Quantifying the Inexactness of Poisson Distributions in Standard Libraries

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

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

Abstract. The Poisson distribution plays a central role in modeling event-driven processes across domains such as network traffic analysis, reliability engineering, and queueing systems. Accordingly, sampling from the Poisson distribution is a core subroutine in many randomized algorithms, with implementations expecting standard numerical libraries such as NumPy to faithfully implement Poisson distributions.
However, these standard libraries often rely on heuristic approximations that trade exactness for performance. Despite their widespread use, there has been limited rigorous analysis of how these approximations may affect the quality of generated samples.
The primary contribution of this work is a rigorous theoretical analysis of the statistical distance between the ideal Poisson distribution and the Poisson distribution implemented via Inverse Transformed Rejection Sampling in standard numerical libraries such as NumPy. Our analysis reveals significant parameter-dependent inexactness that can substantially impact downstream outcomes. We establish an upper bound on statistical distance, in particular, our bound is of the form bound of $\mathcal{O}(\zeta + \lambda\varepsilon)$, where ζ represents intrinsic error from factorial approximations and ε captures the effect of finite-precision arithmetic, with λ being the Poisson distribution parameter. We demonstrate through a case study as how these bounds can be combined with theoretical analysis (modulo assumption to access to ideal Poisson distributions) of the implemented algorithms; thereby, providing confidence to the end user for correctness of their implemented randomized algorithms.

Keywords: Poisson Distributions, Indistinguishability

1 Introduction

Randomized algorithms form the backbone of modern computing, underpinning everything from fundamental data structures like hash tables and skip lists to sophisticated applications in cryptography [31,5], Monte Carlo simulations [39,3,26], and machine learning. The theoretical guarantees of these algorithms—whether correctness proofs for randomized quicksort [13], security proofs for cryptographic protocols [11], or convergence bounds for Monte Carlo methods—fundamentally depend on a crucial assumption: access to high-quality

samples from mathematically precise probability distributions. In practice, these algorithms rely heavily on standard library sampling APIs such as NumPy's `random` module, GSL's random number generators, and similar interfaces across programming languages and scientific computing platforms.

Given the critical importance of samplers in ensuring the reliability of randomized algorithms, one would naturally expect these standard library subroutines to faithfully sample from the intended ideal distributions. Surprisingly, Sarkar, Chakraborty, and Meel [34] showed that such is not the case even for fundamental samplers like binomial samplers. Their analysis demonstrated that widely-used library implementations introduce measurable deviations from the ideal binomial distribution due to finite-precision arithmetic and approximation techniques employed for computational efficiency. To quantify these deviations, Sarkar et al. introduced the framework of statistical distance analysis for samplers. The statistical distance $\Delta(p, q) = \frac{1}{2} \sum_{k \in \Omega} |p(k) - q(k)|$ between distributions p and q offers a compelling interpretation: if two distributions are within δ in statistical distance, then any downstream algorithm using one in place of the other will behave identically with probability at least $1 - \delta$. Furthermore, Sarkar et al. [34] provided the first known formal bounds on the distance between implemented binomial sampler in NumPy and ideal binomial distribution. Recognizing the broader implications of their findings, Sarkar et al. identified the analysis of other fundamental distributions in standard libraries as a major direction for future research.

Our work addresses this call by analyzing the statistical distance between ideal *Poisson* distribution and the *Poisson* distributions implemented in standard Numerical libraries—another cornerstone distribution in randomized algorithms with widespread applications in queueing systems [32], reliability engineering [19], network traffic modeling, and count-based algorithms [37]. However, extending the analytical framework from binomial to Poisson distributions introduces unique technical challenges. While binomial distributions have finite support bounded by the parameter n, Poisson distributions possess unbounded support over all non-negative integers. This fundamental difference necessitates novel techniques for bounding the statistical distance, as the finite-support arguments employed for binomial analysis no longer apply. Our approach overcomes this challenge by leveraging the mathematical properties of the inverse transform function used in Poisson samplers, enabling us to establish rigorous bounds despite the unbounded domain.

Our main theoretical contribution demonstrates that the statistical distance between ideal Poisson distribution and the Poisson distributions implemented in the standard libraries is bounded by $\mathcal{O}(\zeta + \lambda\varepsilon)$, where ζ represents the intrinsic error from factorial approximations (specifically, the Lanczos method) and $\varepsilon = 2^{-\beta}$ captures the effect of finite-precision arithmetic with precision parameter β. This bound reveals several key insights: first, the distance breaks down into two independent sources—factorial approximation and floating-point precision—allowing targeted improvement of each part. Second, the linear dependence on the Poisson parameter λ shows that higher-rate processes need more

precision to maintain the same accuracy. Third, the exponential decrease in statistical distance with increasing precision β provides a clear trade-off between computational cost and sampling accuracy. These findings help practitioners make informed decisions about precision requirements based on their specific λ values and error tolerance, while also showing fundamental limitations of current library implementations for high-rate Poisson processes. We demonstrate through a case study from DNF counting how our bounds can be combined with theoretical analysis (under the assumption of access to ideal Poisson distributions) of algorithms to achieve rigorous guarantees on the implemented systems.

Organisation of the Paper. In Section 2, we present the necessary preliminaries and an overview of related concepts that lay the foundation for the techniques discussed in the rest of the paper. Section 3 surveys related work on samplers, and the evaluation of sampler quality. In Section 4, we outline our results on the analysis of Poisson samplers. Section 5 details the technical concepts behind our main result. Section 6 presents a case study demonstrating the practical implications of our theoretical findings. Finally, in Section 6, we discuss the implications and limitations of our work as well as future directions.

2 Preliminaries

In this work we focus on discrete distributions, specifically Poisson distribution and its samplers. For $\lambda > 0$, the Poisson distribution with parameter λ is defined as: $\mathsf{p}_\lambda(k) = \frac{\lambda^k e^{-\lambda}}{k!}$, for $k \in \mathbb{Z}^+$. The r-th moment of a discrete distribution p over $\mathbb{Z}^+$ is defined as $\mathbb{E}_p(X^r) = \sum_{k\geq 0} k^r p(k)$. We use the shorthand notation μ_r^p to denote the r-th moment of the distribution p.

A *probabilistic transducer* extends the classical Turing machine model by incorporating both an output tape and access to a source of randomness. On input $x \in \{0,1\}^*$, the machine reads an infinite sequence of random bits $u \in \{0,1\}^\infty$ from a read-only random tape and writes a corresponding output $M(x;u)$ to the output tape. For fixed x and u, the computation is deterministic, allowing the transducer to define a distribution over outputs induced by the randomness. This abstraction naturally captures the behavior of randomized algorithms: given access to a randomness source, such algorithms define a distribution over possible outputs. A sampler for a distribution is defined to be a randomized algorithm that outputs samples from the distribution. For instance, a sampler for a distribution p (denoted by Samp_p) is modeled as a transducer that, given a random bit string u, produces a sample $x \in \Omega$ such that $\Pr_u[\mathsf{Samp}_p(u) = x] = p(x)$. Let PoiSamp denote a numerical sampler for Poisson distributions that takes input $\lambda > 0$ and outputs a sample from p_λ. We denote the distribution induced by this sampler as PoiSamp(λ). Our goal is to upper bound the statistical distance between p_λ and PoiSamp(λ).

Definition 1 (Statistical Distance). *Let p and q be two discrete distributions over a countable set Ω. The* statistical distance *(denoted by Δ) between p and q*

is defined as:

$$\Delta(p,q) = \frac{1}{2}\sum_{x\in\Omega}|p(x)-q(x)| = \sup_{A\subseteq\Omega} p(A) - q(A).$$

The statistical distance is 0 if and only if $p = q$, and is maximum, that is 1, if the supports of p and q are disjoint. The next lemma highlights the relevance of statistical distance when a sampler is used within downstream applications.

Lemma 1 ([34]). *Let $\mathcal{A}$ be a randomized algorithm that outputs 1 with probability at least $1-\delta$ when its randomness is drawn from the distribution p. If p is replaced by q, then $\Pr_q[\mathcal{A}$ outputs $0] \leq \delta + \Delta(p,q)$.*

Multiple-precision Arithmetic and Log Approximations

Following [38], we define the set of all definable numbers within the bit precision β by $\mathbb{F} = \{w \cdot 2^e \ : \ \frac{1}{2} \leq |w| < 1,\ e \in \mathbb{Z}\}$, such that w is defined within β bits. Here e is the exponent, and the $\mathsf{ulp}(x) = 2^{e-\beta}$, where ulp denotes the unit in the last place. Let $\mathtt{rnd} : \mathbb{R} \to \mathbb{F}$ be the rounding function that rounds a real number to the nearest representable number in $\mathbb{F}$ within precision β. The relative error of the rounding function can be bounded by $\frac{|\mathtt{rnd}(x)-x|}{x} \leq 2^{-\beta}$, for $x \neq 0$ [38,27,17]. Throughout this work, we use ε to denote the relative error, commonly referred to as the *unit round-off error*. For basic operations $* \in \{+,-,\times,\div,\sqrt{}\}$, the computed value satisfies:

$$|\mathtt{rnd}(x) \circledast \mathtt{rnd}(y) - (x * y)| \leq \varepsilon \cdot |x * y| \tag{1}$$

where $\circledast$ denotes the corresponding operation to $*$ in $\mathbb{F}$. For n real numbers $x_1, \ldots, x_n$, the total summation error satisfies:

$$\left| \bigoplus_{i=1}^{n} \mathtt{rnd}(x_i) - \sum_{i=1}^{n} x_i \right| \leq n\varepsilon \sum_{i=1}^{n} |x_i| \tag{2}$$

Logarithm Computation Logarithm evaluation in high-precision libraries is typically done via the arithmetic-geometric mean (AGM) method [8,38]. AGM method assumes 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, denoted by $AGM(w_0, z_0)$.

Let for $x \in \mathbb{F}$ represented as $x = w \cdot 2^e$, we define exponent$(x) = e$. To compute $\log(x)$ via the AGM method, the log computation algorithm selects an integer m such that $s = x2^m > 2^{\beta/2}$. In particular, the MPFR library [38] sets

$$m = \left\lceil \frac{\beta+3}{2} \right\rceil - \text{exponent}(x),$$

which ensures that $s \in [2^{\beta/2}, 2^{\beta}]$. The logarithm is then computed as:

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

The following lemma provides an error bound for the AGM method.

