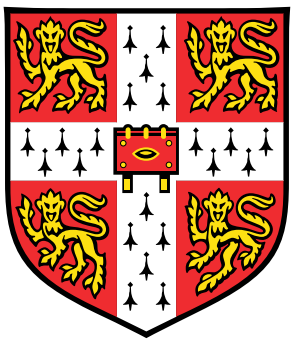
Optimal Importance Sampling in Quantum Monte Carlo for Lattice Models



Blaž Stojanovič

Supervisor: Prof. A. Lamacraft

Department of Physics

This dissertation is submitted for the degree of
Master of Philosophy in Scientific Computing

St. John's College

August 2021

To my family.

Declaration

This dissertation is substantially my own work and conforms to the University of Cambridge's guidelines on plagiarism. Where reference has been made to other research this is acknowledged in the text and bibliography. This dissertation contains fewer than 15,000 words including appendices, figure legends, and tables.

Blaž Stojanovič
August 2021

Acknowledgements

First of all, I would like to thank Zala for her love, support, and for helping me get through the roughest parts of my MPhil journey. I would also like to thank my family: grandparents for their words of wisdom, parents for showing nothing but support for my ambitions, and brother for being the most reliable friend.

Secondly, I would like to acknowledge everyone that helped make studying at Cambridge a reality for me. My primary and secondary school teachers who inspired me, as well as the faculty at FMF for providing the foundation of my education. I am most grateful to dr. Kosec for allowing me to work with him and for everything I learned from him and dr. Slak at IJS. I want to thank dr. Kosec and prof. Prosen, for believing in me and writing my recommendation letters.

Finally, I thank prof. Austen Lamacraft, not just for his supervision, guidance, and advice, but for all the coffees and discussions of Physics at St. Johns, which were exactly how I imagined Cambridge would be.

Abstract

The Feynman-Kac formula provides a bridge between stochastic processes and partial differential equations. It can be used to formulate the ground state problem of a quantum many-body system in terms of evaluating an expectation with respect to the Feynman-Kac path measure. In this work we make use of connections between optimal control, quantum mechanics and stochastic processes to propose a method that obtains the Feynman-Kac measure for stoquastic Hamiltonians. The trajectory distribution of a quantum lattice model is represented with transition rates of a continuous time Markov chain, which are learned by minimising the log RN loss with fixed endpoints. Finding the optimal rates is equivalent to knowing the Feynman-Kac path measure and permits optimal importance sampling from the ground state of the model. The method is implemented and demonstrated on the transverse field Ising model.

Word count: 14,998

Table of contents

List of figures

Nomenclature

Other Symbols

Cov	Covariance
$\mathbb{E}$	Expectation
"$\mathcal{F}$"	σ-field (algebra)
$\mathbb{F}$	Filtration
$\mathfrak{g}$	Group element
$\mathfrak{G}$	Group
D_{KL}	Kullback-Liebler divergence
$\mathbb{P}$	Measure
$\mathcal{N}$	The Gaussian distribution
$\{X\}$	Random variable
Γ	Rate matrix
P	Transition matrix
$\mathbb{R}$	The set of real numbers
S	State space of a Markov process
$\{X_t\}$	Stochastic process
Var	Variance
W_t	Wiener process, mathematical Brownian motion

Acronyms / Abbreviations

cdf	Cumulative density function
e.g.	Exempli gratia ("for the sake of an example")
i.e.	Id est ("it is")
i.i.d	Independent and identically distributed
mdp	Markov Decision Process
p.b.c	Periodic boundary condition

pdf	Probability density function
s.p.	Stochastic process
s.t.	Such that
w.r.t	With respect to
AD	Automatic Differentiation
CNN	Convolutional Neural Network
pCNN	Periodic Convolutional Neural Network
CTMC	Continuous time Markov Chain
DFT	Density Functional Theory
DL	Deep Learning
DMC	Diffusion Quantum Monte Carlo
DMFT	Dynamical Mean Field Theory
DMRG	Density Matrix Renormalization group
DNN	Deep Neural Network
DTMC	Discrete time Markov Chain
FK	Feynman-Kac
FP	Fokker-Planck
G-CNN	Group Equivariant Convolutional Neural Network
GFMC	Green's function Quantum Monte Carlo
KL	Kullback Liebler
MC	Monte Carlo
MCMC	Markov Chain Monte Carlo
ML	Machine Learning
MLP	Multilayer Perceptron
NN	Neural Network
PDE	Partial Differential Equation
QMC	Quantum Monte Carlo
QM	Quantum Mechanics
RN	Radon-Nikodym
SDE	Stochastic Differential Equations
SEP	Symmetric Exclusion Process
TFIM	Transverse Field Ising Model
VMC	Variational Quantum Monte Carlo

