Application of Markov Chain Monte Carlo Algorithms in Solving Feynman Path Integrals

R. Smith; supervised by Dr. C. Zambon
L3 Computing Project (Group C1)
Submitted: March 11, 2022, Dates of investigation: 12/10/2021 - 22/02/2022

Richard Feynman in the 1940s reformulated quantum mechanics in terms of Feynman path integrals. Markov Chain Monte Carlo (MCMC) algorithms provide an efficient way to solve Feynman path integrals numerically. In this paper we visualise and discuss the action of five MCMC algorithms. Then we use the random walk Metropolis (RWM) algorithm and the Hamiltonian Monte Carlo (HMC) algorithm to generate the probability distribution of a particle in a one dimensional harmonic potential well. The RWM performs better than the HMC algorithm. We see reasonable agreement with the Schrödinger solution in the RWM case despite systematic error. The advantages of HMC are minimised due to the reduction in the exploration period by initialising the path at classical solution. We conclude with future areas for research.

I. INTRODUCTION

In 1942 for his PhD thesis titled "The Principle of Least Action in Quantum Mechanics" Richard Feynman introduced the concept of path integrals to reformulate conventional quantum mechanics in terms of the action principle [1]. Path integrals are important in understanding quantum field theory and allow us to solve problems in quantum chromodynamics (QCD) (for example the forces in multi-hadron interactions) which are inaccessible by other methodologies [2].

In this work we explore the use of Markov Chain Monte Carlo algorithms in order to solve these integrals numerically. We start by investigating five different sampling techniques on a simplified problem. Then we compare the Uniform Random Walk Metropolis method to the Hamiltonian Monte Carlo method for use in calculating the ground state probability distribution of a one dimensional harmonic potential well. We conclude (in the case that we know the classical solution) that the random walk method gives a more accurate result. Not considering a small enough time step in our calculation causes large systematic error in our final result. Finally we conclude with areas for further research.

II. THEORY

Path integrals are a generalisation of the principle of least action in classical mechanics where we consider the contribution from all possible paths in the final result. The probability amplitude of propagating from one position eigenstate to another is related to the path integral formulation via Eq.(1), where x_f, x_i are the position vectors at the final and initial states, t_f, t_i are the final and initial times, $\hat{H}$ is the Hamiltonian operator, $\int Dx(t)$ represents the integral over all possible paths and $\tilde{S}[x]$ represents the action across each path considered.

$$\langle x_f | \, e^{-\hat{H}(t_f - t_i)} \, | x_i \rangle = \int Dx(t) e^{i\tilde{S}[x]} \tag{1}$$

The 'numerical sign problem' is encountered due to the highly oscillatory exponential in the RHS in which large numbers of evaluations are needed to observe meaningful results [3]. To prevent this we perform a Wick rotation in which we transform our time variable as $t \longrightarrow -i\tau$ where τ is our new time variable. This changes the action $\tilde{S}[x]$ to the total energy across the path $S[x]$ shown in Eq.(2) [4]. This is standard practice when evaluating these integrals [4].

$$\int Dx(t) e^{i\tilde{S}[x]} \xrightarrow{Wick} \int Dx(t) e^{-S[x]} \tag{2}$$

The probability distribution for a particle in a one dimensional harmonic potential well is found by fixing the start and end positions and letting $T = t_f - t_i$ go to infinity shown in Eq.(3) with x the position, E_0 the ground state energy and the far right term being the probability distribution [4].

$$\lim_{T \to \infty} \langle x | \, e^{-\hat{H}T} \, | x \rangle \propto | \, \langle x | E_0 \rangle \, |^2 = |\Psi(x)|^2 \tag{3}$$

To make this problem numerically solvable we make the following approximations. We discretise time into small time steps, we consider large values of T and a large but finite number of paths. Using these assumptions Eq.(3) becomes Eq.(4) where Δt is the time step and $S[x]$ is now the discretised total energy summed across each path.

$$|\Psi(x)|^2 \propto \sum_{paths} e^{-\Delta t S[x]} \tag{4}$$

For example over nine seconds we discretise time to one second intervals producing ten lattice nodes that can vary in position. Fig.1 shows an example of paths that can be generated on this space.

