A Practical Implementation of the Bernoulli Factory

Anonymous Author 1
Unknown Institution 1

Anonymous Author 2
Unknown Institution 2

Abstract

The Bernoulli Factory is an algorithm that takes as input a series of i.i.d. Bernoulli random variables with an unknown but fixed success probability p, and outputs a corresponding series of Bernoulli random variables with success probability $f(p)$, where the function f is known and defined on the interval $[0, 1]$. While several practical uses of the method have been proposed in Monte Carlo applications, these require an implementation framework that is flexible, general and efficient. We present such a framework for functions that are either strictly linear, concave, or convex on the unit interval using a series of envelope functions defined through a cascade, and show that this method not only greatly reduces the number of input bits needed in practice compared to other currently proposed solutions for more specific problems, and is easy to specify for simple forms, but can be coupled to asymptotically efficient methods to allow for theoretically strong results.

1 INTRODUCTION

First made explicit by Keane and O'Brien [1994], a Bernoulli Factory is defined as an algorithm that takes as its input an i.i.d. sequence of Bernoulli random variables with unknown success probability – call this p_{in} – and outputs a new sequence of Bernoulli random variables whose success probability, p_{out}, is a known function of the input probability. A Bernoulli Factory does not use any approximation for either p_{in} or p_{out}, instead obtaining output draws through a stochastic process with two absorbing states, one of which has terminal probability p_{out}. The specification of the "factory function", by which $p_{out} = f(p_{in})$, is what makes this problem of general interest; an efficient Bernoulli Factory is important to applications of perfect sampling, particularly for taking direct draws from posterior distributions previously approximated by Markov Chain Monte Carlo [Asmussen et al., 1992; Hobert and Robert, 2004; Blanchet and Meng, 2005; Hobert et al., 2006; Blanchet and Thomas, 2007; Flegal and Herbei, 2012].

The main difficulty is creating a general algorithm that is both implementable and effective with various families of factory functions. While previous implementations have "fast" asymptotic convergence properties under particular conditions [Nacu and Peres, 2005], and are special cases of converging super- and sub-Martingale sequences [Latuszynski et al., 2011], they are often functionally inefficient at producing output bits for practical problems. Methods based on enveloping Bernstein polynomial approximations, which are naturally approximated by strings of random bits, are ideal for approximating either the upper or the lower bound of the function $f(p)$ if it is convex or concave respectively, as there is a natural nesting for successive approximations that is guaranteed by the method; as a consequence, the other limit is often extremely difficult to fit efficiently. We design our algorithm to better approximate the "non-natural" limit, through a series of functions that cascade toward this limit. This guarantees the natural nesting of envelopes on both sides of the function.

We develop this scheme for practical purposes first. While other methods have chosen functional forms that have guaranteed convergence properties in the infinite limit, we start with a different premise: specify a finite sequence that is simply specified for a reasonable applied finite limit – typically, the number of available bits, or the machine precision of the particular problem – and show that the method can produce a further sequence of envelopes that satisfy the compatibility equations of Nacu and Peres [2005], at a minimum by manual numerical checking of the conditions. While we can produce an infinite sequence of envelopes in

Preliminary work. Under review by AISTATS 2014. Do not distribute.

most cases, we can produce a sequence with known infinite theoretical limits by "grafting" the envelopes of another factory, whose convergence properties are known, onto the finite sequence of our method, so long as they are shown to be compatible.

We begin by reviewing the roots of the problem, before addressing the Bernstein polynomial approach in Section 2. We then address the implementation of the Cascade Bernoulli Factory method for piecewise-linear concave/convex functions in 3, then demonstrate its potential for more general functions in Section 4.

1.1 von Neumann's Fair Coin Maker

von Neumann [1951] generates a "fair coin", or a draw from a $Be(0.5)$ random variable, from an i.i.d. sequence of Bernoullis with unknown p, so that $f(p) = 0.5$ for all $0 < p < 1$. The corresponding stochastic process has three states, "yes", "no" or "continue". At each step, take two draws from the input sequence. Continue if the outcome is 00 or 11. If the outcome is 10, output "yes"; if the outcome is 01, output "no". Since each has probability $p(1-p)$ or occurring, the outcome can be likened to the flip of a fair coin. The running time of this method is two times a Geometric random variable with success probability $2p(1-p)$, or $\frac{1}{p(1-p)}$.

A similar process can be conducted to turn a series of fair coins into a coin with any success probability p_{out}, by noting that a uniform random variable can be produced through the representation $U = \sum_{i=1}^{\infty} 2^{-i} X_i$ where each $X_i \sim Be(0.5)$. However, a finite number of bits will be needed if the outcome of interest is specified as $Y = \mathbb{I}(U < p_{out})$.