Lemma 2 (Prop. 2 of [8]). *The following holds for any $s \geq 4$:*

$$\left| \frac{\pi}{2AGM(1, 4/s)} - \log(s) \right| \leq \frac{64}{s^2}(10 + |\log s|).$$

Since, for $\beta > 8$, we have $3\log(s) > 10$ and $2^{\beta/2} \leq s \leq 2^{\beta}$, we make the following conclusion from the above lemma:

$$|\mathsf{Log}(x) - \log(x)| \leq \frac{178\beta}{2^{\beta}} \tag{3}$$

We will use the notation $\tau = \frac{178\beta}{2^{\beta}}$ to denote the additive error bound for the logarithm approximation in the rest of the paper. We emphasize that a similar error bound can be derived for the Taylor series method as well [6,7].

Factorial Approximations

Poisson samplers also require computing $k!$, which is approximated using several methods. We focus on the Lanczos approximation, which is widely used in numerical libraries. We denote this approximation by $\mathsf{Fact}^{\mathsf{Lancz}}(k)$:

$$\mathsf{Fact}^{\mathsf{Lancz}}(k) = \sqrt{2\pi}\,(k + g + \tfrac{1}{2})^{k+\frac{1}{2}} e^{-(k+g+\frac{1}{2})} A_{t,g}(k)$$

where $A_{t,g}(k)$ is a degree-t polynomial and g (with $g \geq -1/2$) is a constant affecting convergence. The approximation has a uniform relative error bounded by some $\zeta > 0$ as follows [30],

$$\left| \mathsf{Fact}^{\mathsf{Lancz}}(k) - k! \right| \leq \zeta k!, \tag{4}$$

It is standard practice to evaluate the logarithm of the factorial rather than the factorial itself, due to improved numerical stability and efficiency. We define the corresponding function $\mathsf{LogFactorial}$ that approximates $\log(k!)$ using the Lanczos method with fixed parameters t and g, as follows:

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

3 Related Work

The study of sampling exactly from probability distributions has a rich history, with significant contributions across various domains. Since this work focuses on library implemented samplers, a natural question is whether there exists a sampling mechanism in the existing libraries that can sample exactly from the target distribution. The answer is unfortunately negative. One of the earliest

works by Von Neumann laid the foundation for sampling exactly from distributions, which has been further explored for various distributions, including the normal and exponential distributions [21]. However, such exact sampling techniques remain limited in scope and are typically feasible only for a narrow class of distributions.

Given the inherent difficulty and inefficiency of exact sampling, much of the research has focused on designing efficient algorithms that generate samples from distributions that are close to the target and easier to sample from. Therefore, all the library implementations of samplers, including NumPy and GSL use the transformed rejection sampling framework [12,14,20,35,15]. These algorithms achieve a constant sampling time complexity, but at the cost of approximations, as they need to evaluate the probability mass function, an operation that is computationally expensive unless approximated.

A natural question that arises is whether the inexactness introduced by these approximations can be quantified in terms of some distance metric. The recent work by Sarkar et al. [34] proposed a statistical distance framework for this purpose, due to the importance of statistical distance. By leveraging the fact that the internal workings of the sampling algorithm are observable, they enabled a deeper analysis that allows for tighter, implementation-aware bounds on statistical distance. They provided a detailed case analysis of the Binomial samplers implemented in libraries, identifying the primary sources of error and quantifying their impact on the output distribution. Specifically, their methodology involved dissecting the internal steps of the sampler, including the evaluation of the probability mass function, the impact of approximations and floating-point arithmetic. By systematically evaluating each component, they were able to provide an explicit upper bound on the statistical distance between the distribution of implemented binomial samplers and the true Binomial distribution.

A parallel line of work has focused on the evaluation of samplers in terms of their statistical distance from the target distribution [25,10,29,28,1,24,2]. These works focus on designing provably correct tests to assess the quality of samplers by estimating the statistical distance between the sampler and the target distribution. However, the key limitations with these works are twofold: (1) these tests ideally require $\mathcal{O}\left(Poly\left(1/\Delta\right)\right)$ many samples to approximate the statistical distance Δ between the sampler and the target distribution. In real-world applications, however, the library-provided samplers are often quite close to the target distribution, and therefore, the statistical distance is typically quite small. In such cases, these tests are not practical, as they require a large number of samples to approximate the statistical distance. (2) These tests assume a sample-only access to the samplers, treating the sampler as a black-box from which samples can be drawn, but without insight into its internal workings. In practice, however, we often have access to the implementation of these samplers, which could allow us an easier analysis that goes beyond sample-only access to the samplers.

Finally, the impact of numerical accuracy on computational programs has been extensively studied in the literature of verification. A large body of work has focused on analyzing and bounding errors in fundamental arithmetic op-

erations [9,17,16,18,33]. More recently, researchers have investigated how such numerical errors propagate in higher-level computations. For instance, [4] and [7] have explored the effect of the errors in the performance and stability of functions such as log-sum-exp and softmax. These studies highlight the critical importance of accounting for numerical error when designing algorithms and evaluating their real-world behavior.

4 Analysis of Implementations of Poisson Distributions

In this section, we first provide an overview of the Transformed Rejection Sampling approach, which is the core approach implemented in the standard libraries to enable sampling from Poisson distributions. For concreteness, we use the same constants as in the NumPy's implementation of Poisson distribution (v2.2.0, released Dec 8, 2024), GSL). We then state the *main technical result* of our paper, the upper bound on the statistical distance between the sampler implemented in libraries and the ideal Poisson distribution and discuss the implications. We close off the section with a visualization of our bounds in Section 4.3.

4.1 Transform Rejection Samplers

The sampling subroutine implemented in libraries relies on the standard transformed rejection sampling method [15]. At a high level, we first draw a sample k from a hat distribution h via inverse transform, and accept k with probability proportional to the rejection ratio $r_k = \frac{\mathsf{p}_\lambda(k)}{\alpha \mathsf{h}(k)}$ where $\alpha \geq \max_k \frac{\mathsf{p}_\lambda(k)}{\mathsf{h}(k)}$ is the rejection constant.

The detailed algorithm is depicted in Algorithm 1. The algorithm relies on a pair of functions, collectively referred to as the inverse transform function, denoted $(\mathcal{H}, \mathsf{h})$, where h is the hat distribution and $\mathcal{H}$ is its cumulative distribution function (CDF). $\mathcal{H}$ is provided via its inverse function $\mathcal{H}^{-1}$, so that computing $\mathcal{H}^{-1}(u)$ uses basic arithmetic operations [14,15,12]. Since $(\mathcal{H}^{-1})'(u) = 1/\mathsf{h}(k)$, the hat distribution is typically represented in the reciprocal form $\mathsf{h}^{-1}(u) := 1/\mathsf{h}(k)$. We use $\mathsf{h}(k)$ and $\mathsf{h}^{-1}(u)$ interchangeably, depending on whether we are indexing by an outcome k or a uniform random variable u. The algorithm first samples k from h using the inverse transform method, that is by computing $\lfloor \mathcal{H}^{-1}(u) \rfloor$, where u is drawn uniformly from $[0, 1]$. It then computes the rejection ratio r_k and accepts k with probability r_k. The algorithm uses a uniform random sample v to determine whether to accept or reject the candidate k. Since, computing the rejection ratio r_k requires evaluating the Poisson probability mass function $\mathsf{p}_\lambda(k)$, which is computationally expensive, the algorithm usually computes the logarithm $\log r_k$.

The inverse function $\mathcal{H}$ as used in [15] is defined as follows:

$$\mathcal{H}^{-1}(u) = \left(\frac{2\gamma_1}{1/2 - |u|} + \gamma_2 \right) u + \gamma_3 \tag{6}$$

where the parameters γ_1, γ_2, and γ_3 are functions of the Poisson rate parameter λ [15], given by:

$$\gamma_3 = \lambda + 0.445,\ \gamma_2 = 0.931 + 2.53 \cdot \sqrt{\lambda},\ \text{ and } \gamma_1 = -0.059 + 0.02483 \cdot \gamma_2$$

Likewise, the rejection constant α is also parameterized by λ and following [15] it can be written as $\alpha = 1.1239 + 1.1328/(\gamma_2 - 3.4)$.

Taken together, these steps form a sampler that is inherently approximate: numerical errors influence both the proposal step (computing $\mathcal{H}^{-1}(u)$) and the rejection step (computing $\log r_k$). In the next section, we analyze how these approximations affect the output distribution and present bounds on the statistical distance $\Delta(\mathsf{p}_\lambda^{\mathsf{PoiSamp}}, \mathsf{p}_\lambda)$.

Algorithm 1: PoiSamp(λ)

Input : Parameter λ
Output: Sample k

Initialize inverse-function-pair $(\mathcal{H}, \mathsf{h})$
Initialize rejection parameter α
$l_\lambda \leftarrow \mathsf{Log}(\lambda)$
while *True* **do**
  $v \leftarrow$ uniform random samples within $[0, 1]$
  $u \leftarrow$ uniform random samples within $[-0.5, 0.5]$
  $k \leftarrow \lfloor \mathcal{H}^{-1}(u) \rfloor$
  $l_k \leftarrow \mathsf{LogFactorial}(k)$
  $l_v \leftarrow k \otimes l_\lambda \ominus \lambda \ominus l_k \oplus \mathsf{Log}(\mathsf{h}^{-1}(u)) \ominus \mathsf{Log}(\alpha)$
  if $\mathsf{Log}(v) \leq l_v$ **then**
    return k

4.2 Our Results

We now analyze the deviation between the ideal Poisson distribution p_λ and the distribution $\mathsf{p}_\lambda^{\mathsf{PoiSamp}}$ induced by the sampler PoiSamp. Our objective is to identify the primary sources of error introduced during sampling and to provide a formal bound on the statistical distance $\Delta(\mathsf{p}_\lambda, \mathsf{p}_\lambda^{\mathsf{PoiSamp}})$. Unlike previous analyses for the binomial case [34], our approach must address the unbounded support of the Poisson distribution. To manage this, we leverage the characteristics of the inverse transform function $(\mathcal{H}, \mathsf{h})$ used in the sampler. In particular, we utilize the moments of the hat distribution h, which play a key role in bounding the statistical distance. Our main result is summarized in the following theorem.

