

Reliability of browser-native Poisson probabilities: overflow, cancellation, and the limits of self-consistency checks

JJ Ben-Joseph
TensorSpace, Inc, Tel Aviv, Israel
jj@tensorspace.ai

October 18, 2026

Abstract

Direct evaluation of a Poisson mass can overflow even when the probability is ordinary: along the integer modal sequence, the separate-factor formula first returns infinity at mean and count 144, where the mass is approximately 0.033226. Avoiding that overflow does not by itself establish numerical reliability. We evaluate a frozen browser-native implementation that combines a logarithmic mass, regularized incomplete-gamma cumulative probabilities, and a mass-cumulative-step agreement check. A deterministic corpus of 1,923 rate-count pairs spans zero and very small rates, the overflow boundary, extreme tails, and means through 10^6 . References evaluated at 80 and 120 decimal digits agree after rounding to binary64; a positive-recurrence audit independently checks 20 selected cases without incomplete-gamma evaluation. The direct-factor baseline is nonfinite in 781 cases. The checked implementation returns 1,816 results and refuses 107; among returned results, maximum absolute errors are 7.21×10^{-12} for the mass and 1.44×10^{-10} for the cumulative/tail probabilities. The study also identifies refusals caused by amplification of cumulative rounding errors, accepted results with shared errors, and three avoidable subnormal zeros. A compensated-summation ablation reduces the worst relative mass error over the full corpus from 3.41×10^{-7} to 2.01×10^{-9} for reference masses at least 10^{-300} . The contribution is a reproducible failure analysis and validation protocol for a small numerical application, rather than a new Poisson algorithm or an accuracy certificate for its advertised input range.

Keywords: Poisson distribution; floating-point arithmetic; incomplete gamma; numerical validation; reproducibility.

1 Introduction and contribution

A web calculator that returns a finite number has not necessarily computed the requested probability accurately. The Poisson distribution provides a compact example because its familiar expression contains an exponentially small factor, an enormous power, and a factorial. Intermediate overflow can destroy a probability that is neither small nor ill conditioned. Replacing these factors with logarithms repairs the range problem but introduces other questions: how accurately is a long logarithmic sum accumulated, how are tails evaluated, and what does an agreement check actually establish?

This study examines one fixed version of the AgentCalc Poisson calculator. Its public interface reports the mass, cumulative probability, inclusive upper tail, and a consistency residual. The evaluator is small enough to audit and archive in full. The study asks three questions: (i) where

do elementary evaluation strategies fail; (ii) what accuracy and coverage does the checked implementation achieve on a declared stress corpus; and (iii) which failures remain invisible to, or are introduced by, the consistency check?

The contribution comprises an exact overflow-boundary analysis, a versioned experiment using the actual JavaScript evaluator, an analysis of the information contained in its residual, and a controlled accumulation ablation. Complete inputs, results, reference values, and reproduction code accompany the manuscript. The application is a case study; the known probability identities and numerical techniques are not claimed as new.

1.1 Related numerical methods

Stable Poisson evaluation is an established problem. Fox and Glynn [1] construct a band of nonnegligible probabilities around the mode, with truncation control and work proportional to $\sqrt{\lambda}$ for fixed error requirements. Regularized incomplete-gamma identities provide the cumulative probabilities, while asymptotic methods are important for large parameters [2, 3]. R documents a Loader-based mass evaluation [4]; this is a stronger production alternative than forming a power and factorial separately. SciPy exposes dedicated mass, cumulative, survival, and logarithmic functions, and notes that direct survival evaluation can be more accurate than subtracting a CDF from one [5]. These methods motivate evaluation against established software and higher precision, not replacement of mature libraries by an unqualified short implementation.

2 Probability specification and overflow boundary

For a rate $\lambda > 0$ and integer count $k \geq 0$, define

$$p_k = e^{-\lambda} \lambda^k / k!, \quad F_k = \Pr(X \leq k), \quad S_k = \Pr(X \geq k). \quad (1)$$

The upper tail is inclusive. Consequently $S_k = 1 - F_{k-1}$ for $k \geq 1$, and $F_k + S_k = 1 + p_k$, not one. At zero rate the distribution is concentrated at zero: $F_k = 1$, $p_k = \mathbf{1}_{k=0}$, and $S_k = \mathbf{1}_{k=0}$ for nonnegative integer k .

