Optimally revealing bits for rejection sampling

Louis-Roy Langevin
Dept. of Mathematics and Statistics
McGill University
Montréal, Canada
louis-roy.langevin@mail.mcgill.ca

Alex Waese-Perlman
Dept. of Mathematics and Statistics
McGill University
Montréal, Canada
alex.waese-perlman@mail.mcgill.ca

***Abstract*—Rejection sampling is a popular method used to generate numbers that follow some given distribution. We study the use of this method to generate random numbers in the unit interval from increasing probability density functions. We focus on the problem of sampling from n correlated random variables from a joint distribution whose marginal distributions are all increasing. We show that, in the worst case, the expected number of random bits required to accept or reject a sample grows at least linearly and at most quadratically with n.**

***Keywords*—rejection sampling, random number generation, random bits**

I. Introduction

Understanding how much information we need to learn about a number to answer questions about it is a fundamental problem in computer science. Imagine you pick a random number X uniformly between 0 and 1, but you don't know its exact value. Instead, you can reveal the digits of its binary representation one at a time. For example, if $X = 1/3$, its binary form is $0.010101...$, and each digit gives a clue about where X lies.

Each digit of X is like a fair coin toss — either 0 or 1 with equal chance. If you want to know whether X is less than some fixed number x, on average you need to look only at the two first bits of X to find out.

Now, consider two random numbers, X_1 and X_2, chosen independently and uniformly between 0 and 1. Our goal is to decide if $X_1 < X_2$ by revealing bits from either number, one by one, in any order we choose. The best way is to alternate between revealing bits from X_1 and X_2. With this strategy, we expect to reveal about 4 bits in total before finding out the answer.

We can generalize this question: given a known increasing function f, how many bits from X_1 and X_2 must we reveal to decide if $f(X_1) < X_2$ on average? This question is central to our work and has important applications in a classic technique called rejection sampling.

A. Rejection sampling

The challenge of generating random numbers has been addressed by many, including Devroye [2], and remains an active topic of interest in computer science. Various methods based on quantum computing [3] and spin electronics [4] have been recently developed and have many applications in statistics, physics, finance, etc.

Rejection sampling is a popular method for generating random samples from given probability distributions. First introduced by John von Neumann in 1951 [1], it is widely used in many fields including machine learning [5] [6], cryptography [7], and numerical analysis.

Here is how it works: given a function f that corresponds to a probability density for the points in $[0,1)^n$, we repeatedly pick random points $(X_1, \ldots, X_n)$ and X_{n+1} uniformly and independently in $[0,1)^n$ and $[0,1)$, respectively. We keep doing this until $f(\mathbf{X})$ is less than X_{n+1}. The point $\mathbf{X}$ we stop at then follows a distribution proportional to f.

In the following picture, we ran rejection sampling many times with $n = 1$. The points in blue are the ones that have been accepted and outputted because they lie under the function f. Notice that the values of $\mathbf{X}$ for which f is higher are outputted more often than those where f has low value, as highlighted by the two blue intervals.

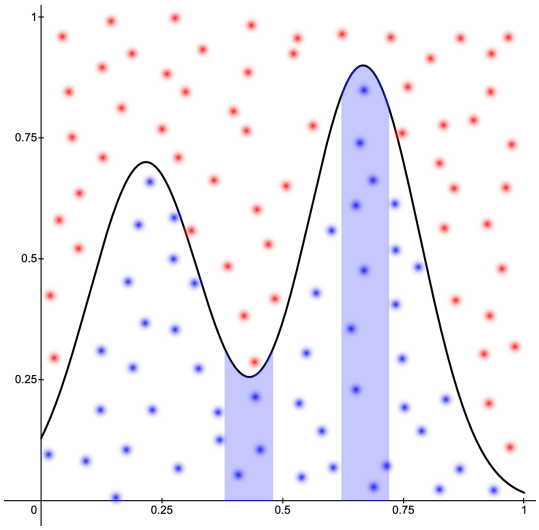

Fig. 1. Visual representation of rejection sampling with $n = 1$

When implementing rejection sampling on a computer, one challenge is to minimize how many bits of these numbers we need to reveal before knowing if $f(\mathbf{X}) < X_{n+1}$. This is the problem we focus on in this paper.

We assume we can compute $f(\mathbf{x})$ instantly for any input $f(\mathbf{x})$, so the challenge is to minimize queries to the bits of the X_j's.

