Exact Simulation and Error-Controlled Sampling via Regeneration and a Bernoulli Factory

Jose Blanchet* Andrew C. Thomas†

April 13, 2007

Abstract

Methods using regeneration have been used to draw approximations to the stationary distribution of Markov Chain Monte Carlo processes. We introduce an algorithm that allows exact sampling of the stationary distribution through the use of a regeneration method and a Bernoulli Factory to select samples within each regeneration cycle that are shown to be from the desired density. We demonstrate the algorithm on a probit model Markov Chain using a known data set for comparison to other approximate methods.

1 Introduction

The objective of Markov Chain Monte Carlo methods is to create such a chain that converges to a stationary distribution, such that draws from it are found to be independent and identically distributed to a random quantity of interest, typically the posterior distribution of a Bayesian probability model.

One difficulty in this approach is knowing when the chain has reached its stationary state, independent of its starting location. By constructing conditions under which the chain regenerates - that is, discovering a transition between states that is independent of the starting state, thereby severing the chain's link with the past - it is possible to set conditions under which an iteration of the chain within each regeneration cycle is equivalent to a draw from the stationary distribution.[2]

Once this approach has been conceived, the conditions on the chain are set, and one value within a cycle may be chosen for consideration as being a draw from the stationary distribution.

*Assistant Professor, Department of Statistics, Harvard University. Email: blanchet@stat.harvard.edu.

†Doctoral Candidate, Department of Statistics, Harvard University. Email: athomas@stat.harvard.edu. The authors thank, among others, Xiao-Li Meng for his editorial comments, David van Dyk for providing the data set, and Jim Hobert and Vivek Roy for preprints of their work.

Because the explicit distributional form is not known, we use acceptance-rejection sampling. Given that one of our conditions is that regeneration has yet to occur for this cycle, which has an unspecified probability, we use a Bernoulli factory to perform the acceptance-rejection step and determine whether a draw is indeed from the stationary distribution.

Two conditions are needed to specify regenerations: the minorization condition, with which the chain is specified in a way that yields regenerations, and the drift condition, which specifies the time until the next regenerations. It should be noted that the example we give, the case of the probit model posterior, has an estimated drift condition. This has the consequence that samples from the posterior will be error-controlled rather than perfect; however, this does not change the algorithm itself to determine the selected draws.

1.1 Drawing from the stationary distribution

In general, the stationary distribution of a positive recurrent Markov process $Y(t)$ is defined as

$$\pi(A) = \lim_{t\to\infty} \frac{1}{t} \int_0^t I(Y(s) \in A) ds.$$

Suppose there exists a sequence of times $T_0 = 0, T_1, T_2, ...$ on to infinity (but where each time is finite.) We can then express the above integral as a piecewise sum

$$\pi(A) = \lim_{t\to\infty} \frac{1}{t} \sum_{k=0}^{N(t)-1} \int_{T_k}^{T_{k+1}} I(Y(s) \in A) ds.$$

If these times represent regeneration times, then the integrals $\int_{T_k}^{T_{k+1}} I(Y(s) \in A) ds$ are iid random variables V_k by the renewal-reward theorem. We then have

$$\begin{aligned} \pi(A) &= \lim_{t\to\infty} \frac{1}{t} \sum_{k=0}^{N(t)-1} V_k & (1) \\ &= \lim_{t\to\infty} \frac{N(t)}{t} \frac{\sum_{k=0}^{N(t)-1} V_k}{N(t)} & (2) \\ &= \frac{1}{ET} EV. & (3) \end{aligned}$$

We express each cycle as its own Markov chain with regeneration happening at time T, such that X_{T-1} is the last iteration in the cycle, we then have

$$\begin{aligned} P(X_\infty \in A) &= \frac{E(\sum_{k=0}^{T-1} I(X_k \in A))}{ET} && (4)\\ &= \frac{E(\sum_{k=0}^{\infty} I(X_k \in A) I(T > k))}{ET} && (5)\\ &= \frac{\sum_{k=0}^{\infty} P(X_k \in A | T > k) P(T > k)}{ET} && (6)\\ &= \frac{\sum_{k=0}^{\infty} P(X_k \in A | T > k) P(T > k)}{\sum_{k=0}^{\infty} P(T > k)}. && (7) \end{aligned}$$

But since T is non-negative, $ET = \sum_{k=0}^{\infty} P(T > k)$. If this sum is finite, we define a new random variable $\tilde{T}$ such that $P(\tilde{T} = k) \propto P(T > k)$; it is clear that the constant of proportionality is then $1/ET$. If we are able to draw $k \sim \tilde{T}$, we can select the value X_k within a regeneration cycle corresponding to the stationary distribution X_∞.

