Adaptive Magnetic Hamiltonian Monte Carlo

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

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

Corresponding author: Wilson Tsakane Mongwe (e-mail: 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 Magnetic Hamiltonian Monte Carlo (MHMC) is a Markov Chain Monte Carlo method that expands on Hamiltonian Monte Carlo (HMC) by adding a magnetic field to Hamiltonian dynamics. This magnetic field offers a great deal of flexibility over HMC and encourages more efficient exploration of the target posterior. This results in faster convergence and lower autocorrelations in the generated samples compared to HMC. However, as with HMC, MHMC is sensitive to the user specified trajectory length and step size. Automatically setting the parameters of MHMC is yet to be considered in the literature. In this work, we present the Adaptive MHMC (A-MHMC) algorithm which extends MHMC in that it automatically sets the parameters of MHMC and thus eliminates the need for the user to manually set a trajectory length and step size. The trajectory length adaptation is based on an extension of the No-U-Turn Sampler (NUTS) methodology to incorporate the magnetic field present in MHMC, while the step size is set via dual averaging during the burn-in period. Empirical results based on experiments performed on jump diffusion processes calibrated to real world financial market data, a simulation study using multivariate Gaussian distributions and real world benchmark datasets modelled using Bayesian Logistic Regression show that A-MHMC outperforms MHMC and NUTS on an effective sample size basis. In addition, A-MHMC provides significant relative speed up (up to 40 times) over MHMC and produces similar time normalised effective samples sizes relative to NUTS.

INDEX TERMS Bayesian Logistic Regression, Jump Diffusion Processes, Adaptive, Magnetic Hamiltonian Monte Carlo, Markov Chain Monte Carlo

I. INTRODUCTION

Markov Chain Monte Carlo (MCMC) methods are a key inference tool in the inference of probabilistic machine learning models [1, 2]. One of the simplest MCMC methods is the random walk Metropolis-Hastings [3] algorithm which suffers from random walk behaviour and consequently generates highly correlated samples [4]. This method has been extended using gradient-free MCMC techniques such as multiple-try Metropolis, delayed rejection techniques and methods that adapt the proposal distribution [5, 6, 4]. It has been further enhanced through methods that employ gradient information of the target posterior such as the Metropolis Adjusted Langevin Algorithm (MALA) and Hamiltonian Monte Carlo (HMC) [7, 8].

HMC has been successfully employed in the inference of probabilistic machine learning models and has been applied in various fields including renewable energy, health and cosmology [8, 2, 9, 7, 10, 11, 12, 13, 14, 15, 16, 10, 15, 17]. This algorithm is the preferred MCMC method in practice due to its ability to incorporate first-order gradient information about the target posterior distribution. This leads to better exploration of the posterior and lower autocorrelations in the generated samples when compared to the Metropolis-Hastings algorithm, Gibbs sampling and MALA [8, 2, 7, 9]. However, HMC still produces samples with relatively high autocorrelations [11, 7].

Various extensions to HMC have been introduced to improve the efficiency of HMC. These include the Riemannian Manifold Hamiltonian Monte Carlo [7] (RMHMC) method that takes into account the local geometry of the target posterior, the No-U-Turn Sampler [18] (NUTS) algorithm which automatically tunes the HMC parameters, thermostat-
assisted continuously-tempered Hamiltonian Monte Carlo [19, 20] which better explores multi-modal target densities and also provides the Bayesian evidence metric which can be used for model selection, as well as Magnetic Hamiltonian Monte Carlo [11, 14] (MHMC) which uses non-canonical Hamiltonian dynamics to better explore the posterior. Mongwe et al. [14] extend MHMC by utilising partial momentum refreshment, which they show enhances the performance of MHMC. These authors also show that using a random mass for the auxiliary momentum variable in MHMC to mimic the behaviour of quantum particles results in improved performance over MHMC and Quantum-Inspired Hamiltonian Monte Carlo [21, 10].

MHMC improves the exploration of HMC by introducing a magnetic field in addition to the force field already in HMC. The magnetic field provides an extra degree of freedom over HMC and results in efficient exploration of the target posterior and consequently faster convergence and lower autocorrelations in the generated samples when compared to HMC [11, 22, 23]. When the magnetic component of MHMC is set to zero, MHMC has exactly the same dynamics as HMC [11, 22, 23]. We leverage this key characteristic of MHMC in developing our proposed method. The main impediment of HMC and MHMC being broadly used in practice is that they are very sensitive to the trajectory length and step size parameters [18, 12]. If these parameters are not set correctly, it can result in very inefficient sampling of the posterior distribution. A trajectory length that is too short leads to random walk behaviour akin to the Metropolis-Hasting method [18]. A trajectory length that is too long results in a trajectory that traces back [18, 9]. On the other hand, small step sizes are computationally inefficient leading to correlated samples and poor mixing, while large step sizes compound discretisation errors leading to low acceptance rates [18, 9, 15, 12]. Tuning these two parameters requires the practitioner to conduct multiple time consuming pilot runs [18, 9, 12].

The No-U-Turn Sampler (NUTS) of Hoffman and Gelman [18] automates the tuning of the step size and trajectory length parameters for HMC. The step size parameter is tuned during the burn-in phase using the primal dual averaging methodology by targeting a user specified level of acceptance rate in the generated samples [18, 12]. The trajectory length is set by iteratively doubling the trajectory length until specific criterion are met [18, 12, 24]. Empirical results show that NUTS performs at least as efficiently as and sometimes more efficiently than a well tuned standard HMC method, without requiring user intervention or costly tuning runs [18]. The tuning of the parameters of MHMC is yet to be explored in the literature. This work aims to fill this gap in the literature.

MHMC is closely related to HMC and only differs with it due to the magnetic term. In this work, we expand on the NUTS methodology of Hoffman and Gelman [18] by extending it to incorporate the magnetic field present in MHMC. This new algorithm will automatically tune the trajectory length as in NUTS, with the step size tuned using dual averaging during the burn-in phase while leaving the target distribution invariant. We refer to this new method as Adaptive MHMC (A-MHMC).