Theorem 1. *Let $\beta \geq 2\lceil \log_2 \lambda \rceil$ be the bit precision used by* PoiSamp*, and let $\mathsf{p}_\lambda^{\mathsf{PoiSamp}}$ be the resulting output distribution. Then the statistical distance between*

$\mathsf{p}_\lambda^{\mathsf{PoiSamp}}$ *and the ideal Poisson distribution* p_λ *is bounded by:*

$$\Delta\left(\mathsf{p}_\lambda, \mathsf{p}_\lambda^{\mathsf{PoiSamp}}\right) \leq 5\zeta + (1098\beta + 5 + 5\log(1+\lambda) + 3c)\,\lambda\varepsilon + 1607\beta\varepsilon + 2\alpha c^2 \mu_2^{\mathsf{h}} \varepsilon^2 + \frac{\alpha c \mu_1^{\mathsf{h}}}{2}\varepsilon + \mathcal{O}(\varepsilon)$$

where $\varepsilon = 2^{-\beta}$ *is the roundoff error,* ζ *is the uniform error bound from the Lanczos approximation and* c *is a constant depending on the inverse sampling function* $(\mathcal{H}, \mathsf{h})$*. The terms* μ_1^{h} *and* μ_2^{h} *are the first and second moments of the hat distribution* h*, respectively. The term* $\mathcal{O}(\varepsilon)$ *represents higher-order contributions that are negligible compared to* ε*.*

The bound can be interpreted as $\mathcal{O}(\zeta + \lambda\varepsilon)$, where ζ denotes the error from Lanczos' factorial approximation and ε captures the finite-precision error. This decomposition shows that the statistical distance between the sampler's distribution and the ideal Poisson distribution is affected by these two sources of error. As a result, improving either source independently leads to a corresponding improvement in inexactness. This separation is particularly valuable from a designer's perspective, as it allows targeted optimization of numerical precision or factorial approximation accuracy independently without requiring a complete redesign of the sampling procedure.

We now turn to observe that the statistical distance decreases exponentially with the bit precision β, specifically as 2^β. For large values of λ, it is often infeasible to explicitly represent the full domain of the Poisson distribution. The parameter β governs the number of domain values that can be accurately represented. Higher precision allows for a more complete and accurate support. Increasing β by 1 doubles the number of representable domain values, thereby bringing the sampling distribution closer to the ideal. Finally, even when the support is representable, the probability mass function (pmf) values of the Poisson distribution may not be expressible under limited precision, and therefore, causing imprecision in rejection step.

Implications of Our Results Our findings offer both theoretical and practical value for the design and deployment of Poisson samplers in real-world systems. In particular, as implied by Lemma 1, when an implementation of a randomized algorithm makes an API call to an implementation of Poisson distribution, say PoiSamp, then probability of failure of the implementation can be bounded in terms of the statistical distance between the PoiSamp's output distribution and the ideal Poisson distribution. Therefore, it is crucial to advocate for the use of PoiSamp in applications that require Poisson sampling.

Building on this, we advocate for an extension to existing Poisson sampler interfaces that allows users to express and monitor accuracy requirements directly. Specifically, we propose two additions to the sampler API (see Figure 1):