The historical evaluation forms λ^k , $e^{-\lambda}$, and $k!$ separately in binary64. Its largest finite positive value is

$$M = (2 - 2^{-52})2^{1023}, \quad \log_{10} M = 308.25471555991675. \quad (2)$$

Along $\lambda = k = n$, the power has logarithm $n \log_{10} n$. This is 308.21305335750384 at $n = 143$, but 310.804198861716 at $n = 144$. The power therefore overflows at 144 while the factorial is still finite. At 171 the factorial also overflows, producing an indeterminate infinity-over-infinity expression. These statements concern this modal sequence and evaluation order, not a universal threshold for every rate-count combination.

For integer $n > 0$, the modes are $n - 1$ and n . Stirling’s formula gives

$$p_n \sim (2\pi n)^{-1/2}. \quad (3)$$

The modal probability decreases smoothly through both overflow boundaries (Figure 1). At 144 its high-precision reference is approximately 0.03322595651555852. The old cumulative sum includes an infinite term and returns infinity; subtracting the old sum through 143 from one returns negative infinity. This is a representational failure, not an extreme probability. General floating-point issues of range, rounding, and cancellation are discussed by Goldberg [6].

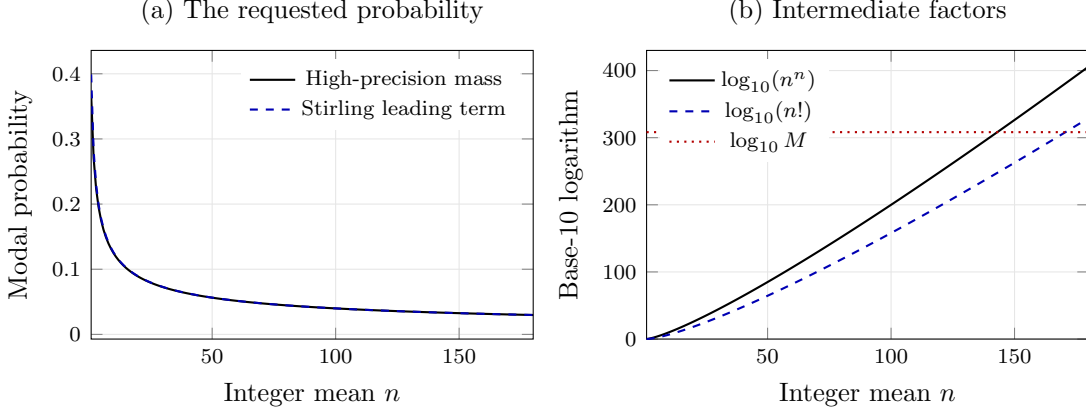


Figure 1: A modest output can require unrepresentable intermediate values. The 180 integer means in this illustration are separate from the declared stress corpus. The reference mass is evaluated at 80 decimal digits; the factor logarithms do not require forming either large factor.

3 The frozen evaluator

The studied artifact is the unmodified calculator HTML from `agentcalc.science` commit 20233d3. Its full commit identifier and SHA-256 checksum are recorded in the accompanying README and experiment summary. The experiment runs its actual inline JavaScript, rather than a Python translation. The source is frozen so that a future change to the live calculator cannot silently change the study.

3.1 Logarithmic mass and incomplete-gamma pair

The mass is evaluated as

$$\hat{p}_k = \exp \left(-\lambda + k \log \lambda - \sum_{i=2}^k \log i \right), \quad (4)$$

with special handling for $k = 0$ and $\lambda = 0$. The sum is accumulated from left to right. For regularized lower and upper incomplete-gamma functions P and Q , respectively, the identities are

$$F_k = Q(k+1, \lambda), \quad S_k = P(k, \lambda) \quad (k \geq 1). \quad (5)$$

They follow from the finite exponential sums in NIST DLMF §8.4 [2]. The implementation evaluates the lower-gamma series when $x < a + 1$ and an upper-gamma continued fraction otherwise. The other member is obtained by complementation. The prefactor uses a $g = 7$ Lanczos-style log-gamma approximation [7]; the exact coefficient list is retained in the snapshot. Neither a factorial nor an unregularized gamma value is formed.

Each expansion has a 10^{-15} relative correction criterion and a limit of 10,000 iterations. This is a stopping rule, not a proved error bound. The branch condition alone also does not prove that the directly computed member is the smaller probability for every parameter pair. The implementation sets a contribution to zero when its computed log value is below -745 .

