Antithetic Magnetic and Shadow Hamiltonian Monte Carlo

WILSON TSAKANE MONGWE, RENDANI MBUVHA, AND TSHILIDZI MARWALA (Senior Member, IEEE)

1 School of Electrical Engineering, University of Johannesburg, Auckland Park 2006, South Africa
2 School of Statistics and Actuarial Science, University of Witwatersrand, Johannesburg 2000, South Africa

Corresponding author: Wilson Tsakane Mongwe (wilsonmongwe@gmail.com)

The work of Wilson Tsakane Mongwe and Rendani Mbuvha was supported by the Google Ph.D. Fellowships in Machine Learning. The work of Tshilidzi Marwala was supported by the National Research Foundation of South Africa.

ABSTRACT Hamiltonian Monte Carlo is a Markov Chain Monte Carlo method that has been widely applied to numerous posterior inference problems within the machine learning literature. Markov Chain Monte Carlo estimators have higher variance than classical Monte Carlo estimators due to autocorrelations present between the generated samples. In this work we present three new methods for tackling the high variance problem in Hamiltonian Monte Carlo based estimators: 1) We combine antithetic and importance sampling techniques where the importance sampler is based on sampling from a modified or shadow Hamiltonian using Separable Shadow Hamiltonian Hybrid Monte Carlo, 2) We present the antithetic Magnetic Hamiltonian Monte Carlo algorithm that is based on performing antithetic sampling on the Magnetic Hamiltonian Monte Carlo algorithm and 3) We propose the antithetic Magnetic Momentum Hamiltonian Monte Carlo algorithm based on performing antithetic sampling on the Magnetic Momentum Hamiltonian Monte Carlo method. We find that the antithetic Separable Shadow Hamiltonian Hybrid Monte Carlo and antithetic Magnetic Momentum Hamiltonian Monte Carlo algorithms produce effective sample sizes that are higher than antithetic Hamiltonian Monte Carlo on all the benchmark datasets. We further find that antithetic Separable Shadow Hamiltonian Hybrid Monte Carlo and antithetic Magnetic Hamiltonian Monte Carlo produce higher effective sample sizes normalised by execution time in higher dimensions than antithetic Hamiltonian Monte Carlo. In addition, the antithetic versions of all the algorithms have higher effective sample sizes than their non-antithetic counterparts, indicating the usefulness of adding antithetic sampling to Markov Chain Monte Carlo algorithms. The methods are assessed on benchmark datasets using Bayesian logistic regression and Bayesian neural network models.

INDEX TERMS Antithetic sampling, Bayesian logistic regression, Bayesian neural network, Hamiltonian Monte Carlo, machine learning, magnetic Hamiltonian Monte Carlo, shadow Hamiltonian, variance reduction.

I. INTRODUCTION

Predictive models of high parameter dimensionality have become the mainstay across a multitude of critical tasks such as medicine, law enforcement, finance and self-driving automobiles [1]–[5]. Bayesian methods such as Bayesian logistic regression (BLR) and Bayesian neural networks (BNNs) allow for principled inference of posterior distributions of model parameters. The Bayesian framework provides a robust approach to predictive uncertainty, theoretically justified interpretations of regularisation and generalisation through prior distributions.

Markov Chain Monte Carlo (MCMC) methods are fundamental in performing inference of probabilistic machine learning models. Hamiltonian Monte Carlo (HMC) and its variants have been a popular choice of inference due to their ability to suppress random-walk behaviour through the use of first-order gradient information [6]–[9]. Although HMC serves as an improvement to other MCMC methods, like other MCMC methods it still suffers from the presence of autocorrelations in the generated samples [10], [11]. This results in the high variance of HMC based estimators. One approach of tackling the high variance of
MCMC estimators is by using results from MCMC coupling theory [12], [13].

The coupling of MCMC methods has been used as a theoretical tool to prove convergence behaviour of Markov chains [12], [14]–[17]. Johnson [15] presents a convergence diagnostic for MCMC methods that is based on coupling a Markov chain with another chain that is periodically restarted from fixed parameter values. The diagnostic provides an informal lower bound on the effective sample sizes (ESS) of a MCMC chain. Jacob et al. [12] remove the bias in MCMC chains by using couplings of Markov chains using a telescopic sum argument. The resulting unbiased estimators can then be computed independently in parallel.

More recently, MCMC couplings have been studied for HMC with good results [18], [19]. Markov chain coupling has also been used to provide unbiased HMC estimators [12], [13], [20]. Heng and Jacob [13] propose an approach that constructs a pair of HMC chains that are coupled in such a way that they meet after some random number of iterations. These chains can then be combined to create unbiased chains. Unlike the approach of Heng and Jacob [13] and other authors [12], [19], Bou-Rabee et al. [17] present a new coupling algorithm where the momentum variable is not shared between the coupled chains.

In this paper we present methods that use the results from coupling theory to create anti-correlated HMC based chains where the momentum variable is shared between the chains. These anti-correlated chains are created by running the second chain with the momentum auxiliary variable having the opposite sign of the momentum of the first chain. These chains also share the random uniform variable in the Metropolis-Hastings acceptance step. When these two chains anti-couple strongly, taking the average of the two chains results in estimators that have lower variance, or equivalently, these chains produce samples that have higher effective sample sizes than their non-antithetic counterparts.

Antithetic coupled chains have a rich history in the literature [19], [21]–[23]. Green and Han [22] reduce the variance of samples from a Markov chain using an alternating sign-flip algorithm, which is akin to antithetic sampling. Antithetic Gibbs sampling is presented in Frigessi et al. [23], and is shown to produce estimators that always have lower variance than Gibbs sampling. The work that is most closely related to ours is the paper by Piponi et al. [19] where the antithetic HMC (A-HMC) and control variate HMC variance reduction techniques are presented. The HMC based antithetic samplers have an advantage over antithetic Gibbs samplers in that they are applicable to problems where conditional distributions are intractable, and where Gibbs sampling may mix slowly [19]. Although the control variate HMC variance reduction technique is more generally applicable than the antithetic approach (which requires the target distribution to be symmetric), it relies on the practitioner being able to construct efficient and accurate approximate distributions, which is difficult for neural networks, and is an impediment to broad use [19]. Thus, this work focuses on the antithetic sampling approach which is straightforward to implement for BNNS as well as for BLR.

We expand on the work of Piponi et al. [19] by presenting three new antithetic sampling MCMC algorithms being the antithetic Separable Shadow Hamiltonian Hybrid Monte Carlo (A-S2HMC), antithetic Magnetic Hamiltonian Monte Carlo (A-MHMC) and the antithetic Magnetic Momentum Hamiltonian Monte Carlo (A-MMHMC) algorithms.