Empirical results on the sampling of jump diffusion processes calibrated to real world financial market data, modelling of real world benchmark classification datasets using Bayesian logistic regression and simulation studies utilising multivariate Gaussian distributions show that A-MHMC outperforms MHMC and NUTS on an effective sample size basis without the user having to manually set the trajectory length and step size. The proposed method is more computationally expensive compared to NUTS due to the extra matrix multiplications required to incorporate the magnetic field. This leads to the new method producing similar effective sample sizes normalised by execution time when compared to NUTS.

Contributions: Our contributions in this work can be summarised as follows:

- We introduce the adaptive Magnetic Hamiltonian Monte Carlo method which automatically tunes the trajectory length and step size parameters of Magnetic Hamiltonian Monte Carlo and leaves the target density invariant.
- Numerical examples using various targets are provided demonstrating significant improvements over Magnetic Hamiltonian Monte Carlo and produces higher effective sample sizes than the No-U-Turn Sampler.

The remainder of this paper is structured as follows: Section II discuss the Markov Chain Monte Carlo methods that form the basis of the new method, Section III presents the proposed method, Section IV outlines the experiments conducted, Section V presents and discusses the results of the experiments and we provide the conclusion in Section VI.

II. THE NO-U-TURN SAMPLER

The Hamiltonian Monte Carlo (HMC) algorithm better explores the target posterior distribution due to its use of Hamiltonian dynamics [8, 1]. Hamiltonian dynamics incorporate the first-order gradient information about the target posterior distribution. The HMC adds an auxiliary momentum variable to the parameter space and the resultant Hamiltonian is given as:

$$H(w, p) = U(w) + K(p)$$

(1)

where $U(w)$ is the negative log-likelihood of the target posterior distribution and $K(p)$ is the kinetic energy defined by the kernel of a Gaussian as

$$K(p) = \frac{1}{2} \log(2\pi)^D |M| + \frac{p^T M^{-1} p}{2},$$

(2)

where $M$ is a positive definite mass matrix.

The the systems of equations governing the path of the chain are defined by Hamilton’s equations at a fictitious time $t$ as [25]:

$$\frac{dw}{dt} = \frac{\partial H(w, p)}{\partial p}; \quad \frac{dp}{dt} = -\frac{\partial H(w, p)}{\partial w}.$$ 

(3)
The leapfrog integration scheme is utilised to explore the space. The update equations for the leapfrog integration scheme are [25]:

\[
\begin{align*}
P_{t+\epsilon} &= P_t + \frac{\epsilon}{2} \partial H \left( P_t, P_{t+\epsilon} \right) \\
W_{t+\epsilon} &= W_t + \epsilon M^{-1} P_{t+\epsilon} \\
\frac{P_{t+\epsilon}}{P_{t+2\epsilon}} &= \frac{\partial H \left( W_{t+\epsilon}, P_{t+\epsilon} \right)}{\partial W}
\end{align*}
\]

(4)

where \( \epsilon \) is the discretisation step size. Due to the discretisation errors arising from the leapfrog integration, a Metropolis-Hastings acceptance step is then performed in order to accept or reject the proposed sample [25, 9]. The leapfrog steps are repeated until the maximum trajectory length \( L \) is reached. For the HMC algorithm, one needs to manually tune the step size and the trajectory length. Tuning these parameters typically requires multiple time consuming pilot runs [18].

The No-U-Turn Sampler (NUTS) of Hoffman and Gelman [18] automates the tuning of the HMC parameters. The step size is tuned through primal dual averaging during an initial burn-in phase by targeting a user specified level of sample acceptance. The trajectory length parameter is automatically tuned by iteratively doubling the trajectory length until the Hamiltonian becomes infinite or the chain starts to trace back [15, 12]. That is, when the last proposed position state \( w^* \) starts becoming closer to the initial position \( w \). This happens if: \( (w^* - w) \times P \leq 0 \). This approach however, violates the detailed balance condition. To overcome this problem in NUTS, a path of states \( B \) is generated such that its size is determined by the termination criterion [12, 18]. The set \( B \) has the following property, which is required for detailed balance [12, 18]:

\[
P(B|z) = P(B|z') \quad \forall z, z' \in B.
\]

(5)

where \( z = (w, P) \) is the current state. That is, the probability of generating \( B \), starting from any of its members is the same [12, 18]. Having \( B \), the current state \( z \) and a slice variable, \( u \), that is uniformly drawn from the interval \([0, \pi_z(z)]\), a set of chosen states \( C \) [12, 18]:

\[
C := \{ z' \in B \mid s.t \pi_z(z) \geq u \}
\]

(6)

is constructed from which the next state is drawn uniformly. Algorithm 1 shows the pseudo-code for NUTS with dual averaging, showing how the trajectory length \( L \) and the step size \( \epsilon \) are set automatically.

III. ADAPTIVE MAGNETIC HAMILTONIAN MONTE CARLO

The recently introduced Magnetic Hamiltonian Monte Carlo (MHHMC) algorithm enhances HMC by equipping it with a magnetic field [11]. This magnetic field encourages better exploration of the posterior [11, 26]. This results in faster convergence and lower autocorrelations in the generated samples when compared to HMC [11, 14].

Algorithm 1: NUTS with dual averaging

Input: \( M, M^{\text{adapt}}, \epsilon_0, w^0 \)

Output: \((w)^N_{m=0}\)

