Binomial distribution can be easily integrated into practical tools. To demonstrate the applicability of these bounds, we utilize them in conjunction with an off-the-shelf Binomial sampler to implement the DNF Counting algorithm `APSEst` [37].

A DNF formula is a disjunction of conjunctions of literals, where each conjunction (clause) represents a set of conditions. For example, $(x_1 \wedge \neg x_2) \vee$

$(x_2 \wedge \neg x_3)$ is a DNF formula with clauses like $(x_1 \wedge \neg x_2)$. A DNF formula $\varphi := \varphi_1 \vee \ldots \vee \varphi_m$ has m clauses, and the number of its solutions is denoted as $|sol(\varphi)|$. The problem of counting the $|sol(\varphi)|$ for a DNF formula φ is #P-hard. To address this challenge, various Fully Polynomial-Time Randomized Approximation Schemes (FPRAS) have been developed [30,31,13]. Given a DNF formula φ and tolerance and confidence parameters $\varepsilon, \delta \in [0,1]$, these FPRAS return $\widehat{n} \in [(1-\varepsilon)|sol(\varphi)|, (1+\varepsilon)|sol(\varphi)|]$ with probability at least $(1-\delta)$.

The most recent progress in sampling-based DNF counting FPRAS is embodied by the algorithm `APSEst` [37]. Given a DNF formula φ, the `APSEst` returns an ε multiplicative approximation of $|sol(\varphi)|$ with high probability. The algorithm maintains a bucket $\mathcal{X}$ to keep sampled solutions from DNF clauses and, also, a probability parameter p such that, for any solution σ of φ, σ belongs to $\mathcal{X}$ with probability p. To achieve this goal, the algorithm removes all the elements σ from bucket $\mathcal{X}$ if σ satisfies φ_i. The algorithm next samples new solutions from φ_i. To determine the number of solutions, the `APSEst` asks for a sample N_i from Binomial distribution $\mathsf{b}_{|sol(\varphi_i)|,p}$ and adds N_i many new satisfying assignments of φ_i to $\mathcal{X}$. If the bucket overflows, the algorithm keeps on removing elements uniformly from the bucket until the bucket size falls under the threshold. The end goal of `APSEst` is to output the ratio $\frac{|\mathcal{X}|}{p}$ which is a good estimate of $|sol(\varphi)|$.

Since `APSEst` heavily relies on the Binomial sampler, the theoretical guarantees of `APSEst` are contingent on the quality of the Binomial sampler. This case study illustrates how our results allow users to maintain the theoretical guarantees of `APSEst`. Computing bounds on $\mathsf{d_{TV}}$ between the Binomial distribution and the sampler, users can adjust the confidence parameter δ in `APSEst` to account for errors from the underlying Binomial sampler, thereby ensuring correctness with theoretical guarantees.

Algorithm Modification We demonstrate how easily the `APSEst` algorithm can be modified to incorporate our statistical distance bounds. We denote this modified version as `APSEst2`, detailed in algorithm 2, with highlighted modifications. The primary difference between `APSEst` and `APSEst2` lies in handling the confidence parameter δ to account for the errors due to the underlying binomial sampler. Therefore, `APSEst2` takes another parameter $\kappa \in [0,1]$ to adjust the error budget for the sampler, such that, $\kappa\delta$ is the error budget for the sampler and $(1-\kappa)\delta$ is the error budget for the algorithm. If the accumulated error due to the sampler exceeds the error budget, `APSEst2` halts immediately and returns Fail. The user can restart the algorithm using a larger value of κ to accommodate an increased error budget.

Analysis Modification We now illustrate how simply the analysis of `APSEst` can be modified to `APSEst2`. Recall that we are concerned with the event: $\frac{|\mathcal{X}|}{p} \notin (1 \pm \varepsilon)|sol(\varphi)|$, which we will refer to as Bad. Note that, $\delta_1 = \kappa\delta$ is the error budget for the sampler and $\delta_2 = (1-\kappa)\delta$ is the error budget for the algorithm. By the correctness guarantee of `APSEst`, $\Pr_{\mathsf{b}_{n,p}}(\mathsf{Bad}) \leq \delta_2$. During the execution of `APSEst2`, if the algorithm invokes the sampler t times with parameters n_i, p_i,

Algorithm 2: `APSEst2`$(\varphi, \varepsilon, \delta, \kappa)$
(The modifications from `APSEst` to `APSEst2` are highlighted)

$\delta_1 \leftarrow \kappa\delta$
$\delta_2 \leftarrow (1-\kappa)\delta$
Initialize $T \leftarrow \left(\frac{\log(4/\delta_2)+\log m}{\varepsilon^2}\right)$
Initialize $p \leftarrow 1$, $\mathcal{X} \leftarrow \emptyset$
$\delta' \leftarrow 0$
for $i = 1$ *to* m **do**
  for *all* $\sigma \in \mathcal{X}$ **do**
    if $\sigma \vDash \varphi_i$ **then**
      remove σ from $\mathcal{X}$
  $N_i \leftarrow \mathsf{BinSamp}(|sol(\varphi_i)|, p)$
  $\delta' \leftarrow \delta' + \delta^i_{|sol(\varphi_i)|,p}$
  if $\delta' > \delta_1$ **then**
    return Fail
  Add N_i distinct random solutions of ϕ_i to $\mathcal{X}$
  while $|\mathcal{X}| > T$ **do**
    $p = p/2$
    Throw away each element of $\mathcal{X}$ with probability $\frac{1}{2}$
Output $\frac{|\mathcal{X}|}{p}$

then from Lemma 2, we have

$$\Pr_{\mathsf{b}^{\mathsf{BinSamp}}_{n,p}}(\mathsf{Bad}) \leq \Pr_{\mathsf{b}_{n,p}}(\mathsf{Bad}) + \sum_{i=1}^{t} \mathsf{d_{TV}}(\mathsf{b}_{n_i,p_i}, \mathsf{b}^{\mathsf{BinSamp}}_{n_i,p_i})$$