We focus on the case where f is increasing: if every coordinate of $\mathbf{x}'$ is less than or equal to the corresponding coordinate of $\mathbf{x}$, then $f(\mathbf{x}') \leq f(\mathbf{x})$. This property lets us use just two points to decide if $f(\mathbf{X})$ is bigger or smaller than X_{n+1}, which makes the algorithm much more realistic to implement.

B. The setup

Given a number $x \in [0,1)$, let $x^{(i)} \in \{0,1\}$ be the i'th bit of its binary expansion, i.e. $x = 0.x^{(1)}x^{(2)}x^{(3)}\ldots$. We assume that

979-8-3315-5937-3/25/$31.00 ©2025 IEEE

$(x^{(i)})_{i\geq 1}$ does not end with a tail of repeating 1's. This way, we have a bijection between $[0,1)$ and all such binary sequences. Finally, denote $\mathcal{F}_n$ to be the set of all increasing functions from $[0,1]^n$ to $[0,1)$.

To represent the algorithms we are studying, we will first consider deterministic inputs rather than random variables. We introduce the alphabet $\{\bar{x}_1,\ldots,\bar{x}_{n+1}\}$, where $\bar{x}_1,\ldots,\bar{x}_{n+1}$ are symbols that we call *cuts*. An algorithm takes in a point $(\mathbf{x},x_{n+1})\in[0,1)^{n+1}$ and an increasing function $f\in\mathcal{F}_n$, and outputs a string in $\{\bar{x}_1,\ldots,\bar{x}_{n+1}\}^*$ (possibly of infinite length) that represents the exact sequence of bits that have been revealed. For example, if the output of the algorithm is $\bar{x}_1\bar{x}_2\bar{x}_1\bar{x}_{n+1}\bar{x}_1$, it means that the bits $x_1^{(1)},x_2^{(1)},x_1^{(2)},x_1^{(n+1)},x_1^{(3)}$ have been revealed in this exact order.

Such an algorithm is thus a function

$$A:\mathcal{F}_n\times[0,1)^{n+1}\to\{\bar{x}_1,\ldots,\bar{x}_{n+1}\}^*\ ,$$

with certain restrictions. Let $p_1,\ldots,p_{n+1}\geq 1$ be integers and $f\in\mathcal{F}_n$. If $(\mathbf{x},x_{n+1}),(\mathbf{x}',x'_{n+1})\in[0,1)^n$ are such that $x_j^{(i)}=x_j'^{(i)}$ for all $1\leq i\leq p_j, 1\leq j\leq n+1$, then if s is a prefix of $A(f,\mathbf{x},x_{n+1})$ that contains $\bar{x}^{(j)}$ at most p_j times for all $1\leq j\leq n+1$, then any prefixes of lengths $|s|+1$ or less of $A(f,\mathbf{x},x_{n+1})$ and $A(f,\mathbf{x}',x'_{n+1})$ must be identical. This restriction ensures that the values of the bits that are not revealed cannot have an impact on which bit is queried next.

The last restriction is that given $(\mathbf{x},x_{n+1})\in[0,1)^{n+1}$ and $f\in\mathcal{F}_n$, the string of revealed bits outputted by these algorithms must provide sufficient information to determine whether $f(\mathbf{x})<x_{n+1}$ or not. Let s be a string of cuts, k_i be the number of $x^{(i)}$'s in s, and $x_i^*=0.x_i^{(1)}x_i^{(2)}\ldots x^{(k_i)}$, i.e. x_i^* is the number consisting of the first k_i bits of x_i. After revealing its first k_i bits, x_i could take any value in $[x_i^*,x_i^*+2^{-k_i})$. We thus have that $f(\mathbf{x})<x_{n+1}$ almost surely[1] if and only if $f(x_1^*+2^{-k_1},\ldots,x_n^*+2^{-k_n})<x_{n+1}^*$, and $f(\mathbf{x})>x_{n+1}$ almost surely if and only if $f(\mathbf{x}^*)>x_{n+1}^*+2^{-k_{n+1}}$. Note that $f(\mathbf{x})\neq x_{n+1}$ almost surely.