(1) an input parameter δ_{in}, which lets users specify a target upper bound on the statistical distance from the ideal p_λ distribution, and

```
def PoiSamp(lambd):
    """
    Input: lambd
    Output: sample
    """
    ...
```

```
def PoiSamp(lambd, delta_in):
    """
    Input: lambd, 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`.

(2) an output parameter δ_{out}, which reports the actual statistical distance achieved by the sampler at runtime.

These extensions enable users to have control over the trade-off between accuracy and efficiency. For instance, when δ_{in} is loose, faster approximate implementations may be employed; when tighter guarantees are required, the sampler can internally adjust precision to meet the specified bound. The output δ_{out} provides runtime visibility of the deviation.

4.3 Visualization

To illustrate the implications of our theoretical results, we visualize, in Fig 2, the upper bound on the statistical distance $\Delta(\mathsf{p}_\lambda^{\mathsf{PoiSamp}}, \mathsf{p}_\lambda)$ derived in our main theorem for various values of λ and precision β. This plot is generated directly from the analytical bound derived in our main theorem. To evaluate the statistical distance empirically, we need two additional components to be specified:

(1) The hat distribution h used in the sampler, which is defined by the inverse function $\mathcal{H}^{-1}$.
(2) The factorial approximation scheme used in the sampler.

For the hat distribution, we fix the inverse transform function defined in Equation (6). Considering this inverse function we can fix the value of $c = 10$. Now we need to understand the moments μ_2^{h} and μ_1^{h} of h. We can compute a trivial upper bound on these moments. Particularly, for $r \in \{1, 2\}$, we can bound μ_r^{h} as follows:

$$\mu_r^{\mathsf{h}} = \mathop{\mathbb{E}}_{\mathsf{h}}[X^r] = \sum_{k=0}^{\infty} k^r \mathsf{h}(k) \leq \max_k(k^r) \cdot \mathop{\mathbb{E}}_{\mathsf{h}}[1] = \max_k(k^r)$$

Since for the hat distribution h the domain is unbounded, the trivial upper bound on the moments is not very useful. However, we can use the fact that maximum mass of the hat distribution h is concentrated within a bounded range. Therefore, μ_r^{h} can be approximately bounded as

$$\mu_r^{\mathsf{h}} = \mathop{\mathbb{E}}_{\mathsf{h}}[X^r] \leq \sum_{k=0}^{K} k^r \mathsf{h}(k) + \delta \leq \max_{k \in [0,K]}(k^r) + \delta \lesssim \max_{k \in [0,K]}(k^r)$$

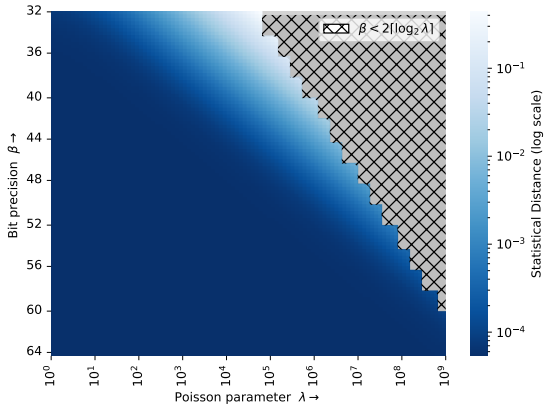


Fig. 2. Heatmap of the upper bound on the statistical distance $\Delta(\mathsf{p}_\lambda^{\mathsf{PoiSamp}}, \mathsf{p}_\lambda)$ as a function of λ and precision β, as predicted by our bound. The color intensity indicates the distance, with darker shades representing larger distances. The results show that the distance increases with λ, indicating better sampling guarantees. The precision β also plays a crucial role, with higher values leading to lower distances. The 'xx' shaded region indicates the illegal region $\beta < 2\lceil \log_2 \lambda \rceil$.

Since 99% of the probability mass of h would be concentrated within $[0,\ \mathcal{H}^{-1}(0.49)]$, we conservatively upper bound the effective support of h by:

$$\mathcal{H}^{-1}(0.49) \leq \lambda + 7.397\sqrt{\lambda} - 2.617$$

Combining these, we can approximately bound the moments as follows:

$$\mu_2^{\mathsf{h}} \lesssim (\lambda + 7.397\sqrt{\lambda} - 2.617)^2, \qquad \mu_1^{\mathsf{h}} \lesssim \lambda + 7.397\sqrt{\lambda} - 2.617$$

This simplifies our bound in Theorem 1 to:

$$\begin{aligned} \Delta\left(\mathsf{p}_\lambda, \mathsf{p}_\lambda^{\mathsf{PoiSamp}}\right) \lesssim\ & 5\zeta + (1098\beta + 35 + 5\log(1+\lambda))\,\lambda\varepsilon + 1607\beta\varepsilon \\ & + 200\alpha(\lambda + 7.397\sqrt{\lambda} - 2.617)^2\varepsilon^2 + 5\alpha(\lambda + 7.397\sqrt{\lambda} - 2.617)\varepsilon \end{aligned} \tag{7}$$

For the factorial approximation, we adopt the Lanczos approximation, which is widely used in libraries and we set the factorial error bound ζ using conservative parameters $t = 6$ and $g = 5$, based on the bounds from [30] (see section 2).

Figure 2 presents a heatmap of the statistical distance as a function of λ and β. The color intensity reflects the magnitude of the distance, with darker regions indicating higher error. Since our theoretical guarantees are only valid when $\beta \geq 2\lceil \log_2 \lambda \rceil$, we explicitly mark the invalid region in the plot. The range of β spans from single-precision floating point ($\beta = 32$) to double precision ($\beta = 64$), reflecting typical usage in practice. The visualization confirms that increasing β leads to a smaller distance, and that higher values of λ require greater precision to maintain accuracy—both aligning with our theoretical insights.

5 Detailed Technical Analysis

In this section, we present a detailed analysis of the errors introduced by the Poisson sampler PoiSamp and prove our main result Theorem 1. We begin by outlining the sources of deviations, followed by a formal proof of the theorem.

As the name, Inverse Transformed Rejection sampling, indicates that there are two sources of deviations from Ideal distribution that we must account for:

`E1` **(Deviation due to Inverse Transform):** Due to the use of finite-precision arithmetic in evaluating the inverse CDF $\mathcal{H}^{-1}(u)$, the drawn value $\tilde{k}$ may differ from the intended value k, introducing a perturbation in the hat distribution h itself.

`E2` **(Deviation due to Rejection Ratio Evaluation):** This component stems from approximating the Poisson mass function $\mathsf{p}_\lambda(k)$, particularly in the evaluation of expressions like $\log(\lambda^k/k!)$ using numerical routines such as Log and $\mathsf{LogFactorial}$. These approximations—combined with finite-precision arithmetic—introduce error into the rejection ratio r_k and, consequently, into the acceptance probability.

Deviation due to Inverse Transform Sampling (`E1`) The first source of deviation, `E1`, originates from numerical inaccuracies in computing the proposal k by $\mathcal{H}^{-1}(u)$, during transformed rejection sampling. Rather than committing to a particular hat distribution, we treat this error abstractly and assume that any implementation of $\mathcal{H}^{-1}$ involves a sequence of basic arithmetic operations $\{+, -, \times, \div, \sqrt{\ }\}$. Each operation introduces a small rounding error, which accumulates multiplicatively. We bound the total relative error by a factor $\varepsilon_{\mathcal{H}} = c\varepsilon$, where $\varepsilon = 2^{-\beta}$ is the machine roundoff and $c > 0$ is a constant dependent on the number of operations involved. In practice, for the standard inverse transform function defined in eq. (6) $c \leq 10$; therefore for $\beta > 5$, we have $\varepsilon_{\mathcal{H}} \leq \frac{1}{2}$.

As a result, the computed value $\widetilde{\mathcal{H}}^{-1}(u)$ deviates from the true value $\mathcal{H}^{-1}(u)$ and lies within the range $[(1-\varepsilon_{\mathcal{H}})\mathcal{H}^{-1}(u), (1+\varepsilon_{\mathcal{H}})\mathcal{H}^{-1}(u)]$. When the final output is obtained by applying $\lfloor\cdot\rfloor$ to this approximate value, it may produce any neighboring integer $k' \in [(1-\varepsilon_{\mathcal{H}})\mathcal{H}^{-1}(u), (1+\varepsilon_{\mathcal{H}})\mathcal{H}^{-1}(u)]$ instead of the correct k. This deviation perturbs the proposal distribution h, resulting in a modified distribution $\widetilde{\mathsf{h}}$ that we analyze in the next lemma.

Lemma 3. *Let $\widetilde{\mathsf{h}}$ be the perturbed hat distribution due to finite-precision errors in evaluating $\mathcal{H}^{-1}(u)$, and let $\varepsilon_{\mathcal{H}} = c\varepsilon$, where $c > 0$ depends on the number of operations in $\mathcal{H}^{-1}(\cdot)$. Then, for all $k \geq 0$,*

$$(1 - \varepsilon_{\mathcal{H}}(3k+1))\,\mathsf{h}(k) \leq \widetilde{\mathsf{h}}(k) \leq \varepsilon_{\mathcal{H}}\big(1 + S_k\big)\mathsf{h}(k),$$

where $S_k = \sum_{t \in L_k \cup U_k} w_{kt}(t+1)$, $w_{kt} = \mathsf{h}(t)/\mathsf{h}(k)$, and

$$L_k = \left[\left\lceil \tfrac{k}{1+\varepsilon_{\mathcal{H}}} \right\rceil, k-1\right], \quad U_k = \left[k+1,\ \left\lfloor \tfrac{k+1}{1-\varepsilon_{\mathcal{H}}} \right\rfloor + 1\right].$$

Error in Rejection Ratio Computation (E2) The second source of error, E2, stems from approximations used in computing the rejection ratio $r_k = \frac{\mathsf{p}_\lambda(k)}{\alpha \mathsf{h}(k)}$. However, in a practical implementation, neither $\mathsf{p}_\lambda(k)$ nor $\mathsf{h}(k)$ are evaluated exactly. The evaluation of $\mathsf{p}_\lambda(k)$ requires computing:$\log \mathsf{p}_\lambda(k) = k \log \lambda - \lambda - \log(k!)$, where $\log(k!)$ is approximated using the Lanczos method, and logarithms and arithmetic operations are performed within the given precision. Let $\widetilde{r}_k$ denote the computed rejection ratio $\widetilde{r}_k = \frac{\mathsf{p}_\lambda^{\mathsf{PoiSamp}}(k)}{\alpha \mathsf{h}(k)}$, where $\mathsf{p}_\lambda^{\mathsf{PoiSamp}}(k)$ is the numerically computed approximation of $\mathsf{p}_\lambda(k)$ using finite-precision logarithms and factorial estimates. The dominant sources of error in computing $\widetilde{r}_k$ are:

(a) **Logarithmic evaluation error:** Logarithmic terms such as $\log \lambda$ and $\log(k + g + \frac{1}{2})$ are evaluated using the finite-precision logarithm function Log. As shown in Equation (3), this introduces an additive error of at most $\tau = \mathcal{O}(\varepsilon)$.

(b) **Lanczos approximation error:** The value of $\log(k!)$ is not computed exactly but approximated using the Lanczos formula, as defined in Equation (5). This involves not only the error due to factorial approximation but also the logarithm errors.

(c) **Arithmetic rounding error:** The final Poisson log-probability expression, given by

