Manuscript version: Author’s Accepted Manuscript
The version presented in WRAP is the author’s accepted manuscript and may differ from the published version or Version of Record.

Persistent WRAP URL:
http://wrap.warwick.ac.uk/152153

How to cite:
Please refer to published version for the most recent bibliographic citation information. If a published version is known of, the repository item page linked to above, will contain details on accessing it.

Copyright and reuse:
The Warwick Research Archive Portal (WRAP) makes this work by researchers of the University of Warwick available open access under the following conditions.

Copyright © and all moral rights to the version of the paper presented here belong to the individual author(s) and/or other copyright owners. To the extent reasonable and practicable the material made available in WRAP has been checked for eligibility before being made available.

Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.

Publisher’s statement:
Please refer to the repository item page, publisher’s statement section, for further information.

For more information, please contact the WRAP Team at: wrap@warwick.ac.uk.
Abstract

Accept-reject based Markov chain Monte Carlo (MCMC) algorithms have traditionally utilised acceptance probabilities that can be explicitly written as a function of the ratio of the target density at the two contested points. This feature is rendered almost useless in Bayesian posteriors with unknown functional forms. We introduce a new family of MCMC acceptance probabilities that has the distinguishing feature of not being a function of the ratio of the target density at the two points. We present two stable Bernoulli factories that generate events within this class of acceptance probabilities. The efficiency of our methods rely on obtaining reasonable local upper or lower bounds on the target density and we present two classes of problems where such bounds are viable: Bayesian inference for diffusions and MCMC on constrained spaces. The resulting portkey Barker’s algorithms are exact and computationally more efficient that the current state-of-the-art.
1 Introduction

Markov chain Monte Carlo (MCMC) is a popular tool for drawing samples from complicated distributions. For Bayesian posteriors, where MCMC is most often used, the target density is usually available only up to a proportionality constant. That is, the target density, $\pi(x)$ is such that, $\pi(x) \propto \pi'(x)$, where the functional form of $\pi'$ is usually known. The popular MCMC algorithm of Hastings (1970); Metropolis et al. (1953) has been immensely useful as unknown normalising constants of Bayesian posteriors play no role in the sampler. Given a proposal density $q(x,y)$, a move from $x$ to $y$ in a Metropolis-Hastings (MH) algorithm is accepted with probability

$$\alpha_{\text{MH}}(x,y) = \min \left\{ 1, \frac{\pi(y)q(y,x)}{\pi(x)q(x,y)} \right\} = \min \left\{ 1, \frac{\pi'(y)q(y,x)}{\pi'(x)q(x,y)} \right\}.$$  

On the other hand, there are many other acceptance probability functions that can be used to ensure the stationarity of $\pi$. For instance, Barker (1965) proposed a similar algorithm with

$$\alpha_{\text{B}}(x,y) = \frac{\pi(y)q(y,x)}{\pi(x)q(x,y) + \pi(y)q(y,x)} = \frac{\pi'(y)q(y,x)}{\pi'(x)q(x,y) + \pi'(y)q(y,x)}.$$  

Barker’s algorithm is used less frequently than MH due to the result of Peskun (1973), which establishes that MH is always better than Barker’s algorithm, in terms of asymptotic variance of ergodic averages. However, as shown in Latuszyński and Roberts (2013), the variance of ergodic averages from Barker’s method are no worse than twice that of MH. For this reason, when the MH algorithm is difficult to implement, Barker’s algorithm can be particularly important.

Gonçalves et al. (2017a,b) used Barker’s acceptance probability for Bayesian posteriors with unknown functional forms. Such intractable posteriors can be due to intractable priors on constrained parameter spaces or intractable likelihoods for complex systems which cannot be conveniently modeled tractably. In either case, the Bayesian posterior is not available up to a normalising constant and $\pi(y)/\pi(x)$ cannot be evaluated making it difficult to implement the MH algorithm. There are various solutions to this proposed in the literature; from inexact algorithms like the double Metropolis-Hastings (Liang et al., 2007) to exact ones like pseudo-marginal MCMC (Andrieu and Roberts, 2009). See Park and Haran (2018) for a comprehensive review.

Nearly all methods modify the Markovian dynamics of an underlying accept-reject based MCMC algorithm. We, instead, focus on the use of Bernoulli factories to avoid explicitly calculating the acceptance probabilities, as in Gonçalves et al. (2017a,b). Their two-coin Bernoulli factory generates events of probability $\alpha_{\text{B}}(x,y)$ without explicitly
evaluating it. Thus, an algorithm which evaluates $\alpha_B(x, y)$ explicitly, and one which embeds a Bernoulli factory within each iteration to determine whether to accept a proposed move, are statistically indistinguishable. However, as we will demonstrate, the two-coin Bernoulli factory can be computationally burdensome.

