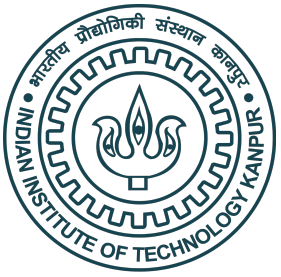
Bernoulli factory based Portkey and Flipped Portkey MCMC Algorithms: Theory and Examples*

Submitted by:

Arkajyoti Bhattacharjee [†‡]
Devyanshi Singh [§†]
Dhruvil Sangani [¶†]
Nitin Garg [‖†]
Rishabh Gupta [*⁂]

Supervised by:

Dr. Arnab Hazra [†]



Submitted on:

22nd April, 2022

Abstract

In this report, we present two stable Bernoulli factories that generate events within those class of acceptance probabilities which do not involve the ratio of intractable target (posterior) distribution evaluated at two points. The efficiency of the methods rely on obtaining a reasonable lower and upper bound on the target density and we present examples where such bounds are viable. The report is primarily based on Vats et al. (2021).

Contents

1 Introduction **3**

2 Barker's method and the two-Coin algorithm **6**

3 Portkey Barker's Method **8**

4 Flipped portkey two-coin algorithm **12**

5 Example applications **15**
5.1 Gamma Mixture of Weibulls 15
5.2 MCMC on constrained spaces 17
5.3 Bayesian inference for the Wright-fisher diffusion 21

6 Conclusion **25**

7 Supplementary Material **25**

8 Acknowledgements **25**

*This report has been prepared towards the partial fulfillment of the requirements of the course *MTH535A: An Introduction To Bayesian Analysis.*

†Department of Mathematics & Statistics, Indian Institute of Kanpur, India.

‡201277, M.Sc. Statistics (Final year).

§180243, B.S. Statistics (Final year).

¶, B.S. Statistics (Final year).

‖180490, B.S. Statistics (Final year).

**180607, B.S. Statistics (Final year).

1 Introduction

Over the course of the 21^{st} century, the use of Markov Chain Monte Carlo (MCMC) algorithms has grown exponentially, with various applications in astronomy (Thrane and Talbot (2019), Sharma (2017)), health science (Sorensen et al. (2002), Vajargah et al. (2021)), cognitive science (Kim et al. (2003)), image compression, optimization (Mahendran et al. (2012), Ma et al. (2015)) and machine learning (Andrieu et al. (2003), Hensman et al. (2015)). It is a sampling methodology widely used in estimating expected values under complicated and high-dimensional distributions, which are known up to a normalizing constant (see Brooks et al. (2011), Liu and Liu (2001), Gilks et al. (1995)).

MCMC consists of two parts - *Markov chain* and *Monte Carlo*. The *Monte Carlo* methods are a broad class of computational algorithms used to compute closed-form analytical solutions of complicated numerical integrals. For example, we may be interested in obtaining an analytical solution of

$$\int_{\pi}^{2\pi} e^{sin(log(x))cos(e^x)}dx.$$

Clearly, finding a closed-form solution of this integral is difficult as no standard anti-derivative exists of the integrand. A Monte Carlo approach to this problem is to sample a large number of $\mathscr{U}(\pi,2\pi)$ variables and compute the above integral as an expectation under the uniform distribution. Mathematically,

$$\int_{\pi}^{2\pi} e^{sin(log(x))cos(e^x)}dx = \pi\mathbb{E}_{\mathscr{U}}(e^{sin(log(X))cos(e^X)}) \hat{=} \frac{1}{N}\sum_{i=1}^{N} e^{sin(log(X_i))cos(e^{X_i})},$$

where $X_1,...,X_N$ is a random sample drawn from $\mathscr{U}(\pi,2\pi)$, $\mathbb{E}_{\mathscr{U}}$ is expectation under $\mathscr{U}(\pi,2\pi)$ and "$\hat{=}$" means 'is estimated by'. The right hand side of the above equation holds due to the weak law of large numbers, assuming N is large enough. The Monte Carlo approach easily provides 3.16 an 'estimated' solution of the otherwise intractable integral. Although other numerical integration techniques may be able to provide an 'approximate' solution, difficulty increases as the dimensionality increases, and MCMC is often the better alternative.

The *Markov chain* part of MCMC uses the Markovian property, which means that the proposed random value depends on the current value and not on the previous values of the sequential process (hence, 'chain').

MCMC is extensively used in Bayesian inference as it involves generating sam-

ples from complicated posterior distributions where the exact form of the likelihood is unknown or difficult to derive analytically. One of most popular MCMC algorithms is the *Metropolis-Hastings* (M-H) algorithm (Metropolis et al. (1953), Hastings (1970), Chib and Greenberg (1995), Robert and Casella (1999)). A survey (Beichl and Sullivan (2000)) placed the MH algorithm among the ten algorithms that have had the greatest influence on the development and practice of science and engineering in the 20^{th} century.

Let $\pi(.)$ be the target density function such that $\pi(x) \propto \pi'(x)$, where the functional form of $\pi'(x)$ is usually known. The aim of the M-H algorithm is to propose values from a proposal density $q(x,.)$, x being the current state of the Markov chain, and store them sequentially in a Markov chain if they are accepted with acceptance probability $\alpha_{MH}(x,y) = \min\{1, \frac{f(y)q(y,x)}{f(x)q(x,y)}\}$. Here, y is a proposed move from x. The M-H algorithm is given in Algorithm 1.

Algorithm 1 Metropolis-Hastings algorithm

Let $X_n = x$. To obtain X_{n+1}:

1. $Y \sim Q(x,.)$ and independently $U \sim \mathscr{U}(0,1)$.
2. If $U < \alpha(x,y) = min\{1, \frac{f(y)q(y,x)}{f(x)q(x,y)}\}$,
   set $X_{n+1} = y$.
3. Else
   set $X_{n+1} = x$.

The class of M-H acceptance probabilities ensure π-stationarity. We state a few definitions and theorems to understand this.

Definition 1 (π**-stationary**). *Let P be a Markov transition kernel and X_0 be the starting value of the associated Markov chain. Let π be a distribution such that $X_0 \sim \pi$. P is said to be π-invariant or π-stationary if $\pi P = \pi$. That is,*

$$\pi P(A) = \int_{\mathscr{X}} P(x,A)\pi(dx) = \pi(A).$$

The above definition means that if we start from π and use P to get another sample, then the sample will also be from π.

One way to construct a Markov kernel P such that it is π-stationary is that it is π-symmetric.

Definition 2 (π**-symmetry**). *The kernel P is π-symmetric or π-reversible if*

$$F(dx)P(x,dy) = F(dy)P(y,dx).$$

The following theorem connects π-symmetry with π-invariance.

Theorem 1. *π-symmetry implies π is invariant for P.*

The Barker's (A. (1965)) acceptance probability, given by:

$$\alpha_B(x,y) = \frac{\pi(y)q(y,x)}{\pi(y)q(y,x)+\pi(x)q(x,y)} = \frac{\pi'(y)q(y,x)}{\pi'(y)q(y,x)+\pi'(x)q(x,y)}, \tag{1}$$

is also π-invariant. It is not as popular as the Metropolis-Hastings acceptance probability, mainly due Peskun's (Peskun (1973)) ordering, by which $\alpha_{MH}(x,y)$ maximizes the acceptance probability for all transitions $x \to y$. However, it can be easily shown that

$$\frac{\alpha_{MH}(x,y)}{2} \leq \alpha_B(x,y) \leq \alpha_{MH}(x,y),$$

which indicates that both the acceptance probabilities have similar performances. Furthermore, the following theorem shows that the variance of ergodic averages from Barker's method is no worse than twice than that of the Metropolis–Hastings:

Theorem 2 (Łatuszyński and Roberts (2013), **Theorem 4(ii)**). *Let $f \in L^2(\pi)$ and denote the i.i.d. Monte Carlo variance by $\sigma_\pi^2 := Var_\pi(f)$. If a square root central limit theorem holds for f and the Metropolis–Hastings chain with the CLT asymptotic variance σ_{MH}^2 i.e.*

$$\frac{\sum_{i=1}^N f(\theta_i) - \pi(f)}{\sqrt{N}} \implies N(0, \sigma_{MH}^2),$$

then a corresponding CLT holds for f and the Barker chain with CLT asymptotic variance σ_B^2 satisfying

$$\sigma_{MH}^2 \leq \sigma_B^2 \leq 2\sigma_{MH}^2 + \sigma_\pi^2.$$

So, Barker's algorithm is a useful alternative when Metropolis–Hastings is difficult to implement. In particular, it is useful in scenarios where $\pi(y)/\pi(x)$ can't be evaluated as π is not tractable upto a normalizing constant. Gonçalves et al. (2017a), Gonçalves et al. (2017b) use the Barker's acceptance probability for intractable Bayesian posteriors. In fact, they used Bernoulli factories, to avoid explicitly calculating $\alpha_B(x,y)$, which we define below.