$$k \cdot \mathsf{Log}(\lambda) - \lambda - \mathsf{LogFactorial}(k) - \mathsf{Log}(\mathsf{h}(k)),$$

involves a sequence of arithmetic operations including addition, subtraction, multiplication, division, and square root. Each operation performed in finite precision arithmetic introduces a small rounding error, and these accumulate proportionally to the number of operations and the unit round-off error ε.

These sources combine to influence the actual rejection ratio $\widetilde{r}_k$. The following lemma formalizes the effect of these approximations on the rejection ratio.

Lemma 4. *Let $r_k = \frac{\mathsf{p}_\lambda(k)}{\alpha \mathsf{h}(k)}$ denote the ideal rejection ratio and $\widetilde{r}_k = \frac{\mathsf{p}_\lambda^{\mathsf{PoiSamp}}(k)}{\alpha \mathsf{h}(k)}$ the rejection ratio computed using finite-precision arithmetic. Then, for all $k \geq 0$, we have $(1 - 3\zeta - \Lambda\varepsilon) \leq \frac{\widetilde{r}_k}{r_k} \leq (1 + 3\zeta + \Lambda\varepsilon)$ where ζ is the uniform bound on the Lanczos approximation error, $\varepsilon = 2^{-\beta}$ is the floating-point unit roundoff, and $\Lambda = (722k + 1068)\beta + 10\lambda + 10k \log k + 10 \log(\mathsf{h}^{-1}(u)) + c$*

6 Case Study: DNF Counting

Now we illustrate how our theoretical bounds can be integrated into practical tools. To showcase the applicability of our bounds, we utilize them in conjunction with an off-the-shelf Poisson sampler to implement a recent proposed DNF counting algorithm [37]. The problem of counting the number of satisfying assignments for a DNF formula is a well-known #P-hard problem. 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 $|sol(\varphi)|$.

Since the problem of DNF counting is #P-complete, the problem of approximately counting the $|sol(\varphi)|$ for a DNF formula φ has been studied extensively in the literature [22,23,36,37]. We outline the algorithm of Soos et al. [37] in algorithm 2. This 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, number of copies of σ belongs to $\mathcal{X}$ follows a Poisson distribution with parameter 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 `pepin` asks for a sample N_i from Poisson distribution $\mathsf{p}_{|\varphi_i|p}$. It is important to note that $|\varphi_i|$ is easy to compute since it is simply 2^{n-k}, where k is the number of literals in clause φ_i. The algorithm then adds N_i many new satisfying assignments (not necessarily distinct) 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 `pepin` is to output the ratio $\frac{|\mathcal{X}|}{p}$ which is a good estimate of $|sol(\varphi)|$, and with high probability, $(1-\varepsilon)|sol(\varphi)| \leq \frac{|\mathcal{X}|}{p} \leq (1+\varepsilon)|sol(\varphi)|$.

Since the core randomness of `pepin` algorithm relies on the sampler PoiSamp, the statistical distance between the distribution $\mathsf{p}_\lambda^{\mathsf{PoiSamp}}$ and the ideal Poisson distribution p_λ directly impacts the quality of the output. Here we illustrate how our bounds can be used to provide guarantees on the output of `pepin` when PoiSamp is used instead of an ideal Poisson sampler. Throughout this section, we will use the simplified version of our bound as given in eq. (7), which is derived from theorem 1. For the completeness of presentation, we restate the bound as a function of λ here:

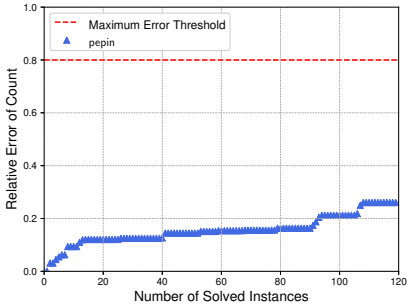
$$\begin{aligned}\mathrm{Err}_{\Delta}(\lambda) = 5\zeta &+ \big(1098\beta + 35 + 5\log(1+\lambda)\big)\,\lambda\varepsilon \\ &+ 1607\beta\varepsilon + 200\alpha\big(\lambda + 7.397\sqrt{\lambda} - 2.617\big)^2\varepsilon^2 \\ &+ 5\alpha\big(\lambda + 7.397\sqrt{\lambda} - 2.617\big)\varepsilon\end{aligned}$$

Following the work of Sarkar et al. [34], we illustrate (1) the algorithmic modifications needed to incorporate the bound on the statistical distance and (2) the analysis modifications needed to provide guarantees on the output of `pepin`. The modifications needed to the algorithm are minimal, as shown in algorithm 2. The only change needed is to compute the statistical distance bound $\mathrm{Err}_{\Delta}(|sol(\varphi_i)|p)$ after sampling from the Poisson distribution in lines 9 and 15 and accumulate it in a variable δ'. This variable keeps track of the total statistical distance accumulated so far. The algorithm returns the estimate $\frac{|\mathcal{X}|}{p}$ along with δ'. If at any point, the accumulated statistical distance δ' exceeds the user-specified error tolerance δ_1, the algorithm returns a failure message. The analysis of the modified algorithm is also straightforward. The only change needed is to account for the additional deviation introduced by the Poisson sampler. By a union bound, the total error of the algorithm due to PoiSamp is at most δ'. Therefore, if $\delta' \geq \delta_1$, the algorithm is no longer guaranteed to provide the

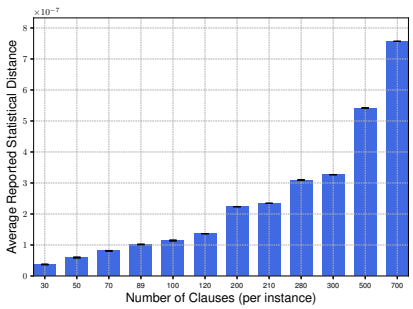
desired approximation with high probability, or in other words, if the algorithm does not return 'Failure', the algorithm provides the desired approximation with probability at least $1 - \delta_2$.

Experimental Setup We implement the `pepin` algorithm as described in algorithm 2. We use the Poisson sampler implemented in NumPy (v2.2.0, released Dec 8, 2024) as PoiSamp. We set the error tolerance parameters $\varepsilon = 0.8$ and $\delta = 0.36$. We set $\kappa = 0.5$ to split the error budget equally between the algorithmic error and the deviation due to PoiSamp. We run our experiments on a machine with AMD EPYC 9654 (Zen 4) @ 2.4 GHz. We use the set of DNF formulas from Soos et al. [37] since they were used as a benchmark in their accuracy evaluation.

Experimental Results We evaluate the performance of the `pepin` algorithm on the benchmark DNF formulas. Our experiments focus on measuring the accuracy of the estimated solution counts and the statistical distance bounds provided by the algorithm. The modified `pepin` algorithm successfully provides accurate estimates for the number of satisfying assignments for all the benchmark DNF formulas as shown in section 6. The reported statistical distance bound δ' by the algorithm for individual instances are shown in section 6. The reported bounds are well within the specified tolerance limit of $\delta_1 = 0.18$. The reported boundss increase with the number of clauses in the DNF formula, as shown in section 6, which is expected since the number of calls to the Poisson sampler increases linearly with the number of clauses. Overall, the experimental results demonstrate that the modified `pepin` algorithm effectively integrates the Poisson sampler while maintaining high accuracy and providing reliable error bounds.



(a) Accuracy experiment results for `pepin`. The red line represents the tolerance factor ($\varepsilon = 0.8$).



(b) Reported statistical distance (δ') by `pepin` averaged over instances with the same number of clauses.

Fig. 3. Experimental results for `pepin`.

Algorithm 2: `pepin` $(\varphi, \varepsilon, \delta, \kappa)$
(The modifications to the original algorithm 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 $p \leftarrow 1$, and multiset $\mathcal{X} \leftarrow \emptyset$
$\delta' \leftarrow 0$
for $i = 1$ *to* m **do**
    for *all* $\sigma \in \mathcal{X}$ **do**
        if $\sigma \vDash \varphi_i$ **then**
            remove σ from $\mathcal{X}$
    $N_i \leftarrow \text{PoiSamp}(|sol(\varphi_i)|p)$
    $\delta^i \leftarrow \text{Err}_\Delta(|sol(\varphi_i)|p)$
    $\delta' \leftarrow \delta' + \delta^i$
    if $N_i + |\mathcal{X}| > T$ **then**
        Randomly remove elements from $\mathcal{X}$ with probability $\frac{1}{2}$
        $p \leftarrow p/2$
        $N_i \leftarrow \text{PoiSamp}(|sol(\varphi_i)|p)$
        $\delta^i \leftarrow \text{Err}_\Delta(|sol(\varphi_i)|p)$
        $\delta' \leftarrow \delta' + \delta^i$
    if $\delta' > \delta_1$ **then**
        return *"Failure: Error tolerance exceeded"*
    Add N_i many (not necessarily distinct) satisfying assignments of φ_i to $\mathcal{X}$
return $|\mathcal{X}|/p$

7 Conclusion

This work presents the first formal analysis of the statistical distance between practical Poisson samplers and the ideal distribution. Extending the framework of Sarkar et al. [34] to the unbounded Poisson case, we analyze NumPy's inverse transform implementation and establish a bound of $\mathcal{O}(\zeta + \lambda\varepsilon)$ on the total variation distance, where ζ captures factorial approximation errors and ε reflects finite-precision effects. The key technical challenge lies in handling the unbounded support of Poisson distributions, which we address by leveraging moments of the hat distribution h. Our analysis reveals two independent error sources: Lanczos factorial approximations and arithmetic rounding, enabling targeted optimization strategies.

Beyond immediate contributions to Poisson sampling, our methodology provides a template for analyzing other unbounded discrete distributions. The proposed API extensions—allowing users to specify accuracy requirements and receive runtime guarantees—point toward formal reliability assurances in statistical computing libraries. As randomized algorithms become increasingly central to computational science, such formal guarantees on fundamental building blocks transition from desirable to essential.

References

1. Banerjee, A., Chakraborty, S., Chakraborty, S., Meel, K.S., Sarkar, U., Sen, S.: Testing of horn samplers. In: AISTATS (2023)
2. Bhattacharyya, R., Chakraborty, S., Pote, Y., Sarkar, U., Sen, S.: Testing self-reducible samplers. In: AAAI (2024)
3. Binder, K., Heermann, D.W., Binder, K.: Monte Carlo simulation in statistical physics, vol. 8. Springer (1992)
4. 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)
5. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Communications of the ACM **13**(7), 422–426 (1970)
6. 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. (2023)