This provides strong motivation to consider other acceptance probabilities that yield $\pi$-stationarity and for which efficient Bernoulli factories can be constructed. To date, it has been customary to consider functions $\alpha(x, y)$ which can be written explicitly in terms of $\pi(y)/\pi(x)$ so that any unknown normalisation constant cancels (see Billera and Diaconis (2001); Peskun (1973); Tierney (1994)). However, when using Bernoulli factories we have far more flexibility. Keeping this in mind we propose an acceptance probability where for $d(x, y) \geq 0$ such that $d(x, y) = d(y, x)$, a move from $x$ to $y$ is accepted with probability

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

We present two choices of $d(x, y)$ for which efficient Bernoulli factories can be constructed to generate events of probability $\alpha(x, y)$. Naturally, $\alpha(x, y) \leq \alpha_B(x, y)$, so the variance ordering of Peskun (1973) applies. However, usually $d(x, y)$ will be small, so as to yield little decrease in efficiency. As we will demonstrate in our examples, any loss in statistical efficiency is made up for in computational efficiency. In addition to a simple illustrative example, Section 5 describes two different directions for applications of the portkey Barker’s method, both stemming from important problems in Bayesian statistics. The first area involves MCMC on constrained spaces where prior normalisation constants are unknown. As an example of this, we give the first exact MCMC method for a well-known Bayesian correlation estimation model. The second class of applications concerns the Bayesian inference for diffusions, which we illustrate on the Wright-Fisher diffusion model. For all examples, our proposed algorithm leads to a far stable sampling process, with significant computational gains.

2 Barker’s method and the two-coin algorithm

Given a current state of the Markov chain $x$, and a proposal density $q(x, y)$, recall the Barker’s acceptance probability in (1). Algorithm 1 presents the Barker’s update for obtaining a realization at time $m + 1$. Usually, Step 2 is implemented by drawing $U \sim U[0, 1]$ and checking if $U \leq \alpha_B(x_m, y)$. However, this is not possible when $\alpha_B(x_m, y)$ cannot be evaluated. Gonçalves et al. (2017a,b) noticed that a Bernoulli factory can be constructed to obtain events of probability $\alpha_B(x_m, y)$ without explicitly evaluating it.
Algorithm 1 Barker’s MCMC for $x_{m+1}$

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

The Bernoulli factory problem is one in which given events that occur with probability $p$, the goal is to simulate an event with probability $h(p)$, for some function of interest, $h$ (Asmussen et al., 1992; Huber, 2017; Keane and O’Brien, 1994; Latuszyński et al., 2011; Morina et al., 2021; Nacu and Peres, 2005). Gonçalves et al. (2017a,b) proposed the following Bernoulli factory to sample events with probability $\alpha_B(x, y)$. Suppose, $
\pi(x)q(x, y) = c_x p_x,
$ where $c_x$ is possibly known and $0 < p_x < 1$. Similarly, $\pi(y)q(y, x) = c_y p_y$. The roles of $c_x$ and $c_y$ are to ensure that $p_x$ and $p_y$ are valid probabilities. We stress that the bound $c_x$ can be a local bound; a global bound over the full support is not required. For asymmetric proposal distributions, $c_x$ and $p_x$ depend on both $x$ on $y$, however we suppress the dependency on $y$ for notational convenience. One way to arrive at $c_x$ and $p_x$ is to find $c_x$ such that
$
\pi(x)q(x, y) \leq c_x \text{ and then set } p_x = \pi(x)q(x, y)c_x^{-1},
$ (2)

with analogous statements for $c_y$ and $p_y$. The two-coin Bernoulli factory of Gonçalves et al. (2017a,b) presented in Algorithm 2 returns events of probability $\alpha_B(x, y)$ with

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

First, both $c_x$ and $c_y$ can be known up to a common normalising constant. Second, it is assumed in Steps 2 and 3 that events of probabilities $p_y$ and $p_x$, respectively, can be simulated. Using laws of conditional probability, it is easy to check that Algorithm 2 returns 1 with probability $\alpha_B(x, y)$. In addition, as Gonçalves et al. (2017a) describe, the number of loops until the algorithm stops is distributed as a $\text{Geom}(c_y p_y + c_x p_x)/(c_y + c_x)$, and the mean execution time is

$$\frac{c_x + c_y}{c_x p_x + c_y p_y} = \frac{c_x + c_y}{\pi(x)q(x, y) + \pi(y)q(y, x)}.$$
Algorithm 2 two-coin algorithm for $\alpha_B(x, y)$

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

Clearly, the computational efficiency of the two-coin algorithm relies heavily on the upper bounds $c_x$ and $c_y$. If the bound is loose, then the algorithm yields a large mean execution time.

3 Portkey Barker’s method

The main source of inefficiency in implementing Barker’s method via Bernoulli factories is the inefficiency of the two-coin algorithm. Motivated by this, we introduce a new family of acceptance probabilities and provide an efficient Bernoulli factory for members of this family. For a proposal density $q(x, y)$, consider accepting a proposed value $y$ with probability,

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

where $d(x, y) = d(y, x) \geq 0$. Then $\alpha(x, y)$ yields a $\pi$-reversible Markov chain and the symmetry of $d(x, y)$ is essential to this.

Theorem 1. For a proposal density $q(x, y)$, a Markov chain with acceptance probability $\alpha(x, y)$ in (3) is $\pi$-reversible if and only if $d(x, y) = d(y, x)$.