The probability distribution function can be considered proportional to the average position across all of the paths [4]. If we sample paths in proportion to the weighting term shown in Eq.(5) where $P[x]$ is the probability of selecting the path, then we can plot a histogram of how many points in every path lie in a certain range of x values. This histogram will be proportional to the probability distribution [4].

$$P[x] \propto e^{-\Delta t S[x]} \tag{5}$$

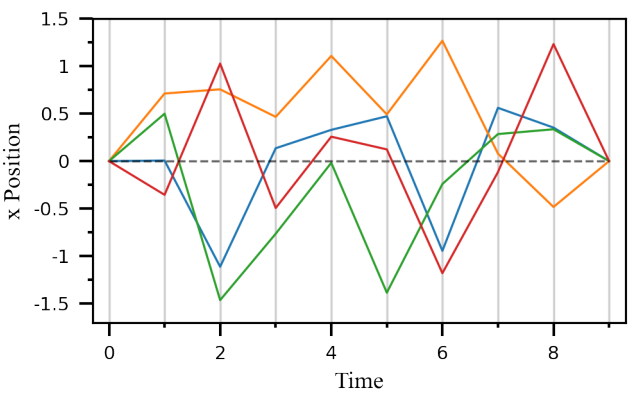

FIG. 1: Propagation of four possible paths over ten lattice points

The classical path of least action is a stationary point so looking at the RHS of Eq.(1) we see that paths close to this have slowly varying phase factors. These paths coherently combine and form a large contribution to the final result. Paths away from the classical path have rapidly varying phase, meaning their contribution will cancel out on average. Eq.(5) is related to this idea as paths near the least action have low energies and so will be selected more frequently and therefore have greater contribution to the probability distribution.

III. NUMERICAL METHODS

Sampling according to a distribution is a frequent problem in Bayesian statistics. When reading literature it is common that our weighting term be called the target distribution or posterior [5].

Markov Chain Monte Carlo (MCMC) algorithms are efficient ways of sampling a distribution. Here Monte Carlo indicates that the procedure uses random numbers. A Markov chain is a sequence of distributions, with the next distribution only depending on the current distribution [5]. This is important in our case as the next path can use the information of the current path to move towards the area of interest.

A stationary state of a Markov chain is where the next distribution and all future distributions are identical to the current one. We wish to generate a Markov chain such that the stationary state is the weighting term of Eq.(5). The Metropolis algorithm is an implementation of this procedure, a complete description of this is found in B. Puza *'Bayesian Method for Statistical Analysis'* (2015) [5].

The flow chart for this procedure is shown in Fig.2, where f(x) is our weighting function, x_i and x_{i+1} are the current and next path and η is a randomly generated number between 0 and 1. The basic idea is as follows: propose a new sample based off the current point. If this sample increases the value of the weighting then move to this point. If not then there is a probability that we reject this point. This holds for symmetric proposal functions like uniform or Gaussian (discussed later). For asymmetrical proposals there is a slight modification of the acceptance criteria [5].

MCMC algorithms are efficient at exploring the target distribution however the whole space is usually not explored in a sparse and multi-modal distribution as the algorithm will get stuck at one of the peaks.

From the Markov property each new path generated will be correlated with the previous path, correlation between time separated points is called autocorrelation [5].

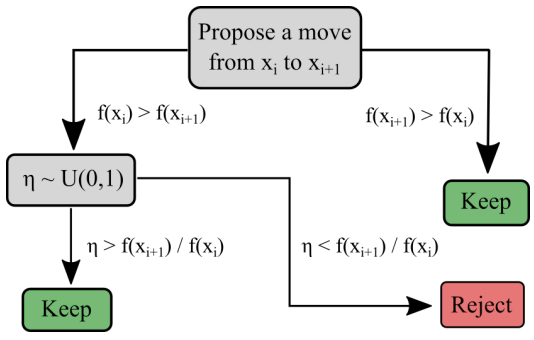

FIG. 2: Flow chart describing one iteration of the Metropolis procedure with weighting function f, current path x_i and η being a random number between zero and one

Autocorrelation affects error calculations, convergence and goodness of fit calculations. For results to be reliable autocorrelation must be reduced [5]. A burn in period is required as initially we are not sampling in the stationary state. The burn in period is the number of path discarded before reaching convergence.

A. Random Walk Metropolis

In the random walk Metropolis algorithm (RWM or URWM) the proposal is a uniform distribution around the current point in a limited range. To visualise the evolution of the paths, we use a simplified target space equivalent to paths with only two nodes (this is called the test potential). In all graphs showing the 2D contour plot the weighting term being used is an elliptical Gaussian. The colour scheme of the contour is such that the white area outside the final ellipse is close to zero not close to one as the colour bar would imply, this is done for clarity.