We cannot draw directly from this distribution, so we use acceptance/rejection sampling. We therefore generate a proposal distribution of an appropriate form, in this case of a form with heavier-than-exponential tails. By Chebyshev's inequality, we therefore have a **drift condition**,

$$P(\tilde{T} = k) ET \leq P(T > k) \leq c e^{-\delta k^l}$$

for a value $c \geq E e^{\delta T^l}$. So it remains to find appropriate values of δ and l such that the expectation exists, and derive the constant c. A discrete proposal distribution is then immediately derived such that $h(j) = c_2 e^{-\delta j^l}$ and $\sum h(j) = 1$. To perform acceptance-rejection sampling, we then draw a sample from the proposal distribution and accept with probability

$$\begin{aligned} p^* &= \frac{P(T > j)}{c e^{-\delta j^l}} && (8)\\ &= \frac{P(T > j)}{k h(j)} && (9) \end{aligned}$$

We do not know the distribution of T explicitly; however, we can sample the random

variable $I(T > j)$ through our explicit calculation of regeneration times. Determining acceptance or rejection then depends on drawing perfectly from a random Bernoulli with probability $\frac{1}{kh(j)}P(T > j)$. If the constant $\frac{1}{kh(j)}$ is less than one, draw from a Bernoulli with that constant as the probability and accept if it is positive and regeneration has not yet occurred ($I(T > j)$). If greater than one, we use the Bernoulli Factory approach to determine acceptance.

2 A Bernoulli Factory, Implemented

A Bernoulli factory takes as input a sequence of Bernoulli random variables with parameter p. It then outputs a sequence of Bernoulli random variables with parameter $f(p)$. The output sequence is invariably shorter than the input, often by a large degree. In this case we require only one output from a series of inputs. We explore the implementation of the **factory function**

$$f(p) = \min(cp, 1 - \epsilon), c > 1, \epsilon > 0$$

where in this case $c = \frac{1}{kh(j)}$. The ceiling term $1-\epsilon$ is required as the algorithm will not converge for $f(p) = 1$, as noted in [4]. (Note that if $c < 1$, producing a transformed Bernoulli random variable is trivial; simply multiply the outcome of a $Be(p)$ with that of a $Be(c)$.

2.1 Method

As suggested by Nacu and Peres [1], we derive envelope functions $g(p)$ and $h(p)$ such that they converge to $f(p)$ from above and below. These functions will be constructed using a Bernstein polynomial expansion, defined as

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

which is equivalent to $E_p f(X/n)$ for $X \sim \text{Binom}(n, p)$.

The approximation is dependent on the chosen value of n. We construct this approximation for each value in a series $(n_1, n_2, ...)$, then evaluate the first n_i bits in the input and determine whether to terminate (outputting a one or zero) or continue. We refer to the first n_i bits as a

word of length n_i; we refer to an n_i-word containing k ones as an (n_i, k)-word.

If the bits in an (n, k) word are independently equal to 1 with probability p, the measure of any n-long word with k ones is equal to $p^k(1-p)^{n-k}$. We can construct an approximation to $f(p)$, from above and below, by determining the correct number of (n, k)-type words in each subset - above the upper envelope, below the lower, and in between - and checking to determine which set our input word is in.

Define three disjoint sets of words of length n: A_n, containing words that generate the output 1, B_n, containing words generating the output 0, and the remainder C_n, which force us to increase n and create a new dictionary and three new sets. In particular, we shall partition these sets according to the number of ones in each word; that is,

- $A_{n,k}$ is a set of (n, k)-words which yield a positive output; likewise $B_{n,k}$ for a negative output and $C_(n, k)$ the signal to continue,
- $\bigcup_{k=0}^{n} A_{n,k} = A_n$, $\bigcup_{k=0}^{n} B_{n,k} = B_n$, $\bigcup_{k=0}^{n} C_{n,k} = C_n$, and
- $A_{n,k} \bigcap A_{n,j} = \emptyset$ for all $k \neq j$.

We need not construct any of these sets explicitly; we only need to determine the size of each subset given by (n, k).

We desire the following two properties:

1. All extensions of any word in A_m or B_m - any additional bits, regardless of value - will produce words in A_n or B_n respectively for $m < n$. We refer to this as the **nesting property** of the envelope scheme, which ensures that dictionaries with higher n do not contradict those for lower n, and that termination is acceptable.