Definition 3 (**Bernoulli factory**, Latuszynski (2010)). *Let $p \in \mathscr{P} \subseteq [0,1]$ be unknown and let $f : \mathscr{P} \to [0,1]$. Then the problem known as the Bernoulli Factory is to generate Y, a single coin toss of an $s = f(p)-$coin, given a sequence $X_1, X_2, \ldots$ of independent tosses of a $p-$coin.*

Motivated by the need to consider π-stationary acceptance probabilities, other than the typical functions of the ratio $\pi(y)/\pi(x)$, and for which efficient Bernoulli factories can be constructed, we consider acceptance probabilities of the form:

$$\alpha(x,y) = \frac{\pi(y)q(y,x)}{\pi(y)q(y,x)+\pi(x)q(x,y)+d(x,y)},$$

where $d(x,y) \geq 0$ is such that $d(x,y) = d(y,x)$. In particular, we will focus on two choices of $d(x,y)$ for which efficient Bernoulli factories can be constructed to generate events of probability $\alpha(x,y)$. Clearly, $\alpha(x,y) \leq \alpha_B(x,y)$. So, by Peskun (1973), $\alpha(x,y)$ is statistically less efficient that $\alpha_B(x,y)$. However, we will be considering small values of $d(x,y)$ so that the decrease in efficiency is not considerable. We provide examples, depicting that any loss in statistical efficiency is compensated by a gain in computational efficiency.

The rest of the report is organized as follows. In Section (1), we present the Barker's method and the two-coin algorithm. In Section (3), we present the theory for the portkey Barker's method. Section (5) deals with the theory for the flipped portkey two-coin algorithm. Finally, examples showing the applicability and efficiency of these algorithms are shown in Section (5).

2 Barker's method and the two-Coin algorithm

Based on the Barker's acceptance probability already defined in (1), we present the Barker's update for a realization at time $m+1$ in Algorithm (2), given the current state of the Markov chain to be at x and the proposal density to be $q(x,y)$.

Algorithm 2 Barker's MCMC for x_{m+1}

1: Draw $y \sim \mathrm{q}(x_m, y)$
2: Draw $A \sim \mathrm{Bern}(\alpha_B(x_m, y))$
3: **if** $A = 0$ **then**
4: $\quad x_{m+1} = x_m$
5: **if** $A = 1$ **then**
6: $\quad x_{m+1} = y$

Usually, in Step (2) of Algorithm (2), we draw $\mathrm{U} \sim \mathscr{U}(0,1)$ and then check if $\mathrm{U} \leq \alpha_B(x,y)$ and then accept or reject the proposed value accordingly. But we can't always calculate $\alpha_B(x,y)$, like in scenarios mentioned in Section 1. Gonçalves et al. (2017a) noticed that a Bernoulli factory can be constructed to obtain events of probability $\alpha_B(x,y)$ without needing to calculate it explicitly and proposed the

following.

Suppose,

$$\pi(x)q(x,y) = c_x p_x,$$
$$\pi(y)q(y,x) = c_y p_y$$

where c_x and c_y are known and p_x and p_y are between 0 and 1. For some cases, c_x and p_x could depend on both x and y; for notational purpose we disregard y. We arrive at c_x and p_x through the following:

$$\pi(x)q(x,y) \leq c_x \text{ and then set } p_x = \pi(x)q(x,y)c_x^{-1}. \tag{2}$$

On using the similar statements for c_y and p_y, the *2-coin* algorithm of Gonçalves et al. (2017a),Gonçalves et al. (2017b) returns events with probability $\alpha_B(x,y)$ with:

$$h(p_x,p_y) = \frac{c_y p_y}{c_y p_y + c_x p_x} = \alpha_B(x,y).$$

The 2-coin algorithm is given in Algorithm (3).

Algorithm 3 The 2-coin algorithm for $\alpha_B(x,y)$

1: Draw $C_1 \sim Ber\left(\frac{c_y}{c_x+c_y}\right)$.
2: **if** $C_1 = 1$ **then**
3: Draw $C_2 \sim Ber(p_y)$.
4: **if** $C_2 = 1$ **then**
5: output 1.
6: **else**
7: go to Step 1.
8: **if** $C_1 = 0$ **then**
9: Draw $C_2 \sim Ber(p_x)$.
10: **if** $C_2 = 1$ **then**
11: output 0.
12: **else**
13: go to Step 1.

We can know both c_x and c_y upto a normalizing constant and assume that p_x and p_y can be simulated.

Theorem 3. *Algorithm (3) yields output 1 with probability* $\alpha_B(x,y)$.

Proof. Let r be the probability of no output in any given loop of the algorithm. Then we can write

$$r = \frac{c_y(1-p_y) + c_x(1-p_x)}{c_x + c_y}$$
$$\Rightarrow \sum_{i=0}^{\infty} r^i = \frac{c_x + c_y}{c_x p_x + c_y p_y}.$$

For the output to be 1, $\forall\, i$ there will be no output before the ith loop and the ith loop will output 1 with probability $\frac{c_y p_y}{c_x + c_y}$. Thus, the probability that the algorithm outputs 1 is

$$\frac{c_y p_y}{c_x + c_y} \sum_{i=0}^{\infty} r^i = \frac{c_y p_y}{c_x p_x + c_y p_y} = \alpha_B(x,y).$$

□

Additionally, the number of loops until the algorithm stops is distributed as a $Geom((c_y p_y + c_x p_x)/(c_y + c_x))$ and hence, the mean execution time is

$$\frac{c_x + c_y}{c_x p_x + c_y p_y} = \frac{c_x + c_y}{\pi(x)q(x,y) + \pi(y)q(x,y)}.$$

We see that the efficiency of the above algorithm depends heavily on the upper bounds c_x and c_y. If the bound is loose, then the algorithm returns high execution time.

3 Portkey Barker's Method

Due to the inefficiency of the two-coin algorithms , a new family of acceptance probabilities is introduced by Vats et al. (2021) along with an efficient Bernoulli factory for members of that family. Say the proposal density is $q(x,y)$, then the acceptance probability for a proposed value y is

$$\alpha(x,y) = \frac{\pi(y)q(y,x)}{\pi(x)q(x,y) + \pi(y)q(y,x) + d(x,y)} \tag{3}$$

where $d(x,y) = d(y,x) > 0$. We prove that $\alpha(x,y)$ gives a π-reversible Markov chain given that the function $d(x,y)$ is symmetric.

Theorem 4. *For a proposal density $q(x,y)$, a Markov chain with acceptance probability α in (3) is π-reversible if and only if $d(x,y) = d(y,x)$.*

Proof. This theorem follows from the fact that an acceptance function results into a π-reversible Markov chain if and only if

$$\pi(y)q(y,x)\alpha(y,x) = \pi(x)q(x,y)\alpha(x,y).$$

For $d(x,y) = d(y,x)$, we can see that

$$\pi(x)q(x,y)\alpha(x,y) = \frac{\pi(y)q(y,x)\pi(x)q(x,y)}{\pi(x)q(x,y)+\pi(y)q(y,x)+d(x,y)} = \pi(y)q(y,x)\alpha(y,x)$$

□

As clearly $\alpha(x,y) \leq \alpha_B(x,y)$, Barker's method is more efficient due to Peskun's ordering (Peskun (1973)). Vats et al. (2021) propose $d(x,y) = \frac{(1-\beta)}{\beta}(c_x + c_y)$ where c_x and c_y are as defined earlier. This choice of $d(x,y)$ results in significant computational gains which supersedes the loss of statistical efficiency. So for $0 < \beta \leq 1$, the *portkey Barker's* acceptance probability is

$$\alpha_{(\beta)}(x,y) := \frac{\pi(y)q(y,x)}{\pi(x)q(x,y)+\pi(y)q(y,x)+\frac{(1-\beta)}{\beta}(c_x+c_y)}. \tag{4}$$

We want $d(x,y)$ to be small, so we take $\beta \approx 1$ in order to prevent much drop in statistical efficiency. The two-coin algorithm is modified to get events with probability $\alpha_{(\beta)}(x,y)$ and is named as the *portkey* method. Algorithm (4) takes the two-coin algorithm and adds another step at the beginning to allow for immediate rejections with probability $1-\beta$. We can observe that if we run Algorithm (4) with $\beta \approx 1$, then we can avoid the large number of loops that are common in Algorithm (3).