We can see the time evolution of the RWM algorithm at various starting points for 1000 time steps in Fig.3. At 10 steps all paths are sampling in the sparse areas of the target distribution and are in the exploration phase, at 100 steps all but the red path have reached the areas of interest and after 1000 steps they have all reached the peak. This emphasises the need for a burn in period as during the exploration phase paths sampled are not representative of the whole target distribution.

To reduce autocorrelation we introduce a reducing factor N_{corr} such that we only sample one in every N_{corr} many paths. N_{corr} and the burn in period are found using trace plots and autocorrelation plots. Fig.4 shows these plots for Node 1 for the five paths of Fig.3.

We visually check convergence is reached when every path varies around a fixed position (in our case zero). The same plots are also be considered for Node 2 and once all nodes reach convergence the paths have converged.

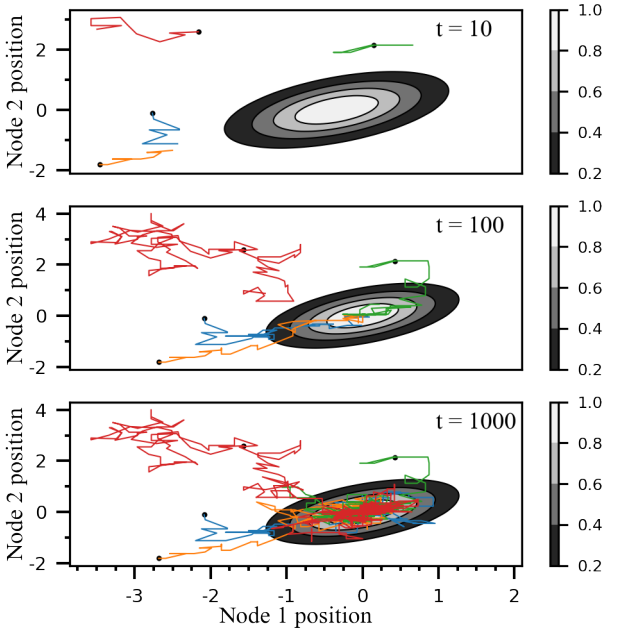

FIG. 3: Evolution of the random walk Metropolis algorithm for 4 different initial positions with a 2D elliptical Gaussian target distribution over 1000 time steps

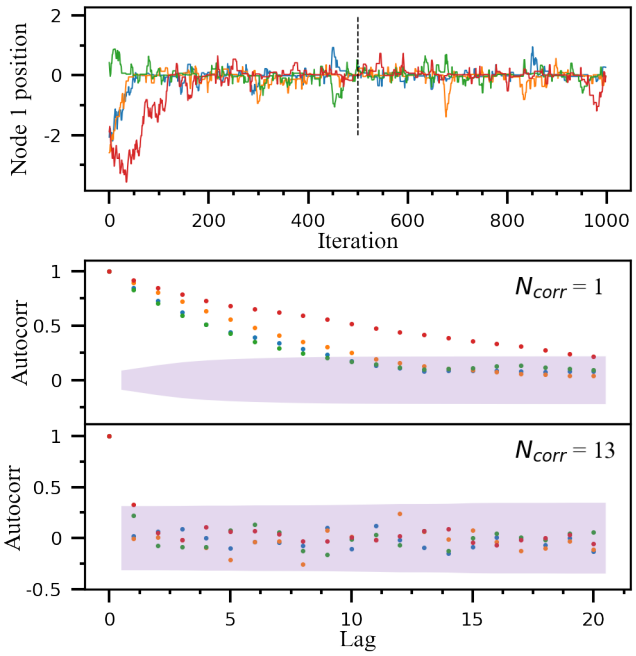

FIG. 4: A trace plot with the dashed line showing convergence point and autocorrelation plots with 5% confidence interval highlighted for a node in the random walk Metropolis algorithm

In this paper the burn in period is calculated before N_{corr}. The Gelman-Rubin diagnostic uses multiple chains (paths starting at different points) to derive a quantitative measure of the convergence, when this value is 1.00 we can consider the paths converged [6]. This point is shown as the black dotted line in Fig.4.

Lag is the delay between time series data. Visual inspection of lag plots identify if we have autocorrelation as shown in Fig.4. The purple shading shows the 5% confidence interval of no autocorrelation. N_{corr} is increased until lags greater than 1 are in this region removing the Markovian effect from the data [5]. High N_{corr} numbers are inefficient as more iterations of the algorithms are needed to obtain paths that are used.

The Gaussian random walk Metropolis (G-RWM) replaces the uniform distribution with a Gaussian distribution centred at the previous point. The evolution of this and all algorithms considered in this paper are shown in Fig.8.