Suppose during the execution, the computed $\mathsf{d_{TV}}$ bounds using Theorem 1 are given by $\delta^1_{n_1,p_1}, \delta^2_{n_2,p_2}, \ldots, \delta^t_{n_t,p_t}$ such that $\delta' = \sum_{i=1}^{t} \delta^i_{n_i,p_i}$. Therefore, if `APSEst2` does not halt then $\Pr_{\mathsf{b}^{\mathsf{BinSamp}}_{n,p}}(\mathsf{Bad}) \leq \Pr_{\mathsf{b}_{n,p}}(\mathsf{Bad}) + \sum_{i=1}^{t} \mathsf{d_{TV}}(\mathsf{b}_{n,p}, \mathsf{b}^{\mathsf{BinSamp}}_{n,p}) \leq \delta_2 + \delta' \leq \delta_2 + \delta_1 \leq \delta$. Thus, by the correctness of `APSEst` the count returned by `APSEst2` is within the ε error bound. If `APSEst2` halts and returns Fail, this implies that the error budget of the sampler has been exceeded.

6.1 Experimental Setup and Evaluation Results

We conducted accuracy experiments following the methodology of [46] to assess the reliability of the counts returned by `APSEst2`. Specifically, we compared the counts from `APSEst2` with those from `Ganak` [45], an exact counting tool. We used a comprehensive benchmark suite from [46] to evaluate the performance and accuracy of the algorithms. This suite consists of DNF formulas with the number of variables ranging from 100 to 700 and clause counts ranging from 30 to 700. Following prior work [46,36,47], we adopted the standard settings for the tolerance parameter ($\varepsilon = 0.8$) and the confidence parameter ($\delta = 0.36$),

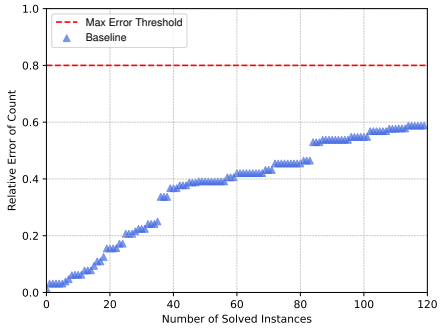

Fig. 2: Accuracy experiment results for `APSEst2`. The red line represents the tolerance factor ($\varepsilon = 0.8$).

which are commonly used in both model counting competitions and practical applications. Finally, we set κ to 0.5.

Summary of Results Our observations reveal that `APSEst2` delivers counts that are nearly as precise as the exact counts obtained from `Ganak`, as demonstrated in fig. 2. The y-axis of fig. 2 represents the relative error of the counts, with the tolerance parameter ($\varepsilon = 0.8$) marked by a red straight line, while the x-axis represents the instances. We observed that for all instances, `APSEst2` computed counts within the tolerance, demonstrating high accuracy.

Figure 3 presents the reported errors δ' of the Binomial sampler during the execution of `APSEst2`. The left plot presents the reported errors for individual instances, while the right plot groups the average error by the number of clauses in the DNF formula. Across our benchmark suite, the errors remain around 10^{-6}. The error increases with the number of clauses as the number of calls to the Binomial sampler increases. Notably, we observe a 10-fold increase in error when the number of clauses grows from 30 to 700.

The results suggest that `APSEst2` is capable of providing highly reliable approximations of counts across a diverse range of DNF instances, maintaining accuracy within the predefined error margin despite its simple design. The integration of our statistical distance bounds into the algorithm allows users to effectively manage the error budget, ensuring that the algorithm remains robust and reliable even in the presence of approximation errors from the underlying Binomial sampler.

7 Conclusion

In this work, we first identified the sources of deviation in the practical implementations of standard binomial samplers. We observed that exact sampling from distributions is infeasible in practice due to high runtime overhead; thus, implementations inevitably introduce deviations. Accordingly, we proposed the

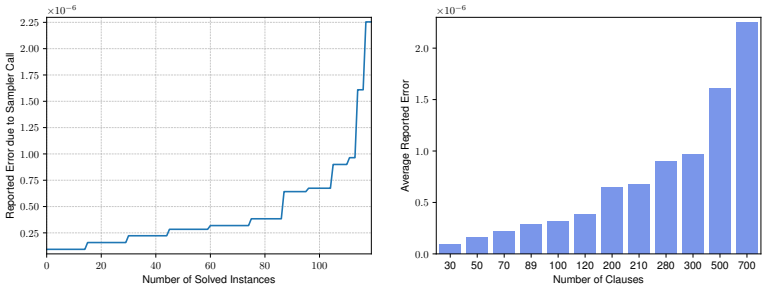

Fig. 3: Left: Reported errors (δ') by `APSEst2` for individual instances. Right: Average error grouped by the number of clauses in the DNF formula, showing a clear upward trend as formula size increases.

usage of statistical distance as the quality metric owing to its ability to allow end users to obtain sound bounds on the *bad* events. We also presented a case study demonstrating the minimal effort required by system designers to incorporate the reported deviation bounds into their systems.

Limitations and Future Work While our current work establishes a foundational framework, there are several limitations and opportunities for future enhancement. The current analysis relies on several simplifying assumptions—for instance, uniformity in the error distribution—which may not hold in more general settings. Additionally, the reported bounds are not yet tight and can be refined for greater accuracy, making this an important avenue for further research. Moreover, the principles of quality measurement can be extended to samplers for other distributions. Developing a general framework for error reporting across various types of samplers would be a valuable contribution to the field. Finally, proposing efficient sampling scheme that can achieve the desired statistical distance with minimal overhead is an open problem that warrants further investigation.

Acknowledgments. Part of this research is supported by DCSW-TAC Project ACMU-24/VPP (SC). Meel acknowledges the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), funding reference number RGPIN-2024-05956. Sarkar acknowledges the support of Google PhD Fellowship. Part of the research was conducted while Sarkar was at the University of Toronto. Computations were performed on the Niagara supercomputer at the SciNet HPC Consortium. SciNet is funded by Innovation, Science and Economic Development Canada; the Digital Research Alliance of Canada; the Ontario Research Fund: Research Excellence; and the University of Toronto.