The stochastic process has an unbounded number of states, but is as simple to specify as the standard von Neumann example. Beginning with $n = 1$:

- At each n, set $U_n = \sum_{i=1}^{n} 2^{-i} X_i$. Note that $U_n \leq U_{n+k}$ for all n and k.
- If $U_n > p$, then $U > p$ as well; output "no".
- If $U_n < p - 2^{-n}$, then no matter what the remaining inputs are, $U < p$; output "yes".
- Otherwise, add one more digit to the expansion and repeat the previous three steps.

These algorithms each converge in geometric time, with rate proportional to the target probability p_{out}. These methods suggestmethods that produce absorbing states of simple Markov chains can transform random bits without loss of information, merely efficiency.

1.2 From Simple Alchemy to the Full Factory

The problem explored by Keane and O'Brien [1994] works on the principle that the input and output success probabilities are possibly unknown, but a function that defines their connection is fully specified. The case where

$$f(p) = \min(cp, 1-\epsilon), \; c > 1, \; \epsilon < 1,$$

henceforth referred to as the "elbow function", is of particular interest to applications in exact sampling of Markov chains [Asmussen et al., 1992; Hobert and Robert, 2004; Blanchet and Meng, 2005; Hobert et al., 2006; Blanchet and Thomas, 2007; Flegal and Herbei, 2012], as this function represents a ratio in general rejection sampling schemes for draws from the stationary distributions of Markov Chains. (In the case where $c < 1$, the problem is trivial: the representation $X \sim Be(c) * Be(p)$ immediately produces the desired result.) The theoretical properties of this method have been described in detail by previous authors, though the algorithmic implementations of these solutions are concerned more with theoretical tractability than flexibility or applicability to real problems. Keane and O'Brien [1994] showed that a factory function must be continuous and satisfy the inequality

$$\min(f(p), 1 - f(p)) \geq \min(p, 1-p)^n$$

for some $n \geq 1$ and all $0 < p < 1$; Nacu and Peres [2005] established conditions for "fast" simulation, such that the required number of input bits decays exponentially, and proved that this held for all real analytic functions bounded away from 0 and 1.

2 BERNSTEIN EXPANSIONS AND SET APPROXIMATIONS

Bernstein polynomials are a set of basis functions defined on the interval $[0, 1]$. The standard Bernstein polynomial approximation to a function is defined as

$$f_n(p) = \sum_{k=0}^{n} \binom{n}{k} f\left(\frac{k}{n}\right) p^k (1-p)^{n-k};$$

their usefulness comes about in this problem because the probability of any one of the sequences of k ones and $n-k$ zeros from n $Be(p)$ random variables is $p^k(1-p)^{n-k}$. If these approximations both converge to the target function in the limit, then we can obtain a draw from the target distribution $Be(f(p))$ in finite time. It is trivial to show that the Bernstein polynomial approximation is the expected value of the function under a Binomial random variable $K \sim Bin(n, p)$;

$$Ef\left(\frac{K}{n}\right) = \sum_{k=0}^{n} \binom{n}{k} f\left(\frac{k}{n}\right) p^k (1-p)^{n-k} = f_n(p).$$

Using this result, a concave function will always be greater than any of its finite Bernstein polynomial expansions; by Jensen's inequality, $f(p) = f(EK/n) \geq Ef(K/n) = f_n(p)$. Conversely, a convex function will always be less than any of its Bernstein expansions.

The general method proposed by Nacu and Peres [2005] estimates $f(p)$ with two approximating functions, one from above, one from below, and ties these directly to the probabilities of observing particular bit strings. With a slight change in notation, consider the following formulation:

- Define a series of functions $a^n(p)$ that approximate the target function $f(p)$ from below (that is, for all n, $a^n(p) \leq f(p)$ and $\lim_{n\to\infty} a^n(p) = f(p)$). Define also another series of functions $b^n(p)$ to approximate $1 - f(p)$, also from below; by construction, $a^n(p) + b^n(p) \leq 1$. These functions are used when the total length of the input bit string is n.

- A_n is a set of bit-words of length n. each $A_{n,k}$ is the subset with exactly k ones; same for B_n.