The acceptance rate is the number of paths accepted divided by the number of paths proposed. The ideal acceptance rate for a 2D Gaussian target distribution is approximately 0.5, decreasing to approximately 0.3 for a high dimensional Gaussian target distribution [7].

In both cases we have a single tuning parameter σ which controls either the uniform range or the Gaussian variance. If σ is too small, the acceptance rate will be high and successive samples will explore the target distribution space slowly. If σ is too large, the acceptance rate will be very low because the proposals are likely to land in sparser regions. In either case, the convergence will be slow.

B. Slice

Slice sampling is a way of sampling from the target distribution without using the Metropolis procedure [8]. The procedure is given in Fig.5. From the current point we sample vertically between zero and function evaluated at this point. Then we expand horizontally outwards in width W from this position until we lie outside the distribution. The new point is then found by uniformly sampling from this slice. If the sample lies outside the curve we move the boundary to this point. In Fig.5 we form a 1D slice but in general we perform hyperrectangle expansion in the dimension of our target distribution.

This algorithm has one tuning parameter W. The disadvantage of slice sampling is when applying to an asymptotic target distributions (which we are in this paper) we can sample from very sparse regions causing numerical rounding errors leading to divergence.

C. Langevin

Langevin sampling is a modification of the random walk process which makes use of the structure of the target distribution space [9]. The evolution is given by Eq.(6)

$$x_{i+1} = x_i + \tau \nabla log f(x_i) + \sqrt{2\tau}\mathcal{N}(\mu, \sigma^2) \qquad (6)$$

were τ is the tuning parameter and $\mathcal{N}(\mu, \sigma^2)$ is a Gaussian noise term. The noise term introduces random walk behavior and the gradient of the negative log of the weighting function acts as a drift term pushing the path into the areas of interest.

The optimal acceptance ratio for a Gaussian proposal is 0.574 [9]. This analogous to a particle in Brownian motion where we have a drift velocity and random fluctuations. Interestingly the Langevin equations that governs the behavior of Brownian motion can be reformulated in terms of path integrals [10].

D. Hamiltonian Monte Carlo

Developed for lattice QCD calculations Hamiltonian Monte Carlo (HMC) aims to reduce the random walk behavior of the previous methods. Each path follows a trajectory through the target distribution space based of Hamiltonian dynamics, with each path conserving energy as it moves. The exact procedure is described in original paper S. Dune et al. *'Hybrid Monte Carlo'* (1987) [10]. There are two tuning parameters a path length and a step size.

An analogous situation describing how this works is a marble in a bowl. The marble's x, y, coordinates make up nodes of the path and the bowl is the negative weighting function (the mound turns into a bowl). The next path is found by flicking the marble in a random direction and tracking its motion for a set distance around the bowl.

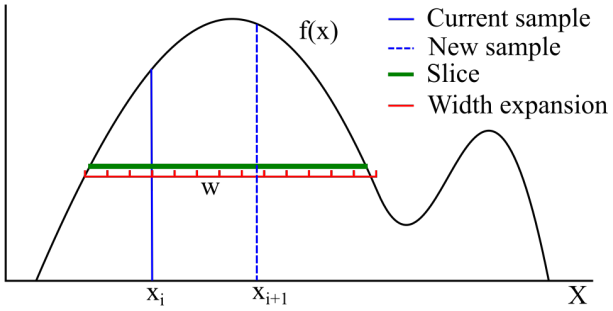

FIG. 5: One iteration of the slice sampling algorithm on a 1D weighting function

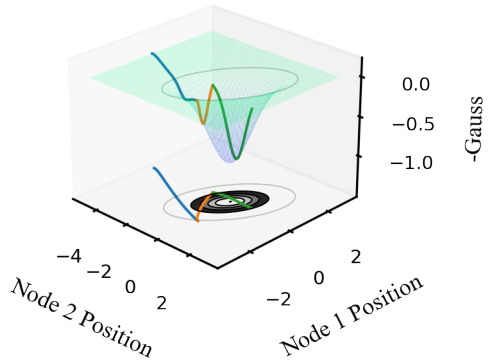

FIG. 6: Initial three trajectories of the path in a 3D representation of the Hamiltonian Monte Carlo algorithm on an elliptical Gaussian

The first three trajectories are shown in Fig.6 in 3D and the first six trajectories are shown in 2D in Fig.7(a). The path position is stored before introducing a new flick into the system. Fig.7(b) shows normalised kinetic energy (related to conjugate momenta) and potential energy (related to weighting value). This is done for clarity as each flick has different total energy. Discontinuities are seen every 100 iterations as we introduce a new flick, this number is a tuning parameter. Compare the energy plots in Fig.7(b) to Fig.6. When we lose height on the 3D plot, we gain kinetic energy such that the total energy of the system is conserved.