Disclosure of Interests. The authors declare that they have no competing interests.

References

1. Arora, S., Barak, B.: Computational complexity: a modern approach. Cambridge University Press (2009)
2. Banerjee, A., Chakraborty, S., Chakraborty, S., Meel, K.S., Sarkar, U., Sen, S.: Testing of horn samplers. In: AISTATS (2023)
3. Bhattacharyya, R., Chakraborty, S., Pote, Y., Sarkar, U., Sen, S.: Testing self-reducible samplers. In: AAAI (2024)
4. Binder, K., Heermann, D.W., Binder, K.: Monte Carlo simulation in statistical physics, vol. 8. Springer (1992)
5. Blanchard, P., Higham, D.J., Higham, N.J.: Accurately computing the log-sum-exp and softmax functions. IMA Journal of Numerical Analysis **41**(4), 2311–2330 (2021)
6. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Communications of the ACM **13**(7), 422–426 (1970)
7. Bonnot, P., Boyer, B., Faissole, F., Marché, C., Rieu-Helft, R.: Formally verified bounds on rounding errors in concrete implementations of logarithm-sum-exponential functions. Tech. rep., Inria (2023)
8. Bonnot, P., Boyer, B., Faissole, F., Marché, C., Rieu-Helft, R.: Formally verified rounding errors of the logarithm sum exponential function. In: FMCAD. pp. 251–260. TU Wien Academic Press (2024)
9. Borwein, J.M., Borwein, P.B.: The arithmetic-geometric mean and fast computation of elementary functions. SIAM review **26**(3), 351–366 (1984)
10. Brent, R., Percival, C., Zimmermann, P.: Error bounds on complex floating-point multiplication. Mathematics of Computation **76**(259), 1469–1481 (2007)
11. Carter, J.L., Wegman, M.N.: Universal classes of hash functions. In: STOC (1977)
12. Chakraborty, S., Meel, K.S.: On testing of uniform samplers. In: AAAI (2019)
13. Chakraborty, S., Meel, K.S., Vardi, M.Y.: Algorithmic improvements in approximate counting for probabilistic inference: From linear to logarithmic sat calls. In: IJCAI. pp. 3569–3576 (2016)
14. Devroye, L.: Generating the maximum of independent identically distributed random variables. Computers & Mathematics with Applications **6**(3), 305–315 (1980)
15. Devroye, L.: Nonuniform random sample generation. Handbooks in operations research and management science **13**, 83–121 (2006)
16. Farach-Colton, M., Tsai, M.T.: Exact sublinear binomial sampling. Algorithmica **73**, 637–651 (2015)
17. Fousse, L., Hanrot, G., Lefèvre, V., Pélissier, P., Zimmermann, P.: Mpfr: A multiple-precision binary floating-point library with correct rounding. ACM Transactions on Mathematical Software (TOMS) **33**(2), 13–es (2007)
18. Galassi, M., Davies, J., Theiler, J., Gough, B., Jungman, G., Alken, P., Booth, M., Rossi, F., Ulerich, R.: GNU scientific library. Network Theory Limited Godalming (2002)
19. Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., del Río, J.F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E.: Array programming with NumPy. Nature **585**, 357–362 (2020)
20. Hoare, C.A.R.: Algorithm 64: quicksort. Communications of the ACM **4**(7), 321 (1961)

21. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to automata theory, languages, and computation. Acm Sigact News **32**(1), 60–65 (2001)
22. Hörmann, W.: The generation of binomial random samples. Journal of statistical computation and simulation **46**(1-2), 101–110 (1993)
23. Hörmann, W., Derflinger, G.: The transformed rejection method for generating random variables, an alternative to the ratio of uniforms method. Communications in Statistics-Simulation and Computation **23**(3), 847–860 (1994)
24. Hörmann, W., Leydold, J., Derflinger, G., Hörmann, W., Leydold, J., Derflinger, G.: Transformed density rejection (tdr). Automatic Nonuniform Random Variate Generation pp. 55–111 (2004)
25. Jeannerod, C.P., Kornerup, P., Louvet, N., Muller, J.M.: Error bounds on complex floating-point multiplication with an fma. Mathematics of Computation **86**(304), 881–898 (2017)
26. Jeannerod, C.P., Rump, S.M.: Improved error bounds for inner products in floating-point arithmetic. SIAM Journal on Matrix Analysis and Applications **34**(2), 338–344 (2013)
27. Jeannerod, C.P., Rump, S.M.: On relative errors of floating-point operations: optimal bounds and applications. Mathematics of computation **87**(310), 803–819 (2018)
28. Kachitvichyanukul, V., Schmeiser, B.W.: Binomial random sample generation. Communications of the ACM **31**(2), 216–222 (1988)
29. Karney, C.F.: Sampling exactly from the normal distribution. ACM Transactions on Mathematical Software (TOMS) **42**(1), 1–14 (2016)
30. Karp, R.M., Luby, M.: Monte-carlo algorithms for enumeration and reliability problems. In: FOCS (1983)
31. Karp, R.M., Luby, M., Madras, N.: Monte-carlo approximation algorithms for enumeration problems. Journal of algorithms **10**(3), 429–448 (1989)
32. Kumar, G., Meel, K.S., Pote, Y.: Tolerant testing of high-dimensional samplers with subcube conditioning (2023)
33. Lanczos, C.: A precision approximation of the gamma function. Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis **1**(1), 86–96 (1964)
34. Meduna, A., Meduna, A.: Turing transducers. Automata and Languages: Theory and Applications pp. 833–887 (2000)
35. Meel, K.S., Pote, Y.P., Chakraborty, S.: On testing of samplers. NeurIPS (2020)
36. Meel, K.S., Shrotri, A.A., Vardi, M.Y.: Not all fprass are equal: demystifying fprass for dnf-counting. Constraints **24**, 211–233 (2019)
37. Meel, K.S., Vinodchandran, N., Chakraborty, S.: Estimating the size of union of sets in streaming models. In: PODS. pp. 126–137 (2021)
38. Muller, J.M., Muller, J.M.: Elementary functions. Springer (2006)
39. Pote, Y., Meel, K.S.: On scalable testing of samplers. NeurIPS (2022)
40. Pote, Y.P., Meel, K.S.: Testing probabilistic circuits. NeurIPS (2021)
41. Pugh, G.R.: An analysis of the Lanczos gamma approximation. Ph.D. thesis, University of British Columbia (2004)
42. Pugh, W.: Concurrent maintenance of skip lists. Citeseer (1990)
43. Rump, S.M., Ogita, T., Oishi, S.: Accurate floating-point summation part i: Faithful rounding. SIAM Journal on Scientific Computing **31**(1), 189–224 (2008)
44. Schmeiser, B., Kachitvichyanukul, V.: Poisson random sample generation. Research memorandum pp. 81–4 (1981)
45. Sharma, S., Roy, S., Soos, M., Meel, K.S.: Ganak: A scalable probabilistic exact model counter. In: IJCAI. vol. 19, pp. 1169–1176 (2019)