7. 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)
8. Borwein, J.M., Borwein, P.B.: The arithmetic-geometric mean and fast computation of elementary functions. SIAM review **26**(3), 351–366 (1984)
9. Brent, R., Percival, C., Zimmermann, P.: Error bounds on complex floating-point multiplication. Mathematics of Computation **76**(259), 1469–1481 (2007)
10. Chakraborty, S., Meel, K.S.: On testing of uniform samplers. In: AAAI (2019)
11. Corbineau, P., Duclos, M., Lakhnech, Y.: Certified security proofs of cryptographic protocols in the computational model: An application to intrusion resilience. In: International Conference on Certified Programs and Proofs. pp. 378–393. Springer (2011)
12. Devroye, L.: Nonuniform random sample generation. Handbooks in operations research and management science **13**, 83–121 (2006)
13. Hoare, C.A.R.: Algorithm 64: quicksort. Communications of the ACM **4**(7), 321 (1961)
14. Hörmann, W.: The generation of binomial random samples. Journal of statistical computation and simulation **46**(1-2), 101–110 (1993)
15. Hörmann, W.: The transformed rejection method for generating poisson random variables. Insurance: Mathematics and Economics **12**(1), 39–45 (1993)
16. 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)
17. 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)
18. 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)
19. Jongbloed, G., Koole, G.: Managing uncertainty in call centres using poisson mixtures. Applied Stochastic Models in Business and Industry **17**(4), 307–318 (2001)
20. Kachitvichyanukul, V., Schmeiser, B.W.: Binomial random sample generation. Communications of the ACM **31**(2), 216–222 (1988)

21. Karney, C.F.: Sampling exactly from the normal distribution. ACM Transactions on Mathematical Software (TOMS) **42**(1), 1–14 (2016)
22. Karp, R.M., Luby, M.: Monte-carlo algorithms for enumeration and reliability problems. In: FOCS (1983)
23. Karp, R.M., Luby, M., Madras, N.: Monte-carlo approximation algorithms for enumeration problems. Journal of algorithms **10**(3), 429–448 (1989)
24. Kumar, G., Meel, K.S., Pote, Y.: Tolerant testing of high-dimensional samplers with subcube conditioning. In: AISTATS (2025)
25. Meel, K.S., Pote, Y.P., Chakraborty, S.: On testing of samplers. NeurIPS (2020)
26. Mooney, C.Z.: Monte carlo simulation. No. 116, Sage (1997)
27. Muller, J.M., Muller, J.M.: Elementary functions. Springer (2006)
28. Pote, Y., Meel, K.S.: On scalable testing of samplers. NeurIPS (2022)
29. Pote, Y.P., Meel, K.S.: Testing probabilistic circuits. NeurIPS (2021)
30. Pugh, G.R.: An analysis of the Lanczos gamma approximation. Ph.D. thesis, University of British Columbia (2004)
31. Pugh, W.: Concurrent maintenance of skip lists. Citeseer (1990)
32. Robert, P.: Queues with poisson arrivals. In: Stochastic Networks and Queues, pp. 177–206. Springer (2003)
33. 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)
34. Sarkar, U., Chakraborty, S., Meel, K.S.: Assessing the quality of binomial samplers: A statistical distance framework. In: International Conference on Computer Aided Verification. pp. 231–253. Springer (2025)
35. Schmeiser, B., Kachitvichyanukul, V.: Poisson random sample generation. Research memorandum pp. 81–4 (1981)
36. Soos, M., Aggarwal, D., Chakraborty, S., Meel, K.S., Obremski, M.: Engineering an efficient approximate dnf-counter. In: IJCAI. pp. 2031–2038 (2023)
37. Soos, M., Sarkar, U., Aggarwal, D., Chakraborty, S., Meel, K.S., Obremski, M.: Engineering an efficient approximate dnf-counter. arXiv preprint arXiv:2407.19946 (2024)
38. The MPFR Team: The mpfr library: Algorithms and proofs, `https://www.mpfr.org/algorithms.pdf/`
39. Thomopoulos, N.T.: Essentials of Monte Carlo simulation: Statistical methods for building simulation models. Springer Science & Business Media (2012)

A Detailed Analysis

A.1 Technical Proof of the Main Result

We begin by presenting the proof of our main result, Theorem 1, leveraging the supporting lemmas Lemma 3 and Lemma 4. The proofs of these lemmas are deferred to later in this section. For completeness, we first restate the theorem below.

Theorem 1. *Let $\beta \geq 2\lceil \log_2 \lambda \rceil$ be the bit precision used by* PoiSamp*, and let $\mathsf{p}_\lambda^{\mathsf{PoiSamp}}$ be the resulting output distribution. Then the statistical distance between $\mathsf{p}_\lambda^{\mathsf{PoiSamp}}$ and the ideal Poisson distribution p_λ is bounded by:*

$$\Delta\left(\mathsf{p}_\lambda, \mathsf{p}_\lambda^{\mathsf{PoiSamp}}\right) \leq 5\zeta + (1098\beta + 5 + 5\log(1+\lambda) + 3c)\,\lambda\varepsilon + 1607\beta\varepsilon + 2\alpha c^2 \mu_2^{\mathsf{h}} \varepsilon^2 + \frac{\alpha c \mu_1^{\mathsf{h}}}{2}\varepsilon + \mathcal{O}(\varepsilon)$$

where $\varepsilon = 2^{-\beta}$ is the roundoff error, ζ is the uniform error bound from the Lanczos approximation and c is a constant depending on the inverse sampling function $(\mathcal{H}, \mathsf{h})$. The terms μ_1^{h} and μ_2^{h} are the first and second moments of the hat distribution h, respectively. The term $\mathcal{O}(\varepsilon)$ represents higher-order contributions that are negligible compared to ε.

Proof. Before we begin, without loss of generality, let us assume $\mathsf{h}(-1) = \mathsf{h}(n+1) = \mathsf{p}_\lambda(-1) = 0$ and $r_{-1} = 0$. Additionally, we define the event 'accept' as the event such that the sample k is sampled by the sampler. Then the probability of a point k in the sampler's distribution is given by $\mathsf{p}_\lambda^{\mathsf{PoiSamp}}(k) = \Pr(k|\mathsf{accept})$. Applying Bayesian rule we get,

$$\mathsf{p}_\lambda^{\mathsf{PoiSamp}}(k) = \frac{\Pr(\mathsf{accept}|k)\Pr(k)}{\Pr(\mathsf{accept})} = \frac{\widetilde{r}_k \widetilde{\mathsf{h}}(k)}{\Pr(\mathsf{accept})}.$$

Therefore, to get a bound on $\mathsf{p}_\lambda^{\mathsf{PoiSamp}}(k)$, we will use lemma 3 and lemma 4 to bound $\widetilde{r}_k$, $\widetilde{\mathsf{h}}(k)$, and $\Pr(\mathsf{accept})$. First we start with lower bounding $\Pr(\mathsf{accept})$. In PoiSamp a sample k being sampled from $\widetilde{\mathsf{h}}$, is accepted with probability $\widetilde{r}_k$. Therefore averaging over all possible k, $\Pr(\mathsf{accept})$ can be expressed as:

$$\Pr(\mathsf{accept}) = \sum_{k\geq 0} \Pr(\mathsf{accept}|k)\Pr(k) = \sum_{k\geq 0} \widetilde{r}_k \widetilde{\mathsf{h}}(k).$$

Substituing, the lower bounds of $\widetilde{r}_k$, $\widetilde{\mathsf{h}}(k)$ from Lemma 4 and Lemma 3 respectively, we get the following lower bound on $\Pr(\mathsf{accept})$:

$$\begin{aligned}
&\Pr(\mathsf{accept}) \\
&\geq \sum_{k\geq 0}(1-3\zeta-(722k+1068)\beta\varepsilon-10\lambda\varepsilon-10k\log k\varepsilon-10\log(\mathsf{h}^{-1}(u))\varepsilon-\mathcal{O}(\varepsilon)) \\
&\qquad\qquad \cdot(1-(3k+1)\varepsilon_{\mathcal{H}})r_k\mathsf{h}(k) \\
&\geq \sum_{k\geq 0}(1-3\zeta-(722k+1068)\beta\varepsilon-10\lambda\varepsilon-10k\log k\varepsilon-10\log(\mathsf{h}^{-1}(u))\varepsilon-\mathcal{O}(\varepsilon)) \\
&\qquad\qquad \cdot(1-(3k+1)\varepsilon_{\mathcal{H}})\frac{\mathsf{p}_\lambda(k)}{\alpha}
\end{aligned}$$

The last inequality follows from the definition of rejection ratio $r_k = \frac{\mathsf{p}_\lambda(k)}{\alpha\mathsf{h}(k)}$. We further simplify the above expression. Since $\mathsf{p}_\lambda(k)\cdot\mathsf{h}^{-1}(u)\leq\alpha$, we have the following inequality:

$$\begin{aligned}
\sum_{k\geq 0}\mathsf{p}_\lambda(k)\log(\mathsf{h}^{-1}(u)) &= \sum_{k\geq 0}\mathsf{p}_\lambda(k)\log\left(\mathsf{p}_\lambda(k)\mathsf{h}^{-1}(u)\right)+\sum_{k\geq 0}\mathsf{p}_\lambda(k)\log\left(\frac{1}{\mathsf{p}_\lambda(k)}\right) \\
&\leq \log\alpha+\frac{1}{2}\log(2\pi e\lambda).
\end{aligned}$$

The last inequality uses the bound on entropy of the Poisson distribution, $\mathbb{E}_{k\sim\mathsf{p}_\lambda}[-\log(\mathsf{p}_\lambda(k))]\leq\frac{1}{2}\log(2\pi e\lambda)\leq\frac{1}{2}\log(2\pi e)+\frac{\beta}{2}$. Also, we have

$$\mathop{\mathbb{E}}_{k\sim\mathsf{p}_\lambda}[k\log k]\leq\lambda\log(1+\lambda).$$

Combining the two inequalities above, we obtain the desired lower bound on $\Pr(\mathsf{accept})$,

$$\Pr(\mathsf{accept})\geq(1-3\zeta-(742\lambda+1073)\beta\varepsilon-(3\lambda+1)\varepsilon_{\mathcal{H}}-\mathcal{O}(\varepsilon))\cdot\frac{1}{\alpha}$$

Now let us move back on to bounding $\mathsf{p}_\lambda^{\mathsf{PoiSamp}}(k)$. Next using the upper bound on $\widetilde{r}_k$ from Lemma 4 and the lower bound on $\Pr(\mathsf{accept})$ from above, we can obtain the following upper bound on $\mathsf{p}_\lambda^{\mathsf{PoiSamp}}(k)$:

$$\begin{aligned}
\mathsf{p}_\lambda^{\mathsf{PoiSamp}}(k)\leq\Big(&1+9\zeta+(2186k+3209)\beta\varepsilon+10\lambda\varepsilon+10k\log k\varepsilon \\
&+10\log(\mathsf{h}^{-1}(u))\varepsilon+(6\lambda+2)\varepsilon_{\mathcal{H}}+\mathcal{O}(\varepsilon)\Big)\cdot\frac{\mathsf{p}_\lambda(k)\widetilde{\mathsf{h}}(k)}{\mathsf{h}(k)}
\end{aligned}$$

Now to finish with the proof we need to bound the ratio $\frac{\widetilde{\mathsf{h}}(k)}{\mathsf{h}(k)}$ which capture the multiplicative deviation between the sampling hat distribution from the ideal hat distribution h. We use Lemma 3 to bound this ratio. For ease of readability we

introduce the notation κ_k (parameterized by k) $\kappa_k = \varepsilon_{\mathcal{H}} \sum_{t \in L_k \cup U_k} w_{kt} \cdot (t+1)$. Therefore the bound on $\mathsf{p}_\lambda^{\mathsf{PoiSamp}}(k)$ reduces down to the following:

$$\begin{aligned}
\mathsf{p}_\lambda^{\mathsf{PoiSamp}}(k) &\leq \Big(1 + 9\zeta + (2186k + 3209)\beta\varepsilon + 10\lambda\varepsilon + 10k\log k\varepsilon \\
&\qquad + 10\log(\mathsf{h}^{-1}(u))\varepsilon + (6\lambda+2)\varepsilon_{\mathcal{H}} + \mathcal{O}(\varepsilon)\Big) \cdot (1+\kappa_k) \cdot \mathsf{p}_\lambda(k) \\
&\leq \Big(1 + 9\zeta + (2186k + 3209)\beta\varepsilon + 10\lambda\varepsilon + 10k\log k\varepsilon \\