Let us call $S=[\mathbf{x}_1^*,\mathbf{x}_1^*+2^{-k_1})\times\ldots\times[\mathbf{x}_{n+1}^*,\mathbf{x}_{n+1}^*+2^{-k_{n+1}})$ the feasible set of A after string s, i.e. the set of possible values of $(\mathbf{x},x_{n+1})$. If s does not satisfy any of the two halting conditions mentioned in the previous paragraph, then the algorithm cannot halt after outputting s. In this case, we say that f *crosses* S, because this means that $S\cap\{(\mathbf{x},x_{n+1}):f(\mathbf{x})<x_{n+1}\}$ and $S\cap\{(\mathbf{x},x_{n+1}):f(\mathbf{x})>x_{n+1}\}$ both have positive measure.

C. The problem and our results

The goal of this paper is to determine which algorithms query the lowest total number of bits from $X_1,\ldots,X_{n+1}$ in the worst case, on average. Formally, letting $A(f)=A(f,X_1,\ldots,X_{n+1})$, we want to study the value and the asymptotic behavior of

$$\mathcal{B}(n)=\inf_{A\in\mathcal{A}_n}\sup_{f\in\mathcal{F}_n}\mathbb{E}|A(f)|$$

and find good algorithms that reach this value for all $n\geq 1$. We also aim to find what functions maximize the expected number of bits these algorithms reveal, i.e. the worst case functions.

In section II, we find a general upper bound on the value of $\mathcal{B}(n)$. We prove in Theorem 1 that this bound grows at most quadratically with n, i.e. that $\mathcal{B}(n)$ is $O(n^2)$. In Section III, we show that $B(n)\geq n+1$, meaning that $B(n)$ is $\Omega(n)$.

[1]Here "almost surely" means that the set of values of $(\mathbf{x},x_{n+1})$ that makes the equation false has measure 0.

D. Preliminary results

Throughout this paper, we will use the fact that if $N\geq 0$ is an integer valued random variable, then

$$\mathbb{E}[N]=\sum_{\ell\geq 0}\mathbb{P}(N>\ell). \tag{1}$$

This can be seen after writing $N=\sum_{\ell\geq 0}\mathbf{1}_{N>\ell}$ and using linearity of expectation on the right hand side.

Next, let $x\geq 0$ be any real number and $n\geq 1$ be an integer. Then,

$$1-(1-x)^n\leq nx \tag{2}$$

with equality if and only if $x=0$. To see this, take the derivatives on both sides to obtain $n(1-x)^{n-1}$ and n, respectively. For every $x>0$, the derivative of the right hand side is strictly bigger, which gives us the inequality.

II. An upper bound for $\mathcal{B}(n)$

First, we bound $\mathcal{B}(n)$ above by constructing the naive alternating algorithm ALT_n defined as follows. $ALT_n(f,\mathbf{x},x_{n+1})$ is equal to the shortest possible prefix of $(\bar{x}_1\bar{x}_2\ldots\bar{x}_{n+1})^*$ that allows the algorithm to halt (of infinite length if and only if $f(\mathbf{x})=x_{n+1}$). Clearly the two conditions for ALT_n to be in $\mathcal{A}_n$ are satisfied. We start the analysis of the efficiency of ALT_n with the following preliminary result.

Proposition 1. *Let $M_1,\ldots,M_{n+1}\geq 1$ be integers and let $f:[0,M_1)\times[0,M_n)\to[0,M_{n+1})$ be an increasing function. For $\mathbf{z}=(z_1,\ldots,z_{n+1})\in\{1,2,\ldots\}^{n+1}$ with $1\leq z_j\leq M_j$, let $\mathbf{I_z}=[z_1-1,z_1)\times\ldots\times[z_{n+1}-1,z_{n+1})$. Then, the number of such $\mathbf{I_z}$'s the function f crosses is at most $\prod_{j=1}^{n+1}M_j-\prod_{j=1}^{n+1}(M_j-1)$.*

Proof. Let $\mathcal{Z}$ be the set of all $\mathbf{I}_{\mathbf{z}^*}$'s such that $z_{n+1}^*=M_{n+1}$ or $z_i^*=1$ for some $1\leq i\leq n$. For each $\mathbf{I}_{\mathbf{z}^*}\in\mathcal{Z}$, let $\mathcal{I}_{\mathbf{z}^*}$ be the set of all $\mathbf{I_z}$ such that there exists an $m\geq 0$ with $z_{n+1}=z_{n+1}^*-m$ and $z_i=z_i^*+m$ for all $1\leq i\leq n$.