Theorem 5. *Algorithm (4) yields output 1 with probability* $\alpha_{(\beta)}(x,y)$.

Proof. Let r be the probability of no output in any given loop of the algorithm. Then we can write

$$r = \beta\frac{c_y(1-p_y)+c_x(1-p_x)}{c_x+c_y}$$
$$\Rightarrow \sum_{i=0}^{\infty} r^i = \frac{c_x+c_y}{(1-\beta)(c_x+c_y)+\beta(c_xp_x+c_yp_y)}.$$

For the output to be 1, $\forall\, i$ there will be no output before the ith loop and the ith loop will output 1 with probability $\beta\frac{c_yp_y}{c_x+c_y}$. Thus, the probability that the algorithm outputs 1 is

$$\beta\frac{c_yp_y}{c_x+c_y}\sum_{i=0}^{\infty} r^i = \frac{c_yp_y}{c_xp_x+c_yp_y+\frac{1-\beta}{\beta}(c_x+c_y)}.$$

which is equal to $\alpha_{(\beta)}(x,y)$. □

Algorithm 4 Portkey two-coin algorithm

Draw $S \sim \text{Bernoulli}(\beta)$
if $S = 0$ **then**
    output 0
if $S = 1$ **then**
    Draw $C_1 \sim \text{Bern}\left(\frac{c_y}{c_x + c_y}\right)$
    if $C_1 = 1$ **then**
        Draw $C_2 \sim \text{Bern}(p_y)$
        if $C_2 = 1$ **then**
            output 1
        if $C_2 = 0$ **then**
            go to Step 1
    if $C_1 = 0$ **then**
        Draw $C_2 \sim \text{Bern}(p_x)$
        if $C_2 = 1$ **then**
            output 0
        if $C_2 = 0$ **then**
            go to Step 1

We have seen that the portkey Barker's algorithm is π-reversible as proven in Theorem (4). Additionally, it is π-ergodic as well since $\alpha_{(\beta)}(x,y) > 0$. The number of loops until Algorithm (4) stops is distributed according to a Geom(s_β), where

$$s_\beta = (1-\beta) + \beta \cdot \frac{c_y p_y + c_x p_x}{c_x + c_y}.$$

For $0 < \beta < 1$, $s_\beta > 1 - \beta$, which is bounded away from zero and hence, the mean execution time is bounded above. This argument does not hold for Barker's original two-coin algorithm. We can compare the ratio of the mean execution time of the two-coin algorithm to the portkey two-coin algorithm; it is

$$\frac{1/s_1}{1/s_\beta} = \frac{s_\beta}{s_1} = (1-\beta) \cdot \left(\frac{c_y p_y + c_x p_x}{c_x + c_y}\right)^{-1} + \beta \;\geq\; 1\,.$$

Here, we can see that if $(c_y p_y + c_x p_x)/(c_x + c_y) \approx 0$, implying that the original two-coin algorithm is highly inefficient, the ratio of the mean execution times diverges to infinity. On the other hand, if $(c_y p_y + c_x p_x)/(c_x + c_y) \approx 1$, implying that the original two-coin algorithm is efficient, the two algorithms have comparable expected number of loops.

As we have stated earlier, the computational efficiency of the portkey method comes at the cost of statistical efficiency. To demonstrate this, let $P_{(\beta)}$ and P_{B} denote the Markov kernels for the portkey Barker's and Barker's algorithms. In addition, for a function g, let $\bar{g}_n$ denote the Monte Carlo estimator of $\int g\pi(dx)$ obtained using a Markov kernel P and denote $\mathrm{var}(g,P) \coloneqq \lim_{n\to\infty} n\mathrm{Var}_\pi(\bar{g}_n)$.

Theorem 6. *For $0 < \beta \leq 1$, $\alpha_{(\beta)}(x,y) \leq \beta\, \alpha_B(x,y)$. As a consequence,*

$$var(g,P_B) \leq \beta\, var(g,P_{(\beta)}) + (\beta-1)Var_\pi(g)\,.$$

Proof. Since $c_x + c_y \geq \pi(x)q(x,y) + \pi(y)q(y,x)$

$$\begin{aligned}
\alpha_{(\beta)}(x,y) &= \frac{\pi(y)q(y,x)}{\pi(x)q(x,y)+\pi(y)q(y,x)+\beta^{-1}(1-\beta)(c_x+c_y)} \\
&\leq \beta\cdot\frac{\pi(y)q(y,x)}{\beta\pi(x)q(x,y)+\beta\pi(y)q(y,x)+(1-\beta)(\pi(x)q(x,y)+\pi(y)q(y,x))} \\
&= \beta\cdot\alpha_{\mathrm{B}}(x,y)\,.
\end{aligned}$$

Combining Łatuszyński and Roberts (2013) (Corollary 1) and the ordering of Peskun (1973) yields,

$$\mathrm{var}(g,P_{\mathrm{B}}) \leq \beta\,\mathrm{var}(g,P_{(\beta)}) + (\beta-1)\mathrm{Var}_\pi(g)\,.$$

□

Under specific conditions, a variance bound in the opposite direction is given in Theorem (7).

Theorem 7. *For $0 < \beta \leq 1$, if there exists $\delta > 0$ such that $p_x > \delta$ and $p_y > \delta$, then*

$$\alpha_B(x,y) \leq \left(1+\frac{1-\beta}{\delta\beta}\right)\cdot\alpha_{(\beta)}(x,y)\,.$$

As a consequence,

$$var\left(g,P_{(\beta)}\right) \leq \left(1+\frac{1-\beta}{\delta\beta}\right) var(g,P_B) + \frac{1-\beta}{\delta\beta}Var_\pi(g)\,.$$

Proof. Since $p_x \geq \delta$ and $p_y \geq \delta$, $c_x + c_y \leq (\pi(x)q(x,y)+\pi(y)q(y,x))/\delta$. So,

$$\begin{aligned}
\alpha_{(\beta)}(x,y) &\geq \frac{\pi(y)q(y,x)}{\pi(x)q(x,y)+\pi(y)q(y,x)+\frac{1-\beta}{\delta\beta}(\pi(x)q(x,y)+\pi(y)q(y,x))} \\
&= \left(1+\frac{1-\beta}{\delta\beta}\right)^{-1}\alpha_{\mathrm{B}}(x,y)\,.
\end{aligned}$$

The variance ordering follows from Łatuszyński and Roberts (2013) (Corollary 1). □

When p_x or p_y is small: δ is small, and hence it is desirable to set β to be large. Theorem (7) suggests that setting $\beta = 1-\delta$ is desirable. However, δ will typically not be available.

4 Flipped portkey two-coin algorithm

Often in problems involving target densities restricted to a constrained space, it is easier to compute a lower bound for $\pi(x)q(x,y)$ rather than its upper bound. To tackle such problems, algorithms that incorporate lower bound of $\pi(x)q(x,y)$, or upper bound of $\pi(x)^{-1}q(x,y)^{-1}$, are more desirable. This motivates Vats et al. (2021) to introduce another acceptance probability $\alpha_{f,(\beta)}(x,y)$ of the form

$$\alpha_{f,(\beta)}(x,y) := \frac{\pi(y)q(y,x)}{\pi(y)q(y,x)+\pi(x)q(x,y)+\dfrac{(1-\beta)}{\beta}\dfrac{(\tilde{c}_x+\tilde{c}_y)}{\tilde{c}_x\tilde{p}_x\tilde{c}_y\tilde{p}_y}},$$

where, for $\tilde{c}_x > 0$, and $0 < \tilde{p}_x < 1$, $\pi(x)^{-1}q(x,y)^{-1} = \tilde{c}_x\tilde{p}_x$.