Chapter 1

Introduction

We are surrounded by intractable problems, and that is even when we consider simplified models of reality. The transverse field Ising model is an example of a simple lattice model that displays very rich physics. Its basic constituents are not particles in continuous space, but spins confined to a grid that only interact with their immediate neighbours and the transverse field. The dynamics of the Ising model can be described by a single vector, but its dimension grows as 2^N with the number N of particles, quickly putting us out of our depth. This exponential explosion can be avoided by using stochastic methods, where our goal is

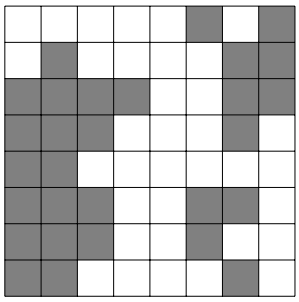


Fig. 1.1 **A configuration of the Ising model.**

to efficiently sample from the probability distribution of spin configurations, which allows us to calculate properties of the system. Some quantum lattice models lend themselves more naturally to stochastic methods than others. Stoquastic Hamiltonians belong right on the border between statistical and quantum mechanics, and this is precisely where we will operate.

The main objective of this thesis is to use concepts from probabilistic machine learning, optimal control, and stochastic processes to devise a method that will be able to learn the Feynman-Kac trajectory distribution in quantum lattice models. Such a method would provide one with the ability to perform optimal importance sampling if a perfect representa-

tion of the distribution was found, and a significant reduction in variance for near-optimal representations.

1.1 Thesis Structure

The thesis is structured as follows, Chapter 2 briefly introduces the quantum many-body problem before focusing on numerical solution approaches to it. In addition to standard methods, recent applications of Machine Learning are highlighted. In the first half of Chapter 3 the focus is turned towards mathematical fundamentals of stochastic processes, which underpin most other parts of the thesis. The rest of the chapter is dedicated to introducing the Feynman-Kac formula and the derivation of the log RN loss. The derivation includes necessary background in optimal control and an analogous derivation in continuous state space. Chapter 4 describes the methodology and implementation of the method, and Chapter 5 describes training and sampling experiments conducted. Chapter 6 contains concluding remarks.

Chapter 2

On the quantum many-body problem

This chapter discusses the quantum many-body problem and numerical approaches to its solution. We begin by introducing the Schrödinger equation and Feynman path integral formulations of quantum mechanics, before briefly discussing lattice models and their significance. We then turn our attention towards solution procedures, providing review of popular methods with emphasis on **Monte Carlo** (MC) approaches. Finally, we highlight recent usage of machine learning methods in this field.

2.1 Schrödinger equation and Feynman path integral

The dynamics of a quantum mechanical system are described with the Schrödinger equation

$$i\hbar \frac{\mathrm{d}}{\mathrm{d}t}|\Psi(t)\rangle = \hat{H}|\Psi(t)\rangle, \tag{2.1}$$

a linear **partial differential equation** (PDE). The state of the quantum system $|\Psi\rangle$ is a vector in a separable Hilbert space $\mathcal{H}$, and the square of its absolute value, e.g. $|\Psi(x,t)|^2$, at each point is interpreted as a **probability density function** (pdf). The Hamiltonian operator is the sum of kinetic and potential energies $\hat{H} = \hat{T} + \hat{V}$ of the system, throughout this thesis we will be interested only in the ground state of the system. Instead of using the time-dependent formulation in eq. (2.1), we use the *stationary* Schrödinger equation

$$\hat{\mathrm{H}}|\Psi\rangle = E|\Psi\rangle, \tag{2.2}$$

an eigenvalue equation, with the lowest energy E_0 corresponding to the ground state $|\Psi_0\rangle$. From this point onward, we use Hartree atomic units $m_e = e = \hbar = a_0 = \mathrm{E}_h = k_e = 1$.