46. Soos, M., Aggarwal, D., Chakraborty, S., Meel, K.S., Obremski, M.: Engineering an efficient approximate dnf-counter. In: IJCAI. pp. 2031–2038 (2023)
47. Soos, M., Meel, K.S.: Bird: engineering an efficient cnf-xor sat solver and its applications to approximate model counting. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 1592–1599 (2019)
48. The MPFR Team: The mpfr library: Algorithms and proofs, `https://www.mpfr.org/algorithms.pdf/`
49. Thomopoulos, N.T.: Essentials of Monte Carlo simulation: Statistical methods for building simulation models. Springer Science & Business Media (2012)

Appendix

A Extended Proofs

Proof of Lemma 5

To prove Lemma 5, we begin by analyzing how $\log n!$ is approximated. In this context, we refer to $\mathsf{LogFactorial}$ (Equation (5)), which presents an approximation of $\log n!$. The following lemma provides a bound on the error introduced by using Log within one $\mathsf{LogFactorial}$ call.

Lemma 7. *The additive error introduced by using* Log *in a single* $\mathsf{LogFactorial}$ *call can be bounded by* $(k+2)\tau$*. Specifically,*

$$\log(\mathsf{Fact}^{\mathsf{Lanczs}}(k)) - (k+2)\tau \leq \mathsf{LogFactorial}(k) \leq \log(\mathsf{Fact}^{\mathsf{Lanczs}}(k)) + (k+2)\tau$$

Proof. Given k, $\mathsf{LogFactorial}(k)$ computes and returns $\frac{1}{2}\mathsf{Log}(2\pi)+\left(k+\frac{1}{2}\right)\mathsf{Log}(k+g+\frac{1}{2})-\left(k+g+\frac{1}{2}\right)+\mathsf{Log}\left(A_{t,g}(k)\right)$. Therefore, from eq. (4) we have the following inequality,

$$\begin{aligned}
&\frac{1}{2}\mathsf{Log}(2\pi) + \left(k+\frac{1}{2}\right)\mathsf{Log}\left(k+g+\frac{1}{2}\right) - \left(k+g+\frac{1}{2}\right) + \mathsf{Log}\left(A_{t,g}(k)\right) \\
&\leq \frac{1}{2}\log(2\pi) + \frac{1}{2}\tau + \left(k+\frac{1}{2}\right)\log\left(k+g+\frac{1}{2}\right) + \left(k+\frac{1}{2}\right)\tau \\
&\qquad - \left(k+g+\frac{1}{2}\right) + \log\left(A_{t,g}(k)\right) + \tau \\
&\leq \log(\mathsf{Fact}^{\mathsf{Lanczs}}(k!)) + (k+2)\tau
\end{aligned}$$

Similarly, $\frac{1}{2}\mathsf{Log}(2\pi) + \left(k+\frac{1}{2}\right)\mathsf{Log}(k+g+\frac{1}{2}) - \left(k+g+\frac{1}{2}\right) + \mathsf{Log}\left(A_{t,g}(k)\right) \geq \log(\mathsf{Fact}^{\mathsf{Lanczs}}(k!)) - (k+2)\tau$. □

Lemma 5. *Let* r_k^{Inter} *denotes the ratio* $r_k^{\mathsf{Inter}} = \frac{\mathsf{b}_{n,p}^{\mathsf{Inter}}(k)}{\alpha\mathsf{h}(k)}$*, and let* $\widetilde{r}_k$ *denote the computed rejection ratio, defined as* $\widetilde{r}_k = \frac{\mathsf{b}_{n,p}^{\mathsf{BinSamp}}(k)}{\alpha\mathsf{h}(k)}$*, for any* k *in* $[n]$*. Then, for all* $k \in [n]$*,* $\left|\frac{\widetilde{r}_k}{r_k^{\mathsf{Inter}}} - 1\right| \leq (1110n + 2540)\beta\varepsilon + 14\varepsilon\log(\mathsf{h}(k)) + o(\varepsilon)$

Proof. $\mathsf{BinSamp}$ uses the hat distribution h. We use $\widetilde{lr}_k$ to denote the log of the computed rejection ratio $\widetilde{r}_k$, which is given by,

$$\widetilde{lr}_k := l_n \ominus l_k \ominus l_{nk} \oplus k \otimes \mathsf{Log}(p) \oplus (n-k) \otimes \mathsf{Log}(1-p) \oplus \mathsf{Log}(\mathsf{h}^{-1}(u))$$

We assume that the computation of $\mathsf{h}^{-1}(u)$ has constant number $c > 0$ of arithmetic operations. Therefore the multiplicative error introduced in the computation of $\mathsf{h}^{-1}(u)$ is bounded by $c\varepsilon$. Therefore, the computed value of $\mathsf{Log}\left(\mathsf{h}^{-1}(u)\right)$ is bounded by $\mathsf{Log}\left((1 \pm c\varepsilon).\mathsf{h}^{-1}(u)\right)$. Consequently, using eq. (4),

$$\mathsf{Log}\left((1+c\varepsilon).\mathsf{h}^{-1}(u)\right) \leq \log\left(\mathsf{h}^{-1}(u)\right) + \log(1+c\varepsilon) + \tau \tag{6}$$