Proof. An acceptance function yields a $\pi$-reversible Markov chain if and only if

$$\pi(y)q(y, x)\alpha(y, x) = \pi(x)q(x, y)\alpha(x, y).$$
Let \( d(x, y) = d(y, x) \). Consider,
\[
\pi(x) q(x, y) \alpha(x, y) = \frac{\pi(y) q(y, x) \pi(x) q(x, y)}{\pi(x) q(x, y) + \pi(y) q(y, x) + d(x, y)} = \pi(y) q(y, x) \alpha(y, x).
\]

Naturally, \( \alpha(x, y) \leq \alpha_B(x, y) \) and by Peskun’s ordering, Barker’s method is more efficient. However, for a particular choice of \( d(x, y) \) we present a Bernoulli factory that provides significant computational gains, enough to supersede the loss of statistical efficiency. For a user-chosen \( 0 < \beta \leq 1 \), consider the following portkey Barker’s acceptance probability:
\[
\alpha(\beta)(x, y) := \frac{\pi(y) q(y, x)}{\pi(y) q(y, x) + \pi(x) q(x, y) + \frac{(1 - \beta)}{\beta} (c_x + c_y)},
\]
(4)
with \( c_x \) and \( c_y \) given by (2). To ensure \( d(x, y) \) is small, \( \beta \approx 1 \).

To yield events of probability \( \alpha(\beta)(x, y) \), we modify the two-coin algorithm via, what we call a portkey method. Our portkey two-coin algorithm in Algorithm 3 introduces a first step in the two-coin algorithm that allows immediate rejections with probability \( 1 - \beta \).

For a given proposal, whenever Algorithm 3 loops on Steps 4a and 5a, a Bern(\( \beta \)) event is drawn, which if zero, rejects the proposed value immediately. Running Algorithm 3 with \( \beta \approx 1 \) avoids the large number of loops often witnessed in Algorithm 2.

**Theorem 2.** Algorithm 3 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,
\[
r = \beta c_y (1 - p_y) + c_x (1 - p_x)
\]
\[
\Rightarrow \sum_{i=0}^{\infty} r^i = \frac{c_x + c_y}{(1 - \beta)(c_x + c_y) + \beta(c_x p_x + c_y p_y)}.
\]

For the algorithm to output 1, for all \( i \), there should be no output in all loops up to \( i \), and the \( i \)th loop should output 1. Thus, the probability that the algorithm outputs 1 is
\[
\frac{\beta c_y p_y}{c_x + c_y} \sum_{i=0}^{\infty} r^i = \frac{c_y p_y}{c_x p_x + c_y p_y + \frac{1 - \beta}{\beta} (c_x + c_y)}.
\]
Algorithm 3 Portkey two-coin algorithm

1: Draw $S \sim \text{Bernoulli}(\beta)$
2: if $S = 0$ then
   3: output 0
3: if $S = 1$ then
   4: Draw $C_1 \sim \text{Bern}(\frac{c_y}{c_x + c_y})$
5: if $C_1 = 1$ then
   6: Draw $C_2 \sim \text{Bern}(p_y)$
7: if $C_2 = 1$ then
   8: output 1
9: if $C_2 = 0$ then
  10: go to Step 1
11: if $C_1 = 0$ then
   12: Draw $C_2 \sim \text{Bern}(p_x)$
13: if $C_2 = 1$ then
   14: output 0
15: if $C_2 = 0$ then
  16: go to Step 1

From Theorem 1, the portkey Barker’s algorithm is $\pi$-reversible, and if $\beta > 0$, it is $\pi$-ergodic since $\alpha(\beta)(x,y) > 0$. It may be intuitive to see that Algorithm 3 will quite obviously lead to a smaller mean execution time. More specifically, the number of loops until the algorithm stops is distributed according to a Geom($s_\beta$), where

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

We note that for $0 < \beta < 1$, $s_\beta > 1 - \beta$, which is bounded away from zero. Thus, the mean execution time is bounded above. A similar argument cannot be made for the original two-coin algorithm. Specifically, the ratio of the mean execution time of the two-coin algorithm to the portkey two-coin algorithm is

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

Thus, if $(c_y p_y + c_x p_x)/(c_x + c_y) \approx 0$, i.e., the original two-coin algorithm is highly inefficient, the ratio diverges to infinity. On the other hand, if $(c_y p_y + c_x p_x)/(c_x + c_y) \approx 1$, i.e., the original two-coin algorithm is efficient, the two algorithms have comparable expected number of loops.

Computational efficiency gained here is at the cost of statistical efficiency. Let $P_{(\beta)}$
and $P_B$ denote the Markov operators for 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 $\text{var}(g, P) := \lim_{n \to \infty} n \text{Var}_\pi(\bar{g}_n)$.

**Theorem 3.** For $0 < \beta \leq 1$, $\alpha(\beta)(x,y) \leq \beta \alpha_B(x,y)$. As a consequence,

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

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

$$\alpha(\beta)(x,y) = \frac{\pi(y)q(y,x)}{\pi(x)q(x,y) + \pi(y)q(y,x) + \beta^{-1}(1 - \beta)(c_x + c_y)} \leq \beta \cdot \frac{\beta \pi(x)q(x,y) + \pi(y)q(y,x) + (1 - \beta)(\pi(x)q(x,y) + \pi(y)q(y,x))}{\pi(y)q(y,x)} = \beta \cdot \alpha_B(x,y).$$

Combining [Latuszyński and Roberts (2013, Corollary 1)] and the ordering of [Peskun (1973)], or equivalently [Zanella (2020, Theorem 2)], yields,

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

A variance bound in the opposite direction can be obtained under specific conditions.