&\qquad + 10\log(\mathsf{h}^{-1}(u))\varepsilon + (6\lambda+2)\varepsilon_{\mathcal{H}} + \kappa_k + \mathcal{O}(\varepsilon)\Big) \cdot \mathsf{p}_\lambda(k)
\end{aligned}$$

To finish with the proof we start by focusing on the expected value of the term κ_k first. Particularly, we begin analyzing the term $\kappa_k \mathsf{p}_\lambda(k)$ to upper bound $\sum_{k\geq 0} \kappa_k \mathsf{p}_\lambda(k)$ as follows:

$$\begin{aligned}
\sum_{k\geq 0} \kappa_k\, \mathsf{p}_\lambda(k) &= \varepsilon_{\mathcal{H}} \sum_{k\geq 0} \left(\sum_{t \in L_k \cup U_k} w_{kt} \cdot (t+1) \right) \mathsf{p}_\lambda(k) \\
&\leq \varepsilon_{\mathcal{H}} \sum_{k\geq 0} \left(\sum_{t \in L_k \cup U_k} w_{kt} \cdot (t+1) \right) \alpha \mathsf{h}(k) && (\mathsf{p}_\lambda(k) \leq \alpha \mathsf{h}(k)) \\
&\leq \varepsilon_{\mathcal{H}} \alpha \sum_{k\geq 0} \left(\sum_{t \in L_k \cup U_k} \mathsf{h}(t) \cdot (t+1) \right) && (\text{Def. of } w_{kt}) \\
&= \varepsilon_{\mathcal{H}} \alpha \sum_{t\geq 0} \mathsf{h}(t) \cdot (t+1) \cdot |\{k : t \in L_k \cup U_k\}| \\
&\leq \varepsilon_{\mathcal{H}} \alpha \sum_{t\geq 0} \mathsf{h}(t) \cdot (t+1) \cdot (4\varepsilon_{\mathcal{H}}(t+1) + 1) && (\text{Bounds on } L_k, U_k) \\
&\leq \alpha \left(4\varepsilon_{\mathcal{H}}^2 \sum_{t\geq 0} \mathsf{h}(t)(t+1)^2 + \varepsilon_{\mathcal{H}} \sum_{t\geq 0} \mathsf{h}(t)(t+1) \right)
\end{aligned}$$

Since, $\mathbb{E}_{\mathsf{h}}(X) = \sum_{k\geq 0} k\mathsf{h}(k)$ and $\mathbb{E}_{\mathsf{h}}(X^2) = \sum_{k\geq 0} k^2 \mathsf{h}(k)$, we can rewrite the above expression as:

$$\sum_{k\geq 0} \kappa_k\, \mathsf{p}_\lambda(k) = 4\alpha\varepsilon_{\mathcal{H}}^2 \mathop{\mathbb{E}}_{\mathsf{h}}(X^2) + \alpha\varepsilon_{\mathcal{H}} \mathop{\mathbb{E}}_{\mathsf{h}}(X) = 4\alpha\varepsilon_{\mathcal{H}}^2 \mu_2^{\mathsf{h}} + \alpha\varepsilon_{\mathcal{H}} \mu_1^{\mathsf{h}}$$

The final inequality follows from the definition of μ_1^{h} and μ_2^{h}. Finally, recall the statistical distance between the sampler's distribution $\mathsf{p}_\lambda^{\mathsf{PoiSamp}}$ and the ideal Poisson distribution p_λ is given by,

$$\Delta(\mathsf{p}_\lambda^{\mathsf{PoiSamp}}, \mathsf{p}_\lambda) = \frac{1}{2} \sum_{k\geq 0} \left| \mathsf{p}_\lambda^{\mathsf{PoiSamp}}(k) - \mathsf{p}_\lambda(k) \right|$$

Substituing the upper bound for $\mathsf{p}_\lambda^{\mathsf{PoiSamp}}(k)$, we get,

$$
\begin{aligned}
&\Delta(\mathsf{p}_\lambda^{\mathsf{PoiSamp}}, \mathsf{p}_\lambda) \\
\leq &\frac{1}{2}\sum_{k\geq 0}\Big|9\zeta + (2186k+3209)\beta\varepsilon + 10\lambda\varepsilon + 10k\log k\varepsilon \\
&\qquad + 10\log(\mathsf{h}^{-1}(u))\varepsilon + (6\lambda+2)\varepsilon_{\mathcal{H}} + \kappa_k + \mathcal{O}(\varepsilon)\Big| \cdot \mathsf{p}_\lambda(k) \\
\leq &5\zeta + (1098\lambda+1607)\beta\varepsilon + 5\lambda\varepsilon + 5\lambda\log(1+\lambda)\varepsilon \\
&\qquad + (3\lambda+1)\varepsilon_{\mathcal{H}} + 2\alpha\varepsilon_{\mathcal{H}}^2\mu_2^{\mathsf{h}} + \frac{\alpha\varepsilon_{\mathcal{H}}}{2}\mu_1^{\mathsf{h}} + \mathcal{O}(\varepsilon) \\
= &5\zeta + \left(1098\beta + 5 + 5\log(1+\lambda) + 3c\right)\lambda\varepsilon + 1607\beta\varepsilon + 2\alpha\varepsilon_{\mathcal{H}}^2\mu_2^{\mathsf{h}} + \frac{\alpha\varepsilon_{\mathcal{H}}}{2}\mu_1^{\mathsf{h}} + \mathcal{O}(\varepsilon)
\end{aligned}
$$

This completes the proof of Theorem 1.

A.2 Analysis of Error Type E1

In this section, we analyze the first source of deviation in the sampling process, denoted as error type E1. As discussed earlier this error arises from the discrepancy in the hat distribution h used in the transformed rejection sampling method. Our goal is to formally bound this deviation. To this end, we prove Lemma 3. For completeness, we begin by restating the lemma below.

Lemma 3. *Let $\widetilde{\mathsf{h}}$ be the perturbed hat distribution due to finite-precision errors in evaluating $\mathcal{H}^{-1}(u)$, and let $\varepsilon_{\mathcal{H}} = c\varepsilon$, where $c > 0$ depends on the number of operations in $\mathcal{H}^{-1}(\cdot)$. Then, for all $k \geq 0$,*

$$(1 - \varepsilon_{\mathcal{H}}(3k+1))\,\mathsf{h}(k) \leq \widetilde{\mathsf{h}}(k) \leq \varepsilon_{\mathcal{H}}\big(1 + S_k\big)\mathsf{h}(k),$$

where $S_k = \sum_{t\in L_k\cup U_k} w_{kt}(t+1)$, $w_{kt} = \mathsf{h}(t)/\mathsf{h}(k)$, and

$$L_k = \left[\left\lceil \tfrac{k}{1+\varepsilon_{\mathcal{H}}}\right\rceil, k-1\right], \quad U_k = \left[k+1,\ \left\lfloor \tfrac{k+1}{1-\varepsilon_{\mathcal{H}}}\right\rfloor + 1\right].$$

Proof. Without loss of generality assume $\mathsf{h}(k) = 0$ for all $k < 0$. Let the "true" value of $\lfloor \mathcal{H}^{-1}(u)\rfloor$ be k, and let k' be the value produced by Algorithm 1. Since $k' \in \big\{ t \in \mathbb{Z} \mid \lfloor(1-\varepsilon_{\mathcal{H}})\mathcal{H}^{-1}(u)\rfloor \leq t \leq \lfloor(1+\varepsilon_{\mathcal{H}})\mathcal{H}^{-1}(u)\rfloor\big\}$. Hence a proposal k can be produced by $t \in \left[\left\lceil \frac{k}{1+\varepsilon_{\mathcal{H}}}\right\rceil, \left\lfloor \frac{k+1}{1-\varepsilon_{\mathcal{H}}}\right\rfloor - 1\right]$. We define sets $L_k := [\lceil k/(1+\varepsilon_{\mathcal{H}})\rceil, k-1]$ and $U_k := [k+1, \lfloor(k+1)/(1-\varepsilon_{\mathcal{H}})\rfloor + 1]$. We can express $\widetilde{\mathsf{h}}(k)$ as,

$$\widetilde{\mathsf{h}}(k) = \mathsf{h}(k) + \sum_{t\in L_k\cup U_k} \Pr_{\varepsilon,u}\big(k' = k \mid \lfloor\mathcal{H}^{-1}(u)\rfloor = t\big)\,\mathsf{h}(t).$$

We will prove the lemma in two parts: we first prove the upper bound and then the lower bound.

Upper bound. An upward error from t to any $t' > t$ can only occur if

$$\mathcal{H}^{-1}(u) \;\geq\; \frac{t'}{1+\varepsilon_{\mathcal{H}}} \;\geq\; \frac{t+1}{1+\varepsilon_{\mathcal{H}}}$$

This implies that the potential error sources lie in the top $\varepsilon_{\mathcal{H}}$-fraction of $[t, t+1)$. Hence under the uniform assumption on $\mathcal{H}^{-1}(u)$ over $[t, t+1)$ we can bound the probability of k' being $t' \geq t+1$ given $\lfloor \mathcal{H}^{-1}(u) \rfloor = t$, which essentially means that the probability mass can 'leak' to $k' \geq t+1$ only if $\mathcal{H}^{-1}(u)$ falls within this upper $\varepsilon_{\mathcal{H}}$-fraction interval. This leakage probability can be explicitly upper bounded as follows:

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

The last inequality follows from the fact that

$$\Pr_{\varepsilon}\left(k' \geq t+1 \mid \lfloor \mathcal{H}^{-1}(u) \rfloor = t, \mathcal{H}^{-1}(u) = v\right) \leq 1.$$

We can also upper bound the probability of k' being $t' < t$ given $\lfloor \mathcal{H}^{-1}(u) \rfloor = t$ in a similar way, by $\Pr_{\varepsilon,u}\left(k' \leq t-1 \mid \lfloor \mathcal{H}^{-1}(u) \rfloor = t\right) \leq \varepsilon_{\mathcal{H}}(t+1)$. Therefore, combining, we can upper bound the probability mass:

$$\widetilde{\mathsf{h}}(k) \leq \left(1 + \varepsilon_{\mathcal{H}} \sum_{t \in L_k \cup U_k} w_{kt} \cdot (t+1)\right) \mathsf{h}(k)$$

Lower bound. We now turn to establishing a lower bound on the probability that the output of the sampler remains accurate, i.e., that the value returned is exactly $k' = k$, given that the (floor of the) inverse CDF approximation is $\lfloor \mathcal{H}^{-1}(u) \rfloor = k$.

Observe that if the value $\mathcal{H}^{-1}(u)$ lies within the interval

$$\left[\tfrac{k}{1-\varepsilon_{\mathcal{H}}},\ \tfrac{k+1}{1+\varepsilon_{\mathcal{H}}}\right],$$

then, even though the rounding behavior and the error bounds on the inverse function, the sampled number will be $k' = k$ with probability 1, i.e., no overshoot or undershoot occurs. Therefore, within this interval, all mass contributes faithfully to $\widetilde{\mathsf{h}}(k)$, the probability that the sampler returns k.

To compute the amount of mass that safely lands in this range, we take the length of the interval:

$$\begin{aligned}\Pr\left(k' = k \mid \lfloor \mathcal{H}^{-1}(u) \rfloor = k\right) &\geq \tfrac{k+1}{1+\varepsilon_{\mathcal{H}}} - \tfrac{k}{1-\varepsilon_{\mathcal{H}}} \\ &= (k+1)\left(1 - \varepsilon_{\mathcal{H}} + \mathcal{O}(\varepsilon_{\mathcal{H}}^2)\right) - k\left(1 + \varepsilon_{\mathcal{H}} + \mathcal{O}(\varepsilon_{\mathcal{H}}^2)\right) \\ &\geq 1 - \varepsilon_{\mathcal{H}}(2k+1)\end{aligned}$$

Thus, at least $(1-\varepsilon_{\mathcal{H}}(2k+1))$-fraction of the probability mass originally assigned to k by h remains intact in the distribution $\widetilde{\mathsf{h}}$, yielding:

$$\widetilde{\mathsf{h}}(k) \;\geq\; (1 - \varepsilon_{\mathcal{H}}(2k+1))\,\mathsf{h}(k).$$

This completes the proof.

A.3 Analysis of Error Type E2

In this section, we analyze the second source of deviation in the sampling process, denoted as error type E2. As discussed earlier, this error arises from the approximation of the factorial terms, logarithms, and basic arithmetic operations in the rejection ratio computation. Our goal is to formally bound this deviation. To that end, we prove Lemma 4. For completeness, we begin by restating the lemma below.

Lemma 4. *Let $r_k = \frac{\mathsf{p}_\lambda(k)}{\alpha \mathsf{h}(k)}$ denote the ideal rejection ratio and $\widetilde{r}_k = \frac{\mathsf{p}_\lambda^{\mathsf{PoiSamp}}(k)}{\alpha \mathsf{h}(k)}$ the rejection ratio computed using finite-precision arithmetic. Then, for all $k \geq 0$, we have $(1 - 3\zeta - \Lambda\varepsilon) \leq \frac{\widetilde{r}_k}{r_k} \leq (1 + 3\zeta + \Lambda\varepsilon)$ where ζ is the uniform bound on the Lanczos approximation error, $\varepsilon = 2^{-\beta}$ is the floating-point unit roundoff, and $\Lambda = (722k + 1068)\beta + 10\lambda + 10k \log k + 10 \log(\mathsf{h}^{-1}(u)) + c$*