We show that f cannot cross more than one rectangle in any $\mathcal{I}_{\mathbf{z}^*}$. Let $\mathbf{I_z},\mathbf{I}_{\mathbf{z}'}\in\mathcal{I}_{\mathbf{z}^*}$ and assume that $z_{n+1}\leq z'_{n+1}-1$. If f crosses $\mathbf{I_z}$, then $f(z_1-1,\ldots,z_n-1)<z_{n+1}$, which implies f does not cross $\mathbf{I}_{\mathbf{z}'}$ since $z'_i\leq z_i-1$ for all $1\leq i\leq n$, and thus $f(z'_1,\ldots,z'_n)<z'_{n+1}-1$ since f is increasing.

Finally, notice that the total number of $\mathbf{I_z}$'s is $\prod_{j=1}^{n+1}M_j$, and the number of $\mathbf{I_z}$'s that are not in $\mathcal{Z}$ is $\prod_{j=1}^{n+1}(M_j-1)$. Furthermore, every $\mathbf{I_z}$ is contained in some $\mathcal{I}_{\mathbf{z}^*}$, and since at most one $\mathbf{I_z}$ per $\mathcal{I}_{\mathbf{z}^*}$ can be crossed by f, this implies that at most $\prod_{j=1}^{n+1}M_j-\prod_{j=1}^{n+1}(M_j-1)$ of the $\mathbf{I_z}$'s can be crossed by f in total, i.e. at most one for each $\mathcal{I}_{\mathbf{z}^*}$. □

Theorem 1. *Let $f:[0,1)^n\to[0,1)$ be any increasing function. Then,*

$$\mathbb{E}\left|ALT_n(f)\right|\leq\sum_{k\geq 0}\sum_{i=0}^{n}\left(1-\frac{(2^{k+1}-1)^i(2^k-1)^{n+1-i}}{2^{k(n+1)+i}}\right).$$

In particular, $\mathcal{B}(n)=O(n^2)$.

Proof. Let $0\leq i\leq n$ and $k\geq 0$ be integers. By observing the algorithm, we see that if ALT_n has not halted after $k(n+1)+i$ queries, then its current string contains $k+1$ occurrences of each of $\bar{x}_1,\ldots,\bar{x}_i$, and k occurrences of each of $\bar{x}_{i+1},\ldots,\bar{x}_n$).

Let $1\leq m_j\leq 2^{k+1}$ be integers for $1\leq j\leq i$ and let $1\leq m_j\leq 2^k$ be integers for $i<j\leq n+1$. Then, let $\mathbf{I_m}=\left[2^{-k-1}(m_1-1),2^{-k-1}m_1\right)\times\ldots\times$

$\left[2^{-k-1}(m_{n+1}-1), 2^{-k-1}m_{n+1}\right)$. Since $X_1, \ldots, X_{n+1}$ are uniform on $[0,1)$, every $\mathbf{I_m}$ is equally likely to contain $(X_1, \ldots, X_{n+1})$. Applying Proposition 1 on a rescaling of $[0,1)^{n+1}$ gives us that at most $2^{k(n+1)+i} - (2^{k+1}-1)^i(2^k-1)^{n+1-i}$ of the $\mathbf{I_m}$ can be crossed by f. Since the feasible set of ALT_n must be equal to some $\mathbf{I_m}$ after $k(n+1)+i$ cuts, this means the probability that f still crosses the feasible set after $k(n+1)+i$ cuts is at most $[2^{k(n+1)+i} - (2^{k+1}-1)^i(2^k-1)^{n+1-i}] \cdot \mathbb{P}(\mathbf{I_m})$. Since $\mathbb{P}(\mathbf{I_m}) = 2^{-k(n+1)-i}$, this means that

$$\mathbb{P}(|ALT_n(f)| > k(n+1)+i) = 1 - \frac{(2^{k+1}-1)^i(2^k-1)^{n+1-i}}{2^{k(n+1)+i}},$$

thus equation 1 gives us the upper bound we are looking for.

To see that $\mathcal{B}(n) = O(n^2)$, note that since $\mathbb{P}(|ALT_n(f)| > \ell)$ decreases when ℓ increases, then we can naively bound the right hand side of this Theorem's inequality with

$$\sum_{k\geq 0}\sum_{i=0}^{n}\left(1 - \frac{(2^k-1)^{n+1}}{2^{k(n+1)}}\right). \tag{3}$$

The terms inside of the double sum can be written as $1-(1-2^{-k})^{n+1}$. We can now apply equation 2 to bound this above with $(n+1)2^{-k}$. In the end, this gives us an upper bound of

$$\sum_{k\geq 0}(n+1)^2 2^{-k} = 2(n+1)^2. \tag{4}$$

□

When we plot original theorem's equation in Desmos and compare it with $2(n+1)^2$, we notice a clear gap between them. This could be due either to the bound in equation 3 or the bound in equation 4. After plotting the three, it looks like equation 3 is very tight and may only alter the upper bound by a constant factor. However, equation 4 seems to make the bound go from a function that seems empirically close to $\Theta(n^{1.175})$ to a quadratic one.