**Theorem 4.** 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,

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

**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,

$$\alpha(\beta)(x,y) \geq \frac{\pi(y)q(y,x)}{\pi(x)q(x,y) + \pi(y)q(y,x) + \frac{1 - \beta}{\delta \beta} (\pi(x)q(x,y) + \pi(y)q(y,x))} = \left( 1 + \frac{1 - \beta}{\delta \beta} \right)^{-1} \alpha_B(x,y).$$

The variance ordering follows from [Latuszyński and Roberts (2013, Corollary 1)].
When \( p_x \) or \( p_y \) is small, it is desirable to set \( \beta \) to be large. Theorem 4 suggests that setting \( \beta = 1 - \delta \) is desirable. However, \( \delta \) will typically not be available.

4 Flipped portkey two-coin algorithm

A significant challenge in implementing the original two-coin algorithm is identifying a suitable \( c_x \). When faced with a target density that is restricted to a constrained space (as in Section 5.2), it may be easier to lower bound \( \pi(x)q(x,y) \), or in other words, upper bound \( \pi(x)^{-1}q(x,y)^{-1} \). That is, suppose for \( \tilde{c}_x > 0 \) and \( 0 < \tilde{p}_x < 1 \),

\[
\pi(x)^{-1}q(x,y)^{-1} = \tilde{c}_x \tilde{p}_x .
\]

Consider an acceptance probability denoted by \( \alpha_{f,(\beta)}(x,y) \) of the form (3), with

\[
d(x,y) := \bar{d}(x,y) = \frac{(1 - \beta)(\tilde{c}_x + \tilde{c}_y)}{\tilde{c}_x \tilde{p}_x \tilde{c}_y \tilde{p}_y}, \tag{5}
\]

Algorithm 4 presents a flipped portkey two-coin algorithm for \( \alpha_{f,(\beta)}(x,y) \).

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

1: Draw \( S \sim \text{Bernoulli}(\beta) \)
2: if \( S = 0 \) then
3: output 0
4: if \( S = 1 \) then
5: Draw \( C_1 \sim \text{Bern}(\frac{\tilde{c}_x}{\tilde{c}_x + \tilde{c}_y}) \)
6: if \( C_1 = 1 \) then
7: Draw \( C_2 \sim \text{Bern}(\tilde{p}_x) \)
8: if \( C_2 = 1 \) then
9: output 1
10: if \( C_2 = 0 \) then
11: go to Step 1
12: if \( C_1 = 0 \) then
13: Draw \( C_2 \sim \text{Bern}(\tilde{p}_y) \)
14: if \( C_2 = 1 \) then
15: output 0
16: if \( C_2 = 0 \) then
17: go to Step 1
Theorem 5. Algorithm 4 yields output 1 with probability

\[ \alpha_f(\beta)(x, y) = \frac{\pi(y)q(y, x)}{\pi(y)q(y, x) + \pi(x)q(x, y) + \frac{(1 - \beta)(\tilde{c}_x + \tilde{c}_y)}{\beta} \tilde{c}_x \tilde{p}_x \tilde{c}_y \tilde{p}_y} \]

Proof. Using \( \pi(x)^{-1}q(x, y)^{-1} = \tilde{c}_x \tilde{p}_x \) and following the steps of Theorem 2, we have

\[ \Pr(\text{output} = 1) = \frac{\tilde{c}_x \tilde{p}_x}{\tilde{c}_x \tilde{p}_x + \tilde{c}_y \tilde{p}_y + \beta^{-1}(1 - \beta)(\tilde{c}_x + \tilde{c}_y)} = \alpha_f(\beta)(x, y). \]

There are two important consequences of Theorem 5. First, by a simple application of Theorem 1, Algorithm 4 yields a \( \pi \)-reversible Markov chain. Second, setting \( \beta = 1 \), yields the usual Barker’s acceptance probability, providing a second Bernoulli factory for Barker’s algorithm! Variance ordering results for the flipped portkey Barker’s acceptance probability can be obtained as before and is given below for completeness.

Theorem 6. For \( 0 < \beta \leq 1 \), \( \alpha_f(\beta)(x, y) \leq \beta \alpha_B(x, y) \). As a consequence,

\[ \text{var}(g, P_B) \leq \beta \text{var}(g, P_f(\beta)) + (\beta - 1) \text{Var}_\pi(g). \]

Further, 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,

\[ \text{var}(g, P_f(\beta)) \leq \left(1 + \frac{1 - \beta}{\delta \beta}\right) \text{var}(g, P_B) + \frac{1 - \beta}{\delta \beta} \text{Var}_\pi(g). \]

In addition to the availability of lower bounds on \( \pi(x)q(y, x) \), the flipped portkey algorithm should be chosen if generating \( \tilde{p}_x \) coins is easier than generating \( p_x \) coins. This advantage is demonstrated clearly in Section 5.2.