Similar to algorithm 4, the acceptance probability $\alpha_{f,(\beta)}(x,y) \leq \alpha_B(x,y)$. Thus, by Peskun's ordering Peskun (1973), Barker's method is more efficient, but this choice of $d(x,y) = \dfrac{(\tilde{c}_x+\tilde{c}_y)}{\tilde{c}_x\tilde{p}_x\tilde{c}_y\tilde{p}_y}$ results in significant computational gains that compensate for the statistical inefficiencies. Thus, the *Flipped Portkey* method modifies algorithm (4) such that the algorithm returns output 1 with probability $\alpha_{f,(\beta)}(x,y)$.

Theorem 8. *Algorithm (5) yields output 1 with probability* $\alpha_{f,(\beta)}(x,y)$.

Proof. Here we have $\pi(x)^{-1}q(x,y)^{-1} = \tilde{c}_x\tilde{p}_x$. Therefore, similar to the proof of

Algorithm 5 Flipped Portkey two-coin algorithm

Draw $S \sim \text{Bernoulli}(\beta)$
if $S = 0$ **then**
    output 0
if $S = 1$ **then**
    Draw $C_1 \sim \text{Bern}\left(\frac{\tilde{c}_x}{\tilde{c}_x + \tilde{c}_y}\right)$
    if $C_1 = 1$ **then**
        Draw $C_2 \sim \text{Bern}(\tilde{p}_x)$
        if $C_2 = 1$ **then**
            output 1
        if $C_2 = 0$ **then**
            go to Step 1
    if $C_1 = 0$ **then**
        Draw $C_2 \sim \text{Bern}(\tilde{p}_y)$
        if $C_2 = 1$ **then**
            output 0
        if $C_2 = 0$ **then**
            go to Step 1

theorem 4, we obtain the probability of output 1 as:

$$\begin{aligned}
\beta \frac{\tilde{c}_x \tilde{p}_x}{\tilde{c}_x + \tilde{c}_y} \sum_{i=0}^{\infty} r^i &= \frac{\tilde{c}_x \tilde{p}_x}{\tilde{c}_x \tilde{p}_x + \tilde{c}_y \tilde{p}_y + \frac{1-\beta}{\beta}(\tilde{c}_x + \tilde{c}_y)} \\
&= \frac{\frac{1}{\tilde{c}_y \tilde{p}_y}}{\frac{1}{\tilde{c}_y \tilde{p}_y} + \frac{1}{\tilde{c}_x \tilde{p}_x} + \frac{1-\beta}{\beta} \frac{\tilde{c}_x + \tilde{c}_y}{\tilde{c}_x \tilde{p}_x \tilde{c}_y \tilde{p}_y}} \\
&= \frac{\pi(y) q(y,x)}{\pi(y) q(y,x) + \pi(x) q(x,y) + \frac{1-\beta}{\beta} \frac{\tilde{c}_x + \tilde{c}_y}{\tilde{c}_x \tilde{p}_x \tilde{c}_y \tilde{p}_y}} \\
&= \alpha_{f,(\beta)}(x,y)
\end{aligned}$$

□

Similar to the portkey algorithm, flipped portkey two-coin algorithm is also $\pi-$reversible and computationally more efficient compared to the Barker's method. Its computational efficiency compensated for the loss in statistical efficiency caused by

$\alpha_{f,(\beta)}(x,y) \leq \alpha_B(x,y)$.

Similar to the theorem (6) in the portkey method, variance ordering for the flipped portkey Barker's acceptance probability can be obtained from the theorem below:

Theorem 9. *For* $0 < \beta \leq 1$, $\alpha_{f,(\beta)}(x,y) \leq \beta\, \alpha_B(x,y)$. *As a consequence,*

$$var(g,P_B) \leq \beta\, var(g,P_{f,(\beta)}) + (\beta-1)Var_\pi(g)\,.$$

Proof. We have,

$$\begin{aligned}\frac{\tilde{c}_x+\tilde{c}_y}{\tilde{c}_x\tilde{p}_x\tilde{c}_y\tilde{p}_y} &= \frac{\pi(y)q(y,x)}{\tilde{p}_x} + \frac{\pi(x)q(x,y)}{\tilde{p}_y} \\ &\geq \pi(y)q(y,x) + \pi(x)q(x,y),\end{aligned}$$

as $0 < \tilde{p}_x < 1$, and $0 < \tilde{p}_y < 1$.

Therefore, the acceptance probability

$$\begin{aligned}\alpha_{f,(\beta)}(x,y) &= \frac{\pi(y)q(y,x)}{\pi(x)q(x,y)+\pi(y)q(y,x)+\beta^{-1}(1-\beta)\dfrac{\tilde{c}_x+\tilde{c}_y}{\tilde{c}_x\tilde{p}_x\tilde{c}_y\tilde{p}_y}} \\ &\leq \beta\cdot\frac{\pi(y)q(y,x)}{\beta\pi(x)q(x,y)+\beta\pi(y)q(y,x)+(1-\beta)(\pi(x)q(x,y)+\pi(y)q(y,x))} \\ &= \beta\cdot\alpha_{\mathrm{B}}(x,y)\,.\end{aligned}$$

Combining Łatuszyński and Roberts (2013) (Corollary 1) and the ordering of Peskun (1973) yields,

$$\mathrm{var}(g,P_{\mathrm{B}}) \leq \beta\,\mathrm{var}(g,P_{(f,\beta)}) + (\beta-1)\mathrm{Var}_\pi(g)\,.$$

□

Theorem 10. *For* $0 < \beta \leq 1$, *if there exists* $\delta > 0$ *such that* $\tilde{p}_x > \delta$ *and* $\tilde{p}_y > \delta$, *then*

$$\alpha_B(x,y) \leq \left(1+\frac{1-\beta}{\delta\beta}\right)\cdot\alpha_{f,(\beta)}(x,y)\,.$$

As a consequence,

$$var\left(g,P_{f,(\beta)}\right) \leq \left(1+\frac{1-\beta}{\delta\beta}\right)var(g,P_B) + \frac{1-\beta}{\delta\beta}Var_\pi(g)\,.$$

Proof. Since $\tilde{p}_x > \delta$ and $\tilde{p}_y > \delta$, similar to theorem 9,

$$\begin{aligned}\frac{\tilde{c}_x + \tilde{c}_y}{\tilde{c}_x\tilde{p}_x\tilde{c}_y\tilde{p}_y} &= \frac{\pi(y)q(y,x)}{\tilde{p}_x} + \frac{\pi(x)q(x,y)}{\tilde{p}_y} \\ &\leq \frac{\pi(y)q(y,x) + \pi(x)q(x,y)}{\delta},\end{aligned}$$

So,

$$\begin{aligned}\alpha_{f,(\beta)}(x,y) &\geq \frac{\pi(y)q(y,x)}{\pi(x)q(x,y) + \pi(y)q(y,x) + \frac{1-\beta}{\delta\beta}(\pi(x)q(x,y) + \pi(y)q(y,x))} \\ &= \left(1 + \frac{1-\beta}{\delta\beta}\right)^{-1} \alpha_{\mathrm{B}}(x,y)\,.\end{aligned}$$

The variance ordering follows from Łatuszyński and Roberts (2013) (Corollary 1). □

5 Example applications

5.1 Gamma Mixture of Weibulls

We consider a target distribution of the form $\pi(\theta) = \int \pi(\theta|\lambda)\nu(d\lambda)$, where ν is a mixing measure on λ. We will assume that $\theta|\lambda$ follows a Weibull distribution with λ as a scale parameter and k as a known shape parameter. We take proposal distribution as the normal distribution, centered at the current step with variance .001. Now, want to calculate p_θ and c_θ. We can write $\pi(\theta) = c_\theta \cdot \pi(\theta)/c_\theta$. We assign c_θ as $c_\theta := \pi(\theta|\lambda) \leq k/(e\theta)$ and we can simulate events of probability $p_\theta = \pi(\theta)/c_\theta$.

Moreover, we would draw $\lambda \sim \nu$ and $U \sim \mathscr{U}[0,1]$ independently. Then $Pr\{U \leq \pi(\theta|\lambda)/c_\theta\} = p_\theta$. We set ν = Gamma(10, 100) and $k = 10$. We run both Barker's and portkey Barker's algorithms for 10^5 steps using the two Bernoulli factories for various values of β.
We repeat the simulation 1000 times to compare the performances. We observe that effective sample size (ESS) (Gong and Flegal (2016)) decreases as β decreases. We also observe that ESS per second is significantly higher for portkey Barker's compared to Barker's. This is due to large mean execution time of the two coin algorithm. Moreover, the two-coin algorithm's loops demonstrates heavy tailed behaviour where the average maximum number of loops for MCMC run is 1.3 million.