The A-S2HMC algorithm is based on sampling from the shadow Hamiltonian using Separable Shadow Hamiltonian Hybrid Monte Carlo (S2HMC) [24], [25]. Sampling using the S2HMC algorithm provides benefits over sampling using HMC. One benefit is that the shadow Hamiltonian is better conserved by the numerical integrator, which allows one to use larger step sizes (and thus reducing autocorrelations) than in HMC without a significant drop in the acceptance rates [24]–[27]. In addition, S2HMC is an importance sampler and already offers variance reduction over HMC. The A-S2HMC algorithm thus combines the benefits of antithetic sampling with importance sampling, which should provide even more variance reduction than is provided by the the S2HMC over HMC. The disadvantage of S2HMC over HMC is that it consumes more computational resources than HMC, which reduces the outperformance on a normalised ESS basis.

The A-MHMC algorithm is based on adding antithetic sampling to the Magnetic Hamiltonian Monte Carlo (MHMC) [28] algorithm. The MHMC algorithms provides an improvement on HMC by adjusting the range of exploration via a magnetic field. This has the effect of enhancing the convergence speed and reducing the autocorrelations of the samples [28], [29]. In the A-MHMC algorithm we combine antithetic sampling with the benefits that MHMC already has over HMC with the aim of creating a sampler that outperforms A-HMC. Unlike S2HMC, MHMC has comparable execution time with HMC, which means that the computational burden should not outweigh the benefits of higher ESSs produced by MHMC and A-MMHMC.

The A-MMHMC algorithm is based on performing antithetic sampling on the Magnetic Momentum Hamiltonian Monte Carlo (MNHMC) algorithm [28], [30]. As with MHMC, MMHMC produces higher ESSs than HMC and we thus leverage this to create the A-MMHMC algorithm which should outperform A-HMC. Unlike MHMC, the MMHMC algorithm requires an implicit integration scheme, which causes the method to be slow when compared to MHMC and HMC [28], [30]. This affects its performance on a normalised ESS basis.

The empirical results show that the proposed algorithms provide higher ESSs than A-HMC on the majority of the datasets, more especially in higher dimensions. In addition, the antithetic versions of all the algorithms we consider in this paper have higher ESSs than their non-antithetic counterparts. This shows the benefit of adding antithetic sampling to MCMC algorithms.
The main contributions of this work are as follows:

- We present three new antithetic algorithms being antithetic Separable Shadow Hamiltonian Hybrid Monte Carlo, antithetic Magnetic Hamiltonian Monte Carlo and the antithetic Magnetic Momentum Hamiltonian Monte Carlo algorithms.
- We provide the first application of antithetic MCMC algorithms to the inference of Bayesian neural networks.

The remainder of this paper is structured as follows: Sections II to IV discuss the HMC, MMHMC, MHHMC and S2HMC algorithms on which the new antithetic algorithms are based, Section V presents the three new antithetic algorithms that we are proposing along with the antithetic HMC method, Section VI outlines the experiments conducted, Section VII presents and discusses the results of the experiments and we provide the conclusion in Section VIII.

II. HAMILTONIAN MONTE CARLO

Hamiltonian Monte Carlo (HMC) reduces random walk behaviour of Metropolis-Hastings by adding auxiliary momentum variables to the parameter space [6], [31]. HMC creates a vector field around the current state using gradient information, which assigns the current state a trajectory towards a next state with high probability [31], [32]. The dynamical system formed by the model parameters \( \mathbf{w} \) and the auxiliary momentum variables \( \mathbf{p} \) is represented by the Hamiltonian \( H(\mathbf{w}, \mathbf{p}) \) written as follows [33]:

\[
H(\mathbf{w}, \mathbf{p}) = U(\mathbf{w}) + K(\mathbf{p})
\]

where \( U(\mathbf{w}) \) is the negative log-likelihood of the target posterior distribution, also referred to as the potential energy and \( K(\mathbf{p}) \) is the kinetic energy defined by the kernel of a Gaussian with a covariance matrix \( \mathbf{M} \) [32]:

\[
K(\mathbf{p}) = \frac{\mathbf{p}^T \mathbf{M}^{-1} \mathbf{p}}{2}.
\]

HMC then samples from the joint canonical distribution defined by the Hamiltonian as:

\[
\pi \propto \exp \left( -H(\mathbf{w}, \mathbf{p}) \right).
\]

The Hamiltonian dynamics are defined by the differential equation

\[
\frac{d}{dt} \begin{bmatrix} \mathbf{w} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} \nabla_w H(\mathbf{w}, \mathbf{p}) \\ \nabla_p H(\mathbf{w}, \mathbf{p}) \end{bmatrix}.
\]

This differential equation cannot be solved analytically in most instances, and one has to resort to a numerical integration scheme. The leapfrog integration scheme is the most commonly used integration scheme. In the leapfrog integrator, to reach the next point in the path, we take half a step in the momentum direction, followed by a full step in the direction of the model parameters - then ending with another half step in the momentum direction [4], [6].

Since numerical integration is not exact and introduces errors, the Metropolis-Hasting acceptance procedure is used to adjust the acceptance rate of the sampler and ensure detailed balance [6], [34]. Algorithm 1 shows the pseudocode for HMC and the leapfrog integration scheme.

The HMC algorithm has multiple parameters that require tuning for efficient sampling, such as the step size and the trajectory length. In terms of trajectory length, a trajectory that is too long results in a trajectory that inefficiently traces the posterior distribution, also referred to as the potential energy and a covariance matrix \( \mathbf{M} \) [32]:

\[
K(\mathbf{p}) = \frac{\mathbf{p}^T \mathbf{p}}{2}.
\]

Algorithm 1 Hamiltonian Monte Carlo

**Input:** \( N, \epsilon, L, w_{\text{init}}, H(w, p) \)

**Output:** \( (w)_{m=0} \)

1: \( w_0 \leftarrow w_{\text{init}} \)
2: for \( m \rightarrow 1 \) to \( N \) do
3: \( p_{m-1} \sim N(0, \mathbf{M}) \)
4: \( p_m, w_m = \text{Leapfrog}(p_{m-1}, w_{m-1}, \epsilon, L, H) \)
5: \( \delta H = H(w_{m-1}, p_{m-1}) - H(w_m, p_m) \)
6: \( \alpha_m = \min(1, \exp(\delta H)) \)
7: \( u_m \sim \text{Unif}(0, 1) \)
8: \( w_m = \text{Metropolis}(\alpha_m, u_m, w_m, w_{m-1}) \)
9: end for

function Leapfrog(\( p, w, \epsilon, L, H \) )

10: for \( t \leftarrow 1 \) to \( L \) do
11: \( p \leftarrow p + \epsilon \frac{\delta H}{\delta w}(w, p) \)
12: \( w \leftarrow w + \epsilon \mathbf{p} \)
13: \( p \leftarrow p + \epsilon \frac{\delta H}{\delta w}(w, p) \)
14: end for

function Metropolis(\( \alpha_m, u_m, w_m, w_{m-1} \) )

15: if \( \alpha_m < u_m \) then
16: \( w_m = w_{m-1} \)
17: else
18: \( w_m = w_m \)
19: end if

return \( w_m \)

The HMC algorithm has multiple parameters that require tuning for efficient sampling, such as the step size and the trajectory length. In terms of trajectory length, a trajectory length that is too short leads to random walk behaviour similar to the Metropolis-Hasting method. While a trajectory length that is too long results in a trajectory that inefficiently traces back [7].