5 Examples

5.1 Gamma mixture of Weibulls

Consider a target distribution of the form \( \pi(\theta) = \int \pi(\theta|\lambda) \nu(d\lambda) \) where \( \nu \) is a mixing measure on \( \lambda \). We assume that \( \theta|\lambda \sim \text{Weibull}(\lambda, k) \), where \( \lambda \) is the scale parameter and
$k$ is a known shape parameter. The proposal distribution is the normal distribution centered at the current step with variance .001. It can be shown that, $\pi(\theta | \lambda) \leq k/(e^{\theta}) := c_\theta$. Then $\pi(\theta) = c_\theta \cdot \pi(\theta)/c_\theta$ and events of probability $p_\theta = \pi(\theta)/c_\theta$ can be simulated. Specifically, draw $\lambda \sim \nu$ and $U \sim U[0, 1]$ independently. Then $\Pr\{U \leq \pi(\theta|\lambda)/c_\theta\} = p_\theta$.

We set $\nu = \text{Gamma}(10, 100)$ and $k = 10$.

We run Barker’s and portkey Barker’s algorithms for $10^5$ steps using the two Bernoulli factories for various values of $\beta$. Trace plots and autocorrelation plots from one such run are shown in Figures 1 and 2 respectively. As expected, for smaller values of $\beta$, the acceptance probability is smaller, but the autocorrelations do not increase drastically.

![Figure 1: Trace plots of the last 1000 steps of the chains for four values of $\beta$, with acceptance probabilities are in text.](image1)

![Figure 2: ACF plots for a run-length of 1e5 for four $\beta$ values.](image2)

We repeat the simulation 1000 times to compare their performance. From the results in Table 1, unsurprisingly the effective sample size (ESS) (Gong and Flegal, 2016) decreases as $\beta$ decreases. However, the ESS per second is significantly higher for portkey Barker’s compared to Barker’s. In fact, for $\beta = .90$, the portkey Barker’s method is almost three times as efficient as the original Barker’s method. This is a clear consequence of the
large mean execution time of the two-coin algorithm. The two-coin algorithm required an average of 32 loops, while the portkey algorithms were significantly smaller. More notably, the two-coin algorithm’s loops demonstrates heavy tailed behavior where the average maximum number of loops for MCMC run is 1.3 million.

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

<table>
<thead>
<tr>
<th></th>
<th>β=1</th>
<th>β=.99</th>
<th>β=.90</th>
<th>β=.75</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESS</td>
<td>7484</td>
<td>6939</td>
<td>4320</td>
<td>2501</td>
</tr>
<tr>
<td>ESS/s</td>
<td>422.47</td>
<td>1052.66</td>
<td>1248.97</td>
<td>1159.68</td>
</tr>
<tr>
<td>Mean loops</td>
<td>32.00 (3.7)</td>
<td>7.63 (0.0)</td>
<td>3.97 (0.0)</td>
<td>2.55 (0.0)</td>
</tr>
<tr>
<td>Max loops</td>
<td>1315683 (366763.76)</td>
<td>604 (3.44)</td>
<td>78 (0.37)</td>
<td>32 (0.12)</td>
</tr>
</tbody>
</table>

5.2 MCMC on constrained spaces

Consider the following general setup of Liechty et al. (2009). Let \( f(y|\theta) \) be a likelihood and for a set \( \mathcal{A} \), let \( \pi(\theta|\eta) \) be the prior on \( \theta \) constrained in \( \mathcal{A} \), so that for a function \( g(\cdot|\eta) \),

\[
\pi(\theta|\eta) = \frac{g(\theta|\eta)I(\theta \in \mathcal{A})}{\int_{\mathcal{A}} g(\theta|\eta) d\theta},
\]

where \( \int_{\mathcal{A}} g(\theta|\eta) d\theta \) is not tractable. If \( \eta \) is a fixed hyperparameter, this intractability is not an issue. However, for a full Bayesian model, it is desirable to assign a hyperprior to \( \eta \), and this leads to intractability in the posterior distribution for \( (\theta, \eta) \). Chen et al. (2010) identify a shadow prior technique that modifies the Bayesian posterior to yield approximate inference. We describe this technique as it relates to a Bayesian correlation estimation model and demonstrate how the flipped portkey algorithm can be used very naturally in this setting.

As a general recipe, if one can find a set \( \mathcal{B} \supset \mathcal{A} \) such that \( \int_{\mathcal{B}} g(\theta|\eta) d\theta \) is tractable, then \( \pi(\theta|\eta) \) can be lower bounded easily. The efficiency of the flipped portkey algorithm depends on the relative sizes of sets \( \mathcal{B} \) and \( \mathcal{A} \). If \( g(\theta|\eta) \) is a well-defined density on a larger space, then \( \int_{\mathcal{A}} g(\theta|\eta) d\theta \leq 1 \) and depending on the constraint set \( \mathcal{A} \), may lead to an efficient flipped portkey algorithm. Such a construction can be done for the correlation estimation models of Hartman et al. (2020); Liechty et al. (2009); Philip et al. (2014); Wang and Pillai (2013), for the Bayesian graphical lasso (Wang et al., 2012), and for Bayesian semiparametric regression (Papageorgiou and Marshall, 2020).