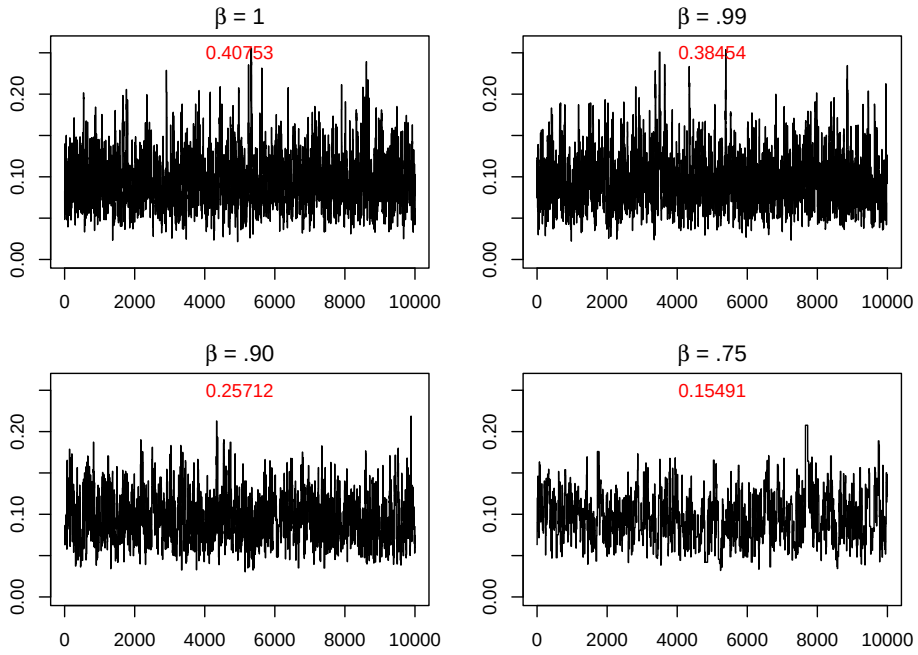


Figure 1: Trace plots of the chains for four values of β; the acceptance probabilities are 0.40, 0.38, 0.26 and 0.16 for $\beta = 1$, $\beta = 0.99$, $\beta = 0.9$ and $\beta = 0.75$ respectively.

Definition 4 (**Effective Sample Size (ESS)**, Gong and Flegal (2016))**.** *An estimate of the sample size required to achieve the same level of precision if that sample was a simple random sample.*

We repeat the simulation 1000 times to compare the performances. We observe that effective sample size (ESS) (Gong and Flegal (2016)) decreases as β decreases. We also observe that ESS per second is significantly higher for portkey Barker's compared to Barker's. This is due to large mean execution time of the two coin algorithm. Moreover, the two-coin algorithm's loops demonstrates heavy tailed behaviour where the average maximum number of loops for MCMC run is 1.3 million.

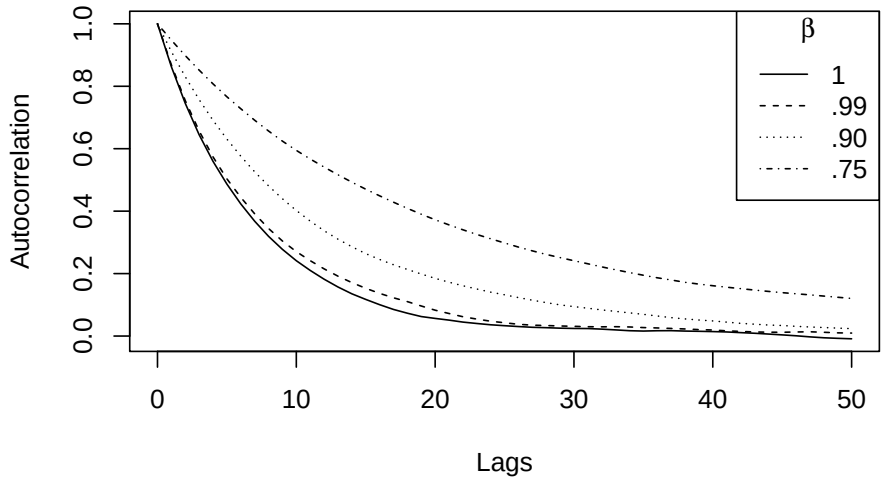


Figure 2: Autocorrelation plots for a run-length of 10^5 for four β values: 1 (solid), 0.99 (dashed), 0.90 (dotted) and 0.75 (dot-dashed).

5.2 MCMC on constrained spaces

Let $f(y|\theta)$ be a likelihood and $\pi(\theta|\eta)$ be the prior on θ constrained in a set $\mathscr{A}$, so that for a function $g(\cdot|\eta)$,

$$\pi(\theta|\eta) = \frac{g(\theta|\eta)I(\theta \in \mathscr{A})}{\int_{\mathscr{A}} g(\theta|\eta)d\theta},$$

where $\int_{\mathscr{A}} g(\theta|\eta)d\theta$ is not tractable. For a full Bayesian model, we assign a hyperprior to η, and this leads to intractability in the posterior distribution for (θ,η) making this an issue. To illustrate the applicability of the methods introduced by Vats et al. (2021), we consider the following model as an example.

Suppose $y_1,\ldots,y_n|R \overset{iid}{\sim} N(0,R)$ where R is a $p \times p$ correlation matrix. Assume the unique elements in R, r_{ij}, to be normally distributed, restricted to R being positive-

Table 1: Averaged results from 1000 replications. Standard errors are in brackets.

β	1	0.99	0.90	0.75
ESS	7484(7.74)	6939(12.18)	4320(13.86)	2501(9.19)
ESS/s	251.40(2.68)	616.60(3.5)	717.79(4.59)	658.05(4.27)
Mean loops μ	32.00(3.7)	7.63(0)	3.97(0)	2.55(0)
Max loops μ	1315683	604(3.44)	78(0.37)	32(0.12)

definite. We get the following prior on R:

$$f(R \mid \mu, \sigma^2) = L(\mu, \sigma^2) \prod_{i<j} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ -\frac{(r_{ij}-\mu)^2}{2\sigma^2} \right\} \mathbb{I}\{R \in S_p^+\}, \text{ where}$$

$$L^{-1}(\mu, \sigma^2) = \int_{R \in S_p^+} \prod_{i<j} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ -\frac{(r_{ij}-\mu)^2}{2\sigma^2} \right\} dr_{ij}$$

where S_p^+ is the set of $p \times p$ positive-definite matrices and $L(\mu, \sigma^2)$ is the normalising constant which is typically not known in closed form. For known τ^2, a_0, b_0, hyperpriors for $\mu \sim N(0, \tau^2)$ and $\sigma^2 \sim IG(a_0, b_0)$ are chosen. To implement a component-wise Metropolis-within-Gibbs sampler, Let $l = p(p-1)/2$. Then full conditional densities are

$$f(r_{ij} \mid r_{-ij}, \mu, \sigma^2) \propto |R|^{-n/2} \exp\left\{ -\frac{\mathrm{tr}(R^{-1}Y^TY)}{2} \right\} \exp\left\{ -\frac{(r_{ij}-\mu)^2}{2\sigma^2} \right\} \mathbb{I}\{l_{ij} \le r_{ij} \le u_{ij}\},$$

$$f(\mu \mid R, \sigma^2) \propto L(\mu, \sigma^2) \prod_{i<j} \exp\left\{ -\frac{(r_{ij}-\mu)^2}{2\sigma^2} \right\} \exp\left\{ -\frac{\mu^2}{2\tau^2} \right\} := L(\mu, \sigma^2) g(\mu, R, \sigma^2),$$

$$f(\sigma^2 \mid R, \mu) \propto L(\mu, \sigma^2) \prod_{i<j} \exp\left\{ -\frac{(r_{ij}-\mu)^2}{2\sigma^2} \right\} \left(\frac{1}{\sigma^2} \right)^{a_0 + l/2 + 1} \exp\left\{ -\frac{b_0}{\sigma^2} \right\},$$

where the indicator variable $\mathbb{I}\{l_{ij} \le r_{ij} \le u_{ij}\}$ ensures positive-definiteness of R. We use Metropolis-Hastings update with a Gaussian proposal for each r_{ij}.
Updating μ and σ^2 requires the knowledge of $L(\mu, \sigma^2)$ which is unavailable. Vats et al. (2021) compare performance for three methods in this example. They use a shadow prior technique that interjects the Bayesian hierarchy such that $r_{ij} \sim N(\delta_{ij}, v^2)$ and $\delta_{ij} \sim N(\mu, \sigma^2)$, with unchanged hyper-priors on μ and σ^2. The resulting marginal posterior of (R, μ, σ^2) is different from the original model. The full conditionals of μ and σ^2 are available in closed-form, but the full conditionals for δ are intractable. However, Liechty et al. (2009) argue that for updating δ_{ij}, if $v^2 \approx 0$, the intractable constants can be ignored. Thus, this methodology is although convenient, but it is not asymptotically exact since the resulting Markov chain targets an approximation of a modified desired distribution.