Similarly, considering the error in the term $k \otimes \mathsf{Log}(p)$, we deduce using eq. (2), eq. (4) that,

$$k \otimes \mathsf{Log}(p) \leq k \log(p) + \varepsilon |k \log(p)| + k\tau + k\tau\varepsilon \tag{7}$$

Similarly, we can bound the term $(n-k) \otimes \mathsf{Log}(1-p)$ as follows,

$$(n-k) \otimes \mathsf{Log}(1-p) \leq (n-k)\log(1-p) + \varepsilon |(n-k)\log(1-p)| + (n-k)\tau + (n-k)\tau\varepsilon \tag{8}$$

Lastly, we are interested in the error in the computation of $l_n - l_k - l_{nk}$. Using lemma 7, we can bound the error as follows:

$$\begin{aligned} l_n - l_k - l_{nk} \leq \log \mathsf{Fact}^{\mathsf{Lancz}}(n) - \log \mathsf{Fact}^{\mathsf{Lancz}}(k) - \log \mathsf{Fact}^{\mathsf{Lancz}}(n-k) \\ + (n+2)\tau + (k+2)\tau + (n-k+2)\tau \end{aligned} \tag{9}$$

Thus, we bound the following sum using eq. (7), eq. (8) and eq. (9):

$$\begin{aligned} & l_n - l_k - l_{nk} + k \otimes \mathsf{Log}(p) + (n-k) \otimes \mathsf{Log}(1-p) \\ \leq & \log \mathsf{Fact}^{\mathsf{Lancz}}(n) - \log \mathsf{Fact}^{\mathsf{Lancz}}(k) - \log \mathsf{Fact}^{\mathsf{Lancz}}(n-k) + k \otimes \mathsf{Log}(p) \\ & + (n-k) \otimes \mathsf{Log}(1-p) + (n+2)\tau + (k+2)\tau + (n-k+2)\tau \\ = & \log \mathsf{Fact}^{\mathsf{Lancz}}(n) - \log \mathsf{Fact}^{\mathsf{Lancz}}(k) - \log \mathsf{Fact}^{\mathsf{Lancz}}(n-k) + k \log p + \varepsilon |k \log p| \\ & + (n-k)\log(1-p) + \varepsilon |(n-k)\log(1-p)| + (3n+6)\tau + n\tau\varepsilon \end{aligned} \tag{10}$$

In a similar way $l_n + l_k + l_{nk} + k \otimes \mathsf{Log}(p) + (n-k) \otimes \mathsf{Log}(1-p)$ is at most,

$$\begin{aligned} & \log \mathsf{Fact}^{\mathsf{Lancz}}(n) + \log \mathsf{Fact}^{\mathsf{Lancz}}(k) + \log \mathsf{Fact}^{\mathsf{Lancz}}(n-k) + k \otimes \mathsf{Log}(p) \\ & + (n-k) \otimes \mathsf{Log}(1-p) + (n+2)\tau + (k+2)\tau + (n-k+2)\tau \\ \leq & n \log n + k \log k + (n-k)\log(n-k) + k \log p + \varepsilon |k \log p| \\ & + +(n-k)\log(1-p) + \varepsilon |(n-k)\log(1-p)| + (3n+6)\tau + n\tau\varepsilon \end{aligned} \tag{11}$$

The last inequality follows from the fact that $\log \mathsf{Fact}^{\mathsf{Lancz}}(n) \leq n \log n$. Next we accumulate all the errors due to basic arithmetic computations in the computation of $\widetilde{lr}_k$. We first bound the compound sum using eq. (3):

$$\begin{aligned} & l_n \ominus l_k \ominus l_{nk} \oplus k \otimes \mathsf{Log}(p) \oplus (n-k) \otimes \mathsf{Log}(1-p) \oplus \mathsf{Log}\left(\mathsf{h}^{-1}(u)\right) \\ \leq & l_n - l_k - l_{nk} + k \otimes \mathsf{Log}(p) + (n-k) \otimes \mathsf{Log}(1-p) + \mathsf{Log}\left(\mathsf{h}^{-1}(u)\right) \\ & + 6\varepsilon \left(l_n + l_k + l_{nk} + k \otimes \mathsf{Log}(p) + (n-k) \otimes \mathsf{Log}(1-p) + \mathsf{Log}\left(\mathsf{h}^{-1}(u)\right)\right) \end{aligned}$$

Using the corresponding bounds from eqs. (10) and (11) we can have,

$$\begin{aligned} & l_n \ominus l_k \ominus l_{nk} \oplus k \otimes \mathsf{Log}(p) \oplus (n-k) \otimes \mathsf{Log}(1-p) \oplus \mathsf{Log}\left(\mathsf{h}^{-1}(u)\right) \\ \leq & \log \mathsf{Fact}^{\mathsf{Lancz}}(n) - \log \mathsf{Fact}^{\mathsf{Lancz}}(k) - \log \mathsf{Fact}^{\mathsf{Lancz}}(n-k) \\ & + k \log(p) + (n-k)\log(1-p) + \log\left(\mathsf{h}^{-1}(u)\right) \\ & + 7\varepsilon \left[n \log n + k \log k + (n-k)\log(n-k) + k|\log(p)| \right. \\ & \left. + (n-k)|\log(1-p)| + \log\left(\mathsf{h}^{-1}(u)\right)\right] + (3n+7)\tau + \log(1+c\varepsilon) + \mathcal{O}(n\tau\varepsilon) \end{aligned}$$