2. The distribution of the number of input bits needed for one output, N, has exponential tails:

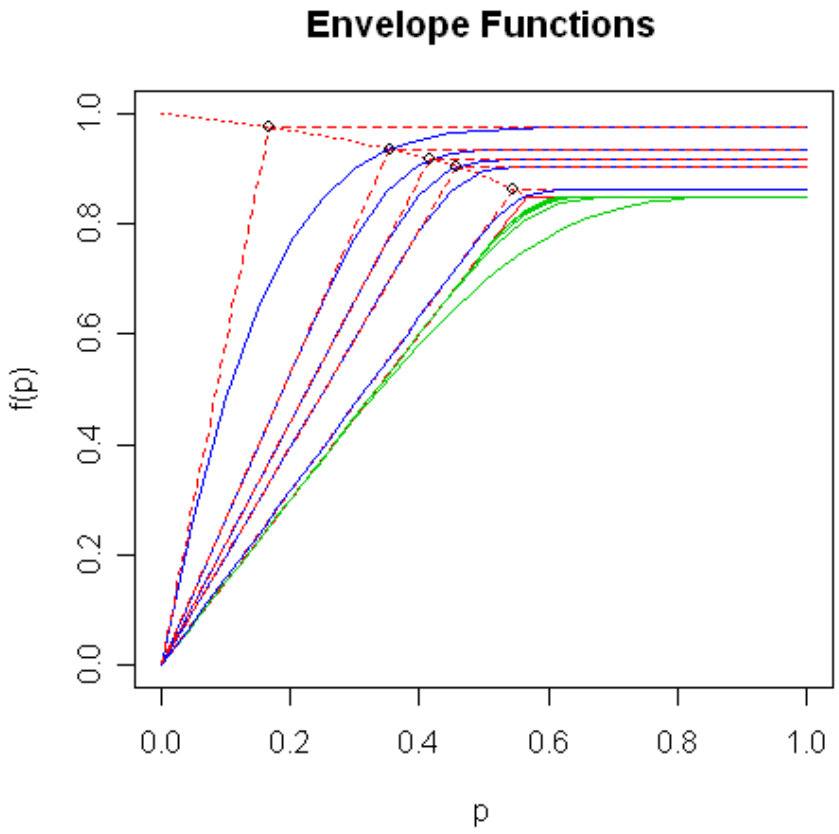
   $P(N > n) \leq C\rho^n$, where $\rho < 1$. This ensures that simulation is "fast" ([1]) - that is, that the generating process terminates exponentially quickly.

Algorithm

Here is one algorithm which incorporates the above requirements to produce a Bernoulli Factory for the function $f(p) = \min(cp, 1-\epsilon)$.

1. **Design the envelope functions.** For our $f(p)$, the under-envelope $g(p)$ can simply be $f(p)$; since $f(p)$ is concave, and $\hat{g}(p;n)$ is an expectation, by Jensen's inequality the condition is immediately satisfied.

   Since Bernstein polynomials of concave functions increase uniformly, constructing a set of decreasing envelopes is more challenging. What follows is the **cascade envelope** method guaranteed to produce a decreasing sequence. Define a path between $(1,0)$ and $((1-\epsilon)/c, 1-\epsilon)$. This path will define a series of "kink points", defining functions whose Bernstein polynomials are strictly less than their predecessors. The first kink point is defined close to $(0,1)$, and each successive kink is defined as where the path intersects the previous Bernstein polynomial, as seen below.



   The final envelope must still be greater than the factory function $f(p)$; since the polyno-

mials are concave, we need to make sure only that they are greater only at the kink point and the end point. Since we will in practicality only choose a finite number of envelopes, we can move the final kink point as far down the path as possible while still producing a viable upper envelope. (This will not prevent us from continuing the algorithm ad infinitum; rather, the number of bits required to produce the next valid envelope will be so great as to render the algorithm impractical.)

Now that we have a defined set of Bernstein polynomials for each n_i, we can build a dictionary starting with $i = 1$.

2. **Calculate** $\lfloor \binom{n_i}{k} * g(k/n_i) \rfloor$ for $k \in [0, ..., n_i]$. These values are the lengths of the sub-dictionaries of A_{n_i} sorted by the number of ones.

3. **Calculate** $\lceil \binom{n_i}{k} * h(k/n_i) \rceil$ $k \in [0, ..., n_i]$. These values are the lengths of the sub-dictionaries of B_{n_i} sorted by the number of ones.