We now provide a simple example of how one would practically use the HMC algorithm and its variants, that we present in later sections of this paper, to infer the parameters of a model. Suppose that one wanted to infer the parameters \( a \) and \( b \) from the function \( f(x) = ax^2 + bx \) for \( x > 0 \). If we have \( n \) independent observations of \( x \) and use a uniform prior on the parameters, we would set

\[
U(\mathbf{w}) = -\log \left[ \prod_{i=1}^{n} f(x_i) \right]
\]

where \( \mathbf{w} = \{ a, b \} \) and

\[
K(\mathbf{p}) = \frac{\mathbf{p}^T \mathbf{p}}{2}.
\]
The Hamiltonian for this problem would then be given as:

\[
H(w, p) = -\log \left[ \prod_{i=1}^{n} f(x_i) \right] + \frac{p^T p}{2} \quad (7)
\]

Algorithm 1 would then be run using equation 7 with \( M = I \).

III. MAGNETIC HAMILTONIAN MONTE CARLO METHODS

Magnetic Hamiltonian Monte Carlo (MHMC) expands on HMC by adding a magnetic field in addition to the force field already in HMC. This magnetic field offers a great deal of flexibility over HMC and encourages more efficient exploration of the posterior [28], [29]. This results in lower autocorrelations in the generated samples when compared to HMC.

MHMC uses the same Hamiltonian as in HMC, but exploits non-canonical Hamiltonian dynamics where the canonical matrix now has a non-zero element on the diagonal. The MHMC dynamics are:

\[
\frac{d}{dt} \begin{bmatrix} w \\ p \end{bmatrix} = \begin{bmatrix} 0 & I \\ -I & G \end{bmatrix} \begin{bmatrix} \nabla_w H(w, p) \\ \nabla_p H(w, p) \end{bmatrix} \quad (8)
\]

where \( G \) is the term that represents the magnetic field. This also shows that MHMC only differs from HMC dynamics by \( G \) being non-zero. When \( G = 0 \), MHMC and HMC have the same dynamics. As with HMC, these dynamics cannot be integrated exactly, and we resort to a numerical integration scheme with a Metropolis-Hastings acceptance step to ensure detailed balance [28]. For MHMC the integrator is not exactly the leapfrog integrator, but is very similar. Algorithm 2 show the pseudo-code for the MHMC algorithm as well as the leapfrog-like integration scheme.

Algorithm 2 Magnetic Hamiltonian Monte Carlo

Input: \( N, \epsilon, L, w_{init}, H(w, p), G \)
Output: \((w)^N_{n=0}\)

1: \( w_0 \leftarrow w_{init} \)
2: for \( m \rightarrow 1 \) to \( N \) do
3: \( p_{m-1} \leftarrow \mathcal{N}(0, M) \)
4: \( p_m, w_m, G \leftarrow \text{Integrator}(p_{m-1}, w_{m-1}, \epsilon, L, G, H) \)
5: \( \delta H = H(w_{m-1}, p_{m-1}) - H(w_m, p_m) \)
6: \( \alpha_m = \min(1, \exp(\delta H)) \)
7: \( u_m \leftarrow \text{Unif}(0, 1) \)
8: \( w_m \leftarrow \text{Metropolis}(\alpha_m, u_m, w_m, w_{m-1}) \)
9: \( p_m = -p_m, G = -G \)
end for

function Integrator(p, w, \epsilon, L, G, H(w, p))

11: for \( t \leftarrow 1 \) to \( L \) do
12: \( p \leftarrow p + \frac{\epsilon}{2} \frac{\partial}{\partial w} H(w, p) \)
13: \( w \leftarrow w + G^{-1}(\exp(\epsilon G) - I)p \)
14: \( p \leftarrow \exp(\epsilon G)p \)
15: \( p \leftarrow p + \frac{\epsilon}{2} \frac{\partial}{\partial w} G(w, p) \)
16: end for
return \(-p, w, -G\)

The matrix \( G \) provides an extra degree of freedom for the MHMC algorithm. It is not clear as to how one should set this matrix. Ideally this matrix should be tuned adaptively [28]. The selection of \( G \) is still an open research problem [28], [30]. In this paper we follow the directions of the authors [28] and select only a few dimensions to be influenced by the magnetic field. In particular, \( G \) was set such that \( G_{ii} = G_{i1} = g, G_{ii} = G_{ig} = -g \) and zero elsewhere for \( g = 0.2 \). We also ensure that \( G \) is anti-symmetric [30].

Although MHMC requires matrix exponentiation and inversion (see Algorithm 2 line 13), this only needs to be computed once upfront and stored [28]. Following this approach results in computation time that is comparable to HMC. This becomes more important in models that have many parameters such as neural networks.

As with MHMC, MMHMC [30] differs with HMC by the canonical matrix having a non-zero diagonal entry. The dynamics of MMHMC are given as:

\[
\frac{d}{dt} \begin{bmatrix} w \\ p \end{bmatrix} = \begin{bmatrix} E & I \\ -I & 0 \end{bmatrix} \begin{bmatrix} \nabla_w H(w, p) \\ \nabla_p H(w, p) \end{bmatrix} \quad (9)
\]

and thus only differ from HMC dynamics by \( E \) being non-zero. When \( E = 0 \), MMHMC and HMC have the same dynamics. As with MHMC, these dynamics cannot be integrated exactly and we resort to numerical integration. However, unlike MHMC, the integration scheme is not leapfrog-like and we resort to an implicit mid-point integration scheme [28], [30].

For the experiments in this paper we set \( E = G \). The selection of \( E \) is still an open research problem which we intend to address in future work. Algorithm 3 show the pseudo-code for the MMHMC algorithm, as well as the implicit mid-point integration scheme. The fixed point iterations in lines 15 to 20 of Algorithm 3 are the reasons why the MMHMC is much slower relative to HMC and MMHC. This reduces the algorithms practical usefulness, especially in problems with high dimensions such as neural networks.

IV. SHADOW HAMILTONIANS

Shadow or modified Hamiltonians are perturbations of the Hamiltonian that are by design exactly conserved by the numerical integrator. In the case of shadow Hamiltonian Hybrid Monte Carlo, we sample from the importance distribution defined by the shadow Hamiltonian as:

\[
\hat{\pi} \propto \exp\left(-\tilde{H}^{[k]}(w, p)\right) \quad (10)
\]

where \( \tilde{H}^{[k]} \) is the shadow Hamiltonian defined using backward error analysis of the numerical integrator.