1: for \( m \to 1 \) to \( M \) do
2: \( p_{m-1} \sim \mathcal{N}(0, I), a \sim \text{Unif}[0, \exp\{-H(w_{m-1}, p_{m-1})\}]\)
3: \( w^- = w_{m-1}, p^- = p_{m-1}, w^+ = w_{m-1}, p^+ = p_{m-1}, j = 0, n = 1, s = 1.\)
4: while \( s \leq 1 \) do
5: \( v \sim \text{Unif}\{[-1, 1]\}\)
6: if \( v = -1 \) then
7: \( w^-, p^-, w^+, p^+, n', s', \alpha, n_{\alpha} = \text{BuildTree}(w^-, p^-, u, v, j, \epsilon_{m-1}, w_{m-1}, p_{m-1})\)
8: else
9: \( w^-, p^-, w^+, p^+, n', s', \alpha, n_{\alpha} = \text{BuildTree}(w^+, p^+, u, v, j, \epsilon_{m-1}, w_{m-1}, p_{m-1})\)
10: end if
11: if \( s' = 1 \) then
12: With prob. \( \min\{1, n'/m\} \), set \( w^m = w', p^m = p' \)
13: end if
14: \( n = n + n', j = j + 1 \)
15: \( s = s' \text{I}(\epsilon_{m-1} w - w') \geq 0 | \text{I}(\epsilon_{m-1} w - w') \geq 0 \)
16: end while
17: if \( m < M^{\text{adapt}} \) then
18: \( H_m = (1 - m/m_{\text{tol}}) H_{m-1} + \frac{1}{m_{\text{tol}}} (\alpha - \alpha_m)\)
19: \( \log \epsilon_m = \mu - \frac{m}{m_{\text{tol}}} H_m, \)
20: \( \log \epsilon_m = m^{-\kappa} \log \epsilon_m + (1 - m^{-\kappa}) \log \epsilon_{m-1} \)
21: \( \epsilon_{m} = \tilde{\epsilon}_{M^{\text{adapt}}} \)
22: end if
23: end for

function BuildTree\((w, p, u, v, j, \epsilon, w^0, p^0)\)

24: if \( j = 0 \) then
25: \( w', p' = \text{Leapfrog}(w, v, \epsilon)\)
26: \( n' = \text{I}(u < \exp(-H(w', p'))\),
27: \( s' = \text{I}(u < \exp(\Delta_{\text{max}} - H(w', p'))\)
28: \( n_{\alpha} = 1, \alpha = \min\{1, \exp(H(w^0, p^0) - H(w', p'))\} \)
29: \( \text{return } w', p', w^+, p^+, n', s', \alpha', n_{\alpha} = \text{BuildTree}(w^0, p^0, u, v, j - 1, \epsilon, w^0, p^0)\)
30: end if
31: if \( s' = 1 \) then
32: if \( v = -1 \) then
33: \( w^-, p^-, w^+, p^+, n', s' = \text{BuildTree}(w^0, p^0, u, v, j - 1, \epsilon, w^0, p^0)\)
34: else
35: \( -w^-, p^-, w^+, p^+, n', s', \alpha', n_{\alpha} = \text{BuildTree}(w^+, p^+, u, v, j - 1, \epsilon, w^0, p^0)\)
36: end if
37: With prob. \( n'/n_{\alpha} \), set \( w^m = w', p^m = p' \)
38: \( \alpha' = \alpha' + \alpha' n'/n' + n', n'_{\alpha} = n'_{\alpha} + n'_{\alpha} \)
39: \( s' = s' \text{I}[(w^m - w^-) p^- \geq 0 | (w^m - w^-) p^+ \geq 0] \)
40: end if
41: \( \text{return } w^-, p^-, w^+, p^+, w', p', n', s', \alpha', n_{\alpha} \)
42: end if
MHMC uses the same Hamiltonian as in HMC in equation (1), but exploits non-canonical Hamiltonian dynamics where the canonical matrix now has a non-zero element on the diagonal. The MHMC dynamics are [27]:

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

(7)

where G is the term that represents the magnetic field. The update equations for the non-canonical MHMC integration scheme are given as [11]:

$$\begin{align*}
    w_{t+\epsilon} &= w_t + G^{-1}(\exp(G\epsilon) - I) p_{t+\frac{\epsilon}{2}} \\
    p_{t+\frac{\epsilon}{2}} &= \exp(G\epsilon) p_{t+\frac{\epsilon}{2}} \\
    p_{t+\epsilon} &= p_{t+\frac{\epsilon}{2}} + \frac{\epsilon}{2} \frac{\partial H}{\partial p}(w_{t+\epsilon}, p_{t+\frac{\epsilon}{2}})
\end{align*}$$

(8)

This means that MHMC only differs from HMC dynamics by G being non-zero [11]. When G = 0, MHMC and HMC have the same dynamics. These MHMC dynamics cannot be integrated exactly and we resort to a numerical integration scheme with a Metropolis-Hastings acceptance step to ensure detailed balance [11]. The integrator for MHMC is not exactly the leapfrog integrator, but is very similar. This leapfrog-like integration scheme is shown as the Integrator function in Algorithm 2. As with HMC, the MHMC algorithm has trajectory length and step size parameters that need to be tuned. Tuning of these parameters often requires expert judgment, which makes the method difficult to use in practice for non-specialists.

We now introduce the adaptive Magnetic Hamiltonian Monte Carlo (A-MHMC) algorithm which automatically tunes the parameters of MHMC. The proposed method addresses a key limitation of MHMC - which is how to tune the step size and trajectory length parameters of MHMC. Without a systematic algorithm for tuning an MHMC algorithm its utility becomes severely limited. The tuning of the parameters of MHMC is yet to be explored in the literature. The proposed algorithm addresses this gap in the literature and makes the MHMC accessible to non-experts.

This new algorithm is based on incorporating the magnetic field in MHMC into the NUTS methodology in Algorithm 1.

In particular, the step size is tuned via primal dual averaging during the burn-in phase, while the trajectory length is set by a recursive doubling procedure until the particle traces back. The A-MHMC method differs from NUTS in that a different integrator is used. Instead of using the leapfrog integrator associated with HMC, the leapfrog-like integrator (which is the explicit and symplectic) corresponding to MHMC is used.

Thus the A-MHMC algorithm combines the non-canonical dynamics of MHMC with the automation benefits of NUTS to create a new sampler.

The algorithmic difference of A-MHMC with NUTS is highlighted in Algorithm 2. The stopping criteria in A-MHMC is the same as in the original NUTS algorithm outlined in 6. It is crucial to note that when G = 0, this new algorithm is equivalent to NUTS. This results in the proposed method inheriting the properties of NUTS such as detailed balance and time-reversibility.

**Theorem III.1.** A-MHMC satisfies detailed balance and leaves the target distribution invariant.

**Proof.** To show that A-MHMC satisfies detailed balance one would need to show that a NUTS algorithm that uses the leapfrog-like numerical integrator is both sympletic and reversible, and hence satisfies detailed balance. This amounts to showing that the leapfrog-like integrator satisfies detailed balance as this is what is required to show that NUTS satisfies detailed balance [18, 12].