Here are the plots we are referring to. In red, we have the first Theorem's inequality. In green, we have the bound from equation 3. In purple, we have the bound from equation 4. Both the green and the red functions look like they follow asymptotic behaviors of order $\Theta(n^{1.175})$. This can be seen by changing the value of b to match the function.

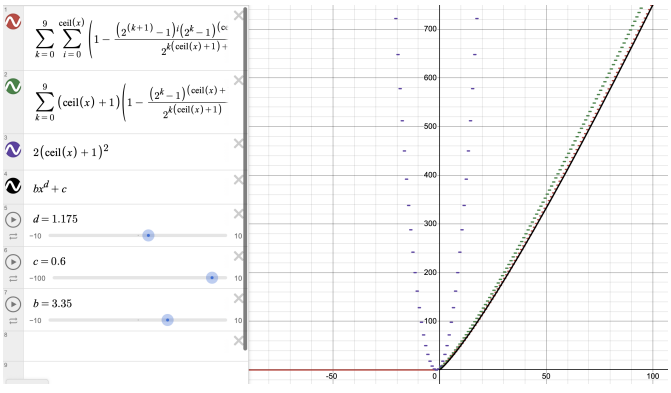

Fig. 2. Graphical representation of the different upper bounds for $\mathcal{B}(n)$.

This leads us to believe that the upper bound can be drastically improved if we find an alternative bound to equation 4.

III. A LOWER BOUND FOR $\mathcal{B}(n)$

In this section, we now prove that $\mathcal{B}(n) \geq n+1$, letting us conclude that $\mathcal{B}(n) = \Omega(n)$. We denote $h_n : [0,1)^n \to [0,1)$ to be the function in $\mathcal{F}_n$ defined as $h_n(\mathbf{x}) = \min\left(1-2^{-(n+1)}, (2^{n+1}-1)\min_{1\leq i\leq n} x_i\right)$. Plots of h_n are shown in the following figure for $n=1$ and $n=2$:

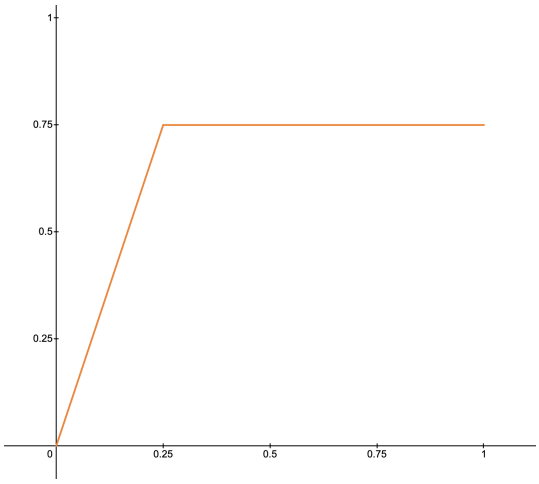

Fig. 3. Graphical representation of h_1.

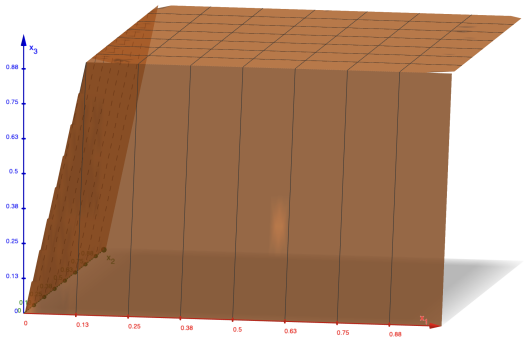

Fig. 4. Graphical representation of h_2.

We use h_n to prove our lower bound in the following Theorem.