Consider the Bayesian common correlation estimation model of Liechty et al. (2004). Suppose \( y_1, \ldots, y_n | R \overset{iid}{\sim} N(0, R) \) where \( R \) is a \( p \times p \) correlation matrix. Liechty et al. (2004) assume the unique elements in \( R, r_{ij} \), are normally distributed, restricted to \( R \)
being positive-definite. That is, let \( S_p^+ \) be the set of \( p \times p \) positive-definite matrices, then

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

where

\[
L^{-1}(\mu, \sigma^2) = \int_{R \in S_p^+} \prod_{i<j} \frac{1}{\sqrt{2\pi \sigma^2}} \exp \left\{ -\frac{(r_{ij} - \mu)^2}{2\sigma^2} \right\} dr_{ij} \tag{6}
\]

is typically not available in closed form due to the complex nature of \( S_p^+ \). For known \( \tau^2, a_0, b_0 \), hyperpriors for \( \mu \sim N(0, \tau^2) \) and \( \sigma^2 \sim IG(a_0, b_0) \) are chosen. Interest is in the posterior distribution for \((R, \mu, \sigma^2)\) and Liechty et al. (2004) implement a component-wise Metropolis-within-Gibbs sampler. Let \( l = p(p-1)/2 \). The full conditional densities are

\[
f(r_{ij} | r_{-ij}, \mu, \sigma^2) \propto |R|^{-n/2} \exp \left\{ -\frac{\text{tr}(R^{-1}Y^TY))}{2} \right\} \exp \left\{ -\frac{(r_{ij} - \mu)^2}{2\sigma^2} \right\} \mathbb{I}\{l_{ij} \leq r_{ij} \leq u_{ij}\},
\]

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

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

where the indicator variable \( \mathbb{I}\{l_{ij} \leq r_{ij} \leq u_{ij}\} \) ensures positive-definiteness of \( R \). The interval \((l_{ij}, u_{ij})\) can be obtained deterministically using the methods of Barnard et al. (2000). We use Metropolis-Hastings update with a Gaussian proposal for each \( r_{ij} \).

Updating \( \mu \) and \( \sigma^2 \) requires the knowledge of \( L(\mu, \sigma^2) \) which is unavailable. Liechty et al. (2004) implement a shadow prior technique that interjects the Bayesian hierarchy such that \( r_{ij} \sim N(\delta_{ij}, \nu^2) \) and \( \delta_{ij} \sim N(\mu, \sigma^2) \), with unchanged hyper-priors on \( \mu \) and \( \sigma^2 \). The resulting marginal posterior of \((R, \mu, \sigma^2)\) is different from the original model. The full conditionals of \( \mu \) and \( \sigma^2 \) are available in closed-form, but the full conditionals for \( \delta \) are intractable. However, Liechty et al. (2009) argue that for updating \( \delta_{ij} \), if \( \nu^2 \approx 0 \), the intractable constants can be ignored. Thus, the methodology, although convenient, 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 \( \mu \) and \( \sigma^2 \) and we only present details for \( \mu \). We use Gaussian random walk proposals for both components so that we can ignore the proposal density in the Bernoulli factories. Although \( L^{-1}(\mu, \sigma^2) \leq 1 \), we can obtain a tighter upper bound by noting that each \( r_{ij} \in [-1, 1] \)
and thus
\[ g(\mu, R, \sigma^2)^{-1}L^{-1}(\mu, \sigma^2) \leq g(\mu, R, \sigma^2)^{-1} \left[ \Phi\left(\sigma^{-1}(1 - \mu)\right) - \Phi\left(\sigma^{-1}(-1 - \mu)\right)\right]^t := \tilde{c}_\mu \]

Set \( f(\mu|R, \sigma^2)^{-1} = \tilde{c}_\mu \tilde{c}_\mu^{-1}f(\mu | R, \sigma^2)^{-1} := \tilde{c}_\mu \tilde{p}_\mu \). Generating coins of probability \( \tilde{p}_\mu \) is straightforward: 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.

Consider 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 \( \mu \) and \( \sigma^2 \) and first compare the estimated posterior density of \( \mu \) and \( \sigma^2 \) from a run of length \( 10^5 \) for both the shadow prior method and the two Bernoulli factory MCMC algorithms. Figure 3 presents the results, where it is evident that the shadow prior method yields a biased posterior distribution.

![Figure 3](image.png)

Figure 3: Estimated posterior density of \( \mu \) and \( \sigma^2 \) from Markov chain lengths of \( 10^5 \) with \( v^2 = .001 \) as recommended by Chen et al. (2010).

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 \( \mu \), the original two-coin algorithm requires an exponentially large number of Bernoulli factory loops whenever unlikely values are proposed. This makes it laborious to tune Barker’S algorithm and encourages short jumps.

We repeat the above experiment 10 times for MCMC runs of length \( 10^4 \). Ideally we would have liked to increase the Monte Carlo replication size, but the original two-coin
algorithm often got stuck in a loop for days. The results are presented in Table 2 with estimated ESS (Vats et al., 2019) calculated using the R package mcmcse (Flegal et al., 2017). 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 \( \mu \).

Table 2: Averaged results from 10 replications. Standard errors are in brackets.