Alternatively to the Schrödinger equation one can use an integral Green's function representation to express the wavefunction Ψ at some future time t given initial condition $\Psi(x', t')$ as

$$\Psi(x,t) = \int \mathcal{K}\left(x,t;x',t'\right)\Psi\left(x',t'\right)\mathrm{d}x'. \tag{2.3}$$

The *propagator* $\mathcal{K}\left(x,t;x',t'\right)$ is the kernel of the Schrödinger equation

$$\left(i\frac{\partial}{\partial t} - H_x\right)\mathcal{K}\left(x,t;x',t'\right) = i\delta\left(x-x'\right)\delta\left(t-t'\right). \tag{2.4}$$

It is related to the fundamental solution or Green's function as

$$\mathcal{G}\left(x,t;x',t'\right) = \frac{1}{i}\Theta\left(t-t'\right)\mathcal{K}\left(x,t;x',t'\right), \tag{2.5}$$

where Θ is the Heaviside function and δ is the Dirac delta. The same propagator can also be expressed using the Feynman path integral

$$\mathcal{K}\left(x,t;x',t'\right) = \int_{\substack{q(t)=x\\ q(t')=x'}} \exp\left(i\int_{t'}^{t}\mathcal{L}(q,\dot{q},t)dt\right)\mathcal{D}[q(t)], \tag{2.6}$$

where $\mathcal{L}$ is the classical Lagrangian function of the system, and the path integral is over all paths that satisfy the endpoint conditions $q(t) = x, q(t') = x'$. This formulation of quantum mechanics (QM) is closely related to the approaches we develop in the later chapters.

2.2 Lattice models

Lattice quantum spin systems, or lattice models for short, play a very important role in the modern condensed matter physics landscape. The study of lattice systems in one and two dimensions has been especially productive, as many exact solutions containing non-trivial physics have been found. Concretely, spin systems that exhibit phase transitions provide insight into thermal and quantum fluctuations driving such transitions [57].

2.2.1 Examples of lattice models

Lattice models have long been used to study magnetism, and perhaps the most famous model is the Heisenberg model [34],

$$\hat{H} = -\sum_{ij}\sum_{\alpha\beta} J_{ij}^{\alpha\beta} S_i^{\alpha} S_j^{\beta} + \sum \vec{B}\cdot\vec{S}_i, \tag{2.7}$$

where $\vec{B}$ is the external magnetic field, i,j are lattice sites, and the spin-spin interaction between spin-vector components $\alpha,\beta \in \{x,y,z\}$ at sites i,j is given by $J_{ij}^{\alpha\beta}$. The general problem in eq. (2.7) cannot be solved, but some special cases in low dimensions have analytical solutions. One such case is the **transverse field Ising model** (TFIM) [23], which is obtained by setting

$$J_{ij}^{\alpha\beta} = J\delta_{\langle i,j\rangle}\delta_{\alpha,\beta}\delta_{\alpha,z}, \quad \vec{B} = -B\vec{e}_x, \quad \text{and} \quad S = \frac{1}{2}, \tag{2.8}$$

thus

$$\hat{H}_{\text{Ising}} = -J\sum_{\langle i,j\rangle}\hat{\sigma}_i^z\hat{\sigma}_j^z - h\sum_i \sigma_i^x, \tag{2.9}$$

where $\langle i,j\rangle$ denotes a sum over nearest neighbours. A spin-$\frac{1}{2}$ system is represented in terms of Pauli matrices $\hat{\sigma}_i^\alpha$ acting on spin at site i

$$\hat{\sigma}_i^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}_i \quad \hat{\sigma}_i^y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}_i \quad \hat{\sigma}_i^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}_i. \tag{2.10}$$

The Hilbert space of a single spin is two-dimensional. We construct its basis in terms of $\hat{\sigma}^z$ eigenvalues $\{|\uparrow\rangle, |\downarrow\rangle\}$, where $\hat{\sigma}^z|\uparrow\rangle = |\uparrow\rangle$ and $\hat{\sigma}^z|\downarrow\rangle = -|\downarrow\rangle$. Pauli matrices at different sites commute

$$\left[\hat{\sigma}_i^\alpha, \hat{\sigma}_{i'}^{\alpha'}\right] = 0 \quad \text{for} \quad i' \neq i, \tag{2.11}$$

and follow

$$\left[\hat{\sigma}_i^\alpha, \hat{\sigma}_k^\alpha\right] = 2i\varepsilon_{jk\ell}\hat{\sigma}_\ell^\alpha \tag{2.12}$$

on the same site i. We define raising and lowering operators $\hat{\sigma}_i^\pm$ as

$$\hat{\sigma}_i^\pm = \left(\hat{\sigma}_i^x \pm i\hat{\sigma}_i^y\right)/2, \quad \text{where} \quad \hat{\sigma}^+|\downarrow\rangle = |\uparrow\rangle \text{ and } \hat{\sigma}^-|\uparrow\rangle = |\downarrow\rangle. \tag{2.13}$$