4. **Subtract** $|A_{n_i,k}|$ and $|B_{n_i,k}|$ from $\binom{n_i}{k}$, which equals $|C_{n_i,k}|$ by definition.

5. If $i > 1$, to have reached this level we must have received a "continue" from the previous one. We must subtract off those elements that led to termination at the previous step. This number is equal to

$$A^{-}_{n_i,k} = \sum_{j=0}^{k} \binom{(n_i - n_{i-1})}{k-i} \left\lfloor \binom{n_{i-1}}{i} * g(k/n_i) \right\rfloor$$

and the formula is similar for the "negative" case.

6. Repeat the previous four steps for all remaining n_i.

7. To obtain a draw, first form a word from the first n_1 input bits. Note the number of ones in the word (call it k) and draw from the trinomial distribution where $N = 1$ and $p = (|A_{n_1,k}|, |B_{n_1,k}|, |C_{n_1,k}|)$. If the result is the first or second, output 1 or 0 respectively. If not, add $n_2 - n_1$ bits and repeat similarly.

3 Implementation

The Bayesian probit regression model takes as its outcome variable either zero or one, and uses $\Phi(\cdot)$, the cumulative distribution function of the standard Normal, as its link function. Each observation takes the form

$$y_i = Be(p_i) \tag{10}$$

$$p_i = \Phi(\mathbf{x_i}'\beta), \tag{11}$$

where $i = 1...n$, and with each β coefficient having an improper prior distribution, equal to 1 across the entire real line.

On its own, the parameter space is difficult to explore, and so several data augmentation methods have been suggested to allow Gibbs sampling of the posterior distribution. One highly successful method is the Parameter Expansion Data Augmentation (PX-DA) chain proposed by Liu and Wu (1999)[1] in which each outcome is augmented with a truncated normal random variable z_i. Let the notation $TN(\mu, \kappa^2, w)$ represent a normal random variable with mean μ and variance κ^2, truncated to the domain $(-\infty, 0)$ if $w = 0$, or $(0, \infty)$ if $w = 1$. Define g as a perturbation parameter

The algorithm involves the repetition of the following three steps:

1. Draw β' from a multivariate normal distribution with mean $g(X'X)^{-1}X'z'$ and variance $(X'X)^{-1}$.

2. Draw $z_1...z_n$ independently from $z_i \sim TN(x_i'\beta, 1, y_i)$.

3. Draw g^2 from the distribution

$$Gamma(\frac{n}{2}, \frac{1}{2}\sum_{i=1}^{n}(z_i - x_i(X'X)^{-1}X'z)^2).$$

Note that we use the square root of this draw in the earlier step.

[1]This chain is a substantial improvement over that proposed by Albert and Chib (1993).

We express the transition kernel as $k(z', g', \beta'|z, g, \beta)$, where (z', g', β') is the distribution after the transition.

3.1 Regeneration in PX-DA

Roy and Hobert [3] propose a method for inducing regenerations in a PX-DA chain, so that each cycle started by a regeneration is independent of all other cycles. It is then desired to draw from the stationary distribution of the chain knowing that draws from within a cycle are fully independent of the beginning of the chain.

Define the transition kernel $k(z', g', \beta'|z, g, \beta)$ according to the mixture distribution

$$k(z', g', \beta'|z, g, \beta) = s(z, g, \beta)d(z', g', \beta') + (1 - s(z, g, \beta))r(z', g', \beta'|z, g, \beta),$$

where $s(z, g, \beta)$ is the probability that we generate the next step using a distribution $d(\cdot)$ independent of the chain's present value. The remainder term $r(\cdot|\cdot)$ can be defined implicitly as that part of the distribution of $k(\cdot)$ not accounted for by the regenerative distribution $d(\cdot)$.

Because all quantities in the transition equation are nonnegative, we have a necessary condition on the distributions $s(\cdot)$ and $d(\cdot)$. This is the minorization condition for the chain,

$$k(z', g', \beta'|z, g, \beta) \geq s(z, g, \beta)d(z', g', \beta').$$

Roy and Hobert [3] derived functional forms for $s(\cdot)$ and $d(\cdot)$ using a fixed "distinguished point" z_* and a hyper-rectangle D on the same space as β nicknamed the "red zone", such that when the chain leads to the condition $\beta' \in D$, there is a nonzero probability that the destination co-ordinate has been generated according to the distribution $d(\cdot)$, and hence that a regeneration has taken place.

