A Practical Implementation of the Bernoulli Factory

Andrew C. Thomas* Jose H. Blanchet†

June 14, 2011

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, but can easily be coupled to more 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 prototypical problem for this comes from von Neumann [1951], which seeks to generate a "fair coin", or a draw from a $Be(0.5)$ random variable, from an i.i.d. sequence of Bernoullis with unknown success probability p. The corresponding stochastic process has three states, labelled "yes", "no" or "continue":

- Begin in state "continue".
- Take two draws from the input sequence. The possible outcomes are grouped as $\{00, 01, 10, 11\}$.

*Department of Statistics, Carnegie Mellon University. Corresponding author; email act@acthomas.ca
†Department of Industrial Engineering and Operations Research, Columbia University.

- While the outcome is 00 or 11, remain in state "continue". Discard these bits and replace them with two new draws.
- If the outcome is 10, output "yes"; if the outcome is 01, output "no".

Because both 01 and 10 have probability $p(1-p)$ or occurring, there is equal probability of the outcome being a "yes" or a "no", and as a result, 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 $2p(1-p)$, so that the expected number of input bits required is $\frac{1}{p(1-p)}$.

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 $X_i \sim Be(0.5)$. 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:

- Begin with $n = 1$.
- At each stage 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 $p - U_n < 2^{n-1}$, 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 rates proportional to the target probability p_{out}. While neither requires a sophisticated implementation in order to produce a correct output, these methods suggest a general trend: that methods that produce absorbing states of simple Markov chains are powerful methods of transforming random bits without a loss of information, merely efficiency.

1.1 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, 2011], 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.)

To make practical use of this, the solution proposed by Nacu and Peres [2005] uses a pair of Bernstein polynomial forms to approximate $f(p)$ from above and below. 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 a draw from the target distribution $Be(f(p))$ can be obtained in finite time.

While the theoretical properties of this method have been described by previous authors, the practical implementations to these solutions are concerned more with theoretical tractability than flexibility or applicability to real problems. Chief among these problems is that for convex or concave functions, while one envelope is trivial to construct, the other is invariably more difficult and requires tuning to the particular expectations of the problem. To strengthen the use of the factory in practical problems, we propose a general strategy for envelope construction that uses a series of cascading envelopes on the "difficult" function, one that is easy to implement and verify on many classes of input function. We first review the Bernstein expansion as introduced by Nacu and Peres [2005] and demonstrate it on purely linear functions, before moving to more general convex or concave functions including piecewise linear functions.

2 Bernstein Polynomial Expansions and Set Approximations

Bernstein polynomials are a set of basis functions defined on the interval $[0, 1]$. A Bernstein approximation of order n contains a total of $n+1$ functions of the form $p^k(1-p)^{n-k}$, which is also the individual probability of any single sequence containing k ones and $n-k$ zeros. Indeed, it is trivial to show that the Bernstein polynomial approximation is the expected value of the function under a Binomial random variable:

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

Using this result, a convex 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 concave function will always be less than any of its Bernstein approximations.

The general method proposed by Nacu and Peres [2005] works with the use of 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 $a^n(x)$ that approximate the target function $f(x)$ from below (that is, for all n, $a^n(x) < f(x) and \lim_{n\to\infty} a^n(x) = f(x)$). Define also another series of functions $b^n(x)$ to approximate $1 - f(x)$, also from below; by construction, $a^n(x) + b^n(x) \leq 1$. These functions are used when the total length of the input bit string is n.

- Let A_n be a set of bit-words of length n, and $A_{n,k}$ be the subset of A_n with exactly k ones; likewise with B_n.

- Note the Bernstein polynomial approximations $a_n^n(x)$ and $b_n^n(x)$ yield natural quantities expressions for $A_{n,k}$ and $B_{n,k}$. Since

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

we introduce

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

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 $B_{n,k}$ (notingthat the probability of observing any one bit-string is $p^k(1-p)^{(n-k)}$.)

- Ensure that $A_n^n(x) \leq f(x)$ and $B_n^n(x) \leq 1 - f(x)$. 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 may also be 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 $f(p) = c + hp$, it is clear that the standard Bernstein polynomial expansion is identical:

$$\begin{aligned} f_n(p) &= \sum_{k=0}^{n} \binom{n}{k} (c + h\frac{k}{n}) \, p^k(1-p)^{n-k} && (1) \\ &= c + \frac{h}{n} \sum_{k=0}^{n} \binom{n}{k} \, k \, p^k(1-p)^{n-k} && (2) \\ &= c + \frac{h}{n} EK = c + \frac{h}{n} np = c + hp. && (3) \end{aligned}$$

As a result, the function can be used as both an upper and a lower envelope. 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 $C_{n,k}$. As a result, the survival function is bounded above by the simple expression

$$P(T > n) \leq \sum_{i=0}^{n} p^i(1-p)^{(n-i)}.$$

To demonstrate, consider the von Neumann problem again, so that $f(p) = 0.5$ for whatever p. Setting $a^n(x) = b^n(x) = 0.5$ for all x, for any n it is clear that the size of any $C_{n,k}$ is either 0 or 1, if $\binom{n}{k}$ is even or odd respectively. In particular, consider the cases where $n = 2$ and $n = 4$, and a potential distribution of bit strings over sets:

k,n=2	$A_{2,k}$	$B_{4,k}$	$C_{2,k}$
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

The sequences in the respective A, B and C sets once again represent "output 1", "output 0" or "add more bits", but under the Bernstein construction, there are two additional bit strings that will terminate the algorithm where von Neumann would not: 0011 and 1100. Note that the "descendents" of A_2 also appear in A_4 and indeed all A_n beyond $n = 2$.