3.2 Acceptance and refusal

The diagnostic compares the logarithmic mass with a cumulative step:

$$d_k = \hat{F}_k - \hat{F}_{k-1}, \quad R_k = |\hat{p}_k - d_k|, \quad A_k = \max(|\hat{p}_k|, |d_k|). \quad (6)$$

At $k = 0$ the step is $\widehat{F}_0$. A result is refused if an expansion does not meet its stopping criterion, if the mass or CDF is nonfinite, or if

$$A_k > 10^{-10} \quad \text{and} \quad R_k > 10^{-8} A_k. \quad (7)$$

Thus the relative agreement test is disabled for small masses. Negative or nonfinite rates, noninteger or negative counts, and either input above 10^6 are outside the implemented input contract. The display uses five significant digits. This study analyzes the valid numeric evaluator inputs; it does not measure browser layout, form parsing, or clipboard behavior.

4 Why consistency cannot certify accuracy

Write the cumulative errors as $\widehat{F}_j = F_j + e_j$ and, temporarily, take the difference of the stored values in exact arithmetic. Then

$$d_k - p_k = e_k - e_{k-1}, \quad \frac{|d_k - p_k|}{p_k} \leq \frac{|e_k| + |e_{k-1}|}{p_k}. \quad (8)$$

If each CDF has a relative perturbation bounded by η , the corresponding first-order amplification factor is

$$\kappa_k = \frac{F_k + F_{k-1}}{p_k}. \quad (9)$$

This is conditioning of the CDF-difference representation, not a claim that the Poisson mass itself is ill conditioned. When both CDFs are near one, κ_k is approximately $2/p_k$. The subtraction of nearby binary64 CDF values may itself be exact; cancellation exposes errors already present in those values. A small mass computed accurately by equation (4) can therefore disagree with their difference.

Conversely, a common additive error c in both CDFs cancels out of d_k , leaving the residual unchanged while the reported CDF is wrong by c . A small residual cannot bound the individual CDF errors. Nor does agreement between two calculations establish a bound on their shared error. The residual is evidence of consistency, conditional on a well-conditioned comparison; it is not an independent accuracy certificate. The stress study below demonstrates both unnecessary refusals and accepted errors.

5 Reproducible experimental design

5.1 Corpus and execution environment

Table 1 partitions 1,923 distinct valid pairs. The small-rate cohort uses thirteen rates from 10^{-300} to 10, with counts 0–20, 50, and 100. The overflow cohort uses integer rates 120–180 and counts 0, 1, $\lambda - 1$, λ , and $\lambda + 1$. The wide-range grid uses rates $10^{i/10}$, $i = 0, \dots, 60$, and counts $\max(0, \lfloor \lambda + z\sqrt{\lambda} \rfloor)$ for $z \in \{-40, -12, -8, -5, -2, -1, 0, 1, 2, 5, 8, 12, 40\}$, plus counts zero and one. Values outside the input contract are omitted. Additional cohorts probe zero rate, subnormal boundaries, and large-mean neighbors.

The stress cohort proposes 512 pairs using the linear congruential generator $s_{j+1} = (1664525s_j + 1013904223) \bmod 2^{32}$, seeded with 20261018. Rates are logarithmically distributed from 10^{-6} to 10^6 ; counts alternate between standardized offsets and the interval $[0, 2\lambda)$. Domain filtering and deduplication leave 511 distinct stress cases; a pair retains its first assigned cohort. Exact construction and ordering are executable in the accompanying generator; the archived case list

Cohort	Attempted	Returned	Refused
Zero-rate limit	4	4	0
Small rates	299	291	8
Overflow boundary	305	305	0
Wide-range grid	786	707	79
Subnormal boundary	11	11	0
Large-mean probes	7	4	3
Seeded stress cases	511	494	17
Total	1,923	1,816	107

Table 1: Coverage of the frozen evaluator. Cohorts are disjoint after deduplication; refusals remain in all archived input and result files.

records every binary64 input. This deliberately selected stress corpus is not a probability sample of calculator use, so percentages describe coverage of this corpus only.

Evaluation used Node.js 26.11.1, V8 14.6.202.34-node.37, Python 3.15.0, mpmath 1.3.0, NumPy 2.5.4, and SciPy 1.18.1 on an Apple M2, macOS 27.0.1. Node’s virtual-machine module supplies a minimal document stub to load the page; probability functions are then called directly. These are tests of browser-native source executed in Node/V8, not measurements in Chrome, Firefox, or Safari. Engine-specific elementary-function rounding can change last digits and threshold decisions.

