Requests and Open Questions

This version of the document is dated 2021-09-12.

Peter Occil

This page lists questions and issues relating to my articles posted on this site. Any answers to these questions will greatly improve those articles. If you can answer any of them, post an issue in the **GitHub issues page**.

1 Contents

- **Contents**
- **Randomization and Sampling Methods**
- **Bernoulli Factory Algorithms**
- **Partially-Sampled Random Numbers for Accurate Sampling of Continuous Distributions**
- **More Algorithms for Arbitrary-Precision Sampling**
- **Randomized Estimation Algorithms**
- **Color Topics for Programmers**
- **Notes**
- **License**

2 Randomization and Sampling Methods

Size Reduction Sought:

Of the articles in this repository, **Randomization and Sampling Methods** and **More Random Sampling Methods** combined are very long (about 230 KB in size combined).

These articles describe numerous algorithms to generate random variates (from discrete and continuous distributions) as well as perform random sampling with and without replacement, shuffling, geometric sampling, and more, assuming a source of "truly" random numbers is available.

I would like to reduce the size of these articles while maintaining the most relevant algorithms for random variate generation.

Here are my goals for both articles:

- To shorten the **Randomization with Real Numbers** section as much as possible, while still describing the most general (and exact) algorithms possible for sampling real numbers of any distribution.
- To put emphasis on algorithms that work with random integers (or, if necessary, rational numbers), rather than random floating-point numbers.
- To put emphasis on algorithms that sample a distribution *exactly*, or at least with a controlled upper bound on the error. For discussion, see "**Exact, Error-Bounded, and Approximate Algorithms**".
- To ensure the documents are easy for programmers to understand and implement.

3 Bernoulli Factory Algorithms

https://peteroupc.github.io/bernoulli.html

This is a page showing algorithms to turn a coin with an unknown probability of heads into a coin with a different probability of heads, also known as *Bernoulli factories*. A *factory function* is a function that relates the old probability to the new one. Roughly speaking, a function can be a factory function only if it is the constant 0 or 1, or if it is continuous on its domain and equals neither 0 nor 1 on the open interval (0, 1) (Keane and O'Brien 1994)[(1)].

Attention is drawn to the requests and open questions on that page:

- **https://peteroupc.github.io/bernoulli.html#Requests_and_Open_Questions**

Among other things, they relate to finding polynomial sequences, probabilities, and other mathematical constructions needed to apply certain Bernoulli factories. These questions are reproduced below.

1. What simulations exist that are "relatively simple" and succeed with an irrational probability between 0 and 1? What about "relatively simple" Bernoulli factory algorithms for factory functions? Here, "relatively simple" means that the algorithm:

    - Should use only uniform random integers (or bits) and integer arithmetic.
    - Does not use floating-point arithmetic or make direct use of square root or transcendental functions.
    - Does not calculate base-n expansions directly.
    - Should not use rational arithmetic or increasingly complex approximations, except as a last resort.

    See also Flajolet et al. (2010)[(2)]. There are many ways to describe the irrational probability or factory function. I seek references to papers or books that describe irrational constants or factory functions in any of the following ways:

    - For irrational constants:
        - Simple **continued fraction** expansions.
        - Closed shapes inside the unit square whose area is an irrational number. (Includes algorithms that tell whether a box lies inside, outside, or partly inside or outside the shape.) **Example.**
        - Generate a uniform (x, y) point inside a closed shape, then return 1 with probability x. For what shapes is the expected value of x an irrational number? **Example.**
        - Functions that map [0, 1] to [0, 1] whose integral (area under curve) is an irrational number.
    - For Bernoulli factory functions:
        - Functions with any of the following series expansions, using rational arithmetic only:
            - Power series where $f(0)$ is 0 and $f(1)$ is rational or vice versa (see "**Certain Power Series**").
            - Series with non-negative terms that can be "tucked" under a discrete probability mass function (see "**Convex Combinations**").
            - Alternating power series whose coefficients are all in the interval [0, 1] and form a nonincreasing sequence (see "**Certain Power Series**").
            - Series with non-negative terms and bounds on the truncation error (see "**Certain Converging Series**").

- A way to compute two sequences of polynomials written in Bernstein form that converge from above and below to a factory function as follows: (a) Each sequence's polynomials must have coefficients lying in [0, 1], and be of increasing degree; (b) the degree-n polynomials' coefficients must lie at or "inside" those of the previous upper polynomial and the previous lower one (once the polynomials are elevated to degree n). For a formal statement of these polynomials, see my **question on Mathematics Stack Exchange**.

  The **supplemental notes** include formulas for computing these polynomials for large classes of factory functions, but none of them ensure a finite expected number of coin flips in general, and it is suspected that a finite number of flips isn't possible unless the factory function is C^2 continuous (has two or more continuous "slope" functions). Thus one question is: Given a C^2 continuous factory function, are there practical algorithms for building polynomials that converge to that function in a manner needed for the Bernoulli factory problem, where the expected number of coin flips is finite (besides the algorithms in this article or the supplemental notes)?