Thus accumulating all the errors, and denoting $\log(r_k^{\mathsf{Inter}})$ by lr_k, we can bound the error in the computation of $\widetilde{lr}_k$ using the following inequalities:

$$\begin{aligned}\widetilde{lr}_k &\leq lr_k + (3n+7)\tau + 7\varepsilon\,[n\log n + k\log k + (n-k)\log(n-k)\\ &\quad + k|\log(p)| + (n-k)|\log(1-p)| + \log\left(\mathsf{h}^{-1}(u)\right)\big] + \log(1+c\varepsilon) + \mathcal{O}(n\tau\varepsilon)\\ &\leq lr_k + (3n+7)\tau + 21n\beta\varepsilon + 7\varepsilon\log\left(\mathsf{h}^{-1}(u)\right) + \log(1+c\varepsilon) + \mathcal{O}(n\tau\varepsilon)\end{aligned}$$

The second inequality follows from $\log n, \log k, \log(n-k), |\log p|, |\log(1-p)| < \beta$ (due to our choice of β). Using a similar set of inequalities, we can bound the error in the computation of $\widetilde{lr}_k$ from below.

$$\widetilde{lr}_k \geq lr_k - (3n+7)\tau - 21n\beta\varepsilon - 7\varepsilon\log\left(\mathsf{h}^{-1}(u)\right) - \log(1-c\varepsilon) - \mathcal{O}(n\tau\varepsilon)$$

Therefore, considering $\widetilde{r}_k = e^{\widetilde{lr}_k}$, and using inequalities $e^x \leq 1+2x, e^{-x} \geq 1-2x$, we get the following bound on the obtained rejection ratio:

$$\begin{aligned}\widetilde{r}_k &\leq r_k \cdot (1+c\varepsilon) \cdot e^{(3n+7)\tau + 21n\beta\varepsilon + 7\varepsilon\log\left(\mathsf{h}^{-1}(u)\right) + \mathcal{O}(n\tau\varepsilon)}\\ &\leq r_k(1 + (6n+14)\tau + 42n\beta\varepsilon + 14\varepsilon\log\left(\mathsf{h}^{-1}(u)\right) + c\varepsilon + \mathcal{O}(n\tau\varepsilon))\\ &\leq r_k(1 + 178(6n+14)\beta\varepsilon + 42n\beta\varepsilon + 14\varepsilon\log\left(\mathsf{h}^{-1}(u)\right) + c\varepsilon + \mathcal{O}(n\varepsilon^2))\\ &= r_k(1 + (1110n+2540)\beta\varepsilon + 14\varepsilon\log\left(\mathsf{h}^{-1}(u)\right) + \mathcal{O}(n\varepsilon^2))\end{aligned}$$

The last inequality follows from substituing the value of τ. Similarly, we have, $\widetilde{r}_k \geq r_k(1 - (1110n+2540)\beta\varepsilon - 14\varepsilon\log\left(\mathsf{h}^{-1}(u)\right) - \mathcal{O}(n\varepsilon^2))$. □

Proof of Lemma 6

Lemma 6. *For all $k \in [n]$, $(1-15\zeta)\mathsf{b}_{n,p}(k) \leq \mathsf{b}^{\mathsf{Inter}}_{n,p}(k) \leq (1+15\zeta)\mathsf{b}_{n,p}(k)$, where ζ denotes the uniform error bound due to the Lanczos approximation and $\mathsf{b}^{\mathsf{Inter}}_{n,p}$ denotes the intermediate distribution.*

Proof. We begin by denoting the approximated probability mass, as calculated by $\mathsf{BinSamp}$, with $\overline{\mathsf{b}}_{n,p}(k)$ for all $k \in [n]$, such that,

$$\overline{\mathsf{b}}_{n,p}(k) = \frac{\mathsf{Fact}^{\mathsf{Lanczs}}(n)}{\mathsf{Fact}^{\mathsf{Lanczs}}(k) \cdot \mathsf{Fact}^{\mathsf{Lanczs}}(n-k)} \cdot p^k q^{n-k}$$

But, here $\overline{\mathsf{b}}_{n,p}$ is not necessarily a well defined distribution, since $\sum_{k=0}^{n} \overline{\mathsf{b}}_{n,p}k$ is not necessarily 1. We normalize $\overline{\mathsf{b}}_{n,p}$ to a distribution and obtain its upper and lower bounds. From eq. (1) we have $(1-\zeta)k! \leq \mathsf{Fact}^{\mathsf{Lanczs}}(k) \leq (1+\zeta)k!$ for all $k \in [n]$. Therefore,

$$\sum_{k=0}^{n} \mathsf{b}_{n,p}(k) \cdot \frac{1-\zeta}{(1+\zeta)^2} \leq \sum_{k=0}^{n} \overline{\mathsf{b}}_{n,p}(k) \leq \sum_{k=0}^{n} \mathsf{b}_{n,p}(k) \cdot \frac{1+\zeta}{(1-\zeta)^2}$$

Therefore for all $k \in [n]$ we have the following sets of inequalities:

$$\begin{aligned}
&\frac{(1-\zeta)}{(1+\zeta)^2}\mathsf{b}_{n,p}(k) \cdot \frac{(1-\zeta)^2}{(1+\zeta)} \leq \frac{\overline{\mathsf{b}}_{n,p}(k)}{\sum_{i=0}^{n}\overline{\mathsf{b}}_{n,p}(i)} &&\leq \frac{(1+\zeta)}{(1-\zeta)^2}\mathsf{b}_{n,p}(k) \cdot \frac{(1+\zeta)^2}{(1-\zeta)} \\
\implies \quad &\frac{(1-\zeta)^3}{(1+\zeta)^3}\mathsf{b}_{n,p}(k) \leq \frac{\overline{\mathsf{b}}_{n,p}(k)}{\sum_{i=0}^{n}\overline{\mathsf{b}}_{n,p}(i)} &&\leq \frac{(1+\zeta)^3}{(1-\zeta)^3}\mathsf{b}_{n,p}(k) \\
\implies \quad &(1-3\zeta)^3\mathsf{b}_{n,p}(k) \leq \frac{\overline{\mathsf{b}}_{n,p}(k)}{\sum_{i=0}^{n}\overline{\mathsf{b}}_{n,p}(i)} &&\leq (1+3\zeta)^3\mathsf{b}_{n,p}(k) \\
\implies \quad &(1-15\zeta)\mathsf{b}_{n,p}(k) \leq \frac{\overline{\mathsf{b}}_{n,p}(k)}{\sum_{i=0}^{n}\overline{\mathsf{b}}_{n,p}(i)} &&\leq (1+15\zeta)\mathsf{b}_{n,p}(k)
\end{aligned}$$

The third and the last inequalities follow due to the fact that for $\zeta < 1/3$, $\frac{1-\zeta}{1+\zeta} > 1-3\zeta$, $(1-3\zeta)^3 > 1-15\zeta$ and $(1+3\zeta)^3 < 1+15\zeta$.