5.2 References and metrics

For every pair, mpmath evaluates the mass and incomplete-gamma expressions at 80 and 120 decimal digits [8]. The reference rate is the exact binary64 input value, rather than an idealized decimal rate. The reference uses the upper gamma integral for $x \geq a$ and the lower integral otherwise, complementing the other member. This avoids a slowly convergent large-argument lower-gamma evaluation without requiring a near-unit lower probability to be evaluated directly. All three references agree between precisions after rounding to binary64. This stability check is not a rigorous interval enclosure.

A second audit uses 100-digit positive Poisson recurrences on 20 cases: five fixed examples and the union of the maximum-error cases for each method and metric. The selection rule is specified in the script. Summation proceeds away from the mode; decreasing term ratios give a geometric bound on the unadded tail, required to be at most 10^{-80} of the partial sum. All rounded audit values agree with the gamma-based references. The audit avoids incomplete-gamma calls but shares mpmath’s log-gamma routine for the initial mass; it is not independent of the entire arithmetic library.

SciPy’s `pmf(k, rate)`, `cdf(k, rate)`, and `sf(k-1, rate)` are additional comparators, not exact references. Absolute errors are measured against the rounded high-precision values. Relative-error maxima are reported only where the reference is at least 10^{-300} ; subnormal losses are counted separately. A “lost zero” means that the method returns zero where the reference rounds to a nonzero binary64 value. Inevitably underflowed values that also round to zero in the reference are not counted as lost zeros.

5.3 Baselines and controlled ablation

Four mass evaluations are compared: the direct-factor expression; the forward recurrence $p_0 = e^{-\lambda}$, $p_j = p_{j-1}\lambda/j$; the frozen logarithmic mass; and that same log formula with Kahan compensated

Mass method	Nonfinite	Lost zeros	Max. absolute error	Max. relative error
Direct factors	781	2	1.11×10^{-16}	1.00×10^0
Forward recurrence	0	472	1.42×10^{-2}	1.00×10^0
Logarithmic mass	0	3	1.33×10^{-10}	3.41×10^{-7}
Compensated log sum	0	3	6.11×10^{-13}	2.01×10^{-9}
SciPy mass	0	0	9.62×10^{-13}	2.84×10^{-9}

Table 2: All 1,923 inputs, without the refusal filter. Relative errors use reference masses $\geq 10^{-300}$. Nonfinite outputs are counted but excluded from the finite-error maxima. Lost zeros use all nonzero rounded references, including subnormals. SciPy is a comparator, not the definition of truth.

Returned quantity	Max. absolute error	Max. relative error	Lost zeros
Mass	7.21×10^{-12}	3.25×10^{-7}	3
Cumulative	1.43×10^{-10}	2.52×10^{-9}	3
Inclusive upper tail	1.44×10^{-10}	2.35×10^{-9}	0

Table 3: Errors on the 1,816 returned cases only; the 107 refusals are not successful evaluations. Relative errors use reference values $\geq 10^{-300}$.

accumulation [9]. The ablation changes only the log-factorial sum. It preserves the -745 cutoff and does not modify the cumulative functions or the archived calculator. The cumulative-step check is applied only in the frozen evaluator. Its returned-case errors are therefore reported separately from unconditional baseline errors.

6 Results

6.1 Overflow avoidance and the remaining mass error

Table 2 reports the full-corpus mass comparisons. Direct factors are nonfinite in 781 cases; their small maximum finite absolute error must not be read as good coverage. The forward recurrence avoids large powers, but its seed underflows at large rates and it loses 472 representable probabilities. The logarithmic mass is finite throughout, yet its accumulated error increases with count. Near the mode, equation (4) subtracts quantities of order $k \log k$ to obtain a logarithm of order $\log k$. Avoiding overflow does not eliminate rounding in this cancellation or in the preceding sum.

Compensated accumulation lowers the maximum full-corpus relative mass error from 3.41×10^{-7} to 2.01×10^{-9} , approximately a factor of 170. Figure 2(a) shows the effect along modal-neighbor grid points. This isolates long-sum accumulation as an important contributor, but does not establish compensation as a complete repair: it leaves the gamma evaluation, residual conditioning, and subnormal cutoff untouched.

6.2 Coverage and errors after the agreement check

The evaluator returns 1,816 results and refuses 107 (5.56% of the declared corpus). Every refusal is due to the mass-step test; no gamma expansion reaches its iteration cap on this corpus. The largest combined reported expansion count is 14,827; the iteration cap applies separately to each expansion. All returned probabilities are finite and inside $[0, 1]$, but that necessary condition does not certify their accuracy. Table 3 retains the conditional nature of the returned-case maxima.