To prove Lemma 4, we begin by analyzing the effect of factorial approximation on the rejection ratio and consequently we will show how $\log n!$ is approximated using LogFactorial, and their effect on the rejection ratio. For the purpose of the proof, we introduce a new notation $\mathsf{p}_\lambda^{\mathsf{Inter}}$ as the intermediate distribution, which denotes the distribution where only the factorial computations are approximated and all other computations are exact. Similarly we define $r_k^{\mathsf{Inter}} = \frac{\mathsf{p}_\lambda^{\mathsf{Inter}}(k)}{\alpha \mathsf{h}(k)}$ as the rejection ratio of if we were to use the intermediate distribution $\mathsf{p}_\lambda^{\mathsf{Inter}}$ instead of the ideal Poisson distribution. We formalize the notion of intermediate distribution using the following lemma.

Lemma 5. *For all $k \geq 0$, $(1 - 3\zeta)\mathsf{p}_\lambda(k) \leq \mathsf{p}_\lambda^{\mathsf{Inter}}(k) \leq (1 + 3\zeta)\mathsf{p}_\lambda(k)$, where ζ denotes the uniform error bound due to the Lanczos approximation and $\mathsf{p}_\lambda^{\mathsf{Inter}}$ denotes the intermediate distribution.*

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

$$\overline{\mathsf{p}}_\lambda(k) = \frac{e^{-\lambda}\lambda^k}{\mathsf{Fact}^{\mathsf{Lancz}}(k)}$$

But, here $\overline{\mathsf{p}}_\lambda$ is not necessarily a well defined distribution, since $\sum_{k=0}^{n} \overline{\mathsf{p}}_\lambda(k)$ is not necessarily 1. We normalize $\overline{\mathsf{p}}_\lambda$ to a distribution and observe that the normalized distribution corresponds to the intermediate distribution $\mathsf{p}_\lambda^{\mathsf{Inter}}$, that is,

$$\frac{\overline{\mathsf{p}}_\lambda(k)}{\sum_{i\geq 0} \overline{\mathsf{p}}_\lambda(i)} = \mathsf{p}_\lambda^{\mathsf{Inter}}(k)$$

From eq. (4) we have $(1-\zeta)k! \leq \mathsf{Fact}^{\mathsf{Lanczs}}(k) \leq (1+\zeta)k!$ for all $k \in [n]$. Therefore,

$$\sum_{k\geq 0} \mathsf{p}_\lambda(k) \cdot \frac{1}{(1+\zeta)} \leq \sum_{k\geq 0} \overline{\mathsf{p}}_\lambda(k) \leq \sum_{k\geq 0} \mathsf{p}_\lambda(k) \cdot \frac{1}{(1-\zeta)}$$

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

$$\begin{aligned} & \frac{(1-\zeta)}{(1+\zeta)}\mathsf{p}_\lambda(k) \leq \frac{\overline{\mathsf{p}}_\lambda(k)}{\sum_{i\geq 0} \overline{\mathsf{p}}_\lambda(i)} \quad \leq \frac{(1+\zeta)}{(1-\zeta)}\mathsf{p}_\lambda(k) \\ \implies \qquad & (1-3\zeta)\mathsf{p}_\lambda(k) \leq \frac{\overline{\mathsf{p}}_\lambda(k)}{\sum_{i\geq 0} \overline{\mathsf{p}}_\lambda(i)} \quad \leq (1+3\zeta)\mathsf{p}_\lambda(k) \end{aligned}$$

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

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

Before proceeding with the proof of Lemma 4, we need to address one remaining component: formalizing the error bound introduced by the $\mathsf{LogFactorial}$ function. We state this formally in the following lemma.

Lemma 6. *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 a fixed integer k, the function $\mathsf{LogFactorial}(k)$ evaluates the following expression:

$$\frac{1}{2}\mathsf{Log}(2\pi) + \left(k+\frac{1}{2}\right)\mathsf{Log}\left(k+g+\frac{1}{2}\right) - \left(k+g+\frac{1}{2}\right) + \mathsf{Log}\left(A_{t,g}(k)\right),$$

where all logarithmic computations are performed using the approximate logarithm function $\mathsf{Log}(\cdot)$ based on the AGM method. From the approximation guarantee in Equation (3), we know that for any $x > 0$,

$$|\mathsf{Log}(x) - \log(x)| \leq \tau,$$

where $\tau = \frac{178\beta}{2^\beta}$ is the additive approximation error incurred in computing each logarithm.

We now analyze the cumulative error when applying Log to the components of the Lanczos approximation. Using the above bound, we can replace each $\mathsf{Log}(x)$

term in the expression with its corresponding $\log(x)$ plus or minus τ, depending on whether we are computing the upper or the lower bound. Specifically, we upper bound $\mathsf{LogFactorial}(k)$ as follows:

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

An analogous argument provides a lower bound:

$$\mathsf{LogFactorial}(k) \geq \log(\mathsf{Fact}^{\mathsf{Lancz}}(k!)) - (k+2)\tau.$$

Combining both bounds, we conclude that:

$$|\mathsf{LogFactorial}(k) - \log(\mathsf{Fact}^{\mathsf{Lancz}}(k!))| \leq (k+2)\tau,$$

which completes the proof.

We are now ready to prove Lemma 4, which provides a bound on the rejection ratio computed by PoiSamp. The proof will leverage the bounds established in Lemma 5 and Lemma 6.

Proof (Proof of Lemma 4). We begin by the definition of the rejection ratio $\widetilde{r}_k$ computed by PoiSamp, which is given by,

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

We bound each term in the above expression separately and then combine the bounds to obtain the final bound on $\widetilde{lr}_k$. We start with the term $\mathsf{Log}(\mathsf{h}^{-1}(u))$. 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. (3) we can upper bound the term by,

$$\mathsf{Log}\left((1+c\varepsilon).\mathsf{h}^{-1}(u)\right) \leq \log\left(\mathsf{h}^{-1}(u)\right) + \log(1+c\varepsilon) + \tau \tag{8}$$

Similarly, considering the error in the term $k \otimes l_\lambda$, we deduce using eq. (1), eq. (3) that,

$$k \otimes l_\lambda \leq (1+\varepsilon)k\log\lambda + k\tau + k\tau\varepsilon \tag{9}$$

Lastly, we are interested in the error in the computation of l_k. Using lemma 6, we can lower bound the term as follows:

$$\log \mathsf{Fact}^{\mathsf{Lancz}}(k) - (k+2)\tau \leq l_k \tag{10}$$

Thus, we can bound $k \otimes l_\lambda$ using eq. (9) and eq. (10) as follows:

$$\begin{aligned} & k \otimes l_\lambda - l_k \\ \leq & (1+\varepsilon)k \log \lambda - \log \mathsf{Fact}^{\mathsf{Lancz}}(k) + k\tau + k\tau\varepsilon + (k+2)\tau \end{aligned} \tag{11}$$

Similarly, using the fact that $\log \mathsf{Fact}^{\mathsf{Lancz}}(k) \leq k \log k$, we can bound $k \otimes l_\lambda + l_k$ to be at most,

$$\begin{aligned} & (1+\varepsilon)k \log \lambda + \log \mathsf{Fact}^{\mathsf{Lancz}}(k) + k\tau + k\tau\varepsilon + (k+2)\tau \\ \leq & (1+\varepsilon)k \log \lambda + k \log k + k\tau + k\tau\varepsilon + (k+2)\tau \end{aligned} \tag{12}$$

Until now we have taken care of the errors due to the computation of the logarithms and factorials. 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. (2):

$$\begin{aligned} & k \otimes l_\lambda \ominus \lambda \ominus l_k \oplus \mathsf{Log}(\mathsf{h}^{-1}(u)) \\ \leq & k \otimes l_\lambda - \lambda - l_k + \mathsf{Log}(\mathsf{h}^{-1}(u)) + \\ & \qquad 4\varepsilon \left(k \otimes l_\lambda + \lambda + l_k + \mathsf{Log}(\mathsf{h}^{-1}(u)) \right) \end{aligned}$$

Using the corresponding bounds from eqs. (11) and (12) we can bound the above expression by,

$$\begin{aligned} & k \otimes l_\lambda \ominus \lambda \ominus l_k \oplus \mathsf{Log}(\mathsf{h}^{-1}(u)) \\ \leq & k \log \lambda - \lambda - \log \mathsf{Fact}^{\mathsf{Lancz}}(k) + \log(\mathsf{h}^{-1}(u)) \\ & \quad + k\tau + (k+2)\tau + \varepsilon k \log \lambda + \log(1+c\varepsilon) + \tau \\ & \quad + 4\varepsilon \left(k \log \lambda + \lambda + k \log k + \log(\mathsf{h}^{-1}(u)) \right) + \mathcal{O}(k\tau\varepsilon) \end{aligned}$$

This accumulates all the errors due to the computation of the logarithms, factorials, and basic arithmetic operations. Define $lr_k := \log(r_k^{\mathsf{Inter}})$. Therefore we now upper bound $\widetilde{lr}_k$ using the above expression as follows:

$$\begin{aligned} \widetilde{lr}_k &= k \otimes l_\lambda \ominus \lambda \ominus l_k \oplus \mathsf{Log}(\mathsf{h}^{-1}(u)) \\ &\leq lr_k + (2k+3)\tau \\ &\qquad + 5\varepsilon \left[k \log \lambda + \lambda + k \log k + \log(\mathsf{h}^{-1}(u)) \right] \\ &\qquad + \log(1+c\varepsilon) + \mathcal{O}(k\tau\varepsilon) \\ &\leq lr_k + (2k+3)\tau + 5k\beta\varepsilon + 5\lambda\varepsilon + 5k \log k\varepsilon \\ &\qquad + 5 \log(\mathsf{h}^{-1}(u))\varepsilon + \log(1+c\varepsilon) + \mathcal{O}(k\tau\varepsilon) \end{aligned}$$

The second inequality follows from $\log\lambda, \log k \leq \beta$ (due to our choice of β). Let $\Gamma := (2k+3)\tau + 5k\beta\varepsilon + 5\lambda\varepsilon + 5k\log k\varepsilon + 5\log(\mathsf{h}^{-1}(u))\varepsilon + \mathcal{O}(k\tau\varepsilon)$. Using a similar set of inequalities, we can also lower bound $\widetilde{lr}_k$ by $\widetilde{lr}_k \geq lr_k - \log(1+c\varepsilon) - \Gamma$. 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:

$$\begin{aligned}
\widetilde{r}_k &\leq r_k^{\mathsf{Inter}} \cdot (1 + c\varepsilon) \cdot e^{\Gamma} \\
&\leq r_k^{\mathsf{Inter}}(1 + 2\Gamma + c\varepsilon + \mathcal{O}(k\tau\varepsilon)) \\
&\leq r_k^{\mathsf{Inter}}(1 + 178(4k+6)\beta\varepsilon + 10k\beta\varepsilon + 10\lambda\varepsilon + 10k\log k\varepsilon \\
&\qquad\qquad + 10\log(\mathsf{h}^{-1}(u))\varepsilon + c\varepsilon + \mathcal{O}(k\varepsilon^2)) \\
&= r_k^{\mathsf{Inter}}(1 + (722k+1068)\beta\varepsilon + 10\lambda\varepsilon + 10k\log k\varepsilon \\
&\qquad\qquad + 10\log(\mathsf{h}^{-1}(u))\varepsilon + c\varepsilon + \mathcal{O}(k\varepsilon^2))
\end{aligned}$$

The last inequality follows from substituing the value of τ. Similarly, using the lower bound $\widetilde{lr}_k \geq lr_k - \log(1+c\varepsilon) - \Gamma$ we have, $\widetilde{r}_k \geq r_k^{\mathsf{Inter}}(1-(722k+1068)\beta\varepsilon - 10\lambda\varepsilon - 10k\log k\varepsilon - 10\log(\mathsf{h}^{-1}(u))\varepsilon - c\varepsilon - \mathcal{O}(k\varepsilon^2))$. Finally, applying the bound on the ratio $\frac{r_k^{\mathsf{Inter}}}{r_k}$ from lemma 5 we have,

$$(1 - 3\zeta - \Lambda\varepsilon - \mathcal{O}(\varepsilon)) \leq \frac{\widetilde{r}_k}{r_k} \leq (1 + 3\zeta + \Lambda\varepsilon + \mathcal{O}(\varepsilon))$$

The final statistical distance between the output distribution $\mathsf{p}_\lambda^{\mathsf{PoiSamp}}$ and the true distribution p_λ accounts for both sources of error. By combining Lemmas 3 and 4, and bounding the cumulative error over all values of k, we obtain the main theorem (see Theorem 1) showing that the total deviation scales as $\mathcal{O}(\zeta + \lambda\varepsilon)$, with constants depending on the inverse sampling implementation.