We now argue that the leapfrog-like integrator used in A-MHMC satisfies detailed balance. The map defined by integrating the non-canonical Hamiltonian system [11]

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

(9)

with initial conditions (w, p), where A is any square invertible, antisymmetric matrix induces a flow on the coordinates (w, p) that is still energy conserving with respect to A which also implies volume-preservation of the flow [11]. In addition, setting

$$A = \begin{bmatrix} 0 & I \\ -I & G \end{bmatrix}$$

leads to MHMC dynamics and ensures that non-canonical dynamics have a (pseudo) time-reversibility symmetry as shown in [11]. These properties of being energy conserving, volume preserving and time-reversible guarantee that the leapfrog-like scheme satisfies detailed balance as required. Thus A-MHMC satisfies detailed balance and hence leaves the target invariant.

It should be noted that the approach undertaken in this paper is not the only method that could have been used to tune the parameters of MHMC. Wang et al. [28, 29] introduce a Bayesian optimisation framework for tuning the parameters of Hamiltonian and Riemann Manifold Hamiltonian Monte Carlo samplers. The authors show that their approach is ergodic and in some instances precludes the need for more complex samplers. Buchholz et al. [30] adapt Hamiltonian Monte Carlo kernels using sequential Monte Carlo. They show that their approach improves as the dimensionality of the problem increases. Hoffman et al. [31] propose an adaptive MCMC scheme for tuning the trajectory length parameter in HMC and show that this new technique typically yields higher effective samples sizes normalised by the number of gradient evaluation when compared to NUTS. We plan to consider these and other approaches for tuning the parameters of MHMC in future work.

The key advantage of the proposed A-MHMC algorithm over MHMC is that it removes the need to manually tune the MHMC parameters. This makes MHMC more accessible as it removes the need for expert judgement in tuning the
It is not immediately clear as to how one should set or computational cost to HMC. Thus on targets where NUTS that of NUTS as MHMC has similar, albeit slightly higher, computational cost of the proposed method is inline with depends on the target posterior under consideration. The number of function evaluations is not deterministic and trajectory length used to generate each sample varies. As component is still an open area of research. In this work, it is important to note that the number of likelihood function evaluations for NUTS is not deterministic as the trajectory length used to generate each sample varies. As our proposed method is based on the NUTS methodology, the number of function evaluations is not deterministic and depends on the target posterior under consideration. The computational cost of the proposed method is inline with that of NUTS as MHMC has similar, albeit slightly higher, computational cost to HMC. Thus on targets where NUTS has high execution times, we would expect A-MHMC to also have high execution times.

The magnetic field G in MHMC and A-MHMC provides an extra degree of freedom for both these algorithms [11]. It is not immediately clear as to how one should set or tune this matrix, but this should ideally be conducted in an automated manner [11, 27, 14]. Tuning the magnetic component is still an open area of research. In this work, we follow the directions of the authors [11, 27, 14, 21] and select only a few dimensions to be influenced by the magnetic field. In particular, G was set such that \( G_{11} = G_{31} = g \), \( G_{11} = G_{15} = -g \) and zero elsewhere for \( g = 0.2 \). Note that this particular form of G was chosen as G has to be antisymmetric [11]. In addition, we chose the first and fifth dimensions as the target posteriors used in this paper include jump diffusion processes, which have only five parameters. Thus we did not consider higher dimensions as we wanted to use the same setting for G for all the target posterior distributions considered in this paper.

This is means that the choice of G is not necessarily the optimal choice for all the target distributions considered, but was sufficient for our purposes as this basic setting still leads to good performance of the algorithm. Tuning G for each target posterior should result in improved performance compared to the results presented in this manuscript. An alternative approach to the selection of G would have been to follow [23] and selecting G to be random antisymmetric matrix. We plan to explore this approach in future work.

### IV. EXPERIMENTS

We consider three test problems to demonstrate the performance of A-MHMC over MHMC and NUTS. We first study a jump diffusion process [32], whose transition density is an infinite mixture of Gaussian distributions with the mixing weights being probabilities from a Poisson distribution. We then perform a simulation study using multivariate Gaussian distributions with increasing dimensionality as in [33, 14]. We proceed to consider performance on Bayesian Logistic Regression over various benchmark datasets outlined in [7].

Performance of the methods is measured via multivariate effective sample size (ESS) and ESS per second. We also assess the convergence of the methods using the \( R \) metric of Gelman and Rubin [34, 35]. For all the algorithms, the step size was set by targeting an acceptance rate of 80% during the burn-in period. A total of 10 independent chains were run for each method on each target distribution. All experiments were conducted on a 64bit CPU using PyTorch. For MHMC, we considered the trajectory lengths to be in the set \( \{10, 50, 100\} \). This allowed us to assess the sensitivity of the results to the trajectory length.

The \( R \) diagnostic of Gelman and Rubin [34] is a popular method for establishing the convergence of MCMC chains [35]. This diagnostic relies on running multiple chains \( \{X_{i0}, X_{i1}, \ldots, X_{im} \} \) for \( i \in \{1, 2, 3, \ldots, m\} \) staring at various initial states with \( m \) being the number of chains and \( n \) being the sample size. Using these parallel chains, two estimators of the variance can be constructed. The estimators are the between-the-chain variance estimate and the within-the-chain variance. When the chain has converged, the ratio of these two estimators should be one. The \( R \) metric, which is formally known as the potential scale reduction factor, is
defined as:

$$\hat{R} = \frac{\hat{V}}{W}$$  \hspace{1cm} (10)

where

$$W = \sum_{i=1}^{m} \sum_{j=0}^{n-1} \frac{(X_{ij} - \bar{X}_i)^2}{m(n-1)}$$  \hspace{1cm} (11)

is the within-chain variance estimate and $\hat{V} = \frac{n-1}{n}W + \frac{B}{n}$ is the pooled variance estimate which incorporates the between-chains

$$B = \sum_{j=0}^{n-1} \frac{(\bar{X}_i - \bar{X})^2}{m - 1}$$  \hspace{1cm} (12)

and within-chain $W$ variance estimates with $\bar{X}_i$ and $\bar{X}$ being the $i^{th}$ chain mean and overall mean respectively for $i \in \{1, 2, 3, ..., m\}$. Values larger than the convergence threshold of 1.05 for the $\hat{R}$ metric indicate divergence of the chain [23, 34].