The minorization condition components are defined as

$$s(\beta) \quad = \quad w \inf_{\beta \in D} \frac{\pi(\beta|gz, y)}{\pi(\beta|z_*, y)}, \tag{12}$$

$$d(z', g', \beta') = \frac{1}{w}\pi(\beta'|z_*, y)\pi(z'|\beta', y)h(g'|z')I(\beta' \in D), \quad (13)$$

$$w = \int_D \pi(\beta|z_*, y)d\beta, \quad (14)$$

$$h(g|z) = \frac{(z'(I - X(X'X)^{-1}X')z)^{n/2}}{2^{n/2-1}\Gamma(n/2)}g^{n-1}exp(-g^2z'(I - X(X'X)^{-1}X')z/2) \quad (15)$$

where we have now explicitly defined the distribution function of g.

This gives us two methods to simulate the Markov chain. We can proceed as originally designed, or we can draw a Bernoulli random variable with probability $s(\cdot)$; if the result is positive, we generate from the random variable $d(\cdot)$, otherwise, we draw from $r(\cdot)$. This has the unfortunate consequence that drawing from the exact distribution of $r(\cdot|\cdot)$ may be troublesome. However, another approach is easy to implement.

Given the minorization condition, it can be determined after the fact that a regeneration took place during a transition $(\beta, z, g) \rightarrow (\beta', z', g')$ by drawing a Bernoulli random variable with probability

$$\frac{s(\beta)d(z', g', \beta')}{k(z', g', \beta'|z, g, \beta)}$$

which we know to be less than one by definition. Also embedded is the notion that β' must lie within the "red zone" in order that regeneration take place.

3.2 Data

To test the model, we use the lupus data set provided by David van Dyk and used in various probit sampling applications ([5]). The positive outcome is the expression of the condition lupus nephritis in 55 patients. Two explanatory variables, the level of antibody IA and the difference in levels of antibodies IG3 and IG4, are included.

The Markov chain was initialized with each z_i drawn from a normal distribution, truncated to its domain as determined by each y_i; g, the perturbation parameter, is initialized at 1; and β is drawn from the distribution $N((X'X)^{-1}X'z, (X'X)^{-1})$. The chain was run for 2,000,000 iterations.

To determine regenerations, the distinguished point z_* was designated as the respective

means of the draws for z numbered [1001,20000] in the sequence. The red zone D was chosen to be a hyper-rectangle defined as a series of intervals centered at the mean of each β_i and reaching 0.09 of a standard error in either direction.[2]

As we preserved the entire chain as it was run, we were able to repeatedly investigate the chain for regenerations by preserving the *ex post facto* probability that a regeneration occurred. On one such investigation we obtained 138 regenerations; the longest cycle was 72,382 iterations. We then calculated the expectation $Ee^{\delta T^l}$ for various $\delta > 1/ET$ and $l < 1$, and ultimately chose $(\delta, l) = (0.003, 0.64)$, such that the mean of the proposal distribution is comparable to that of the regenerations.

3.3 All Together: Running the Sampling Algorithm

At this point we have all the tools to demonstrate the full algorithm in motion. Beginning with a sequence of regeneration times, and another sequence of proposal times, take the first proposal time, and do the following:

1. Calculate the Bernoulli constant to be used in the factory, and determine whether the proposal is below the first regeneration time.

2. If the constant c is less than one, multiply $I(t_{prop} < \tau)$ by the outcome of $Be(c)$. If this is positive, output the iteration in the sequence corresponding to that draw (including all the other used cycles from earlier attempts.)

3. If the constant c is greater than one, input a series of bits corresponding to whether the proposal time is less than all remaining regeneration times. These then all correspond to realizations of a Bernoulli random variable with probability p; use the factory with these bits to determine the outcome. If positive, output an iteration corresponding to a cycle whose length is greater than the proposal.

4. Remove the used regeneration(s) and the proposal, and repeat with a new proposal.

[2]The smaller the zone, the more likely that location within it will spawn a regeneration; this is counterbalanced by the consideration that a smaller zone is hit less often.

Note that in this method the Bernoulli factory is very inefficient when it comes to the number of regenerations consumed. It exists largely as a means of completing the apparatus, so that there can always be a solution found given an unlimited number of regenerations. In execution it's recommended to minimize the number of regenerations being consumed by choosing a proposal distribution wisely.