When performing backward error analysis, the shadow Hamiltonian can be defined by an asymptotic expansion in the powers of the discretisation step size \( \epsilon \) around the Hamiltonian:

\[
\tilde{H} = H + \epsilon H_2 + \epsilon^2 H_3 + \epsilon^3 H_4 + \ldots \quad (11)
\]
true Hamiltonian. The fourth-order shadow Hamiltonian for \( \epsilon \)ing components of the Taylor series in terms of 

The leapfrog is second-order accurate (\( \mathcal{O}^2 \)), the fourth-order truncation is conserved with higher accuracy (\( \mathcal{O}^4 \)) than the true Hamiltonian. The fourth-order shadow Hamiltonian for the leapfrog after matching coefficients from the flow and the asymptotic expansion becomes:

\[
\tilde{H}^{[4]} = U(w) + K(p) + \frac{\epsilon^2}{12} K_T U_{ww} K_p - \frac{\epsilon^4}{24} U_{pp} U_{pp} + \mathcal{O}(\epsilon^4) \quad (13)
\]

where \( U_w, U_{ww}, K_p \) and \( K_{pp} \) are Jacobians and Hessians of the potential and kinetic energies respectively. 

The shadow Hamiltonian in equation (13) is non-separable in terms of \( w \) and \( p \), which necessitates computational expensive momenta acceptance criteria for momenta, and the potential tuning of additional parameters [24], [25]. This additional computational overhead is overcome by pre-processing the positions and momenta before propagating through the leapfrog integrator.

A. SEPARABLE SHADOW HAMILTONIAN HYBRID MONTE CARLO

Separable Shadow Hamiltonian Hybrid Monte Carlo (S2HMC) utilises a processed leapfrog integrator to create a separable Hamiltonian [25]. The separable Hamiltonian in S2HMC is:

\[
\tilde{H}(w, p) = U(w) + K(p) + \frac{\epsilon^2}{24} U_{w} M^{-1} U_w + \mathcal{O}(\epsilon^4) \quad (14)
\]

Propagation of positions and momenta on this shadow Hamiltonian is performed after performing this reversible mapping \((\tilde{w}, \tilde{p}) = \chi(w, p)\), where \((\tilde{w}, \tilde{p})\) is found through fixed point iterations as:

\[
\hat{p} = p - \frac{\epsilon}{24} [U_w(w + \epsilon M^{-1} \hat{p}) - U_w(w - \epsilon M^{-1} \hat{p})] \quad (15)
\]

\[
\hat{w} = w + \frac{\epsilon^2 M^{-1}}{24} [U_w(w + \epsilon M^{-1} \hat{p}) + U_w(w - \epsilon M^{-1} \hat{p})] \quad (16)
\]

After the leapfrog is performed, this mapping is reversed using post-processing the following fixed point iterations:

\[
\hat{w} = \hat{w} - \frac{\epsilon^2 M^{-1}}{24} [U_w(w + \epsilon M^{-1} \hat{p}) + U_w(w - \epsilon M^{-1} \hat{p})] \quad (17)
\]

\[
\hat{p} = \hat{p} + \frac{\epsilon}{24} [U_w(w + \epsilon M^{-1} \hat{p}) - U_w(w - \epsilon M^{-1} \hat{p})] \quad (18)
\]

Once the samples are obtained from S2HMC as depicted in Algorithm 4, importance weights are calculated to allow for the use of the shadow canonical density rather than the true density. These weights are based on the differences between the true and shadow Hamiltonians as follows:

\[
b_m = \exp(-H(w, p) - \tilde{H}(w, p)) \quad (19)
\]

Mean estimates of observables \( f(w) \) which are functions of the parameters \( w \) can then be computed as a weighted average.
necessarily independent, with the same marginal distribution. Suppose we have two random variables $V_1$ and $V_2$.

The performance of the algorithms is compared on real world regression benchmark datasets. The BNNs considered in this paper have one hidden layer with ten hidden units. The datasets, their number of features and their associated results are detailed in Table 1.

The full benefit of antithetic sampling is greatest when the distribution $U$ is symmetrical about some vector $\bar{w}$, [19]. For the majority of target distribution of interest the symmetry holds only approximately [19]. The approximate symmetry results in approximate anti-coupling, which nonetheless still provides good variance reduction [19].

The pseudo-code for the new antithetic algorithms that we are proposing is presented in Algorithms 6 - 8, with the algorithm for A-HMC shown in Algorithm 5. The difference between the antithetic algorithms and the original algorithms is that the momentum variable and the uniform random variable in the Metropolis-Hastings acceptance step are shared between the two chains.

Algorithm 5 Antithetic HMC

| Input: $N$, $\epsilon$, $L$, $w_0^x$, $w_0^y$, $H(w, p)$ |
| Output: $(w^x)^N_{m=0}$, $(w^y)^N_{m=0}$ |

1: $w_0^x \leftarrow w_{init}$
2: $w_0^y \leftarrow w_{init}$
3: for $m \rightarrow 1$ to $N$ do
4: $p_{m-1}^x \sim \mathcal{N}(0, M)$
5: $p_{m-1}^y = -p_{m-1}^x$
6: $p_m^x, w_m^x = \text{Leapfrog}(p_{m-1}^x, w_{m-1}^x, \epsilon, L, H)$
7: $p_m^y, w_m^y = \text{Leapfrog}(p_{m-1}^y, w_{m-1}^y, \epsilon, L, H)$
8: $\delta H^x = H(w_{m-1}^x, p_{m-1}^x) - H(w_m^x, p_m^x)$
9: $\delta H^y = H(w_{m-1}^y, p_{m-1}^y) - H(w_m^y, p_m^y)$
10: $\alpha_m^x = \min(1, \exp(\delta H^x))$
11: $\alpha_m^y = \min(1, \exp(\delta H^y))$
12: $u_m \sim \text{Unif}(0, 1)$
13: $w_m^x = \text{Metropolis}(\alpha_m^x, u_m, w_{m-1}^x, w_{m-1}^y)$
14: $w_m^y = \text{Metropolis}(\alpha_m^y, u_m, w_{m-1}^x, w_{m-1}^y)$
end for
Algorithm 7 Antithetic MMHMC