Initially energy conservation is violated. When traveling over large distances the integrator used to solve Hamilton's equations cannot sample enough points, leading to compounding errors. The optimal acceptance ratio is 0.65 for a multivariate Gaussian distributions [12].

IV. RESULTS AND DISCUSSION IN MODEL TESTING

Evolution of all algorithms are shown in Fig.8. Convergence and N_{corr} were evaluated as in the RWM case. Each algorithm was tested with 10 chains, with convergence determined by the Gelman-Rubin diagnostic along side visual inspection of all trace plots. N_{corr} was also determined by visual inspection of the autocorrelation plots for all nodes. Parameters of each algorithm were tuned such that the acceptance ratio was the theoretical optimum shown in Tab.1. In this paper, values of N_{corr} and burn in period are upper bounds of the true value. As this is the case, they are shown without errors. These numbers give no autocorrelation and converge in all trial tests.

The RWM has the largest burn in period and N_{corr} due to its limited travel distance. It is the most inefficient algorithm generating few used paths.

TABLE I: Markov chain statistics for the two node testing case for the five MCMC algorithms

Algorithm	Burn-in	N_{corr}	Acceptance
RWM	600	200	0.50
G-RWM	500	120	0.50
Slice	200	3	-
Langevin	320	40	0.57
HMC	40	2	0.65

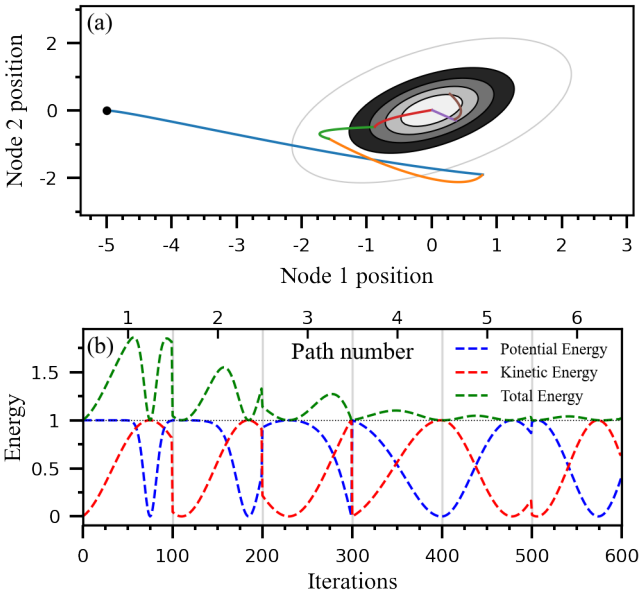

FIG. 7: (a) Initial six trajectories of the Hamiltonian Monte Carlo algorithm on the 2D projection of Fig.6. (b) Evolution of normalised kinetic, potential and total energy for the trajectories in (a)

The Gaussian variant performs slightly better, requiring less burn in and N_{corr} as it can sometimes take large jumps.

Slice sampling is a powerful tool when it converges. We can see from Fig.8 it reaches the area of interest in very few steps. The acceptance cell in Tab.1 is blank as this algorithm does not use the Metropolis procedure. As we increase the dimension of the asymptotic weighting term, the divergence becomes more likely.

Langevin is an improvement on the random walk algorithms. The simple gradient modification greatly increases convergence rate as seen in Fig.8. The drift term increases the number of samples accepted, which increases autocorrelation, giving the large N_{corr} value.

HMC has increased efficiency by reducing the random walk effect. Autocorrelation and burn in period are the smallest. However it is the most computationally expensive and uses two tuning parameters creating more trial runs. Now we consider the simple inefficient algorithm (RWM) and complex efficient algorithm (HMC) on a 1D harmonic potential well.