Since the analytical form of the regeneration time distribution is extremely difficult to calculate, we estimate the upper bound on $P(T > k)$ from the data available by determining the empirical distribution of $e^{\delta T^l}$, measuring its mean and variance, and using the Central Limit Theorem to produce a normal estimate and find a high-percentile value; in particular, we set the desired condition to be that our estimate gives an improper acceptance/rejection ratio 0.5% of the time, for a total variation distance of no greater than 0.005. This calculation was repeated after each trial, as we used only the observed information at each point to estimate the upper bound. Since the distribution is heavily right-skewed, this estimate for the error is conservative; indeed, the standard error is be positively biased leading to a higher minimum value.

We repeated the investigation of the chain 18 times. During three trials, the first observed stationary points were at iteration 19101, 17160 and 23254 respectively. During another the first stationary point was at iteration 1054952. Twelve times, the algorithm finished without producing a stationary point, and twice the proposed value was so high that it imposed a Bernoulli factory that took more than the length of the chain, subsequently breaking it.

4 Discussion and Further Work

This paper is intended to be a primer to a method of perfect sampling that, while inefficient, has been show to give stationary draws under the right conditions.

Several large questions remain to be answered. The inefficiency of the Bernoulli factory, often swallowing a large number of iterations (often 50 regenerations worth at a time) will no doubt have an effect on the drawing process. Cutting this down would have a major impact on the number of pure draws that can be taken as well. Additionally, we have not decided on the appropriate action to be taken with cycles that do not yield a sampled draw. As of now, we

discard any cycles whose regeneration times have been used, and lack a justification for reusing those cycles within the same draw.

Because we do not have an exact drift condition, we introduced the error control in our upper bound formulation. However, even in those circumstances where the exact drift is know, we may yet gain a great deal by allowing a small degree of error in our calculations, rather than chasing the goal of perfection and losing a great deal of efficiency in the process.

References

[1] Nacu, Serban and Peres, Yuval. "Fast Simulation of New Coins from Old". The Annals of Applied Probability, 15(1A). [2005]

[2] Hobert, James P. and Robert, Christian P. "A mixture representation of π with Applications in Markov Chain Monte Carlo and Perfect Sampling". The Annals of Applied Probability, 14(3). [2004]

[3] Roy, Vivekananda and Hobert, James P. "Convergence rates and asymptotic standard errors for MCMC Algorithms for Bayesian probit regression". (provided by authors.) [2007]

[4] Keane, M.S. and O'Brien, George L. "A Bernoulli Factory". ACM Transactions on Modeling and Computer Simulation, 4(2). [1994]

[5] van Dyk, D.A. and Meng, X.L., "The Art of Data Augmentation". Journal of Computational and Graphical Statistics, 10(1). [2001]

A More on the Bernoulli Factory

Example

Consider the factory function $f(p) = min(3p/2, 0.9)$. As a test, I generate a series of coin flips with probability $p = 0.4$ and enter them into the factory. I would then expect to generate a series of independent Bernoulli random variables with probability of success 0.6.

Table 1: The locations of the kink points of the upper envelope functions for $f(p) = \min(3p/2, 0.9)$. Note that as n increases (by 16 each time) each envelope's Bernstein realization gets closer to its theoretical form with each iteration.

x	y	n
0.259	0.977	24
0.358	0.965	40
0.434	0.954	56
0.494	0.944	72
0.544	0.934	88
0.585	0.925	104
0.618	0.916	120
0.647	0.907	136
0.667	0.900	232

For the purposes of this demonstration, I would like a large number of inputs to be used in generating one output.

Bernstein polynomial envelopes are then constructed for each value n_i; the first expansion is given in table 2 for $n_1 = 24$.

k	output 1	output 0	continue
0	0	1	0
1	1	15	8
2	31	91	154
3	341	20	1663
4	2390	105	8131
5	11954	423	30127
6	45426	1340	87830
7	136278	3448	206378
8	330961	7327	397183
9	661923	13026	632555
10	1103206	19539	838511
11	1544489	24868	926787
12	1825305	26940	851911
13	1825305	24868	645971
14	1544489	19539	397228
15	1103206	13026	191272
16	661923	7327	66221
17	311493	3448	31163
18	121136	1340	12120
19	38253	423	3828
20	9563	105	958
21	1821	20	183
22	248	2	26
23	21	0	3
24	0	0	1

Table 2: The Bernstein polynomial expansion for the envelope function corresponding to $n = 24$. Given the number of ones in the input sequence (column 1, "k"), each value represents the number of elements in each subset, to output one or zero or to continue.