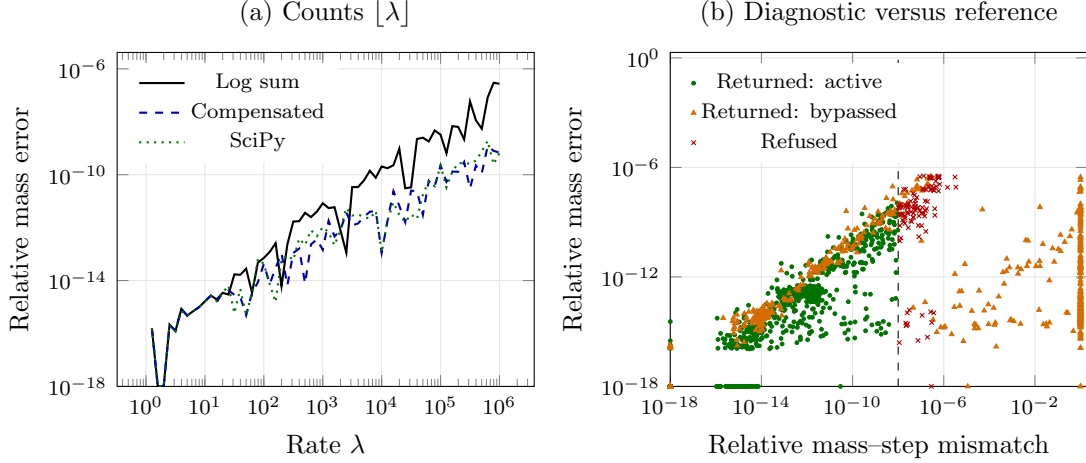


Figure 2: Accumulation and diagnostic behavior. Panel (a) uses the wide-range grid’s modal-neighbor points and the unfiltered mass formulas. Panel (b) uses all cases with reference mass $\geq 10^{-300}$; the vertical line is the relative agreement threshold, active only above the scale floor. Values below 10^{-18} , including exact zero differences, are placed at the plot floor. This graphical floor does not change the reported maxima.

At $(\lambda, k) = (0.001, 3)$ the mass agrees with the rounded high-precision reference, approximately $1.665000833055625 \times 10^{-10}$. Nevertheless the CDF difference disagrees by 4.73×10^{-17} , or about 2.84×10^{-7} relative, causing refusal. Here both CDFs are near one and $\kappa_k \approx 1.20 \times 10^{10}$. The diagnostic is responding to amplified CDF rounding, not a demonstrated mass error. At $(10^6, 10^6)$ it also refuses, but the logarithmic mass itself then has relative error about 2.73×10^{-7} . Identical refusal messages therefore cover different numerical mechanisms.

An accepted case uses the binary64 inputs printed as $\lambda = 398107.1705534969$ and $k = 398107$. The relative mass-step mismatch is about 8.96×10^{-9} , below the threshold, while the mass’s reference-relative error is about 1.14×10^{-8} and the CDF absolute error is 1.43×10^{-10} . Passing the diagnostic has not bounded the true errors by its 10^{-8} agreement parameter. The test is bypassed on 849 inputs because its scale is at most 10^{-10} ; Figure 2(b) distinguishes these cases from active checks.

6.3 Subnormal loss and comparator limits

The smallest positive binary64 subnormal is 2^{-1074} , approximately 4.94×10^{-324} . Under round-to-nearest, the zero boundary for an exact positive exponential is at log value $-1075 \log 2 \approx -745.1332$. The frozen cutoff at -745 is premature. For example, with $\lambda = 745.1$ and $k = 0$, both reference p_0 and F_0 round to 2^{-1074} , while the checked implementation returns zero with zero residual. Three such mass/CDF losses occur in the boundary probes. They are distinct from values so small that binary64 must round them to zero.

SciPy is substantially more accurate in absolute error on this corpus, but also has limits. Its largest absolute errors are approximately 9.62×10^{-13} for mass and 2.34×10^{-13} for either tail probability. Its largest relative inclusive-upper-tail error above the reporting floor is 9.53×10^{-7} , at $\lambda = 832237.6233113663$, $k = 836881$. The high-precision tail is approximately $1.8398261117090812 \times 10^{-7}$; SciPy returns $1.8398243578711704 \times 10^{-7}$. This extremal case is included in the positive-recurrence audit. The observation concerns this version, input, and metric; it is neither evidence against

dedicated survival functions in general nor a proof of a library-wide defect. It illustrates why library agreement alone is an incomplete reference strategy.