The whole Hamiltonian then acts on the tensor product space $\left(\mathbb{C}^2\right)^{\otimes N}$ with dimension 2^N. TFIM has been exhaustively studied [74, 75] and the one-dimensional case was solved by Pfeuty [61]. The solution uses Jordan-Wigner transformation of spin operators to spinless fermions [49, 53]. The one dimensional case has two distinct states, the ordered phase for $|h| < J$ is either ferromagnetically $J > 0$ or anti-ferromagnetically $J < 0$ ordered. Since this state breaks spin-flip symmetry it is doubly degenerate, for $h = 0$ and $J > 0$ we have either $|\uparrow\uparrow\cdots\uparrow\rangle$ or $|\downarrow\downarrow\cdots\downarrow\rangle$. The system undergoes a quantum state transition at $|h| = J$ into a disordered phase for $|h| > J$, in the limit $h \longrightarrow \infty$ we have $|\longrightarrow\longrightarrow\cdots\longrightarrow\rangle$.

The **XY model** [49] is another spin model that can be solved using the Jordan-Wigner transformation, its Hamiltonian is

$$\hat{H}_{\mathrm{XY}} = -\frac{1}{2}\sum_{\langle i,j\rangle} \hat{\sigma}_i^x \hat{\sigma}_j^x + \hat{\sigma}_i^y \hat{\sigma}_j^y. \tag{2.14}$$

More generally, we can write a Hamiltonian for a chain of spins by setting $J_{ij}^{\alpha\beta} = J_\alpha \delta_{\alpha\beta}\delta_{i,j+1}$ and $\vec{B} = -h\vec{e}_z$

$$\hat{H}_{\mathrm{XYZ}} = -\frac{1}{2}\left[\sum_{j=1}^{N} J_x \hat{\sigma}_j^x \hat{\sigma}_{j+1}^x + J_y \hat{\sigma}_j^y \hat{\sigma}_{j+1}^y + J_z \hat{\sigma}_j^z \hat{\sigma}_{j+1}^z + h\sigma_j^z\right], \tag{2.15}$$

this is the **XYZ Heisenberg chain**. It is another well studied model with solutions for various settings of the coupling constants J_α. The isotropic ferromagnetic XXX chain ($J_x = J_y = J_z = J < 0$) was solved by Bethe with his famous ansatz [9]. The ground state [38] and excitations [24] of the isotropic antiferromagnetic XXX chain ($J_x = J_y = J_z = J < 0$) are known, and so are the solutions to the XXZ chain ($J_x = J_y = J \neq J_z$) [88–90] and the fully anisotropic XYZ chain ($J_x \neq J_y \neq J_z$) [7].

Models discussed so far only include nearest-neighbour interactions,. A model which includes next nearest-neighbour interactions is the **J1-J2 model**

$$\hat{H}_{J_1\text{-}J_2} = J_1 \sum_{\langle i,j\rangle} \hat{\sigma}_i \hat{\sigma}_j + J_2 \sum_{\langle\langle i,j\rangle\rangle} \hat{\sigma}_i \hat{\sigma}_j. \tag{2.16}$$

We now turn our attention to general approaches to the quantum many body problem, we will return to models again when discussing stoquastic Hamiltonians in sections 3.2.2 and 3.2.3.

2.3 Approaches to the quantum many-body problem

The quantum many-body problem, which amounts to solving the $3N$-dimensional Schrödinger equation, underpins a large part of quantum chemistry, condensed matter physics and materials science. The problem is notoriously hard to solve and very few systems with analytical solutions exist, most of them constrained in some artificial way such that they lend themselves to mathematical analysis. Great efforts have been made in the nearly 100 years since the conception of the Schrödinger equation, in developing both analytical and numerical techniques to produce insights into quantum systems. Perhaps the most impactful was the development of various approximate methods that solve the many-body problem with limited computational resources. While there is ongoing work on quantum

simulators and computers that could greatly speed-up solving quantum problems [28, 18], we here discuss methods one can use with a classical computer. The commonality of all mentioned methods is that they try to tame the exponential growth of the underlying Hilbert space w.r.t the system size, but they differ in how they achieve this.