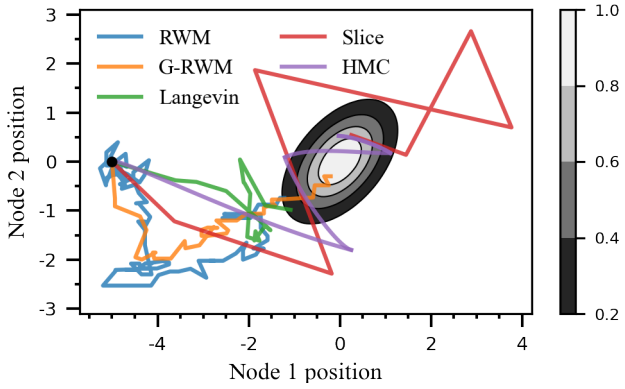

FIG. 8: Evolution of five different MCMC algorithms on an elliptical Gaussian target distribution

V. RESULTS FOR APPLIED CASE

Calculations were performed using 10 lattice points equivalent to 9 time steps. Initial position of all paths were set to zero (the classical solution). The simulation was set up to gather 10,000 paths from each algorithm. Trial runs were used to set tuning parameters, determine burn in period, find the N_{corr} value and to check convergence. A comparison of the RWM and HMC algorithms are shown for a 1D harmonic potential well in Fig.9(a). This plot is formed from connecting the mid points of 40 histogram bins. This was found to produce a smooth curves for both algorithms, the analytical Schrödinger solution is also plotted.

The trace plots for the 5th node are shown in Fig.9(b) along side the trace plot histograms. The trace plots are considered after the burn in period. A visual inspection of trace plots for all nodes was used to assess convergence. The autocorrelation plots with the 5% confidence interval highlighted is shown in Fig.9(c).

The errors at each lattice point are calculated using the bootstrap method (see in Error Appendix) with 100 new data sets generated. The goodness of fit statistics: reduced chi squared, chi squared p-value and the Durbin Watson statistic are shown in Tab.2. The Durbin Watson statistic is a measure of correlation within the residuals with a value of 2.00 representing randomly distributed Gaussian residuals [13]. All goodness of fit statistics have no associated error as we perform our calculation on a single long run.

Further diagnostics of such as the N_{corr} value, burn in period and generating time are given in Tab.2, these values were calculated during the trial runs whilst tuning the parameters. Generating time is given by Eq.(7) and is a measure of the relative speed of generating a used path. The standard error associated with generating time is ± 0.02 s which is calculated over the trial runs. Lower generating time implies a faster generation of the desired distribution.

A comparison of the different Markov chain diagnostics for the two different potentials are shown in Tab.3. Here the test potential is the elliptical Gaussian with two node points where we start chain at random positions. The Potential well is the 1D harmonic potential with ten node points where we start the chains at the classical solution centred at zero.

$$\text{Generating Time} = \frac{\text{Run time} \times N_{corr}}{\text{Number of iterations}} \qquad (7)$$

.

VI. DISCUSSION FOR APPLIED CASE

A visual inspection of Fig.9(a) shows that the random walk algorithm is closer to the Schrödinger solution than the Hamiltonian model. The reduced chi squared values given in Tab.2 show confirmation of this as the RWM is closer to 1 [13].

From the chi squared p-values we question the null hypothesis (that the model fits the data) for RWM and reject the null hypothesis for HMC [13]. Incorrect bootstrapping errors can cause this p-value to be inaccurate however increasing the number of bootstraps by a factor of ten does not change these statistics. This error is due to systematic or random error.

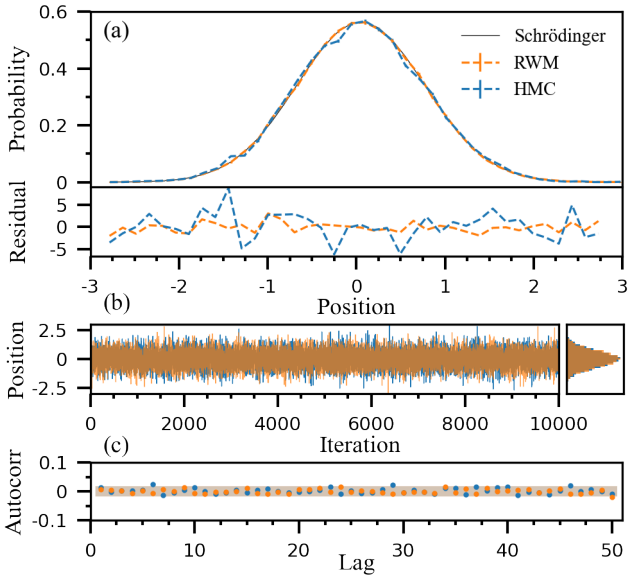

FIG. 9: (a) Probability distributions by random walk Metropolis and Hamiltonian Monte Carlo for a particle in a 1D harmonic potential well together with the analytic Schrödinger solution which is not visible due to overlap (b) Trace plot with histogram plot after burn in period for one node (c) Auto correlation plot with 5% confidence interval highlighted for one node

The residuals in Fig.9(a) in both cases look randomly distributed around 0 which is confirmed by the Durbin Watson statistic. In both cases the statistic is around 2 implying our residuals are randomly distributed in a Gaussian distribution. If this assumption is violated then the error bars are likely underestimated leading the incorrect conclusions about statistical significance [13].