- The Bernstein polynomial approximations $a_n^n(p)$ and $b_n^n(p)$ yield natural derivations for subsets $A_{n,k}$ and $B_{n,k}$. Since

$$a_n^n(p) = \sum_{k=0}^{n} \binom{n}{k} a^n\left(\frac{k}{n}\right) p^k (1-p)^{n-k},$$

we introduce

$$A_n^n(p) = \sum_{k=0}^{n} \left\lfloor \binom{n}{k} a^n\left(\frac{k}{n}\right) \right\rfloor p^k (1-p)^{n-k};$$

and define a set $A_{n,k}$ containing $\lfloor \binom{n}{k} a^n(\frac{k}{n}) \rfloor$ distinct $n-$length bit strings with k ones. From those bit strings that remain, choose $\lfloor \binom{n}{k} b^n(\frac{k}{n}) \rfloor$ to form $B_{n,k}$ (noting that the probability of observing any one bit-string is $p^k(1-p)^{(n-k)}$.)

- Ensure that $A_n^n(p) \leq f(p)$ and $B_n^n(p) \leq 1 - f(p)$. For whatever sequence of functions is used, also ensure that $A_{n+m,k} \geq \sum_{j=0}^{m} \binom{m}{j} A_{n,k-j}$, so that any lower-length bit string in A_n is in A_{n+m}.

- Collecting the remaining unaccounted items, define $C_{n,k}$ to be all those $n-$length bit strings with k ones that were not included in $A_{n,k}$ or $B_{n,k}$.

Using these tools we can now build the Bernoulli factory for a great number of classes of functions; details of the convergence properties of this method for various proposed envelope functions are addressed by Keane and O'Brien [1994] and Nacu and Peres [2005].

2.1 Example: $f(p)$ is Constant or Linear

With a linear factory function $f(p) = c + hp$, the standard Bernstein polynomial expansion is identical:

$$\begin{aligned} f_n(p) &= \sum_{k=0}^{n} \binom{n}{k} (c + h\frac{k}{n})\, p^k (1-p)^{n-k} \\ &= c + \frac{h}{n} \sum_{k=0}^{n} \binom{n}{k}\, k\, p^k (1-p)^{n-k} \\ &= c + \frac{h}{n} EK = c + \frac{h}{n} np = c + hp. \end{aligned}$$

As a result, the function can be used as both an upper and a lower envelope, so long as $0 < f(p) < 1$ for all $0 < p < 1$. This means that for those cases where $\binom{n}{k}$ $(c + h\frac{k}{n})$ is an integer, $C_{n,k}$ is empty, and the algorithm will terminate if k ones are observed. For the case when it is not an integer, there will be only one member of $C_{n,k}$.

Table 1 demonstrates that this is more efficient than the original von Neumann [1951] approach. In practice, it is not necessary to construct this table for all n, or to select a particular partitioning of all bit-strings, only to note the size of the sets themselves; for any string in A_n, all of its "descendants" are obtained by adding any m-length bit string are members of A_{n+m}, and similarly for B_n and B_{n+m}. This can always be done numerically; an analytical proof for convex/concave cases is found in the supplemental material.

3 THE CANONICAL CASE: $f(p)$ IS PIECEWISE LINEAR AND CONCAVE/CONVEX

The key motivating problem for the practical use of the Bernoulli factory is the aforementioned elbow function made famous by Keane and O'Brien [1994] and proposed in connection with Asmussen et al. [1992],

$$f(p) = \min(cp, 1-\epsilon),\ c > 1,\ \epsilon < 1,$$

which is concave. Due to Jensen's inequality, the Bernstein polynomial approximation to this function will always be less than the target function, so the lower bound function is simply $a^n(p) = f(p)$.

k,n=2	$A_{2,k}$	$B_{2,k}$	$C_{2,k}$
0			00
1	10	01	
2			11
k,n=4	$A_{4,k}$	$B_{4,k}$	$C_{4,k}$
0			0000
1	0010,1000	0001,0100	
2	**0011**,1010,1001	0101,0110,**1100**	
3	1110,1011	1101,0111	
4			1111

Table 1: The bit strings needed for a Bernoulli factory with function $f(p) = 0.5$ in the first two steps. This algorithm is slightly more efficient than von Neumann's algorithm for manufacturing fair coins.

The upper envelope function is considerably more difficult to design effectively. A single function cannot be used, as the Bernstein approximation to a concave function will increase as the length of the bit string increases. The envelope functions chosen by Nacu and Peres [2005] are functions of the bit-string n and are sufficient to prove the convergence properties of the algorithm under particular constraints, but are markedly inefficient at producing output draws; Flegal and Herbei [2012] show that a minimum of 2^{16} bits are required for the function $f(p) = \min(2p, 0.8)$. The capability of current computing hardware to efficiently calculate Bernstein expansions for functions above this count renders the algorithm not only inefficient in terms of input, but hopeless for practical implementation.