Theorem 2. *For any algorithm $A \in \mathcal{A}_n$ and any $(\mathbf{x}, x_{n+1}) \in [0,1)^{n+1}$, $|A(h_n, \mathbf{x}, x_{n+1})| \geq n+1$. In particular, $\mathcal{B}(n) \geq n+1$.*

Proof. Let $A \in \mathcal{A}_n$ be any algorithm and let s be a prefix of length less than n of $A(f, \mathbf{x}, x_{n+1})$. By the pigeonhole principle, there must be one of $\bar{x}^{(1)}, \ldots, \bar{x}^{(n+1)}$ that isn't included in s. Let S be the current feasible set of A. If $\bar{x}^{(n+1)} \notin s$, then S contains points with x_{n+1}-value 0 and points with x_{n+1}-value bigger than $1-2^{-(n+1)}$. Since $0 < h_n \leq 1-2^{-(n+1)}$ almost everywhere, this implies that f crosses S, meaning that A cannot halt.

Now, if $\bar{x}^{(n+1)}$ is included in s, then choose $1 \leq i \leq n$ such that $\bar{x}^{(i)}$ is not included in s. This means that S contains a point $(\mathbf{x}, x_{n+1})$ with $x_i = 0$ and $x_{n+1} > 0$, thus $h_n(\mathbf{x}) < x_{n+1}$ since $h_n(\mathbf{x}) = 0$ by observation. Furthermore, since s contains at most n cuts, then S must contain a point $(\mathbf{x}', x'_{n+1})$ such that $x'_i > 2^{-(n+1)}$ for all $1 \leq i \leq n$ and $x'_{n+1} < 1-2^{-(n+1)}$. Thus, $h_n(\mathbf{x}') > x_{n+1}$ since $h_n(\mathbf{x}') = 1-2^{-(n+1)}$ by observation. The existence of $\mathbf{x}$ and $\mathbf{x}'$ implies that h_n strictly crosses S, and so that A cannot halt. Since

s is an arbitrary string of length at most n, this implies that A needs at least $n+1$ cuts to halt, i.e. $|A(h_n, \mathbf{x}, x_{n+1})| \geq n+1$ for any $(\mathbf{x}, x_{n+1}) \in [0,1)^{n+1}$, proving the desired result.

We can use this deterministic result and replace $\mathbf{x}$ and x_{n+1} by $\mathbf{X}$ and X_{n+1}, respectively, to get that $A(h_n) \geq n+1$. Taking the supremum over any function $f \in \mathcal{F}_n$ gives that $\sup_{f \in \mathcal{F}_n} A(f) \geq n+1$, and since this inequality is true for any algorithm $A \in \mathcal{A}_n$, we can conclude that $\mathcal{B}(n) \geq n+1$. □

IV. Conclusion

To generate random numbers that follow a given distribution on $[0,1)^n$, we use rejection sampling driven by sequences of independent random bits. We show that the expected total number of bits that must be revealed to implement this method grows at least linearly and at most quadratically in n in the worst case.

Closing the gap between these lower and upper bounds remains an open problem. In particular, we conjecture that both the lower and upper bounds can be improved. It would also be interesting to determine the exact value of $\mathcal{B}(n)$ for small values of n.

Acknowledgment

The authors would like to thank Luc Devroye from McGill university for bringing this problem to their attention and suggesting to write a paper about it if results were found as well as for providing feedback and suggestions that improved the writing of this paper. The authors would also like to thank Jun Kai Liao from McGill university for actively participating in the early stages of this research project.

References

[1] J. von Neumann. *Various techniques used in connection with random digits.* National Bureau of Standards, Applied Math Series, 12:36–38, (1951)

[2] L. Devroye. *Non-uniform random variate generation*, Springer, New York, (1986)

[3] M. Collantes, J. Escartin, *Quantum random number generators*, Reviews of Modern Physics 89, 015004 (2017)

[4] A. Dubovskiy, T. Criss, A. Valli, L. Rehm, A. Kent, A. Haas, *One trillion true random bits generated with a field programmable gate array actuated magnetic tunnel junction*, IEEE Magnetics Letters (2024)

[5] K. Keith, S. Feldman, D. Jurgens, J. Bragg, R. Bhattacharya, *RCT Rejection Sampling for Causal Estimation Evaluation*, Transactions on Machine Learning Research (TMLR) (2023)

[6] W. Xiong, J. Yao, Y. Xu, B. Pang, L. Wang, D. Sahoo, J. Li, N. Jiang, T. Zhang, C. Xiong, H. Dong, *A Minimalist Approach to LLM Reasoning: from Rejection Sampling to Reinforce*, arXiv:2504.11343 (2025)

[7] J. Awan, V. Rao, *Privacy-Aware Rejection Sampling*, arXiv:2108.00965 (2021)