2. Let a permutation class (such as numbers in descending order) and two continuous probability distributions D and E be given. Consider the following algorithm: Generate a sequence of independent random numbers (where the first is distributed as D and the rest as E) until the sequence no longer follows the permutation class, then return n, which is how many numbers were generated this way, minus 1. In this case:

   1. What is the probability that n is returned?
   2. What is the probability that n is odd or even or belongs to a certain class of numbers?
   3. What is the distribution function (CDF) of the first generated number given that n is odd, or that n is even?

   Obviously, these answers depend on the specific permutation class and/or distributions D and E. Thus, answers that work only for particular classes and/or distributions are welcome. See also my Stack Exchange question **Probabilities arising from permutations**.

3. Is there a simpler or faster way to implement the base-2 or natural logarithm of binomial coefficients? See the example in the section "**Certain Converging Series**".

4. Part of the reverse-time martingale algorithm of Łatuszyński et al. (2009/2011)**(3)** (see "**General Factory Functions**") to simulate a factory function $f(\lambda)$ is as follows. For each n starting with 1:

   1. Flip the input coin, and compute the n^{th} upper and lower bounds of f given the number of heads so far, call them L and U.
   2. Compute the $(n-1)^{\text{th}}$ upper and lower bounds of f given the number of heads so far, call them L' and U'. (These bounds must be the same regardless of the outcomes of future coin flips, and the interval $[L', U']$ must equal or entirely contain the interval $[L, U]$.)

   These parts of the algorithm appear to work for any two sequences of functions (not just polynomials) that converge to f, where L or L' and U or U' are their lower and upper bound approximations. The section on general factory functions shows how

this algorithm can be implemented for polynomials. But how do these steps work when the approximating functions (the functions that converge to f) are rational functions whose coefficients are integers? Rational functions whose coefficients are rational numbers? Arbitrary approximating functions?

5. A *pushdown automaton* is a state machine that holds a stack of symbols. Mossel and Peres (2005)[4] investigated which functions ($f(\lambda)$) can be simulated by these machines when they're given an infinite "tape" of flips of a coin that shows heads with probability λ. They showed that pushdown automata can simulate only *algebraic functions*, but perhaps not all of them. The question is: What is the exact class of algebraic functions a pushdown automaton can simulate? Can it simulate the functions $\min(\lambda, 1-\lambda)$ and λ^p where $p>2$ is a prime number? I have written an **article appendix** showing my progress, but are there other results on this question?

6. The following is an open question in Nacu and Peres 2005. Let J be a closed interval on (0, 1), such as [1/100, 99/100], and let $f(\lambda)$ be a function that admits a Bernoulli factory. Suppose there is an algorithm that takes a coin with unknown probability of heads λ and produces one or more samples of the probability $f(\lambda)$. When the probability λ can be any value in J, is it possible for this algorithm to have an expected number of input coin flips per sample that is arbitrarily close to the so-called *entropy bound*? The entropy bound is $h(f(\lambda))/h(\lambda)$ where $h(x) = -x*\ln(x) -(1-x)*\ln(1-x)$ is the Shannon entropy function. Does the answer change if the algorithm can also use a separate source of unbiased random bits? See my section "**Multiple-Output Bernoulli Factory**".

4 Partially-Sampled Random Numbers for Accurate Sampling of Continuous Distributions

https://peteroupc.github.io/exporand.html

A *partially-sampled random number* (PSRN) is a data structure holding the initial digits of a random number that is built up digit by digit. The following is an open question on PSRNs.

Doing an arithmetic operation between two PSRNs is akin to doing an interval operation between those PSRNs, since a PSRN is ultimately a random number that lies in an interval. However, as explained in "**Arithmetic and Comparisons with PSRNs**", the result of the operation is an interval that bounds a random number that is *not* always uniformly distributed in that interval. For example, in the case of addition this distribution is triangular with a peak in the middle, and in the case of multiplication this distribution resembles a trapezoid. What are the exact distributions of this kind for other interval arithmetic operations, such as division, ln, exp, sin, or other mathematical functions?

5 More Algorithms for Arbitrary-Precision Sampling

https://peteroupc.github.io/morealg.html

This page has more algorithms for sampling using partially-sampled random numbers, as well as more Bernoulli factory algorithms. The following are requests and open questions

for this article.

1. We would like to see new implementations of the following:
    - Algorithms that implement **InShape** for specific closed curves, specific closed surfaces, and specific signed distance functions. Recall that **InShape** determines whether a box lies inside, outside, or partly inside or outside a given curve or surface.
    - Descriptions of new arbitrary-precision algorithms that use the skeleton given in the section "Building an Arbitrary-Precision Sampler".