The alternative specification of Flegal and Herbei [2012] changes the problem slightly by specifying a new objective function that is twice-differentiable, but still linear on the domain $[0, (1-\epsilon)/c]$, as Nacu and Peres [2005] proved that certain twice-differentiable factory functions converge at a rate of $1/n$ (which yields an infinite expected time to termination, even though the termination time is finite.) The number of bits needed is considerably reduced from the original case, but still requires a minimum of many hundreds of bits to operate on these target functions.

This expansion does have nice theoretical convergence properties across the entire domain, but practical applications of this function typically take place when p is small (as demonstrated by Flegal and Herbei [2012]); the flattening-out of the function is to ensure continuity rather than as a particular range of interest. Because convergence at small p is typically the goal, we choose our sampler to visibly converge well in this range.

Rather than create a new functional form to add to the target function in creating an upper envelope, albeit one that would not noticeably affect the output of the algorithm for known inputs, our approach is to use the existing function to create a series of cascading envelopes that will converge to the target function from above (approaching $1 - f(p)$ from below). In particular, the manner in which the functions cascade is governed by their own Bernstein polynomial expansions, and the sequence can easily be constructed to converge to the target function $f(p)$ in the limit. (The rate at which this convergence occurs will depend on the chosen envelopes. If an asymptotically "fast" version exists, we prove that we may couple our finite sequence to this version in the supporting material.)

We require a series of "checkpoints" $\{m_1, m_2, ...\}$ at which the envelopes will be constructed and used. As Nacu and Peres [2005] point out, it is straightforward to define a partition $(A_{n+\Delta n}, B_{n+\Delta n}, C_{n+\Delta n})$ by starting with a partition (A_n, B_n, C_n) and adding all possible $2^{\Delta n}$ bit strings to each member of each set; this freedom allows us to minimize computation by choosing a smaller set of tests to conduct. The choice of checkpoints can be defined in any number of ways but may also be chosen to minimize the running time of the algorithm.

The method takes the following steps:

a) Choose a potential series of functions that converge toward $f(p)$. As shown in Figure 1, we select a series of elbow functions whose elbow points lie along a preset curve (two such curves are demonstrated).

   While any curve can be chosen that will produce a nested set of envelopes, the second curve, a fifth-degree polynomial whose slope at the elbow point matches the elbow function, fits the diagonal ascent of the function more tightly and will therefore be more efficient when the input bits have very small probability. For this function, higher-degree polynomials will flatten out the curve with respect to the elbow and force each Bernstein approximation to be closer to these small values.

b) Choose an initial elbow point along this curve, and initial bit-string length m_1. In this case it is simple to verify that $1 - b^{m_1}_{m_1}(p) > f(p)$ for all p by evaluating $b^{m_1}_{m_1}$ at the target function's elbow point $(1-\epsilon)/c$, as the Bernstein expansion's concavity ensures that we only need check the connecting points of the piecewise linear $f(p)$.

c) Retrieve m_1 bits from the input bit stream and set k_1 to equal the number of ones. Note the sizes

of each subset A_{m_1,k_1}, B_{m_1,k_1} and C_{m_1,k_1}. If desired, one can generate the actual corresponding bit strings, but this is unnecessary to run the algorithm itself.

d) If the bit string memberships of each groups have been specified exactly, note which of the subsets contains the observed string; if not, generate a trinomial random variable with probabilities proportional to $(|A_{m_1,k_1}|, |B_{m_1,k_1}|, |C_{m_1,k_1}|)$. Terminate the algorithm with output 1 or 0 if this trinomial is in each of the first two bins; If not, continue.

For all subsequent steps indexed by i:

e) Choose the next elbow point to be at (or below) the point where the previous Bernstein approximation $1 - b_{m_{i-1}}^{m_{i-1}}(p)$ intersects the elbow point curve; this ensures that the next envelope is less than or equal to the previous envelope's Bernstein approximation. Additionally, choose a value $m_i > m_{i-1}$ such that $1 - b_{m_i}^{m_i}(p) > f(p)$, and so that sufficient room is left for future iterations (since if the envelope is too close to the target, an extreme value of m will be needed to produce a Bernstein expansion greater than the target function.) This is the next upper envelope in the cascade; by construction, it is less than the previous envelope and produces a Bernstein expansion that is less than its precursor. Figure 1 contains two examples, one designed to approach the function relatively uniformly (a quadratic function) and one designed to hew close to the ascending line (a rotated fifth-degree polynomial.)