Fig.9(b) is representative of all the other nodes trace plots. Here we observe the position varying around 0 with constant variance over the 10,000 iterations. The histogram in Fig.9(b) shows us that in both cases the positions are normally distributed, showing we have reached convergence. During trial runs the Gelman-Rubin statistic is used to determine N_{corr} and burn in period, these are then used when generating the large trials.

TABLE II: Goodness of fit statistics and Markov chain statistics for potential well case

Statistic	RWM	HMC
$\chi^2_{reduced}$	1.14	1.96
χ^2 p value	0.03	0.00
D-W	1.80	2.14
N_{corr}	20	4
Burn in	4000	200
Generating Time (s)	0.10	0.01

TABLE III: Comparison of the random walk Metropolis and Hamiltonian Monte Carlo Markov chain statistics between the two potentials

Trial	Test potential		Potential well	
Algorithm	RWM	HMC	RWM	HMC
N_{corr}	200	2	20	4
Burn in	600	40	4000	200

Fig.9(c) shows that to 5% significance there is no autocorrelation in our samples with their corresponding N_{corr} values. This increases confidence in our test statistics.

The resulting error is unlikely to be only random error as we have reached convergence, minimised autocorrelation and checked bootstrap errors. Systematic error is likely as we have only considered 10 node points. 10 points were considered to give reasonable computation time. Increasing the number of node points would have have increased the accuracy of the results but at the cost of higher computation time.

In the test potential and the potential well case we see that N_{corr} for HMC is less than RWM shown in Tab.3. N_{corr} increases between the test potential and potential well in the HMC case but decreases in the RWM case. This is unexpected as we expect N_{corr} to increase with the number of lattice points [4]. Different potentials are used in each case and importantly the starting path is the classical solution in the potential well case. This reduces the exploration period of RWM minimising the advantage of the HMC.

There is a much larger burn in increase in proportion to the N_{corr} when going from the test to the potential well in the RWM case. This is due in part to the decreasing optimal acceptance ratio for RWM from 0.5 to 0.3 which increases the number of samples that are rejected, inducing a larger burn in [7].

The generating time in Tab.2 show RWM is faster than HMC despite having to sample five times as many points to gather a single uncorrelated sample. This is because each loop of the HMC is very computationally expensive.

Despite HMC having many advantages over RWM in terms of convergence and autocorrelation it produces a worse fit and is slower than RWM. In the potential well case, the advantages of HMC are minimised as we are starting at the classical solution. The important lesson from this is that we should start to consider new algorithms only if our original is struggling to converge or is slow to run. For more complex potentials with more nodes and in higher dimensions as in QCD HMC becomes a more viable algorithm.

VII. CONCLUSIONS

The main goal of this paper was to investigate ways of solving the Feynman path integral for a 1D harmonic potential well. We discussed five different MCMC algorithms on a simplified problem calculating their burn in period and N_{corr} values. The random walk Metropolis algorithm was found to be the most inefficient and the Hamiltonian Monte Carlo algorithm initially designed for use in solving lattice QCD problems was found to be the most efficient.

These were tested on the harmonic potential well and it was found that the RWM performed better than the HMC. This was due to starting at the known classical solution of the harmonic potential well, eliminating the need for an exploration period, minimising the advantages of the computationally more expensive HMC.

We found systematic error in the results caused by the large time steps used, increasing number of time steps would increase computation time.

In a further investigation, the method could be improved by increasing the number of time steps considered. The simulations in this paper were performed on a Raspberry Pi Model 4B, increasing computational power would allow for larger and more accurate simulations. Techniques to increase computation speed should be investigated such as multi threading and JIT compilers in which the computationally expensive for loops of HMC are precompiled into machine code allowing for much faster iterations, this would potentially minimise the speed difference between HMC and RWM [14]. More complex potentials could be studied and the number of dimensions increased to perform simple lattice QCD calculations [4]. Further adaptive variants of the algorithms discussed could be considered. In these, parameters are tuned dynamically during the run greatly reducing the number of trial runs [15]. A powerful algorithm to consider is the Riemann Manifold Hamiltonian Monte Carlo which further exploits the curvature of the target distribution space leading to even faster convergence [16].

[1] R. P. Feynman, *'The Principle of Least Action in Quantum Mechanics'*, Laurie M. Brown (ed.), Princeton University (2005).