**Input:** $N, E, l, w^x_{init}, w^y_{init}, h(w, p)$

**Output:** $(w^x)_N, (w^y)_N$

1. \( w^0 \leftarrow w^x_{init} \)
2. \( w^0 \leftarrow w^y_{init} \)
3. for \( m \rightarrow 1 \) to \( N \) do
4. \( p^x_{m-1} \sim N(0, M) \)
5. \( p^y_{m-1} = -p^x_{m-1} \)
6. \( p^x_{m}, w^x_{m} = \text{Integrator} \left( p^x_{m-1}, w^x_{m-1}, E, l, H \right) \)
7. \( p^y_{m}, w^y_{m} = \text{Integrator} \left( p^y_{m-1}, w^y_{m-1}, E, l, H \right) \)
8. \( \delta H^x = H(w^x_{m-1}, p^x_{m}) - H(w^x_{m}, p^x_{m}) \)
9. \( \delta H^y = H(w^y_{m-1}, p^y_{m}) - H(w^y_{m}, p^y_{m}) \)
10. \( \sigma^x_m = \min \left( 1, \exp \left( \delta H^x \right) \right) \)
11. \( \sigma^y_m = \min \left( 1, \exp \left( \delta H^y \right) \right) \)
12. \( u_m \sim \text{Unif} \left( 0, 1 \right) \)
13. \( w^x_{m} = \text{Metropolis} \left( \sigma^x_m, u_m, w^x_{m}, w^x_{m-1} \right) \)
14. \( w^y_{m} = \text{Metropolis} \left( \sigma^y_m, u_m, w^y_{m}, w^y_{m-1} \right) \)
end for

Algorithm 8 Antithetic S2HMC

**Input:** $N, e, l, w^x_{init}, w^y_{init}, h(w, p)$

**Output:** $(w^x)_N, (w^y)_N, (b^x)_N, (b^y)_N$

1. \( w^0 \leftarrow w^x_{init} \)
2. \( w^0 \leftarrow w^y_{init} \)
3. for \( m \rightarrow 1 \) to \( N \) do
4. \( p^x_{m-1} \sim N(0, M) \)
5. \( p^y_{m-1} = -p^x_{m-1} \)
6. Apply the pre-processing mapping to both chains
7. \( p^x_{m}, w^x_{m} = \text{Leapfrog} \left( p^x_{m-1}, w^x_{m-1}, e, l, \hat{H} \right) \)
8. \( p^y_{m}, w^y_{m} = \text{Leapfrog} \left( p^y_{m-1}, w^y_{m-1}, e, l, \hat{H} \right) \)
9. Apply the pre-processing mapping to both chains
10. \( \delta H^x = \hat{H}(w^x_{m-1}, p^x_{m}) - \hat{H}(w^x_{m}, p^x_{m}) \)
11. \( \delta H^y = \hat{H}(w^y_{m-1}, p^y_{m}) - \hat{H}(w^y_{m}, p^y_{m}) \)
12. \( \sigma^x_m = \min \left( 1, \exp \left( \delta H^x \right) \right) \)
13. \( \sigma^y_m = \min \left( 1, \exp \left( \delta H^y \right) \right) \)
14. \( u_m \sim \text{Unif} \left( 0, 1 \right) \)
15. \( w^x_{m} = \text{Metropolis} \left( \sigma^x_m, u_m, w^x_{m}, w^x_{m-1} \right) \)
16. \( w^y_{m} = \text{Metropolis} \left( \sigma^y_m, u_m, w^y_{m}, w^y_{m-1} \right) \)
17. \( b^x_{m} = \exp \left( -H(w^x_{m}, p^x_{m}) - \hat{H}(w^x_{m}, p^x_{m}) \right) \)
18. \( b^y_{m} = \exp \left( -H(w^y_{m}, p^y_{m}) - \hat{H}(w^y_{m}, p^y_{m}) \right) \)
end for

A. PERFORMANCE METRICS

The performance metrics used in this paper are the multivariate Effective Sample Size (mESS), the mESS normalised by the execution time as well predictive performance on unseen data. The predictive performance metric used for classification was the Area Under the Curve (AUC), while for regression the Mean Square Error (MSE) was used. For all the algorithms we set the pre-conditioning matrix $M = I$, which is the common approach in practice.

The execution time is the time taken to generate the samples after the burn-in period. The ESS calculation used in this paper is the multivariate ESS calculation outlined in [36]. This metric takes into account the correlations between the parameters as well, which is unlike the minimum univariate ESS measure that is typically used to analyse MCMC results [10], [36]. The minimum univariate ESS calculation results in the estimate of the ESS being dominated by the parameter dimensions that mix the slowest, and ignoring all other dimensions [36]. The multivariate ESS is calculated as:

$$ mESS = N \times \left( \frac{|\Lambda|}{|\Sigma|} \right)^{\frac{1}{2}} $$

where $N$ is the number of generated samples, $D$ is the number of parameters, $\Lambda$ is the sample covariance matrix and $\Sigma$ is the estimate of the Markov chain standard error. When $D = 1$, $mESS$ is equivalent to the univariate ESS. For S2HMC, which is an importance sampler, the $mESS$ is adjusted by taking into account the possibility of non-uniform importance weights ($b^N_m$) through the thinning algorithm outlined in Radivojevic et al. [26].

The $mESS$ for the variance reduced chain is related to the $mESS$ of the original chain as [19]:

$$ mESS_{antithetic} = \frac{2 \times mESS_{original}}{1 + \rho} $$

(20)

where $\rho$ is the correlation coefficient between the corresponding pairs of chains. In this paper, the correlation is taken as the maximum correlation across all the parameter dimensions, which creates a lower bound for $mESS_{antithetic}$. Equation (20) implies that we can increase the ESS by ensuring that the correlation is as close as possible to -1, which is what the anti-coupling methodology aims to do. This also means that the $mESS_{antithetic}$ can be greater than the total number of generated samples $N$.

For all the datasets, we generated 10 000 samples with a burn-in period of 5 000 samples. This was sufficient for all the algorithms to converge on all the datasets. The number

| Dataset | Features | $N$ | Model | $D$ |
|---------|----------|-----|-------|-----|
| Pima Indian | 7 | 532 | BLR | 8 |
| Heart | 13 | 270 | BLR | 14 |
| Australian credit | 14 | 690 | BLR | 15 |
| German credit | 24 | 1 000 | BLR | 25 |
| Power | 4 | 9 568 | BNN | 61 |
| Airfoil | 5 | 1 503 | BNN | 71 |
| Concrete | 8 | 1 030 | BNN | 101 |
| Protein | 9 | 45 730 | BNN | 111 |

TABLE 1. Real world datasets. $N$ represents the number of observations. $D$ represents the number of model parameters.

VOLUME 9, 2021
of steps used to generate each sample was fixed at 100. The step-sizes were chosen by targeting an acceptance rate of 70% through the dual averaging [7] methodology explained below.