This particular method of descent is chosen for this function because it guarantees this nesting behavior while being easy to understand and implement; aside from this condition, there are no particular restrictions on the method by which envelopes are produced. A default descent function merely makes it more automated to the end user, though this can change depending on the desired convergence properties of the method. Figure 1 shows that a function that quickly approaches the ascending portion of the function will yield more early terminations, by closing the distance between functions; the downside is that if the algorithm does not terminate quickly, it will take a far greater number of bits to terminate.

f) Retrieve $m_i - m_{i-1}$ bits and add them to the current bit string; let the number of ones equal k_i. Calculate the sizes of sets A_{m_i,k_i}, B_{m_i,k_i} and C_{m_i,k_i} for k_i. For each element in $A_{m_{i-1},k_{i-1}}$, there are $\binom{m_i - m_{i-1}}{k_i - k_{i-1}}$ elements that are produced by adding $(m_i - m_{i-1})$-bit strings with $(k_i - k_{i-1})$

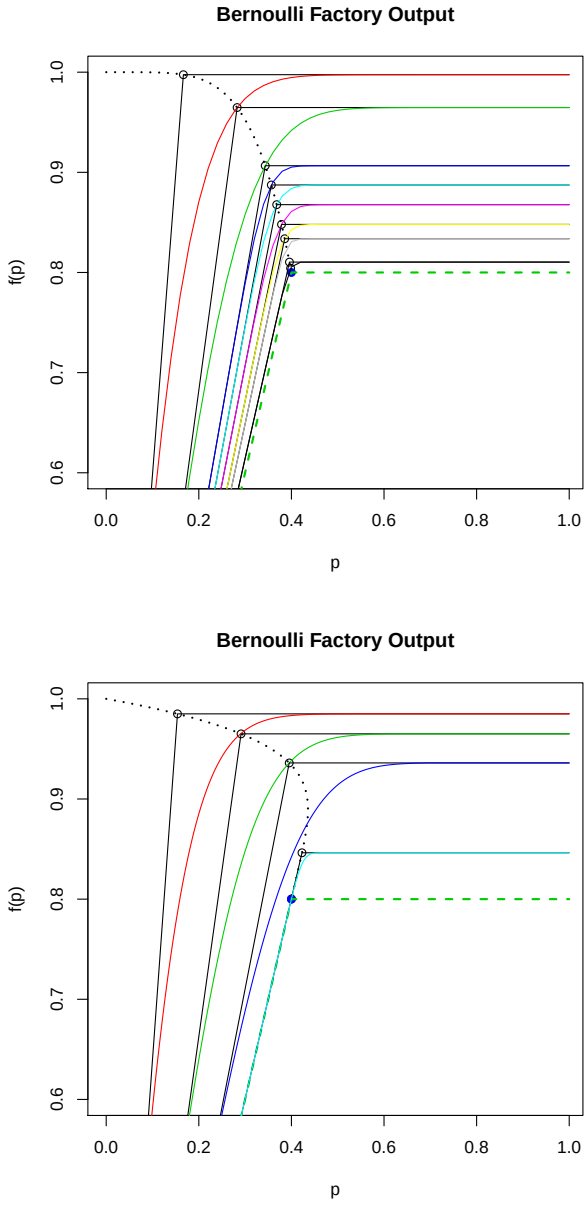

Figure 1: Two methods for generating upper envelope functions for $f(p) = \min(2p, 0.8)$; each successive function is defined by the intersection of a curve with the previous Bernstein approximation. Left, the elbow points are generated with a simple polynomial descent (dotted line), as each successive curve approaches the target function (dashed line). Right, the descent curve is a fifth-degree polynomial in y with respect to x, with derivative at the main elbow point matching that of the target function, designed to minimize the distance between the Bernstein polynomial and the target function for smaller values of p.

i	m_i	elbow (x,y)	0.02-P(1)	P(0)	P(cont.)
1	20	(0.15, 0.99)	1.5E-14	0.93	0.049
2	21	(0.29, 0.97)	1.5E-14	0.96	0.019
3	222	(0.40, 0.94)	8.6E-15	0.97	0.009
4	1223	(0.42, 0.85)	1.4E-16	0.979	1.6E-5

Table 2: Convergence characteristics for the execution of a Cascade Bernoulli Factory with function $f(p) = min(2p, 0.8)$ and $p_{in} = 0.01$. Because this function is concave, the lower curve approximation is far more efficient than the upper curve.

ones; since these would have produced termination in the prior step, remove these from A_{m_i,k_i}; similarly, remove the redundant descendants from B_{m_i,k_i}.