Since, $\mathsf{b}^{\mathsf{Inter}}_{n,p}(k) = \frac{\overline{\mathsf{b}}_{n,p}(k)}{\sum_{i=0}^{n}\overline{\mathsf{b}}_{n,p}(i)}$, the result follows directly. □

Proof of Lemma 4 using Lemma 5 and Lemma 6

Lemma 4. *Let r_k denote the actual rejection ratio, defined as $r_k = \frac{\mathsf{b}_{n,p}(k)}{\alpha\mathsf{h}(k)}$, and let $\widetilde{r}_k$ denote the computed rejection ratio, defined as $\widetilde{r}_k = \frac{\mathsf{b}^{\mathsf{BinSamp}}_{n,p}(k)}{\alpha\mathsf{h}(k)}$, for any k in $[n]$. Then, for any $k \in [0,n]$, we have:*

$$\left|\frac{\widetilde{r}_k}{r_k} - 1\right| \leq (1110n + 2540)\beta\varepsilon + 14\varepsilon\log\left(\mathsf{h}(k)\right) + 15\zeta + o(\varepsilon)$$

Proof (of Lemma 4). Now we complete the proof of Lemma 4. The rejection ratio r_k is defined as $\frac{\mathsf{b}_{n,p}(k)}{\alpha\mathsf{h}(k)}$. Using Lemma 5 and Lemma 6, we can derive the following bound on the rejection ratio: $\widetilde{r}_k = \frac{\mathsf{b}^{\mathsf{Inter}}_{n,p}(k)}{\alpha\mathsf{h}(k)} \cdot \frac{\mathsf{b}_{n,p}(k)}{\mathsf{b}^{\mathsf{Inter}}_{n,p}(k)} \leq (1 + (1110n + 2540)\beta\varepsilon + 14\varepsilon\log\left(\mathsf{h}(k)\right) + 15\zeta + o(\varepsilon))$. Similarly we can lower bound $\widetilde{r}_k$ by $(1 - (1110n + 2540)\beta\varepsilon - 14\varepsilon\log\left(\mathsf{h}(k)\right) - 15\zeta - o(\varepsilon))$. □

Proof of Lemma 3

Lemma 3. *Suppose $\widetilde{\mathsf{h}}$ is the deviated version of the hat distribution h due to the error in the computation of $\mathcal{H}^{-1}(u)$. Then, if the error is distributed uniformly over the range $[-\varepsilon_{\mathcal{H}}, \varepsilon_{\mathcal{H}}]$ and $\varepsilon_{\mathcal{H}} \leq \frac{1}{2}$, then for any $k \in [n]$, $(1-\varepsilon_{\mathcal{H}}(3k+1))\mathsf{h}(k) \leq \widetilde{\mathsf{h}}(k) \leq (1 + w_k\varepsilon_{\mathcal{H}}(k+2))\mathsf{h}(k)$ where, $w_k = \max\left(\frac{\mathsf{h}(k-1)}{\mathsf{h}(k)}, \frac{\mathsf{h}(k+1)}{\mathsf{h}(k)}\right)$*

Proof. Without loss of generality, let us assume that $\mathsf{h}(-1) = \mathsf{h}(n+1) = 0$. Let us also assume that the ideal output of $\lfloor\mathcal{H}^{-1}(u)\rfloor$ is k, and let k' be the value of $\lfloor\mathcal{H}^{-1}(u)\rfloor$ as computed in Algorithm 1. Note that $k' \in (1 \pm \varepsilon_{\mathcal{H}})\mathcal{H}^{-1}(u)$.

Now, because, $\beta \geq 2\lceil \log_2 n \rceil$, we have $\varepsilon_{\mathcal{H}} \mathcal{H}^{-1}(u) \leq c\varepsilon n \leq 1$, and hence, $k' \in \{k-1, k, k+1\}$. Since for all $t \in [n]$, $\Pr_u\left(\lfloor \mathcal{H}^{-1}(u) \rfloor = t\right) = \mathsf{h}(t)$,

$$\begin{aligned}\widetilde{\mathsf{h}}(k) &= \sum_{t \in \{k-1,k,k+1\}} \Pr_{\varepsilon,u}\left(k' = k \mid \lfloor \mathcal{H}^{-1}(u) \rfloor = t\right) \cdot \Pr_u\left(\lfloor \mathcal{H}^{-1}(u) \rfloor = t\right) \\ &= \sum_{t \in \{k-1,k,k+1\}} \Pr_{\varepsilon,u}\left(k' = k \mid \lfloor \mathcal{H}^{-1}(u) \rfloor = t\right) \cdot \mathsf{h}(t)\end{aligned}$$