The flipped portkey two-coin algorithm can easily be implemented for both μ and σ^2 and we only present details for μ. We use Gaussian random walk proposals for both components so that we can ignore the proposal density in the Bernoulli factories.
We can obtain a tight upper bound by noting that each $r_{ij} \in [-1,1]$ and thus

$$g(\mu,R,\sigma^2)^{-1}L^{-1}(\mu,\sigma^2) \leq g(\mu,R,\sigma^2)^{-1}\left[\Phi\left(\sigma^{-1}(1-\mu)\right)-\Phi\left(\sigma^{-1}(-1-\mu)\right)\right]^l := \tilde{c}_\mu$$

Set $f(\mu|R,\sigma^2)^{-1} = \tilde{c}_\mu \tilde{c}_\mu^{-1} f(\mu \mid R,\sigma^2)^{-1} := \tilde{c}_\mu \tilde{p}_\mu$. To generate coins of probability $\tilde{p}_\mu$: draw $z_{ij} \sim \text{TN}(-1,1,\mu,\sigma^2)$ for $i<j$, and construct matrix Z with z_{ij} as the lower-triangular entries. If Z is positive-definite, return 1, else return 0. Notice here that the use of the flipped portkey algorithm over the portkey algorithm makes it much easier to simulate the p coin since $L^{-1}(\mu,\sigma^2)$ takes the form of an integral.
They also use Barker's method for the simulation by taking $\beta = 1$ in the flipped portkey algorithm.
The dataset used is the daily closing prices of major European stocks: Germany DAX, Switzerland SMI, France CAC, and United Kingdom FTSE on each day, not including weekends and holidays from 1991-1998. The data are available in the `datasets` R package and has 1860 observations. The goal is to estimate the correlation matrix of the four stock prices using the model specified above.
We set $\beta = .90$ for both the updates for μ and σ^2 and first compare the estimated posterior density of μ and σ^2 from a run of length 10^5 for the shadow prior method, flipped portkey method and Barker's method. Figure 3 presents the results, where it can be seen that the shadow prior method yields a biased posterior distribution for both μ and σ^2.

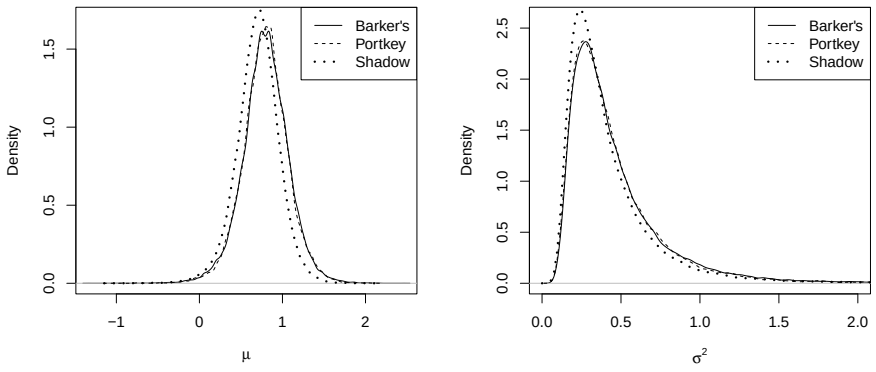


Figure 3: Estimated posterior density of μ and σ^2 from Markov chain lengths of 105 with v2 = 0.001 as recommended Chen et al. (2010) for Barker's (solid), portkey (dashed) and shadow (dotted).

Figure 4 presents the autocorrelation and trace plots for standard Barker's ($\beta = 1$) and the flipped portkey Barker's ($\beta = .90$). Portkey Barker's algorithm leads to only slightly

slower mixing of the chain. However, the (log of) the number of Bernoulli factory loops required for the portkey two-coin algorithm is far less. Particularly for updating μ, the original two-coin algorithm requires an exponentially large number of Bernoulli factory loops whenever unlikely values are proposed.

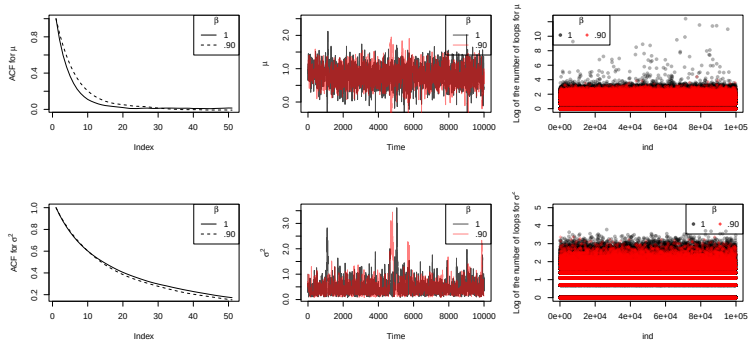


Figure 4: Autocorrelation, trace and log of Bernoulli factory loops for Barker's and portkey Barker's for μ (right), $\beta = 1$(grey), $\beta = 0.9$ (red), and σ^2 (left), $\beta = 1$(solid), $\beta = 0.90$ (dashed).

We repeat the above experiment 10 times for MCMC runs of length 10^4. They could do only 10 runs as the original two-coin algorithm often got stuck in a loop for days. The results are presented in Table 2 with estimated ESS calculated using the `R` package `mcmcse`. Although a regular Barker's implementation yields a slightly higher ESS, the ESS per second for the flipped portkey Barker's algorithm is about 1.5 times higher on average. As further demonstrated in the table, this is clearly a consequence of the large number of loops required for the two-coin algorithm for updating μ.

Table 2: Averaged results from 1000 replications. Standard errors are in brackets.

β	1	.90
ESS	542	496
ESS/s	4.297	6.902
Mean loops μ	218.43	2.99
Mean loops σ^2	3.21	2.49
Max loops μ	2084195	34
Max loops σ^2	38	27

They use the same proposal distributions for both Markov chains for a fair comparison. Since the original two-coin algorithm gets stuck in seemingly infinite loops for proposals

encouraging larger jumps, Vats et al. (2021) were unable to tune the Barker's algorithm to the recommended 15.8% acceptance rate of Agrawal et al. (2021). One run of approximately optimally tuned Barker's two-coin algorithm did not produce even 10^3 samples in 24 hours, whereas the same proposal for the portkey two-coin algorithm yielded 10^4 draws in ≈ 40 seconds with an ESS of 514.
In general, the flipped two-coin algorithm with $\beta = 1$ may work reasonably well if the proposal distribution makes small jumps. This, of course, increases the autocorrelation of the resulting Markov chain. If jump sizes are increased, the original two-coin algorithm will require a large number of loops in the Bernoulli factory. The portkey trick stabilizes this behavior, with little loss in statistical efficiency, and often an overall gain in the robustness of the algorithm.

5.3 Bayesian inference for the Wright-fisher diffusion

In this section, we look into a particular family of diffusion models called the Wright-fisher family of diffusions which is widely used in genetics to model the evolution of the frequency of a genetic variant or allele, in a large randomly mating population. A diffusion process is a solution to a stochastic differential equation. It is a continuous-time Markov process where the sample paths are continuous almost surely.

Gonçalves et al. (2017b), Gonçalves et al. (2017a) demonstrate that their two-coin Barker's MCMC algorithm performs exact inference for diffusions without requiring certain practically infeasible and restrictive model restrictions on diffusion processes.

We show here that the Portkey two-coin Barker's MCMC proposed by Vats et al. (2021) will typically provide considerable gain when compared to the original two-coin Barker's, allowing for a wider applicability of the proposed methodology in terms of model complexity and data size.