2. The **appendix** contains implementation notes for **InShape**, which determines whether a box is outside or partially or fully inside a shape. However, practical implementations of **InShape** will generally only be able to evaluate a shape pointwise. What are necessary and/or sufficient conditions that allow an implementation to correctly classify a box just by evaluating the shape pointwise? See also my related Stack Exchange question: **How can we check if an arbitrary shape covers a box (partially, fully, or not) if we can only evaluate the shape pointwise?**.

3. Take a polynomial $f(\lambda)$ of even degree n of the form choose$(n,n/2)*\lambda^{n/2}*(1-\lambda)^{n/2}*k$, where k is greater than 1 (thus all f's Bernstein coefficients are 0 except for the middle one, which equals k). Suppose $f(1/2)$ lies in the interval (0, 1). If we do the degree elevation, described in the **appendix**, enough times (at least r times), then f's Bernstein coefficients will all lie in [0, 1]. The question is: how many degree elevations are enough? A **MathOverflow answer** showed that r is at least $m = (n/f(1/2)^2)/(1-f(1/2)^2)$, but is it true that floor$(m)+1$ elevations are enough?

4. A ***finite-state generator*** is a finite-state machine that generates a real number's base-2 expansion such as 0.110101100..., driven by flips of a coin. A *pushdown generator* is a finite-state generator with a stack of memory. Both generators produce real numbers with a given probability distribution. For example, a generator with a loop that outputs 0 or 1 at an equal chance produces a *uniform distribution*. The following questions ask what kinds of distributions are possible with these generators.
    1. Of the probability distributions that a finite-state generator can generate, what is the exact class of:
        - *Discrete distributions* (those that cover a finite or countably infinite set of values)?
        - *Absolutely continuous distributions* (those with a probability density function such as the uniform or triangular distribution)?
        - *Singular distributions* (covering an uncountable but zero-volume set)?
        - Distributions with *continuous* density functions?
    2. Same question as 1, but for pushdown generators.
    3. Of the probability distributions that a pushdown generator can generate, what is the exact class of distributions with piecewise smooth density functions? (The answer is known for finite-state generators.)

6 Randomized Estimation Algorithms

https://peteroupc.github.io/estimation.html

Let X be a stream of random numbers and let $f(x)$ be a known continuous function.

1. Is there an algorithm, besides *Algorithm C* or *Algorithm F* in the article, that can find $\mathbf{E}[X]$ (or $f(\mathbf{E}[X])$) with either a high probability of a "small" absolute error or one of a "small" relative error, when the distribution of X is unbounded, and additional assumptions on the distribution of X apply, such as—

    - being unimodal (having one peak) and symmetric (mirrored on each side of the peak), and/or
    - following a geometric distribution, and/or
    - having decreasing or nonincreasing probabilities?

    Notice that merely having finite moments is not enough (Theorem 3.4, Kunsch et al.). Here, the accuracy tolerances for small error and high probability are user-specified.

2. How can *Algorithm D* or *Algorithm E* in the article be adapted to discontinuous functions g, so that the algorithm finds $g(\mathbf{E}[X])$ with either a high probability of a "small" absolute error or one of a "small" relative error at all points in [0, 1] except at a "negligible" area around g's discontinuities? Is it enough to replace g with a continuous function f that equals g everywhere except at that "negligible" area? Here, the accuracy tolerances for small error, high probability, and "negligible" area are user-specified.

3. Is it true that *Algorithm F* in the article remains valid when the sample size n is ceil(abs(M)/(δ*γ^k)), given that the stream's distribution is known to have a maximum k^{th} central absolute moment of M?

7 Color Topics for Programmers

https://peteroupc.github.io/colorgen.html

Should this document cover the following topics, and if so, how?

- The CAM02 color appearance model.
- Color rendering metrics for light sources, including color rendering index (CRI) and the metrics given in TM-30-15 by the Illuminating Engineering Society.

Does any of the following exist?

- A method for performing color calibration and color matching using a smartphone's camera and, possibly, a color calibration card and/or white balance card, provided that method is not covered by any active patents or pending patent applications.
- Reference source code for a method to match a desired color on paper given spectral reflectance curves of the paper and of the inks being used in various concentrations, provided that method is not covered by any active patents or pending patent applications.

8 Notes

- (1) Keane, M. S., and O'Brien, G. L., "A Bernoulli factory", ACM Transactions on Modeling and Computer Simulation 4(2), 1994.
- (2) Flajolet, P., Pelletier, M., Soria, M., "**On Buffon machines and numbers**", arXiv:0906.5560 [math.PR], 2010.
- (3) Łatuszyński, K., Kosmidis, I., Papaspiliopoulos, O., Roberts, G.O., "**Simulating events of unknown probabilities via reverse time martingales**", arXiv:0907.4018v2 [stat.CO], 2009/2011.

- [4] Mossel, Elchanan, and Yuval Peres. New coins from old: computing with unknown bias. Combinatorica, 25(6), pp.707-724.

9 License