**B. INTEGRATION STEP SIZE TUNING**

We address the issue of step size adaptation in the algorithms through dual averaging in the burn-in phase [7]. We target a Metropolis-Hastings acceptance rate $\delta$ using dual averaging updates as follows:

$$
epsilon_{t+1} \leftarrow \mu - \frac{\sqrt{t}}{\gamma} \frac{1}{\t + l_0} \sum_{i=1}^{t} H_i$$

$$
\tilde{\epsilon}_{t+1} \leftarrow \eta_t \epsilon_{t+1} + (1 - \eta_t) \bar{\epsilon}_t
$$

where $\mu$ is a free parameter that $\epsilon_t$ gravitates to, $\gamma$ controls the convergence towards $\mu$, $\eta_t$ is a decaying rate of adaptation in line with [37] and $H_i$ is the difference between the target acceptance rate and the actual acceptance rate. The dual averaging updates are such that

$$E[H_t] = E[\delta - \alpha_t] = 0.$$

This has the effect of updating the step size towards the target acceptance rate $\delta$. The dual averaging algorithm for HMC is shown in Algorithm 9, with the implementation for the other algorithms following similarly. More information on the dual averaging algorithm can be found in Hoffman and Gelman [7]. In this paper we use the same parameter settings as in Hoffman and Gelman [7], and these settings are shown in Algorithm 9. It is important to note that for the antithetic chains, we only adapt the step size of the original chain, and feed this step size to the other chain.

Algorithm 9: HMC With Dual Averaging

*Input:* $N$, $w_{init}$, $\epsilon_0$, $L$, $M^{adapt}$, $\mu = \log(10\epsilon_0)$, $\bar{\epsilon}_0 = 1$, $\bar{H}_0 = 0$, $\gamma = 0.05$, $l_0 = 10$, $\kappa = 0.75$

*Output:* $(\bar{w}_t)^N_{t=0}$, final step size $= \epsilon^{M^{adapt}}$

1. $w_0 \leftarrow w_{init}$
2. for $m \rightarrow 1$ to $N$ do
3. \hspace{1em} $p_{m-1} \sim N(0, M)$
4. \hspace{1em} $p_m, w_m = \text{Leapfrog}(p_{m-1}, w_{m-1}, \epsilon_{m-1}, L, H)$
5. \hspace{1em} $\delta H = H(w_{m-1}, p_{m-1}) - H(w_m, p_m)$
6. \hspace{1em} $\alpha_m = \min(1, \exp(\delta H))$
7. \hspace{1em} if $m < M^{adapt}$ then
8. \hspace{2em} $\bar{H}_m = (1 - \frac{1}{m + l_0}) \bar{H}_{m-1} + \frac{1}{m + l_0} (\delta - \alpha_m)$
9. \hspace{2em} $\log \epsilon_m = \mu - \frac{\sqrt{m}}{\gamma} \bar{H}_m$
10. \hspace{2em} $\log \tilde{\epsilon}_m = m^{-x} \log \epsilon_m + (1 - m^{-x}) \log \tilde{\epsilon}_{m-1}$
11. \hspace{1em} else
12. \hspace{2em} $\epsilon_m = \tilde{\epsilon}^{M^{adapt}}$
13. \hspace{1em} end if
14. \hspace{1em} $u_m \sim \text{Unif}(0, 1)$
15. \hspace{1em} $w_m = \text{Metropolis}(\alpha_m, u_m, w_m, w_{m-1})$
16. end for

\section{C. DATASETS}

The details of the real world datasets used in this paper are displayed in Table 1. The logistic regression datasets are all approached as binary classification tasks as they all have two classes. The datasets modeled using neural networks all have one continuous response variable and are thus approached as regression problems. A 90-10 train-test split was used and the features were normalised for all the datasets. For each dataset, the chain was run 10 times from different starting positions and the results recorded. The prior distribution over the parameters for the logistic regression datasets was Gaussian with standard deviation equal to 10. The above settings are in line with those in Girolami and Calderhead [10]. For the neural network datasets the prior was a standard normal distribution.

\section{VII. RESULTS AND DISCUSSION}

The experiments were implemented in PyTorch [38] and were carried out on a 64-bit precision CPU. In evaluating the S2HMC and MMHMC algorithms and their antithetic variants, we set a convergence tolerance of $10^{-6}$ or the completion of ten fixed point iterations.

The performance of the algorithms across different metrics is shown in Figure 1 and Table 2. In Figure 1, the plots on the first row for each dataset show the $m\text{ESS}$, and the plots on the second row show the $m\text{ESS}$ normalised by execution time (which is in seconds). The results are for the 10 runs of each algorithm.

As with Figure 1, the execution time $t$ in Table 2 is in seconds. The results in Table 2 are the mean results over the 10 runs for each algorithm. When calculating the predictive performance metrics (that is AUC and MSE) for S2HMC and A-S2HMC, the results are weighted by the importance weights in equation (19). Note that we use the mean values over the 10 runs in Table 2 to form our conclusions about the performance of the algorithms.

As expected, HMC has the lowest execution time $t$ on the majority of the datasets, with MHMC being a close second. The execution time for MHMC is similar to that of HMC on the BLR datasets and outperforms HMC on the Concrete and Protein datasets. The increased computation time of the shadow algorithms can be attributed to the fixed-point iterations in equations (9) to (12). The MMHMC algorithm is the slowest method due to the fixed-point iterations required in the implicit mid-point integration scheme used for this algorithm. Note that the execution time for the antithetic variants is twice the execution time for the non-antithetic algorithms. This is because two chains are run for the antithetic versions.

The execution time of all the algorithms increases with the dimensionality of the problem, which is expected.

On the majority of the datasets, we find that MHMC, MMHMC and S2HMC have higher ESSs than HMC, with the out-performance being more pronounced on the the BNN datasets which have more parameters. The A-MMHMC and A-S2HMC methods produce superior ESSs on all
TABLE 2. Means results over 10 runs of each algorithm. Each column represents the mean value for the specific method. For example, column three shows the mean results for HMC. The execution time $t$ is in seconds. The values in bold indicate that the particular method outperforms the other methods on that specific metric. For example, HMC outperforms in terms of $t$ on the Pima dataset. The AUC metric is used for the BLR classification datasets (which are the first four datasets) while the MSE metric is used for the BNN regression datasets (which are the last four datasets). The methods have similar predictive performance across all the datasets with MMHMC and A-MMHMC outperforming the other methods on all the datasets. For the AUC metric higher is better, while for the MSE metric lower is better.