We consider the neutral Wright–Fisher family of diffusions with mutations

$$dY_s = 0.5(\theta_1(1-Y_s) - \theta_2 Y_s) + \sqrt{Y_s}(1-Y_s)dW_s, Y_0 = y_0, \theta_1, \theta_2 > 0, \tag{5}$$

where W_s is a standard Brownian motion. We define what a standard Brownian motion is following definition.

Definition 5. *A standard Brownian motion is a random process $\boldsymbol{W} = \{W_t\}_{t\in[0,\infty]}$ with state space $\mathbb{R}$ that satisfies the following properties:*

1. *$W_0 = 0$ (w.p. 1).*
2. *The map $t \mapsto W_t$ is continuous in t w.p. 1.*
3. *$\boldsymbol{W}$ has independent, stationary increments.*
4. *$W_t \sim \mathcal{N}(0,1)$ for each $t \in (0,\infty)$.*

The process $Y \in [0,1]$. It is assumed that Y is discretely observed at a finite set of time-points, $0 = t_0, t_1, \ldots, t_n = T$ so that the observed data are $Y_obs = (y_0, y_1, \ldots, y_n)'$. Here, (θ_1, θ_2), which describe the drift coefficient of the process, are the parameters of interest. Although Bayesian estimation yields desirable theoretical properties (see Sant et al. (2020)), their transition densities $P(\gamma|y_{i-1}, y_i) \stackrel{def}{=} P_\gamma(Y_{t_i} \in dy_i | Y_{t_i-1} = y_{i-1})/dy_i$, which determine the likelihoods, are not available in closed form and are characterized by poor numerical properties at the boundaries (Jenkins and Spano (2017)). These undermine reliable Bayesian inference based on approximations and hence, provides another motivation for doing exact inference.

For the purpose of inference, we first reparametrize θ_1 and θ_2 to γ_1 and γ_2, where $\gamma_1 = \theta_1 + \theta_2$ and $\gamma_2 = \theta_1/(\theta_1 + \theta_2)$, with uniform priors on $\gamma = (\gamma_1, \gamma_2)$. The first step is to apply the Lamperti transform to Y in (5). This transformation basically transforms Y to a unit diffusion coefficient process, X. Next, Gibbs sampling is used for updating γ and updating the missing paths, X_{mis}, of the transformed diffusion X given the transformed observations X_{obs}. X_{mis} is required as we have discretely observed diffusion process. We use standard Brownian bridge proposals restricted to [0, 1] for the X_{mis} update and implement both Barker's and portkey Barker's MCMC. We define what a Brownian bridge is in the following definition.

Definition 6. *A Brownian bridge is a continuous-time stochastic process $W^*(t)$ whose probability distribution is the conditional probability distribution of a standard Brownian motion W(t) given the condition (when standardized) that $W(T) = 0$, so that the process is pinned to the same value at both $t = 0$ and $t = T$.*
Mathematically:

$$W_t^* := (W_t \mid W_T = 0),\ t \in [0, T].$$

The Brownian bridges are sampled using a layered Brownian bridge construction. This allows feasible lower and upper bounds on the likelihood to be obtained which are important for an efficient Bernoulli factory, as earlier stated.

Component-wise updates are done for γ: $\gamma_1 \sim \mathscr{U}(\gamma_1 - 0.3, \gamma_2 + 0.3)$ and $\gamma_2 \sim \mathscr{U}(\gamma_2 - 0.01, \gamma_2 + 0.01)$. The specific choices of 0.01 and 0.3 are such that the two-coin Barker's algorithm doesn't get trapped in very long loops.Gonçalves et al. (2017a) show that $\pi(\gamma|X_{mis}, Xobs) = c_\gamma p_\gamma$, where they provide the forms of c_γ and p_γ.The layer refinement strategy (see Gonçalves et al. (2017b) for details) is used to obtain a realization of p_γ.Although without the refinement, the bound c_γ is too loose and thus, p_γ is often too small for the Bernoulli factory to be efficient, finer refinements require simulating more layered Brownian bridges. As expected, this leads to much larger computation times to even draw from the proposal distribution. It is precisely this characteristic of the sampling process that dictates the superior performance of the portkey two-coin algorithm (Vats et al. (2021)).

We simulate a Wright–Fisher diffusion with $\gamma_1 = 8$ and $\gamma_2 = 0.5$, and observe the process at times $\{0, 1, ..., 50\}$. We obtain a sample from the posterior distribution of (γ_1, γ_2) via the methodology described above. The two samplers are run for $10,000$ steps. For

updating the parameters, the βs are set to be 0.99995 and 0.9995 for for γ_1 and γ_2, respectively. Figure (5) compares the number of loops of the two Bernoulli factories for each iteration of the two Markov chains for both γ_1 and γ_2.

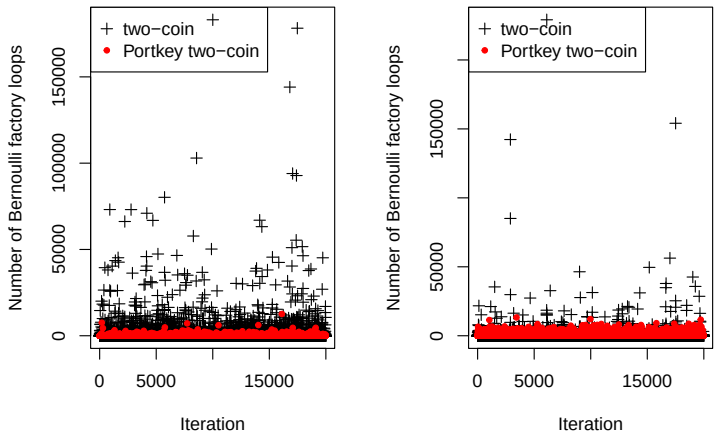


Figure 5: Comparing the number of loops of the Bernoulli factory for two-coin (cross) and portkey (red dot) two-coin algorithm. (a) The γ_1 update. (b) The γ_2 update.

We see that an average of 489 loops with a maximum of 183 077 loops are taken by the γ_1 update of the Barker's two-coin Markov chain. In contrast, the portkey two-coin algorithm runs an average of 43 loops with a maximum of 12 622. However, the computational gain in only with respect to the Bernoulli factories and does not affect the Markov chain considerably as we can see in from the autocorrelation plots in Figure 6. The estimated effective sample sizes for the posterior mean of γ are about 339 and 197, respectively, for Barker's and portkey Barker's. The effective sample sizes per hour, however, are about 9 and 56, respectively. Thus, the portkey Barker's algorithm is approximately 6 times more efficient than the original Barker's algorithm.

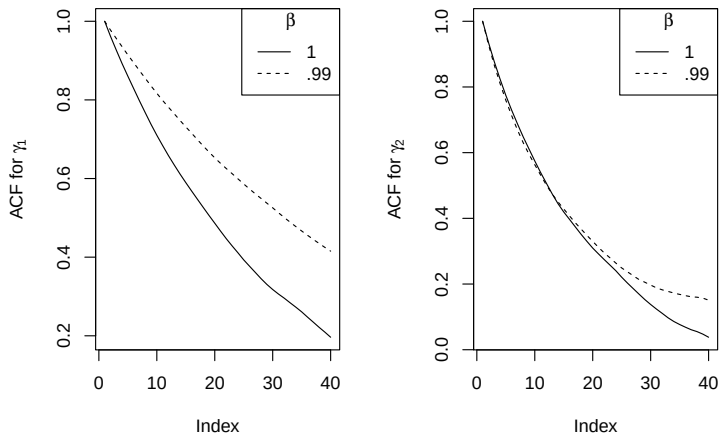


Figure 6: Autocorrelation plots for γ_1 and γ_2; $\beta = 1$ (solid), $\beta = 0.99$ (dashed).

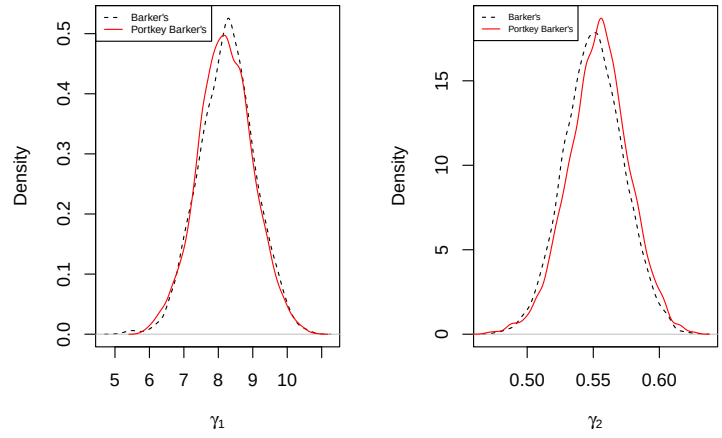


Figure 7: Posterior density estimates of γ_1 and γ_2 using the Barker's and portkey Barker's MCMC method.