This is where the nesting condition on set sizes is explicitly checked. While it is the case that the envelope is less than the previous Bernstein approximation, it is unproven that the new resulting Bernstein expansion must have uniformly nested coefficients for all $k_i \in \{0, ..., m_i\}$ given all possible extensions of the previous bit string set. It is, however, the case that such a series can always be constructed using this method (see supporting material).

g) Generate a trinomial random variable with probabilities proportional to the set sizes,

$$Pr(1, 0, \text{continue}) \propto (|A_{m_i,k_i}|, |B_{m_1,k_i}|, |C_{m_1,k_i}|).$$

Terminate the algorithm with output 1 or 0 if this trinomial is in each of the first two bins, and repeat these steps if not for the next checkpoint m_{i+1}.

In general, these set sizes can all be pre-calculated for as many cascade steps i as is desired, and for all possible k; the enumeration of bit strings into sets is not required, since if the algorithm continues at each $i > 1$, it must be that the previous bit string belonged to group C.

3.1 Practical Justification and Improvement

For any envelope cascade, series of checkpoints, and an input probability value, one can calculate the marginal probability of each output bit's manifestation simply by adding up the total probability of all bit strings in each set (A_m, B_m, C_m); this is done on the graphs for demonstration purposes, but can also be produced in table form in order to gauge the running time of any particular envelope set.

Consider the envelope cascade from Figure 1, which has a target elbow function with $(c, \epsilon) = (2, 0.2)$, using the right-hand descent function. For a given p_{in}, the probabilities of belonging to each group at each stage are calculated and listed in Table 3. Note that these probabilities represent the ultimate termination probability of the algorithm, not the conditional probability of the outcome given that said number of steps was necessary. In this case, survival past the first round all but secures an eventual output of zero; the price paid for perfection of the algorithm is the high number of additional bits needed to guarantee that result.

A lower bound on the expected value for the number of bits required per output can be estimated with the standard formula $EN = \sum_{n=0}^{m_{max}} P(N > n)$, since the marginal probability of non-termination is equal to the size of C_m. Note that in this example, we have engineered a four-tiered envelope structure, and in the last tier shown we have greatly increased the number of input bits, as well as chosen a closer point to the target on the descent curve to increase the probability of termination within these steps; still, if the factory were to be run one million times, we can expect that the algorithm would not terminate at this point in 16 of them. If this is the case, the number of required bits at this point would skyrocket, as we have pinned ourselves into a situation where a vast increase in bits would be necessary to keep the algorithm valid; we call this "termination by gluttony" to distinguish from a standard exhaustion of bits due only to the main cascade.

This is as much a feature of design as any other. Optimizing this factory for practical use takes several directions:

- Does the user wish to optimize the method with respect to a single input probability, or across a range? The above table can be used for various input probabilities, though how to combine them appropriately will depend on the problem being considered. (See Section 4 for an example.)

- Can we use the input bits twice? For a particular factory function, if the bit string input probability is fixed but unknown, we can estimate $\widehat{p} = \frac{1}{N}\sum_{i=1}^{N} Y_i$ and choose an envelope cascade that quickly approaches this point in the function, since only the output speed is associated with the cascade, not the output probability.

- Given that the algorithm is not guaranteed to terminate at the final terrace, do we wish to minimize the probability of not terminating before the massive expense, at the cost of an increased number of bits in those cases that do terminate? This

c	Cascade min(bits)	Method E(bits)	sd(bits)
2	20	66	512
5	100	246	1215
10	200	614	1851
20	400	1410	3047
c	Best min(bits)	Alternative E(bits)	sd(bits)
2	256	562.9	2104.6
5	2048	2439.8	7287.6
10	8192	10373	54836
20	32768	43771	390800

Table 3: Properties of the Cascade Bernoulli Factory against that proposed by Flegal and Herbei [2012].

may prove to be prudent given that our algorithm is not interruptable [Fill, 1998] – that is, we can not terminate any run without introducing bias into the resulting bit stream, since as the iterations progress, we are far more likely to obtain a zero than a one.

- This should also cause a user to consider whether the number of available input bits is comparatively small, in which case the probability that the algorithm would terminate due to lack of supply would far exceed the probability of termination by gluttony.