[2] W. Detmold and M. J. Savage, *'Method to study complex systems of mesons in lattice QCD'*, American Physical Society (2010), Phys. Rev. D 82, 014511

[3] S. Chandrasekharan and U. J. Wiese, *'Meron-Cluster Solution of Fermion Sign Problems'*, American Physical Society (1999), Phys. Rev. Lett. 83, 3116

[4] G. P. Lepage, *'Lattice QCD for Novices'*, World Scientific (2000), arXiv:hep-lat/0506036v1

[5] B. Puza, *'Bayesian Method for Statistical Analysis'*, 2015 ed., Australian National University (2015)

[6] A. Gelman and D. B. Rubin, *'Inference from Iterative Simulation Using Multiple Sequences'*, Statistical Science (1992), VOL. 7 · NO. 4

[7] G. Roberts et al., *'Weak convergence and optimal scaling of random walk Metropolis algorithms'*, Annals of Applied Probability (1997), VOL. 7 · NO. 1

[8] R. M. Neal, *'Slice Sampling'*, Annals of Statistics (2003), VOL. 31 · NO. 3

[9] G. O. Roberts and R. L. Tweedie, *'Optimal scaling of discrete approximations to Langevin diffusions'*, Royal Statistical Society Series B (1998), VOL. 60 · NO. 1

[10] A. K. Das et al., *'A path integral approach to the Langevin equation'*, International Journal of Modern Physics A (2015), VOL. 30 · NO. 07

[11] S. Duane et al., *'Hybrid Monte Carlo'*, Physics Letters B (1987), VOL. 60 · NO. 1

[12] S. Brooks et al., *'Handbook of Markov Chain Monte Carlo'*, Chapman and Hall (2014)

[13] I. G. Hughes and T. P. A. Hase,*'Measurements and their Uncertainties'*, Oxford University Press, Oxford (2010)

[14] Numba Python Compiler. Available at: (https://numba.pydata.org) [Accessed 11 February 2022]

[15] M. D. Hoffman and A. Gelman, *'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo'*, Journal of Machine Learning Research (2014), VOL. 15 · NO. 47

[16] M. Girolami and B. Calderhead, *'Riemann manifold Langevin and Hamiltonian Monte Carlo methods'*, Royal Statistical Society Statistical Methodology Series B (2011), VOL. 73 · NO. 2

VIII. ERROR APPENDIX

The errors on individual valued expressions such as generating time were calculated using the equations in [I. G. Hughes and T. P. A. Hase,*'Measurements and their Uncertainties'*, Oxford University Press, Oxford (2010)].

Errors in node position in Fig.9(a) were calculated using a bootstrapping method. A simple way of calculating mean values and standard deviation would be to rerun the algorithms many times, however this is computationally expensive. Bootstrapping provides a faster way of calculating these errors.

The process is described in Fig.10. We start by considering the original data set of paths generated from a long run of one of the algorithms. Next we sample with replacement from the original data set to generate new data sets. We generate new data sets of the same length as the original for significantly less computation time. From these new data sets we calculate mean values of each point. From these, we can calculate the average of the mean value at each point and a standard deviation associated with each point. Standard error is not used in this case as no new information is gained about the system when we resample. Otherwise we could get arbitrarily small errors just by increasing the bootstrap number.

One hundred new data sets were used to obtain the mean and standard deviation. Experimentation showed that increasing the number of new data sets did not significantly change the mean values or standard deviations obtained. This agrees with the literature [D. L. Goodhue et al., *'Does PLS have advantages for small sample size or non-normal data?'* MIS Quarterly (2012), VOL. 36 · NO. 3]

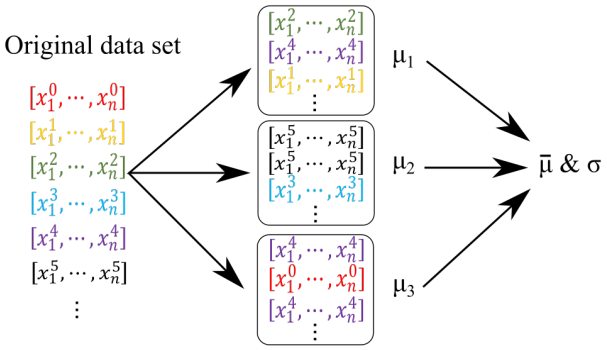

FIG. 10: Explanation of bootstrapping where μ is the list of means for each point, $\bar{\mu}$ is the mean between data sets at each point and σ being the associated standard deviation at each point