| Pima Indian dataset            | MHMC | A-MHMC | HMC | A-HMC | MMHMC | A-MMHMC | S2HMC | A-S2HMC |
|--------------------------------|------|--------|-----|-------|-------|---------|-------|---------|
| $m$ESS                         | 4.952| 29.315 | 5.715| 17.925| 5.754 | 118.419 | 3.704 | 26.519  |
| $t$ (in sec)                    | 333  | 670    | 314 | 625   | 3.967 | 9.353   | 388   | 777     |
| $m$ESS/t                       | 14.86| 43.74  | 18.19| 28.65 | 1.45  | 14.92   | 9.54  | 34.12   |
| AUC                            | 0.826| 0.826  | 0.827| 0.827 | 0.831 | 0.831   | 0.827 | 0.827   |

| Heart dataset                  |      |        |     |       |       |         |       |         |
|--------------------------------|------|--------|-----|-------|-------|---------|-------|---------|
| $m$ESS                         | 4.031| 9.976  | 2.675| 7.091 | 5.709 | 24.273  | 4.956 | 14.215  |
| $t$ (in sec)                    | 309  | 617    | 297 | 575   | 3.876 | 7.752   | 366   | 733     |
| $m$ESS/t                       | 13.04| 16.14  | 8.99 | 11.91 | 1.47  | 3.13    | 13.51 | 19.37   |
| AUC                            | 0.898| 0.897  | 0.896| 0.896 | 0.911 | 0.911   | 0.896 | 0.896   |

| Australia credit dataset        |      |        |     |       |       |         |       |         |
|--------------------------------|------|--------|-----|-------|-------|---------|-------|---------|
| $m$ESS                         | 4.770| 12.322 | 3.467| 8.530 | 5.493 | 16.218  | 4.943 | 10.671  |
| $t$ (in sec)                    | 317  | 633    | 317 | 635   | 3.960 | 7.922   | 395   | 791     |
| $m$ESS/t                       | 15.04| 19.43  | 10.92| 13.42 | 1.38  | 2.04    | 12.48 | 13.48   |
| AUC                            | 0.955| 0.955  | 0.954| 0.954 | 0.961 | 0.961   | 0.955 | 0.955   |

| German credit dataset           |      |        |     |       |       |         |       |         |
|--------------------------------|------|--------|-----|-------|-------|---------|-------|---------|
| $m$ESS                         | 3.996| 11.431 | 4.529| 12.662| 5.980 | 14.468  | 4.932 | 18.809  |
| $t$ (in sec)                    | 377  | 755    | 357 | 715   | 4.508 | 9.026   | 445   | 892     |
| $m$ESS/t                       | 10.57| 15.12  | 12.66| 17.70 | 1.32  | 1.60    | 11.06 | 21.07   |
| AUC                            | 0.763| 0.763  | 0.763| 0.763 | 0.769 | 0.769   | 0.764 | 0.763   |

| Power dataset                  |      |        |     |       |       |         |       |         |
|--------------------------------|------|--------|-----|-------|-------|---------|-------|---------|
| $m$ESS                         | 1.192| 6.196  | 810 | 3.375 | 1.802 | 9.247   | 999   | 4.966   |
| $t$ (in sec)                    | 586  | 1.722  | 580 | 1.160 | 5.195 | 10.391  | 718   | 1.437   |
| $m$ESS/t                       | 2.03 | 5.28   | 1.39 | 2.90  | 0.34  | 0.88    | 1.39  | 3.45    |
| MSE                            | 23.07| 23.07  | 23.08| 23.08 | 22.28 | 22.27   | 23.03 | 23.04   |

| Airfoil dataset                |      |        |     |       |       |         |       |         |
|--------------------------------|------|--------|-----|-------|-------|---------|-------|---------|
| $m$ESS                         | 2.441| 11.672 | 1.516| 7.589 | 3.108 | 15.109  | 2.083 | 9.829   |
| $t$ (in sec)                    | 1.362| 2.731  | 1.304| 2.608 | 8.794 | 17.387  | 1.447 | 2.895   |
| $m$ESS/t                       | 1.79 | 4.27   | 1.16 | 2.90  | 0.35  | 0.85    | 1.43  | 3.39    |
| MSE                            | 27.67| 27.67  | 27.67| 27.68 | 26.84 | 26.91   | 27.68 | 27.68   |

| Concrete dataset               |      |        |     |       |       |         |       |         |
|--------------------------------|------|--------|-----|-------|-------|---------|-------|---------|
| $m$ESS                         | 2.600| 7.191  | 1.607| 4.269 | 3.296 | 8.979   | 1.666 | 4.572   |
| $t$ (in sec)                    | 1.156| 2.313  | 1.330| 2.660 | 6.912 | 13.825  | 1.389 | 2.779   |
| $m$ESS/t                       | 2.24 | 3.11   | 1.20 | 1.60  | 0.47  | 0.64    | 1.19  | 1.64    |
| MSE                            | 75.06| 75.12  | 75.29| 75.15 | 74.54 | 74.55   | 75.01 | 75.20   |

| Protein dataset                |      |        |     |       |       |         |       |         |
|--------------------------------|------|--------|-----|-------|-------|---------|-------|---------|
| $m$ESS                         | 5.006| 21.529 | 3.468| 15.267| 5.868 | 24.973  | 4.999 | 22.650  |
| $t$ (in sec)                    | 1.292| 2.585  | 1.351| 2.697 | 7.670 | 15.341  | 1.591 | 3.193   |
| $m$ESS/t                       | 3.87 | 8.32   | 2.56 | 5.65  | 0.76  | 1.62    | 3.14  | 7.09    |
| MSE                            | 11.58| 11.56  | 11.58| 11.58 | 11.23 | 11.24   | 11.62 | 11.58   |

8 benchmark datasets when compared to A-HMC. A-MHMC produces higher ESSs than A-HMC on 7 of the 8 benchmark datasets, with the under-performance being on the German credit dataset. These results show that the proposed methods outperform in low dimensions, with the outperformance becoming more pronounced in larger dimensions. This will be particularly important when performing inference on deeper neural networks.

On a normalised ESS basis, A-S2HMC and A-MMHMC outperform A-HMC on all the datasets, except that A-MHMC under-performs A-HMC on the German credit dataset. A-MMHMC outperforms all the algorithms on the BNN datasets on a normalised ESS basis. This shows that although the proposed methods have in general a higher computational cost than HMC, they still provide improved normalised ESS rates.

The drop in normalised ESS performance for the A-MMHMC algorithm can be attributed to the increase in execution time of the mid-point integration scheme. Using an explicit integration scheme such as that proposed in...
FIGURE 1. Results for the datasets over 10 runs of each method. For each dataset, the plots on the first row show the multivariate effective sample size and the plots on the second row show the multivariate effective sample size normalised by execution time (in seconds). For all the plots, the larger the value the better the method. The dark horizontal line in each violin plot represents the mean value over 10 runs of each algorithm.

Brofos and Lederman [30] for MMHMC would provide an improvement on a normalised ESS basis.

The algorithms have similar predictive performance on the benchmark datasets, with MMHMC and A-MMHMC algorithms consistently outperforming the other methods on all the datasets. The predictive performance is also similar between the antithetic and non-antithetic counterparts.

VIII. CONCLUSION

We present the antithetic Separable Shadow Hamiltonian Hybrid Monte Carlo, the antithetic Magnetic Hamiltonian Monte Carlo and the antithetic Magnetic Momentum Hamiltonian Monte Carlo variance reduction schemes based on approximate Markov chain coupling. We compare these three new algorithms to the antithetic Hamiltonian Monte Carlo algorithm on classification tasks using Bayesian logistic regression, and on regression tasks using Bayesian neural network models.