The ESS calculation used in this work is the multivariate ESS calculation outlined in [36, 37]. 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 [7, 36, 27]. 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:

$$m\text{ESS} = n \times \left( \frac{|A|}{|\Sigma|} \right)^{\frac{1}{2}}$$  \hspace{1cm} (13)

where $n$ is the number of generated samples, $D$ is the number of parameters, $|A|$ is the determinant of the sample covariance matrix and $|\Sigma|$ is the determinant of the estimate of the Markov chain standard error. When $D = 1$, $m\text{ESS}$ is equivalent to the univariate ESS measure [36]. Note that when there are no correlations in the chain, we have that $|A| = |\Sigma|$ and $m\text{ESS} = n$.

A. Jump Diffusion Process

The modelling of asset returns as typically been undertaken by models that explicitly or implicitly assume that returns are normally distributed [38, 39, 16]. Jump diffusion processes are appealing to use to model asset returns, instead of the traditional diffusion models such as geometric Brownian motion, because they produce returns which are leptokurtic [38, 39, 37, 16]. The jump diffusion model that is analysed in this work is the [32] one-dimensional Markov process $\{S_t, t \geq 0\}$ which is characterised by the following stochastic differential equation (SDE):

$$d \ln S_t = \left( \mu - \frac{1}{2} \sigma^2 \right) dt + \sigma dB_t + \frac{\mu}{\lambda} d\Lambda_t$$  \hspace{1cm} (14)

where $\mu$ is the drift coefficient, $\sigma$ is the diffusion coefficient, $\{B_t, t \geq 0\}$ is a standard Brownian motion process, $Y_i$ is the random size of the $i$th jump and $\{N_t, t \geq 0\}$ is a Poisson process with intensity $\lambda$. Furthermore, the jump sizes $Y_i$ are assumed to be independent and identically distributed random variables, and independent of both $\{N_t, t \geq 0\}$ and $\{B_t, t \geq 0\}$. In addition, it is assumed that $Y_i \sim N(\mu_{jump}, \sigma_{jump}^2)$. The SDE in equation (14) implies the following transition density:

$$\frac{1}{\sqrt{2\pi \sigma^2 \tau}} \int_{w}^{\infty} e^{-\frac{(w-x)^2}{2\sigma^2 \tau}} d\phi$$

where $w$ and $x$ are real numbers, $\tau$ is the time difference between $S(t+\tau)$ and $S(t)$ and $\phi$ is the probability density function of a standard normal random variable. Given that the transition density in equation (15) is an infinite mixture of normal distributions. In this work we truncate the infinite summation in equation (15) to the first 10 terms. The prior distribution over the parameters was the standard normal distribution.

We calibrate jump diffusion processes to real world financial datasets. The datasets consists of daily prices that were obtained from Google Finance. The specifics of the datasets are shown in Table 1 and are as follows:

- **Bitcoin dataset**: Daily data for the crypto-currency index from 1 Jan 2017 to 31 Dec 2020. The prices were converted into log returns.
- **USDZAR dataset**: Daily data for the currency from 1 Jan 2017 to 31 Dec 2020. The prices were converted into log returns.

The formula used to calculate the log returns is given as:

$$r_i = \log \left( \frac{S_i}{S_{i-1}} \right)$$

where $r_i$ is the log return on day $i$ and $S_i$ is the stock or currency level on day $i$. The descriptive statistics of the datasets is presented in Table 2. This table shows that the USDZAR dataset has a very low kurtosis, suggesting that it has very few, if any, jumps when compared to the Bitcoin dataset.

A 90-10 train-test split was used for both datasets. A total of 2 000 samples were generated after a burn-in period of 500 samples.

B. Multivariate Gaussian Distributions and Bayesian Logistic Regression

We first undertake a simulation study using multivariate Gaussian distributions. In this simulation study, the task is to sample from $D$-dimensional Gaussian distributions. The covariance matrix is set to be diagonal, with the standard deviations simulated from a log normal distribution with mean zero and unit standard deviation. For the simulation study, we consider the number of dimensions $D$ to be in the set $\{10, 20, 50, 100\}$. A total of 10 000 samples were generated for each value of $D$, with the first 5 000 being discarded as the burn-in.

The details of the real world datasets modeled using logistic regression are displayed in Table 1. These datasets...
TABLE 1: Real world datasets used in this paper. \( N \) represents the number of observations. BJD is Bayesian jump diffusion process and BLR is Bayesian Logistic Regression. \( D \) represents the number of model parameters.

| Dataset       | Features | \( N \) | Model | \( D \) |
|---------------|----------|---------|-------|--------|
| Bitcoin       | 1        | 1,461   | BJD   | 5      |
| USDZAR        | 1        | 1,425   | BJD   | 5      |
| Pima          | 7        | 532     | BLR   | 8      |
| Heart         | 13       | 270     | BLR   | 14     |
| Australian    | 14       | 690     | BLR   | 15     |
| German        | 24       | 1,000   | BLR   | 25     |

TABLE 2: Descriptive statistics for the real world jump diffusion process financial market datasets considered in this work.

| Dataset       | mean     | standard deviation | skew   | kurtosis |
|---------------|----------|--------------------|--------|----------|
| Bitcoin       | 0.00187  | 0.00447            | -0.925 | 13.586   |
| USDZAR        | 0.00019  | 0.00854            | 0.117  | 1.673    |

are binary classification tasks and are thus modeled using Bayesian logistic regression. The specifics of the datasets are:

- **Pima Indian Diabetes dataset** - This dataset has 7 features and a total of 532 observations. The aim for this dataset is to predict if a patient has diabetes based on diagnostic measurements mad on the patient [40].

- **Heart dataset** - This dataset has 13 features and 270 data points. The purpose of the dataset is to predict the presence of heart disease based on medical tests performed on a patient [40].

- **Australian credit dataset** - This dataset has 14 features and 690 data points. The objective for this dataset is to assess applications for credit cards [40].

- **German credit dataset** - This dataset has 25 features and 1,000 data points. The aim for this dataset was to classify a customer as either good or bad credit [40].