The practical improvement over Nacu and Peres [2005] is clear, for an algorithm whose design was theoretically elegant but known to be impractical. The method also yields considerable practical improvement over that proposed by Flegal and Herbei [2012] in terms of both the minimum number and the expected number of input bits required for a single output, mainly because our method for this function always has $|B_{n,0}| = 1$ (as $b^n(0) = 1$), if the first considered bit sequence contains no ones, the algorithm will output a zero on the first round. This result is perfectly valid if the envelope is shown to be strictly greater than the target function, which requires that the number of input bits exceed some minimum value.

For $\epsilon = 0.2$, the number of input bits for a standard implementation of the Cascade Bernoulli Factory is given in Table 3, comparing this method against the Flegal and Herbei [2012] implementation for various multipliers c, over 10^4 trials, with $p_{in} = 0.01$. The termination probability was chosen to be 1 in 10^6 for most examples, though for the case when $c = 2$, it is easy to find a "small" value of n, on the order of 10000, where the probability of continuing past the last pre-calculated envelope is so small that the computer cannot distinguish it from zero.

The minimum number of draws shown in this table is not a strict property of the method, but simply a consideration to be made in the choice of envelopes, since there needs to be a comfortable distance between each Bernstein expansion and the target function so that future steps do not require a vast number of additional input bits. The fewer bits that are required at the first checkpoint, the less this distance will be, and the more bits will be required in further steps. Likewise, choosing a higher number of bits and a closer envelope function will decrease the probability that a comparably large number of bits will be required for the algorithm to terminate, but greatly increase the number of bits required if termination does not occur quickly.

4 EXTENDING TO GENERAL CONCAVE/CONVEX FUNCTIONS

As defined, the cascading envelope method works identically for general convex and concave functions beyond the simple piecewise linear construction we have used so far. The only difference is that the method for guaranteeing that the Bernstein expansion for the upper envelope is greater than the target function is not as simple as checking a finite number of points. A numerical method may need to be used to guarantee that the envelope does not cross the target function.

4.1 The Smoothed Elbow, Revisited

The proposed factory function of Flegal and Herbei [2012] was chosen partly for its asymptotic properties. The factory function is

$$f(p|c,\epsilon,\delta) = \mathbb{I}\left(p < \frac{1-\epsilon}{c}\right) cp +$$

$$\mathbb{I}\left(p \geq \frac{1-\epsilon}{c}\right)\left((1-\epsilon) + q\left(p - \frac{1-\epsilon}{c}\right)\right)$$

where

$$q(p|c,\epsilon,\delta) = \delta \int_0^{cp/\delta} e^{-t^2}\, dt$$

which is twice differentiable, with $|f''(p)| \leq C = c^2 \frac{\sqrt{2}}{\delta\sqrt{e}}$. The authors choose $a^n(p) = f(p)$ as before, and

$$b^n(p) = f(p) + \frac{C}{2n}$$

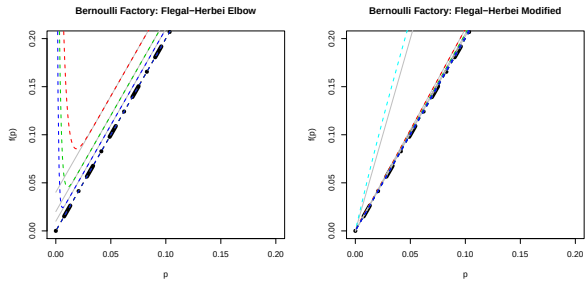

Figure 2: The Flegal-Herbei smoothed elbow function, with the original implementation (left) and an envelope-based modification (right), displayed near the origin for comparison. The improved version considerably speeds up the algorithm, particularly for small p, as the function is quickly optimized for that region of the curve. While the first checkpoint of the original algorithm has 256 bits (512 and 1024 follow), an additional envelope pair is added with only 20 bits that allows for even greater speedup.

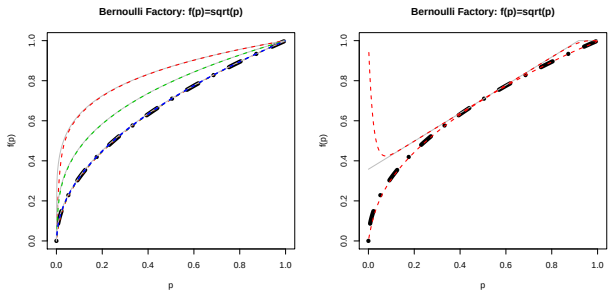

Figure 3: Two choices of envelopes for the factory function $f(p) = \sqrt{p}$. Left, a cascade with $h^m(p) = p^{1/q_m}$, where $n = (100, 200, 300)$ and $q = (5, 3, 2.02)$, for which the envelope conditions are shown to hold manually. Right, a single envelope designed to lie nearly tangent to the target from above, $h^1(p) = \min(0.358 + 0.7p, 1)$ for $n = 50$.

for the upper envelope. This has the property of converging at a rate of $\frac{1}{n}$ across the whole function.