<table>
<thead>
<tr>
<th>( \beta )</th>
<th>1</th>
<th>.90</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESS</td>
<td>542 (13.50)</td>
<td>496 (9.00)</td>
</tr>
<tr>
<td>ESS/s</td>
<td>9.63 (1.992)</td>
<td>14.83 (0.279)</td>
</tr>
<tr>
<td>Mean loops ( \mu )</td>
<td>218.43 (148.89)</td>
<td>2.99 (0.010)</td>
</tr>
<tr>
<td>Mean loops ( \sigma^2 )</td>
<td>3.21 (0.02)</td>
<td>2.49 (0.010)</td>
</tr>
<tr>
<td>Max loops ( \mu )</td>
<td>2084195 (1491777)</td>
<td>34 (2.94)</td>
</tr>
<tr>
<td>Max loops ( \sigma^2 )</td>
<td>38 (1.13)</td>
<td>27 (1.51)</td>
</tr>
</tbody>
</table>

We stress that for a fair comparison, we use the same proposal distributions for both Markov chains. Since the original two-coin algorithm gets stuck in seemingly infinite loops for proposals encouraging larger jumps, we are unable to tune the Barker’s algorithm to the recommended 15.8% acceptance rate of Agrawal et al. (2021). To highlight
At this point, 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 $\approx 40$ seconds with an ESS of 514.

As described previously, such a Bernoulli factory MCMC algorithm can also be constructed for other models. For instance, consider the Bayesian graphical lasso problem of Wang et al. (2012). Let $\Omega$ be a $p \times p$ covariance matrix with elements $w_{ij}$; the Bayesian graphical lasso models the observed data as $y_i|\Omega \sim \text{N}(0, \Omega^{-1})$ for $i = 1, \ldots, n$ with the prior

$$
\Omega \mid \lambda \sim C_{\lambda}^{-1} \left\{ \prod_{i<j} \text{DE}(w_{ij}|\lambda_{ij}) \prod_{i=1}^p \text{Exp}(w_{ii}|\lambda_{ii}/2) \right\} \mathbb{1}\{\Omega \in S_p^+\},
$$

where $\text{DE}$ denotes the double exponential distribution. For a fully Bayesian paradigm and element-dependent penalization, Wang et al. (2012) discuss the challenges in assigning separate hyper-priors on each of the $\lambda_{ij}$, in that the normalizing constant $C_{\lambda}$ remains unknown. Since $C_{\lambda} \leq 1$, we can obtain a lower bound on the joint posterior distribution of $(\Omega, \lambda)$ to implement the flipped portkey two-coin algorithms. This bound is likely to not scale well with $p$, so that the efficiency of the Bernoulli factory algorithms will be affected in high-dimensions, however, the data size $n$, will typically have no affect on the computational efficiency.

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

A collection of methodologies to perform exact inference for discretely-observed diffusions has been proposed in the last 15 years (see Beskos et al., 2009, 2006; Sermaidis et al., 2013). These are all based on algorithms for exact simulation of diffusions (Beskos et al., 2008). Although these methodologies can be applied to a wide class of univariate diffusion models, some important ones are left out. For example, the Cox-Ingersoll-Ross model used to describe the evolution of interest rates, and the Wright-Fisher diffusion, 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.

The methodologies for exact inference for jump-diffusion models based on the exact
simulation of jump-diffusions are considerably more restrict (see Gonçalves et al., 2017b; Gonçalves and Roberts, 2014). Let $\alpha$ be the drift function of the Lamperti transform of the original diffusion. Whilst for diffusion models $\alpha^2 + \alpha'$ is required to be bounded below, for jump-diffusions it is also required to have uniformly bounded drift and jump rate of the Lamperti transformed process.

As demonstrated in Gonçalves et al. (2017a,b), the two-coin Barker’s MCMC algorithm performs exact inference for diffusions and jump-diffusions without requiring the above conditions. The Portkey two-coin Barker’s MCMC proposed here will typically provide considerable gain when compared to the original two-coin Barker’s, allowing for a wider applicability of the methodology in terms of model complexity and data size.

Consider the neutral Wright-Fisher family of diffusions with mutations

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

where $W_s$ is a standard Brownian motion. The process $Y$ lives in $[0,1]$ and we assume it is observed at a finite set of time points, $0 = t_0, t_1, \ldots, t_n = T$ so that the observed data are $Y_{\text{obs}} = (y_0, y_1, \ldots, y_n)^T$. The parameters of interest are $(\theta_1, \theta_2)$ which describe the drift coefficient of the process and Bayesian estimation yields desirable theoretical properties (Sant et al., 2020). Yet, their transition densities $p(\gamma|y_{i-1}, y_i) := \Pr(Y_{t_i} \in dy_i|Y_{t_{i-1}} = y_{i-1})/dy_i$, that determine the likelihoods, are not available in closed-form and are characterised by poor numerical properties near the 0 and 1 barriers (see Jenkins and Spano, 2017) that undermine reliable Bayesian inference based on approximations.

Inference will be done for the following reparameterization $\gamma_1 = \theta_1 + \theta_2$ 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 (7), which leads to a unit diffusion coefficient process $X$. Now we specify a Gibbs sampler alternating between updating $\gamma$ and updating the missing paths, $X_{\text{mis}}$, of the transformed diffusion $X$ given the transformed observations $X_{\text{obs}}$. For the $X_{\text{mis}}$ update, we use standard Brownian bridge proposals restricted to $[0,1]$ and implement both Barker’s and portkey Barker’s MCMC. The Brownian bridges are sampled using a layered Brownian bridge construction, which allows lower and upper bounds on the likelihood that are crucial to devise an efficient Bernoulli factory.