A 90-10 train-test split was used and the input features were normalised for all the datasets. 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 [7]. For the logistic regression benchmark datasets, we generated 10,000 samples with a burn-in period of 5,000 samples.

V. RESULTS AND DISCUSSION

The experiments were implemented in PyTorch and were carried out on a 64-bit precision CPU. The performance, in terms of \( \hat{m}\text{ESS} \), of the algorithms for the simulation study and logistic regression datasets is shown in Figure 1. The detailed results for the real world jump diffusion and logistic regression datasets across different metrics are shown in Tables 3 and 4. Note that the execution time \( t \) in Tables 3 and 4 is in seconds.

The first row in Figure 1 shows the \( \hat{m}\text{ESS} \) for the multivariate Gaussian distributions in the simulation study with increasing dimensionality \( D \). The second row in Figure 1 shows the \( \hat{m}\text{ESS} \) for the logistic regression datasets. The results in in Tables 3 and 4 are the mean results over the 10 runs for each algorithm. Note that we use the mean values over the 10 runs to form our conclusions about the performance of the algorithms.

The first row of Figure 1 shows that in all the values of \( D \), A-MHMC performs comparably or better than the MHMC method with fixed trajectory lengths. The results also show that the optimal trajectory lengths to use for the MHMC for each \( D \) differs with no obvious pattern, illustrating how difficult it is to tune the trajectory length parameter. A-MHMC tunes this trajectory length in an automated fashion, thus removing the manual process that would have been required by the practitioner to select the optimal trajectory length for each \( D \). The results also show that for all the values of \( D \), A-MHMC outperforms NUTS.

The second row in Figure 1 shows that across all the real world logistic regression datasets, A-MHMC outperforms all the methods considered, with NUTS being second. This is also confirmed by the mean results, over 10 runs of the algorithms, shown in Tables 3 and 4. The A-MHMC produces significantly more effective sample sizes, even after taking into account the execution time, than the other MHMC with different trajectory lengths - except on the Bitcoin dataset where the adaptive algorithms (NUTS and A-MHMC) have very large execution times which causes poor normalised effective sample size performance. The A-MHMC method can provide up to 40 times relative speed up when compared to MHMC. This shows the benefit of A-MHMC’s ability to automatically tune the trajectory length without user intervention. A-MHMC and NUTS produce similar effective sample sizes normalised by execution time, and relative speed up, across all the datasets. With exception on the Bitcoin dataset, NUTS has the lowest execution time \( t \) on all the datasets, with A-MHMC being a close second. The low execution time was to be expected as A-MHMC does additional matrix operations, to take into account the magnetic term, when compared to NUTS.

Tables 3 and 4 show that the algorithms have similar predictive performance, with the MHMC algorithm with trajectory length equal to 100 marginally outperforming the other methods on all the logistic regression datasets. A-MHMC outperforms on the jump diffusion datasets. It is also worth noting that A-MHMC and NUTS have large executive times on the Bitcoin dataset and low execution times on the USDZAR dataset. The high execution time on the Bitcoin dataset is due to the high kurtosis of this dataset.

Figure 2 shows that all the MCMC methods have converged on all the target posteriors. This is further confirmed in Tables 3 and 4 where \( \hat{R} \) is close to one for all the algorithms with A-MHMC outperforming on the \( \hat{R} \) metric on the majority of the datasets.

VI. CONCLUSIONS

We present the Adaptive Magnetic Hamiltonian Monte Carlo algorithm that automatically tunes the trajectory length and step size parameters of Magnetic Hamiltonian Monte Carlo...
TABLE 3: Jump diffusion results averaged over 10 runs. The time $t$ is in seconds. NLL stands for negative log-likelihood. The values in **bold** indicate that the particular method outperforms the other methods on that specific metric. The relative speed up metric is with reference to the method with the lowest time normalised multivariate effective sample size.

| NUTS         | MHMC-10 | MHMC-50 | MHMC-100 | A-MHMC |
|--------------|---------|---------|----------|--------|
| **mESS**     | 598     | 104     | 834      | 889    | **1,068** |
| **t (in secs)** | 1,555   | **178** | 684      | 1,397  | 2,374    |
| **mESS/t**   | 0.384   | 0.584   | **1.219**| 0.647  | 0.450    |
| Test NLL     | -2,699  | -2,699  | -2,699   | -2,699 | **-2,767**|
| Relative Speed Up | 1.00    | 1.52    | **3.17** | 1.68   | 1.17     |
| $\hat{R}$    | **1.00**| 1.01    | 1.01     | 1.02   | **1.00** |

| **USDZAR** dataset |
|---------------------|
| **mESS** | 40 | 137 | 43 | 172 | **215** |
| **t (in secs)** | **45** | 172 | 710 | 1,402 | 117 |
| **mESS/t** | 0.939 | 0.800 | 0.061 | 0.123 | **1.835** |
| Test NLL | -4,830 | -4,758 | -4,816 | -4,860 | **-5,121**|
| Relative Speed Up | 15.39 | 13.11 | **1.00** | 2.02 | **30.08** |
| $\hat{R}$ | **0.99** | 1.01 | 1.02 | 1.01 | **1.00** |

TABLE 4: Mean 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 MMHC-50. The execution time $t$ is in seconds. The values in **bold** indicate that the particular method outperforms the other methods on that specific metric. The relative speed up metric is with reference to the method with the lowest time normalised multivariate effective sample size. AUC represents the area under the ROC curve.