Observe that, given $\lfloor \mathcal{H}^{-1}(u) \rfloor = t$, $k' = t+1$ is possible only when $(1+\varepsilon_{\mathcal{H}})\mathcal{H}^{-1}(u) \geq t+1$, that is, $\mathcal{H}^{-1}(u) \geq \frac{t+1}{(1+\varepsilon_{\mathcal{H}})}$. Therefore, given $\lfloor \mathcal{H}^{-1}(u) \rfloor = t$, and assuming $\mathcal{H}^{-1}(u)$ is uniformly distributed over the range $[t, t+1)$, we have,

$$\begin{aligned}&\Pr_{\varepsilon,u}\left(k' = t+1 \mid \lfloor \mathcal{H}^{-1}(u) \rfloor = t\right) \\ &= \int_{v=t}^{v=t+1} \Pr_{\varepsilon}\left(k' = t+1 \mid \lfloor \mathcal{H}^{-1}(u) \rfloor = t, \mathcal{H}^{-1}(u) = v\right) \cdot \\ &\qquad\qquad f\left(\mathcal{H}^{-1}(u) = v \mid \lfloor \mathcal{H}^{-1}(u) \rfloor = t\right)\, dv \\ &= \int_{v=\frac{t+1}{1+\varepsilon_{\mathcal{H}}}}^{v=t+1} \Pr_{\varepsilon}\left(k' = t+1 \mid \lfloor \mathcal{H}^{-1}(u) \rfloor = t, \mathcal{H}^{-1}(u) = v\right)\, dv\end{aligned}$$

where f is the uniform probability density function. The last equality follows from the fact that, we have $\Pr_{\varepsilon}\left(k' = t+1 \mid \lfloor \mathcal{H}^{-1}(u) \rfloor = t, \mathcal{H}^{-1}(u) = v\right) = 0$ for $v \in \left[\frac{t}{1-\varepsilon_{\mathcal{H}}}, \frac{t+1}{1+\varepsilon_{\mathcal{H}}}\right]$.

Since the error in the computation of $\mathcal{H}^{-1}(u)$ is uniformly distributed over the range $[-\varepsilon_{\mathcal{H}}, \varepsilon_{\mathcal{H}}]$, therefore, $\Pr_{\varepsilon}\left(k' = t+1 \mid \lfloor \mathcal{H}^{-1}(u) \rfloor = t, \mathcal{H}^{-1}(u) = v\right) \leq 1/2$ for all $v \in \left[\frac{t+1}{1+\varepsilon_{\mathcal{H}}}, t+1\right]$. Thus, we have $\Pr_{\varepsilon,u}\left(k' = t+1 \mid \lfloor \mathcal{H}^{-1}(u) \rfloor = t\right) \leq \int_{v=\frac{t+1}{1+\varepsilon_{\mathcal{H}}}}^{v=t+1} \frac{1}{2}\, dv = \frac{\varepsilon_{\mathcal{H}}(t+1)}{2(1+\varepsilon_{\mathcal{H}})} \leq \frac{\varepsilon_{\mathcal{H}}(t+1)}{3}$. Similarly, we can upper bound the probability of k' being $t-1$ given $\lfloor \mathcal{H}^{-1}(u) \rfloor = t$, by $\Pr_{\varepsilon,u}\left(k' = t-1 \mid \lfloor \mathcal{H}^{-1}(u) \rfloor = t\right) \leq \frac{\varepsilon_{\mathcal{H}}(t+1)}{3}$. Therefore, combining, we can upper bound the probability mass:

$$\widetilde{\mathsf{h}}(k) \leq \frac{\varepsilon_{\mathcal{H}} k}{3}\mathsf{h}(k-1) + \mathsf{h}(k) + \frac{\varepsilon_{\mathcal{H}}(k+2)}{3}\mathsf{h}(k+1) \leq \left(1 + w_k \varepsilon_{\mathcal{H}}(k+2)\right)\mathsf{h}(k)$$

Next, observing that $\mathcal{H}^{-1}(u) \in \left[\frac{k}{1-\varepsilon_{\mathcal{H}}}, \frac{k+1}{1+\varepsilon_{\mathcal{H}}}\right]$ ensures no possibility of error, we derive a lower bound for the probability of $k' = k$ given $\lfloor \mathcal{H}^{-1}(u) \rfloor = k$ as

$$\begin{aligned}\Pr_{\varepsilon,u}\left(k' = k \mid \lfloor \mathcal{H}^{-1}(u) \rfloor = k\right) &= \int_{v=k}^{v=k+1} \Pr_{\varepsilon,u}\left(k' = k \mid \lfloor \mathcal{H}^{-1}(u) \rfloor = k, \mathcal{H}^{-1}(u) = v\right) \cdot \\ &\qquad\qquad f\left(\mathcal{H}^{-1}(u) = v \mid \lfloor \mathcal{H}^{-1}(u) \rfloor = k\right)\, dv \\ &\geq \int_{v=\frac{k}{1-\varepsilon_{\mathcal{H}}}}^{v=\frac{k+1}{1+\varepsilon_{\mathcal{H}}}} \Pr_{\varepsilon,u}\left(k' = k \mid \lfloor \mathcal{H}^{-1}(u) \rfloor = k, \mathcal{H}^{-1}(u) = v\right)\, dv\end{aligned}$$

Since for all $v \in \left[\frac{k}{1-\varepsilon_{\mathcal{H}}}, \frac{k+1}{1+\varepsilon_{\mathcal{H}}}\right]$, $\Pr_{\varepsilon,u}\left[k' = k \mid \lfloor \mathcal{H}^{-1}(u) \rfloor = k, \mathcal{H}^{-1}(u) = v\right] = 1$,

$$\Pr_{\varepsilon,u}\left(k' = k \mid \lfloor \mathcal{H}^{-1}(u) \rfloor = k\right) \geq \frac{k+1}{(1+\varepsilon_{\mathcal{H}})} - \frac{k}{(1-\varepsilon_{\mathcal{H}})} \geq (1 - \varepsilon_{\mathcal{H}}(3k+1))$$

Where the last inequality follows from the fact that $\frac{1}{1+\varepsilon_{\mathcal{H}}} \geq 1 - \varepsilon_{\mathcal{H}}$ and $\frac{1}{1-\varepsilon_{\mathcal{H}}} \leq 1+2\varepsilon_{\mathcal{H}}$ for $\varepsilon_{\mathcal{H}} \leq \frac{1}{2}$. Consequently, $\widetilde{\mathsf{h}}(k)$ is at least $\Pr_{\varepsilon,u}\left(k' = k \mid \lfloor \mathcal{H}^{-1}(u) \rfloor = k\right) \cdot \mathsf{h}(k) = (1 - \varepsilon_{\mathcal{H}}(3k+1))\mathsf{h}(k)$. This completes the proof. □