For updating $\gamma$, component-wise updates are done with uniform random walk proposal distributions, $U(\gamma_1 \pm 0.3)$ and $U(\gamma_2 \pm 0.01)$ for $\gamma_1$ and $\gamma_2$, respectively. It can be shown that $\pi(\gamma|X_{\text{mis}}, X_{\text{obs}}) = c_\gamma p_\gamma$, where the forms of $c_\gamma$ and $p_\gamma$ can be found in Gonçalves et al. (2017a). A realization of probability $p_\gamma$ is obtained using the layer refinement strategy described in Gonçalves et al. (2017b). Here each interval $(t_{i-1}, t_i)$ is broken into refinements, and a layered Brownian bridge is constructed within each refinement.
Without the refinement, the bound $c_{\gamma}$ is too loose and thus $p_{\gamma}$ is often too small for the Bernoulli factory to be efficient. However, finer refinements require simulating more layered Brownian bridges leading 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.

We simulate a Wright-Fisher diffusion with $\gamma_1 = 8$ and $\gamma_2 = 0.5$ and observe the process at times $\{0, 1, \ldots, 50\}$. We use the component-wise sampler described above to sample from the posterior distribution of $(\gamma_1, \gamma_2)$. We run the two samplers for 10000 steps where the Barker’s algorithm takes about 34.5 hours and the portkey Barker’s algorithm takes about 3.5 hours. First, the update for $X_{\text{mis}}$ performs similarly for both Barker’s and portkey Barker’s, where we set $\beta = 0.98$. The average number of loops over 1e4 samples for the two-coin and the portkey two-coin algorithms are 1.698 and 1.645, respectively, indicating that the original two-coin algorithm already works well enough for this component. We note specifically that close to nothing is lost by using portkey Barker’s instead of Barker’s acceptance probability.

For updating the parameters, the $\beta$s are set to be .99995 and .9995 for $\gamma_1$ and $\gamma_2$, respectively. In Figure 5, are the number of loops of the two Bernoulli factories for each iteration of the two Markov chains for both $\gamma_1$ and $\gamma_2$. The $\gamma_1$ update of the Barker’s two-coin Markov chain requires an average of 489 loops with a maximum of 183077 loops. In comparison, the portkey two-coin algorithm runs an average of 43 loops with a maximum of 12622. The long tails of the number of loops for the original two-coin algorithm mean that the average run-time is quite slow, as evidenced from the computation times. However, since both $\beta$s are close to 1, this gain in computational

![Figure 5](image-url)

Figure 5: Comparing the number of loops of the Bernoulli factory for two-coin and portkey two-coin algorithm. Left is for the $\gamma_1$ update and right is for the $\gamma_2$ update.

efficiency mainly impacts the Bernoulli factory, and has only a small effect on the Markov chain as witnessed by the autocorrelation plots in Figure 6. The estimated ESS for the posterior mean of $\gamma$ is 255 and 197, respectively for Barker’s and portkey Barker’s.
However, the ESS per hour is 7.36 and 56.45, respectively. Thus, the portkey Barker’s algorithm is $\approx 7$ times more efficient than the original Barker’s algorithm. We stress that implementation of the portkey two-coin algorithm requires only minor changes to the code for any two-coin algorithm.

6 Discussion

Motivated by Bernoulli factories, we introduce a family of MCMC acceptance probabilities for intractable target distributions. We argue that the (flipped) portkey two-coin algorithm is a robust alternative to the two-coin algorithm; when the two-coin algorithm is efficient, the portkey two-coin algorithm is essentially similar to the two-coin algorithm. However, when the two-coin algorithm is computationally infeasible, the portkey two-coin algorithm is significantly more stable, has a smaller mean execution time, and allows for larger proposal variances. The resulting Markov chain is exact and for values of $\beta$ close to 1, the loss in statistical efficiency is small compared to the gain in computational efficiency. Moreover, tuning the portkey Barker’s sampler is far easier since feedback from the algorithm is significantly faster than the original Barker’s sampler.

Finally, the (portkey) two-coin algorithm finds immediate use in Bayesian inference for diffusions and jump-diffusions, as illustrated in Section 5.3 and Bayesian models with intractable priors. To implement these algorithms more generally requires reasonable bounds on the target distribution. We believe the solution to this is to construct local bounds compatible with these algorithms and this makes for important future work.

---

$^2$An R package that implements the Bernoulli factories is here: [https://github.com/dvats/portkey](https://github.com/dvats/portkey) and complete reproducible codes for our examples can be found here: [https://github.com/dvats/PortkeyPaperCode](https://github.com/dvats/PortkeyPaperCode).
7 Acknowledgements

Dootika Vats was supported by the National Science Foundation (DMS/1703691). Flávio Gonçalves thanks FAPEMIG, CNPq and the University of Warwick for financial support. Krzysztof Latuszyński is supported by the Royal Society through the Royal Society University Research Fellowship. Gareth Roberts is supported by the EPSRC grants: *ilike* (EP/K014463/1), CoSInES (EP/R034710/1) and Bayes for Health (EP/R018561/1).

References