| **Pima** dataset |
|------------------|
| **mESS** | 4,601 | 4,028 | 1,686 | 2,028 | **5,273** |
| **t (in secs)** | **18** | 49 | 197 | 400 | 26 |
| **mESS/t** | 245 | 81 | 8 | 5 | 203 |
| AUC | 0.827 | 0.827 | 0.826 | 0.828 | 0.827 |
| Relative Speed Up | **49.0** | 16.2 | 1.6 | 1.0 | 40.2 |
| $\hat{R}$ | 0.99 | 1.01 | 1.02 | 1.01 | **1.00** |

| **Heart** dataset |
|--------------------|
| **mESS** | 4,491 | 2,539 | 2,504 | 4,011 | **6,621** |
| **t (in secs)** | **22** | 48 | 194 | 388 | 31 |
| **mESS/t** | 199 | 51 | 12 | 10 | **212** |
| AUC | 0.896 | 0.896 | 0.895 | 0.898 | 0.896 |
| Relative Speed Up | 19.9 | 5.1 | 1.2 | 1.0 | **21.2** |
| $\hat{R}$ | 1.01 | **1.00** | 1.01 | 1.00 | **1.00** |

| **Australian credit** dataset |
|-----------------------------|
| **mESS** | 4,311 | 2,682 | 3,359 | 3,552 | **5,940** |
| **t (in secs)** | 30 | 51 | 203 | 407 | 43 |
| **mESS/t** | **143** | 52 | 16 | 9 | 135 |
| AUC | 0.954 | 0.954 | 0.954 | **0.955** | 0.954 |
| Relative Speed Up | 15.9 | 5.8 | 1.8 | 1.0 | 15.0 |
| $\hat{R}$ | 1.01 | 1.01 | 1.01 | 1.00 | **0.99** |

| **German credit** dataset |
|---------------------------|
| **mESS** | 5,362 | 2,307 | 3,809 | 2,428 | **6,945** |
| **t (in secs)** | **33** | 58 | 228 | 471 | 40 |
| **mESS/t** | 161 | 39 | 16 | 5 | **172** |
| AUC | 0.764 | 0.764 | 0.763 | **0.765** | 0.763 |
| Relative Speed Up | 32.2 | 7.8 | 3.2 | 1.0 | **34.4** |
| $\hat{R}$ | **1.00** | 1.02 | 1.01 | 1.01 | **1.00** |
FIGURE 1: Results for the datasets over 10 runs of each method. For each dataset, the plots show the multivariate effective sample size. 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.

without the user having to manually specify these parameters. We assess the performance of this new method against the No-U-Turn Sampler and Magnetic Hamiltonian Monte Carlo methods using 10, 50 and 100 trajectory lengths respectively. The methods are compared on the calibration of jump diffusion processes, Bayesian logistic regression and multivariate Gaussian targets.

The A-MHMC algorithm significantly outperforms MHMC on both an effective sample size basis and normalised effective sample size basis on the majority of the datasets. The empirical results show that A-MHMC outperforms NUTS on an effective sample size basis and produces similar normalised effective sample sizes when compared to NUTS.

This work can be improved by assessing the performance of the proposed method on deep neural networks and larger datasets such as MNIST. Comparing the performance of the method with Riemannian manifold based Markov Chain Monte Carlo methods would be an area of interest. Incorporating the adaptation of the magnetic field term in the proposed algorithm could also serve as a benefit on the method. We plan to also assess other methods for tuning parameters of MHMC, such as those based on utilising the variance of the change in the shadow Hamiltonian corresponding to the leapfrog-like integrator used in MHMC.

ACKNOWLEDGEMENTS

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

REFERENCES

[1] R. M. Neal et al., “Mcmc using hamiltonian dynamics,” Handbook of Markov Chain Monte Carlo, vol. 2, no. 11, p. 2, 2011.
[2] R. M. Neal, Bayesian learning for neural networks. Springer Science & Business Media, 2012, vol. 118.
[3] W. K. Hastings, “Monte carlo sampling methods using markov chains and their applications,” 1970.
[4] H. Haario, E. Saksman, and J. Tamminen, “Adaptive proposal distribution for random walk metropolis algorithm,” Computational Statistics, vol. 14, no. 3, pp. 375–395, 1999.
[5] M. Bédard, R. Douc, and E. Moulines, “Scaling analysis of multiple-try mcmc methods,” Stochastic Processes and their Applications, vol. 122, no. 3, pp. 758–786, 2012.
[6] L. Martino, “A review of multiple try mcmc algorithms for signal processing,” Digital Signal Processing, vol. 75, pp. 134–152, 2018.
[7] M. Girolami and B. Calderhead, “Riemann manifold langevin and hamiltonian monte carlo methods,” Journal of the Royal Statistical Society: Series B (Statistical Methodology), vol. 73, no. 2, pp. 123–214, 2011.
[8] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, “Hybrid monte carlo,” Physics letters B, vol. 195, no. 2, pp. 216–222, 1987.
FIGURE 2: Diagnostic negative log-likelihood trace plots for various targets. Similar results are observed for targets not included in the plot. Note that HMC-NUTS is the No-U-Turn Sampler while MHMC-NUTS is the A-MHMC method.

[9] M. Betancourt, “A conceptual introduction to hamiltonian monte carlo,” arXiv preprint arXiv:1701.02434, 2017.
[10] Z. Liu and Z. Zhang, “Quantum-inspired hamiltonian monte carlo for bayesian sampling,” arXiv preprint arXiv:1912.01937, 2019.
[11] N. Tripuraneni, M. Rowland, Z. Ghahramani, and R. Turner, “Magnetic hamiltonian monte carlo,” in International Conference on Machine Learning. PMLR, 2017, pp. 3453–3461.
[12] H. M. Afshar, R. Oliveira, and S. Cripps, “Non-volume preserving hamiltonian monte carlo and no-u-turnsamplers,” in International Conference on Artificial Intelligence and Statistics. PMLR, 2021, pp. 1675–1683.
[13] C. Heide, F. Roosta, L. Hodgkinson, and D. Kroese, “Shadow manifold hamiltonian monte carlo,” in International Conference on Artificial Intelligence and Statistics. PMLR, 2021, pp. 1477–1485.
[14] W. T. Mongwe, R. Mbuvha, and T. Marwala, “Magnetic hamiltonian monte carlo with partial momentum refreshment,” IEEE Access, vol. 9, pp. 108 009–108 016, 2021.
[15] R. Mbuvha and T. Marwala, “Bayesian inference of covid-19 spreading rates in south africa,” PloS one, vol. 15, no. 8, p. e0237126, 2020.
[16] W. T. Mongwe, R. Mbuvha, and T. Marwala, “Adaptively setting the path length for separable shadow...
hamiltonian hybrid monte carlo,” IEEE Access, vol. 9, pp. 138,598–138,607, 2021.