We can improve upon this in two ways. First, we can choose an envelope function that equals zero at zero, so that small values of p will have quick rates [Nacu and Peres, 2005], while still respecting the condition on the second derivative; the function

$$b^n(p) = f(p) * \left(1 + \frac{C}{2n(1-\epsilon)}\right)$$

maintains this condition. Second, once this is in place, we can preface this sequence with looser envelopes at lower values of n that will ensure earlier termination without interfering with the original sequence.

Figure 2 illustrates these modifications with respect to $(c, \epsilon, \delta) = (2, 0.2, 1/6)$. These envelopes lie closer to the target function for the same number of input bits. The "preface" envelope, taking 20 input bits, also substantially cuts down on the running time without compromising the integrity of the original algorithm.

4.2 Differential Targetting: The Square Root Function

Consider the square root factory function

$$f(p) = \sqrt{p}$$

which is concave and bounded on the interval $(0, 1)$. Because of this concavity, the only supplemental envelopes that are required are above the function. (Another example for parabolic functions is included in the supporting material.)

Figure 3 shows how a choice of envelope functions may be made with differing objectives. On the left is a sequence of envelopes designed to approximate the entire target function, by taking power functions that approach $f(p) = p^{0.5}$ from above; on the right, a single envelope function is shown that lies nearly tangent to the factory function at $p = 0.5$. If we have knowledge that the input bits have probability in that neighborhood, then this function will be far more efficient than the standard envelope sequence alone. To ensure a balance of efficiency, we can also design a cascade that starts with the tangent envelope, followed by a set of power envelopes that lie between the tangent and the factory function target.

The convergence time properties of these methods are unknown; however, they can be shown to terminate in finite time, since for any c_i there must exist a finite n under which the upper envelope is greater than the target (as the Bernstein polynomial to the upper envelope converges to the envelope from below), so that we can always choose a cascade series $\{c_i\}$ that will converge toward the target function from above. Because these functions both have finite second derivatives on the open interval $(0, 1)$, there exists an envelope cascade across the whole domain where the expected time to convergence for any particular p decays as $1/n$, though in the case of the square root, this convergence will be much slower near zero where the second derivative approaches infinity.

References

Asmussen, S., Glynn, P. and Thorisson, H. (1992). Stationary Detection in the Initial Transient Problem. *ACM Transactions on Modeling and Computer Simulation*, **2** 130–157.

Blanchet, J. H. and Meng, X.-L. (2005). Exact Sampling, Regeneration and Minorization Conditions. Tech. rep., Columbia University.
URL http://www.columbia.edu/~b2814/papers/JSMsent.pdf.

Blanchet, J. H. and Thomas, A. C. (2007). Exact Simulation and Error-Controlled Sampling via Regeneration and a Bernoulli Factory. Working paper,
URL http://acthomas.ca/academic/papers/factory-sampling.pdf.

Fill, J. (1998). An Interruptable Algorithm for Perfect Sampling via Markov Chains. *Ann. Appl. Probab*, **8** 131–162.

Flegal, J. and Herbei, R. (2012). Exact Sampling for Intractable Probability Distributions via a Bernoulli Factory. *Electronic Journal of Statistics*, **6** 10–37.

Hobert, J., Jones, G. and Robert, C. (2006). Using a Markov Chain to Construct a Tractable Approximation of an Intractable Probability Distribution. *Scandinavian Journal of Statistics*, **33** 37–51.

Hobert, J. P. and Robert, C. P. (2004). A Mixture Representation of (pi) with Applications in Markov Chain Monte Carlo and Perfect Sampling. *Annals of Applied Probability*, **14** 1295–1305.

Keane, M. and O'Brien, G. (1994). A Bernoulli Factory. *ACM Transactions on Modelling and Computer Simulation*, **4** 213–219.

Latuszynski, K., Kosmidis, I., Papaspiliopoulos, O. and Roberts, G. O. (2011). Simulating Events of Unknown Probabilities via Reverse Time Martingales. *Random Structures and Algorithms*, **38** 441–452.
URL http://arxiv.org/abs/0907.4018.

Nacu, S. and Peres, Y. (2005). Fast Simulation of New Coins from Old. *Annals of Applied Probability*, **15** 93–115.

von Neumann, J. (1951). Various Techniques Used in Connection with Random Digits. *Applied Math Series*, **12** 36–38.