7 Implications and limitations

The repaired evaluator successfully avoids the original overflow on the reported cases, including mean 144. Its remaining limitations suggest a specific improvement sequence: use a stable mass formulation or compensated accumulation; distinguish agreement failure from demonstrated probability error; evaluate small tails directly; and let the numerical format, or a documented log-probability interface, determine underflow handling. These are recommendations, not changes silently applied to the studied artifact.

The log-factorial loop takes $O(k)$ work per query, and the cumulative expansions require additional parameter-dependent work. No timing or throughput advantage over mature libraries is established. A mode-centered method with truncation bounds, or an established special-function library, is preferable when guarantees or substantially larger parameters are needed. The present agreement test and finite stress corpus supply neither.

The study is limited to one application snapshot and one V8 runtime. The corpus emphasizes known numerical hazards and is not exhaustive over the continuous rate domain or all allowed counts. Agreement at two precisions and the selected recurrence audit increase confidence in the reported references but do not replace verified interval arithmetic. Five-significant-digit display rounding can conceal small internal errors; conversely, the large relative errors highlighted here can correspond to tiny absolute probabilities. Both metrics and the refusal coverage are therefore necessary. No observational count data were analyzed, and numerical agreement says nothing about whether a Poisson model fits a real process.

8 Conclusion

This case study separates four questions often conflated in a small calculator: whether intermediate values are representable, whether the computed probability is accurate, whether two evaluations agree, and whether the program returns a result. A logarithmic mass repairs the explicit-factor overflow but does not settle the other three. The declared corpus, higher-precision references, retained refusals, and accumulation ablation make those distinctions measurable and reproducible. The principal lesson is to report the evidence and limits of a diagnostic rather than interpreting its success as an accuracy certificate.

Data, code, and reproducibility

The supplementary `poisson_validation/` directory contains the frozen HTML evaluator, Node harness, deterministic case generator, all result rows, 80/120-digit reference values, the positive-recurrence audit, generated plot and table inputs, environment details, and pinned Python requirements. Running `python run_validation.py` from that directory regenerates the study; the README gives complete setup and manuscript-build commands. The snapshot derives from https://github.com/tensorspace-ai/agentcalc_science and retains its AGPL-3.0 license. The accompanying harness and analysis code use the same license. The live interface at <https://agentcalc.com/poisson-distribution-calculator> is contextual; reproduction uses the archived snapshot rather than the live site.

Use of AI tools

OpenAI Codex assisted with experimental design, implementation of the verification scripts, mathematical exposition, literature-source checks, and manuscript drafting and revision. The numerical results were generated by the archived executable procedures, including the higher-precision and positive-recurrence checks. Earlier drafting provenance has not been fully established; the author must verify the complete disclosure before submission.

References

- [1] Bennett L. Fox and Peter W. Glynn. Computing Poisson probabilities. *Communications of the ACM*, 31(4):440–445, 1988. doi: 10.1145/42404.42409.
- [2] National Institute of Standards and Technology. NIST digital library of mathematical functions. dlmf.nist.gov/8.4, 2026. Sections 8.4 and 8.9, Chapter 8, Incomplete Gamma and Related Functions. Accessed 18 October 2026.
- [3] National Institute of Standards and Technology. DLMF, chapter 8: Incomplete gamma and related functions, methods of computation. <https://dlmf.nist.gov/8.25>, 2026. Accessed 18 October 2026.
- [4] R Core Team. The Poisson distribution. <https://stat.ethz.ch/R-manual/R-devel/library/stats/html/Poisson.html>, 2026. R stats reference manual. Accessed 18 October 2026.
- [5] SciPy community. `scipy.stats.poisson`. <https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.poisson.html>, 2026. Online documentation accessed 18 October 2026; experiments use SciPy 1.18.1.
- [6] David Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48, 1991. doi: 10.1145/103162.103163.
- [7] Cornelius Lanczos. 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. doi: 10.1137/0701008.
- [8] Fredrik Johansson et al. `mpmath`: A Python library for arbitrary-precision floating-point arithmetic, version 1.3.0. <https://mpmath.org/doc/1.3.0/>, 2023. Experiment dependency pinned to version 1.3.0.
- [9] W. Kahan. Pracniques: Further remarks on reducing truncation errors. *Communications of the ACM*, 8(1):40, 1965. doi: 10.1145/363707.363723.