We show the posterior density estimates of γ_1 and γ_2 using the Barker's and portkey Barker's method in Figure (7). We can see that the estimates obtained via the portkey's Barker's method is almost as good as that obtained using the Barker's MCMC method.

6 Conclusion

In this report, we have shown that the Barker's acceptance probability can be effectively used when the Metropolis-Hastings acceptance probability can't be used due to the unavailability of the functional form of the target distribution up to a normalizing constant. We have shown how the computational efficiency of the two-coin algorithm heavily relies on the bounds c_x and c_y and motivated the development of portkey and flipped portkey Barker algorithms. We have presented their theoretical background and showed via examples that they can be efficiently used in Bayesian models with intractable priors and in Bayesian inference for diffusion processes. In the examples, we have also presented the significant gain in terms of computation in lieu of some negligible statistical efficiency. To implement these algorithms more generally may require reasonable bounds on the target distribution.

An interesting application of the algorithms presented here may be to apply them to a different acceptance probability that lie in between the Barker's acceptance probability and the Metropolis-Hastings acceptance probability. Furthermore, it may be interesting to see the application of these algorithms in high-dimensional settings using the MH algorithm to see the extent of computational efficiency gained.

Application of these algorithms in jump-diffsuion processes would also be worth exploration because Gonçalves et al. (2017a), Gonçalves et al. (2017b) show that their two-coin Barker's MCMC algorithm performs exact inference for jump diffusions without requiring the above conditions. Hence, the portkey Barker should exhibit a gain in computational efficiency as is shown in the case of the Wright-Fisher diffusion process in Example (5.3). For more exact simulation problems for jump-diffusions, we refer the interested reader to Gonçalves and Roberts (2014).

An interesting future work may be to find the optimal acceptance probabilities, $\alpha^*_{(\beta)}(x,y)$ and $\alpha^*_{f,(\beta)}(x,y)$, for dimensions $d \geq 1$ following the works of Roberts and Rosenthal (2001), Agrawal et al. (2021).

7 Supplementary Material

The interested reader is directed to https://github.com/ArkaB-DS/BayesProj which contains all the figures present here in the directory `figures`, the data files and the R codes to generate them in the `Data` and `R codes` directory, respectively.

8 Acknowledgements

We take this opportunity to heartily thank our supervisor Prof. Arnab Hazra for his valuable feedback and constant guidance on this project. The contributions of each of the members for this project are as follows:

1. **Arkajyoti Bhattacharjee** wrote Sections (1), (5.3) and wrote R code for Section (5.3).
2. **Devyanshi Singh** wrote Section (4).
3. **Dhruvil Sangani** wrote Sections (2), (5.1) and wrote R Code for Section (5.1).
4. **Nitin Garg** wrote Sections (3), (5.2) and wrote R code for Section (5.2).
5. **Rishabh Gupta** wrote Section (3).

References

A., B. A. (1965). Monte carlo calculations of the radial distribution functions for a proton-electron plasma. *Computing in Science & Engineering*, pages 119–134.

Agrawal, S., Vats, D., Łatuszyński, K., and Roberts, G. O. (2021). Optimal scaling of mcmc beyond metropolis. *arXiv preprint arXiv:2104.02020*.

Andrieu, C., De Freitas, N., Doucet, A., and Jordan, M. I. (2003). An introduction to MCMC for machine learning. *Machine learning*, 50(1):5–43.

Beichl, I. and Sullivan, F. (2000). The metropolis algorithm. *Computing in Science & Engineering*, 2(1):65–69.

Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. (2011). *Handbook of markov chain monte carlo*. CRC press.

Chib, S. and Greenberg, E. (1995). Understanding the metropolis-hastings algorithm. *The american statistician*, 49(4):327–335.

Gilks, W. R., Richardson, S., and Spiegelhalter, D. (1995). *Markov chain Monte Carlo in practice*. CRC press.

Gonçalves, F. B., Łatuszyński, K., and Roberts, G. O. (2017a). Barker's algorithm for bayesian inference with intractable likelihoods. *Brazilian Journal of Probability and Statistics*, 31(4):732–745.

Gonçalves, F. B., Łatuszyński, K. G., and Roberts, G. O. (2017b). Exact monte carlo likelihood-based inference for jump-diffusion processes. *arXiv preprint arXiv:1707.00332*.

Gonçalves, F. B. and Roberts, G. O. (2014). Exact simulation problems for jump-diffusions. *Methodology and Computing in Applied Probability*, 16(4):907–930.

Gong, L. and Flegal, J. M. (2016). A practical sequential stopping rule for highdimensional markov chain monte carlo. *Computing in Science & Engineering*, 2(1):684–700.

Hastings, W. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109.

Hensman, J., Matthews, A. G. d. G., Filippone, M., and Ghahramani, Z. (2015). Mcmc for variationally sparse Gaussian processes. *arXiv preprint arXiv:1506.04000*.

Jenkins, P. A. and Spano, D. (2017). Exact simulation of the wright–fisher diffusion. *The Annals of Applied Probability*, 27(3):1478–1509.

Kim, W., Navarro, D. J., Pitt, M. A., Myung, I. J., Thrun, S., and Saul, L. (2003). An MCMC-Based Method of Comparing Connectionist Models in Cognitive Science. In *NIPS*, pages 937–944. Citeseer.

Latuszynski, K. (2010). The bernoulli factory, its extensions and applications. *Proceedings of IWAP*, pages 1–5.

Łatuszyński, K. and Roberts, G. O. (2013). Clts and asymptotic variance of time-sampled markov chains. *Methodology and Computing in Applied Probability*, 15(1):237–247.

Liechty, M. W., Liechty, J. C., and Müller, P. (2009). The shadow prior. *Journal of Computational and Graphical Statistics*, 18(2):368–383.

Liu, J. S. and Liu, J. S. (2001). *Monte Carlo strategies in scientific computing*, volume 10. Springer.

Ma, Y.-A., Chen, T., and Fox, E. B. (2015). A complete recipe for stochastic gradient mcmc. *arXiv preprint arXiv:1506.04696*.

Mahendran, N., Wang, Z., Hamze, F., and De Freitas, N. (2012). Adaptive MCMC with Bayesian optimization. In *Artificial Intelligence and Statistics*, pages 751–760. PMLR.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092.

Peskun, P. H. (1973). Optimum monte-carlo sampling using markov chains. *Biometrika*, 60(3):607–612.

Robert, C. P. and Casella, G. (1999). The metropolis—hastings algorithm. In *Monte Carlo Statistical Methods*, pages 231–283. Springer.

Roberts, G. O. and Rosenthal, J. S. (2001). Optimal scaling for various metropolis-hastings algorithms. *Statistical science*, 16(4):351–367.

Sant, J., Jenkins, P. A., Koskela, J., and Spano, D. (2020). Convergence of bayesian estimators for diffusions in genetics. *arXiv preprint arXiv:2001.03527*.

Sharma, S. (2017). Markov chain Monte Carlo methods for Bayesian data analysis in astronomy. *Annual Review of Astronomy and Astrophysics*, 55:213–259.

Sorensen, D., Gianola, D., et al. (2002). Likelihood, bayesian, and mcmc methods in quantitative genetics. *Likelihood, Bayesian, and MCMC methods in quantitative genetics.*

Thrane, E. and Talbot, C. (2019). An introduction to Bayesian inference in gravitational-wave astronomy: parameter estimation, model selection, and hierarchical models. *Publications of the Astronomical Society of Australia*, 36.

Vajargah, K. F., Benis, S. G., and Golshan, H. M. (2021). Detection of the quality of vital signals by the Monte Carlo Markov Chain (mcmc) method and noise deleting. *Health Information Science and Systems*, 9(1):1–10.

Vats, D., Gonçalves, F., Łatuszyński, K., and Roberts, G. (2021). Efficient bernoulli factory markov chain monte carlo for intractable posteriors. *Biometrika*.