Although the proposed methods require more computational resources than Hamiltonian Monte Carlo in general, they result in improved effective sample sizes when compared to Hamiltonian Monte Carlo and antithetic Hamiltonian Monte Carlo, especially in higher parameter dimensions. We find that the antithetic versions of all the algorithms have higher effective sample sizes than their non-antithetic variants, which shows the usefulness of antithetic Markov Chain Monte Carlo methods.

This work can be improved by assessing the sensitivity of the methods to the path length as, in this study, the path length was fixed and the step size tuned. In addition, the adaptive tuning of the magnetic term, which is an open research problem, in Magnetic Hamiltonian Monte Carlo could also improve on the results presented in this paper. In future work we plan to consider larger datasets such as the MNIST dataset and deeper neural networks, albeit the computation cost will be higher. A comparison of the performance of the proposed algorithms to their control-variate counterparts, as well as exploration of the possibility of using Riemannian manifold based Monte Carlo algorithms are also of interest.

ACKNOWLEDGMENT

The computations in this work were performed on resources provided by the Center for High Performance Computing (CHPC) with the Council of Scientific and Industrial Research (CSIR) South Africa.

REFERENCES

[1] L. V. Jospin, W. Buntine, F. Boussaid, H. Laga, and M. Bennamoun, “Hands-on Bayesian neural networks—a tutorial for deep learning users,” 2020, arXiv:2007.06823. [Online]. Available: http://arxiv.org/abs/2007.06823
[2] R. Mbuvha and T. Marwala, “Bayesian inference of COVID-19 spreading rates in South Africa,” PLoS ONE, vol. 15, no. 8, Aug. 2020, Art. no. e0237126.
[3] W. Mongwe and K. Malan, “A survey of automated financial statement fraud detection with relevance to the South African context,” South Afr. Comput. J., vol. 32, no. 1, pp. 74–112, Jul. 2020.
[4] R. Mbuvha, I. Bouklaibet, and T. Marwala, “Bayesian automatic relevance determination for feature selection in credit default modelling,” in Proc. Int. Conf. Artif. Neural Netw. Cham, Switzerland: Springer, 2019, pp. 420–425.
[5] W. T. Mongwe and K. M. Malan, “The efficacy of financial ratios for fraud detection using self-organising maps,” in Proc. IEEE Symp. Ser. Comput. Intell. (SSCI), Dec. 2020, pp. 1100–1106.
[6] R. Neal, “MCMC using Hamiltonian dynamics,” in Handbook of Markov Chain Monte Carlo, S. Brooks, A. Gelman, G. L. Jones, and X. L. Meng, Eds. London, U.K.: Chapman & Hall, 2011, pp. 116–162.
[7] M. D. Hoffman and A. Gelman, “The no-U-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo,” J. Mach. Learn. Res., vol. 15, pp. 1933–1937, 2014.
[8] M. Betancourt, “A conceptual introduction to Hamiltonian Monte Carlo,” 2017, arXiv:1701.02434. [Online]. Available: http://arxiv.org/abs/1701.02434

[9] Z. Wang, S. Mohamed, and N. Freitas, “Adaptive Hamiltonian and Riemann manifold Monte Carlo,” in Proc. Int. Conf. Mach. Learn., 2013, pp. 1462–1470.

[10] M. Girolami and B. Calderhead, “Riemann manifold Langevin and Hamiltonian Monte Carlo methods,” J. Roy. Statist. Soc. B, Stat. Methodol., vol. 73, no. 2, pp. 123–134, Mar. 2011.

[11] D. Levy, M. D. Hoffman, and J. Sohl-Dickstein, “Generalizing Hamiltonian Monte Carlo with neural networks,” 2017, arXiv:1711.09268. [Online]. Available: http://arxiv.org/abs/1711.09268

[12] P. E. Jacob, J. O’Leary, and Y. F. Atchadé, “Unbiased Markov chain Monte Carlo swindles,” in Proc. 23rd Int. Conf. Artif. Intell. Statist., 2020, pp. 3774–3783.

[13] W. T. Mongwe et al., “Antithetic Magnetic and Shadow Hamiltonian Monte Carlo" VOLUME 9, 2021 49867

WILSON TSAKANE MONGWE was born in Tembisa, Gauteng, South Africa. He received the B.Sc. degree in computing from the University of South Africa, and the B.Bus.Sci. degree in actuarial science and the master’s degree in mathematical finance from the University of Cape Town. He is currently pursuing the Ph.D. degree in machine learning with the University of Johannesburg. His research interests include Bayesian machine learning and Markov Chain Monte Carlo methods.

He was a recipient of the Google Ph.D. Fellowship for his Ph.D. degree.

RENDANI MBUVHA was born in Venda, Limpopo, South Africa. He received the B.Sc. degree (Hons.) in actuarial science and statistics from the University of Cape Town and M.Sc. degree in machine learning from the KTH Royal Institute of Technology, Sweden. He is currently pursuing the Ph.D. degree in machine learning with the University of Johannesburg. He is also a Lecturer in statistics and actuarial science with the University of Witwatersrand. He is also a Qualified Actuary and a Holder of the Chartered Enterprise Risk Actuary Designation. He was a recipient of the Google Ph.D. Fellowship for his Ph.D. degree.

TSHILODI MARWALA (Senior Member, IEEE) was born in Duthuni, Venda, Limpopo, South Africa, in 28 July 1971. He received the bachelor’s degree in mechanical engineering from Case Western Reserve University, Cleveland, OH, USA, in 1995, the master’s degree in mechanical engineering from the University of Pretoria, Pretoria, South Africa, in 1997, and the Ph.D. degree in engineering from the University of Cambridge (St Johns College), Cambridge, U.K., in 2000. He is currently a Registered Professional Engineer (Pr.Eng.) with the Engineering Council of South Africa. He has also completed other leadership courses from the Columbia Business School, the National University of Singapore, the GIBS University of Pretoria, the Harvard Business School, and the University of South Africa. He is currently the Vice Chancellor and Principal of the University of Johannesburg. He is affiliated with the School of Electrical Engineering, University of Johannesburg. His contributions in artificial intelligence field come in forms of over 15 books. He has authored over 50 peer-reviewed chapters, over 50 journal publications, and over 150 conference publications. Some of his accomplishments include being a fellow of the Cambridge Commonwealth Trust in 1997, CSIR in 2005, the South African Academy of Engineering in 2007, TWAS, The World Academy of Science in 2010, the African Academy of Science in 2013, and the South African Institute of Electrical Engineers in 2016. He has won many awards including notably, the Bronze Order of Mapungubwe awarded by the President of the Republic of South Africa, in 2004, and the TWAS-AAS-Microsoft award for Young Scientists in 2009. He also holds the Deputy Chairman of the Presidential Fourth Industrial Revolution Commission of South Africa position and has contributed to over 50 articles to local press and journals on the subject of Fourth Industrial Revolution and Economics.

* * *

VOLUME 9, 2021 49867