In practice, it is not necessary to construct this table for all n, or even to select a particular partitioning of all bit-strings, only to note the size of the sets themselves. As we demonstrate, this is done by ensuring that for any string in A_n, all of its "descendants" obtained by adding any k-length bit string are members of A_{n+k}, and similarly for B_n and B_{n+k}.

2.2 $f(p)$ is Piecewise Linear, and Concave (or Convex)

One motivating problem for the practical use of the Bernoulli factory is the aforementioned elbow function

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

which is concave on the interval $[0, 1]$. 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 $a_n(p) = f(p)$.

The upper envelope function is considerably more difficult to design. 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 [2011] shows that a minimum of 2^{16} bits are required for the function $f(p) = \min(2p, 0.8)$.

The alternative specification of Flegal and Herbei [2011] changes the problem slightly by specifying a new objective function that is twice-differentiable, but still linear on the domain $[0, (1-\epsilon)/c]$,

using the method of Latuszynski et al. [2011]. 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.

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 basic form of the 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 converges to the target function $f(p)$ in the limit.

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 trivial 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:

- 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).

- 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 $f(p)$.

- 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.

- 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:

- Choose the next elbow point to be where the previous Bernstein approximation $1 - b_{m_{i-1}}^{m_{i-1}}(p)$ intersects the elbow point curve. 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

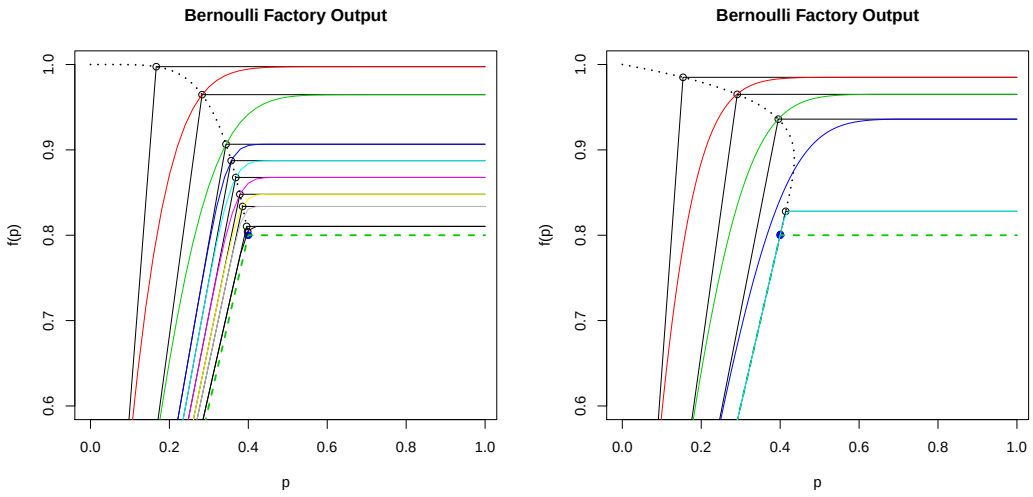

Figure 1: Two methods for generating upper envelope functions for $f(p) = \min(2p, 0.8)$; each successive function is defined by the intersection of a curve with the previous Bernstein approximation. Left, the elbow points are generated with a simple polynomial descent (dotted line), as each successive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

the target, an extreme value of n 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.

- 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 $(k_i - k_{i-1})$ 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}.

- Generate a trinomial random variable with probabilities proportional to $(|A_{m_i,k_i}|, |B_{m_1,k_i}|, |C_{m_1,k_i}|)$. 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 is known that the previous bit string belonged to group C.

Due in part to its design as a practical implementation, the algorithm as stated does not have conveniently calculable convergence properties. However, at worst, there exists some n for which

the envelope functions for Nacu and Peres [2005] can be used instead of the cascade, since as n gets arbitrarily large for any given cascade envelope, it approaches that envelope and not the target function; by the properties of the Bernstein approximation, there must exist some finite n for which $1 - b_n^n(p)$, maintaining the same elbow position, is greater than the NP-envelope $h_n(p)$ for all $0 < p < 1$.

The method yields considerable practical improvement over that proposed by Flegal and Herbei [2011] in terms of both the minimum number and the expected number of input bits required for a single output. Much of the speed increase can be attributed to the fact that because $|B_{n,0}| = 1$ (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 following table represents the number of input bits for a standard implementation of the Cascade Bernoulli Factory against the Flegal and Herbei [2011] implementation for various multipliers c, over 10^4 trials, with $p_{in} = 0.01$:

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

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. The choice of what quantity to optimize is therefore an option that practitioners may configure. If the required number of output bits is small, one can choose a sufficiently close envelope with a relatively high probability of termination – in this example, the chosen probability was 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 is so small that the computer cannot distinguish it from zero.

3 General Convex or Concave Functions

It is not a stretch to say that these methods can be combined to produce an even more efficient Bernoulli factory in the same way that the original Nacu and Peres [2005] envelopes can be introduced for high n. The use of the Latuszynski et al. [2011]/Flegal and Herbei [2011] envelope construction can be used instead, with the change that the target function is no longer piecewise linear.

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 will need to be used to guarantee that the envelope does not cross the target function.

Acknowledgements

We thank Peter Glynn, Jim Hobert, Xiao-Li Meng and Christian Robert for their introduction of the problem to us, Serban Nacu for various discussions, and Krzysztof Latuszynski for making his improved implementation known to us.

Supplemental Code

Currently available from the corresponding author's web site.

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.

Flegal, J. M. and Herbei, R. (2011). Exact sampling for intractable probability distributions via a Bernoulli factory.
URL http://arxiv.org/abs/1012.3768.

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/1012.3768.

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.