[17] R. Mbuvha, W. T. Mongwe, and T. Marwala, “Separable shadow hamiltonian hybrid monte carlo for bayesian neural network inference in wind speed forecasting,” Energy and AI, p. 100,108, 2021.

[18] M. D. Hoffman and A. Gelman, “The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo,” Journal of Machine Learning Research, vol. 15, no. 1, pp. 1593–1623, 2014.

[19] R. Luo, J. Wang, Y. Yang, and Z. Zhu, “Thermostat-assisted continuously-tempered hamiltonian monte carlo for bayesian learning,” in Advances In Neural Information Processing Systems 31 (NIPS 2018), vol. 31. Neural Information Processing Systems Foundation, Inc., 2018.

[20] M. Graham and A. Storkey, “Continuously tempered hamiltonian monte carlo,” in Advances in Approximate Bayesian Inference: NIPS 2016 Workshop, 2016.

[21] W. T. Mongwe, R. Mbuvha, and T. Marwala, “Quantum-inspired magnetic hamiltonian monte carlo,” Plos one, vol. 16, no. 10, p. e0258277, 2021.

[22] J. A. Brofos and R. R. Lederman, “Magnetic manifold hamiltonian monte carlo,” arXiv preprint arXiv:2010.07753, 2020.

[23] ———, “Non-canonical hamiltonian monte carlo,” arXiv preprint arXiv:2008.08191, 2020.

[24] R. Mbuvha, “Bayesian neural networks for short term wind power forecasting,” 2017.

[25] R. M. Neal, “Bayesian learning via stochastic dynamics,” in Advances in neural information processing systems, 1993, pp. 475–482.

[26] M. Gu and S. Sun, “Neural langevin dynamical sampling.” IEEE Access, vol. 8, pp. 31,595–31,605, 2020.

[27] W. T. Mongwe, R. Mbuvha, and T. Marwala, “Antithetic magnetic and shadow hamiltonian monte carlo,” IEEE ACCESS, vol. 9, pp. 1–11, 2021.

[28] Z. Wang, S. Mohamed, and N. Freitas, “Adaptive hamiltonian and riemann manifold monte carlo,” in International conference on machine learning. PMLR, 2013, pp. 1462–1470.

[29] Z. Wang and N. de Freitas, “Predictive adaptation of hybrid monte carlo with bayesian parametric bandits,” in NIPS Deep Learning and Unsupervised Feature Learning Workshop, vol. 30, 2011.

[30] A. Buchholz, N. Chopin, and P. E. Jacob, “Adaptive tuning of hamiltonian monte carlo within sequential monte carlo,” Bayesian Analysis, vol. 1, no. 1, pp. 1–27, 2021.

[31] M. Hoffman, A. Radul, and P. Sountsov, “An adaptive-mcmc scheme for setting trajectory lengths in hamiltonian monte carlo,” in International Conference on Artificial Intelligence and Statistics. PMLR, 2021, pp. 3907–3915.

[32] R. C. Merton, “Option pricing when underlying stock returns are discontinuous,” Journal of Financial Economics, vol. 3, no. 1–2, pp. 125 – 144, 1976.

[33] T. Radivojević and E. Akhmatskaya, “Mix & match hamiltonian monte carlo,” arXiv preprint arXiv:1706.04032, 2017.

[34] A. Gelman and D. B. Rubin, “Inference from iterative simulation using multiple sequences,” Statistical science, vol. 7, no. 4, pp. 457–472, 1992.

[35] V. Roy, “Convergence diagnostics for markov chain monte carlo,” Annual Review of Statistics and Its Application, vol. 7, pp. 387–412, 2020.

[36] D. Vats, J. M. Flegal, and G. L. Jones, “Multivariate output analysis for Markov chain Monte Carlo,” Biometrika, vol. 106, no. 2, pp. 321–337, 04 2019.

[37] W. T. Mongwe, R. Mbuvha, and T. Marwala, “Antithetic riemannian manifold and quantum-inspired hamiltonian monte carlo,” arXiv preprint arXiv:2107.02070, 2021.

[38] R. Cont, “Empirical properties of asset returns: stylized facts and statistical issues,” Quantitative Finance, vol. 1, pp. 223 – 236, 2001.

[39] W. T. Mongwe, “Analysis of equity and interest rate returns in south africa under the context of jump diffusion processes,” Masters Thesis at University of Cape Town, Dec 2015, last accessed: 18 March 2021. [Online]. Available: https://open.uct.ac.za/bitstream/handle/11427/16600/thesis_com_2015_mongwe_wilson_tsakane.pdf?isAllowed=y&sequence=1

[40] D. Michie, D. J. Spiegelhalter, C. C. Taylor, and J. Campbell, Eds., Machine Learning, Neural and Statistical Classification. USA: Ellis Horwood, 1995.
WILSON TSAKANE MONGWE was born in Tembisa (Gauteng, South Africa). He holds a BSc in Computing from the University of South Africa, BBusSc in Actuarial Science from the University of Cape Town and a Masters in Mathematical Finance from the University of Cape Town. He is a recipient of the Google Ph.D. fellowship in machine learning which supports his current Ph.D. research at the University of Johannesburg. His research interests include Bayesian machine learning and Markov Chain Monte Carlo methods.

RENDANI MBUVHA was born in Venda (Limpopo, South Africa) is a lecturer in Statistics and Actuarial Science at the University of Witwatersrand. He is a qualified Actuary and a holder of the Chartered Enterprise Risk Actuary designation. He holds a BSc with Honours in Actuarial Science and Statistics from the University of Cape Town and an MSc in Machine Learning from KTH, Royal Institute of Technology in Sweden. He is a recipient of the Google Ph.D. fellowship in machine learning which supports his current Ph.D. research at the University of Johannesburg.

TSHILIDZI 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 Columbia Business School, National University of Singapore, GIBS University of Pretoria, Harvard Business School, and University of South Africa. Some of his accomplishments include being a fellow of Cambridge Commonwealth Trust (1997), CSIR (2005), South African Academy of Engineering (2007), TWAS, The World Academy of Science (2010), African Academy of Science (2013), South African Institute of Electrical Engineers (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(2009). 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. 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. He is the current Vice-Chancellor and Principal of the University of Johannesburg.