Hartree-Fock

One of the most common approaches to the many-body problem is to replace the original interacting many-body problem with a set of independent-particle problems with effective potential. **Hartree-Fock** (HF) approaches solve an auxiliary system of independent electrons in a self-consistent field and assume that the wave function (for fermions) can be represented as a single Slater determinant. The HF method does not include electron correlation, which makes it a good approximation only in systems where correlation contributions are small.

Post-Hartree-Fock methods

Post-HF methods, such as Coupled Cluster, Configuration interaction and Møller-Plesset theory include correlation by considering a linear combination of Slater determinants. They can be extremely accurate but come at a high computational cost.

Density Funcitonal Theory

Alternatively **Density Functional Theory** (DFT) reformulates the many-body electron problem in terms of the 3-dimensional electron density $n(\mathbf{r})$, which is found by minimising the total energy functional $E[n(\mathbf{r})]$ [36]. In practice this is done by solving the Kohn-Sham auxiliary system. DFT is in theory exact, however only if the true energy functional $E[n(\mathbf{r})]$ is known. As this is not the case, much research has been done in constructing different energy functionals with varying degrees of accuracy, starting with local functionals e.g. LSDA and continuing towards more heavily parameterised, non-local formulations. DFT provides a good trade-off between accuracy and computation time, it is used extensively for simulating large systems as linear scaling variants of DFT exist [70].

Dynamical Mean Field Theory (DMFT)

DMFT [35] is a framework that is specialised in solving strongly correlated systems. It is intuitively similar to Weiss Mean Field Theory in classical statistical physics. The main idea is to map an intractable lattice problem into an impurity model in an effective medium, a many-body local problem which can be solved with any standard approach (QMC, DFT,

exact diagonalisation, etc.). This mapping between lattice and impurity model is exact and the approximation comes from neglecting spatial fluctuations of the lattice self-energy Σ, the contribution to energy due to particle interaction with medium. DMFT assumes that Σ is a function of frequency and not momentum $\Sigma(k, \omega) = \Sigma(\omega)$, which only holds in the infinite coordination case. Time fluctuations are taken into account, i.e. the effective medium is not static in DMFT, which is an advantage over other static mean field theories.

Density Matrix Renormalization group

DMRG [85] is considered the state of the art method for solving one-dimensional lattice problems and it has been widely adopted in condensed matter physics, first used to solve the system of a spin-0 particle in a box. It is an iterative method based on the renormalization group [86], and uses matrix product states as the variational ansatz. The method has also been extended for time evolution of systems [27], and higher dimensions [83].

2.3.1 Stochastic methods - Quantum Monte Carlo

Quantum Monte Carlo (QMC) is a class of methods that uses statistical sampling to directly deal with high-dimensional integration that arises from working with the many-body wave function. QMC methods are among the most accurate, achieving chemical accuracy for smaller systems [29], and can in principle achieve any degree of statistical precision sought. A large ecosystem of QMC methods exists, and they have been adapted to study almost any quantum system imaginable, from discrete to continuous state space, fermionic and bosonic systems, as well as systems at both finite and zero temperature. Even though QMC methods are not computationally the cheapest, they have reasonable storage requirements as the wave function does not need to be stored directly. Moreover, the high computational cost of QMC methods can be aided by paralellisation and use of hardware acceleration, as the core calculation is repetitive.

Variational quantum Monte Carlo (VMC)

The most straightforward QMC approach is based on the variational principle, which provides a clear path towards a solution to the ground state problem. Simply use a *trial wave function* Ψ_T to parameterise the ground state and optimise the parameters of Ψ_T to reach the lowest-energy state. This lowest variational state should capture the behaviour of the ground state if the ansatz is expressive enough. Moreover, given that the variational wave function should encapsulate the main aspects of the system studied it provides intuition into the system itself. Development of trial functions has played a key role in the

applicability of VMC. Famous examples of trial wave functions include the Slater-Jastrow and Backflow wave functions. The drawback of VMC is that the variational wave function might contain a bias that cannot be avoided through optimisation of the parameters alone, see Fig. 2.1. VMC necessarily contains two steps, first is the estimation of the variational

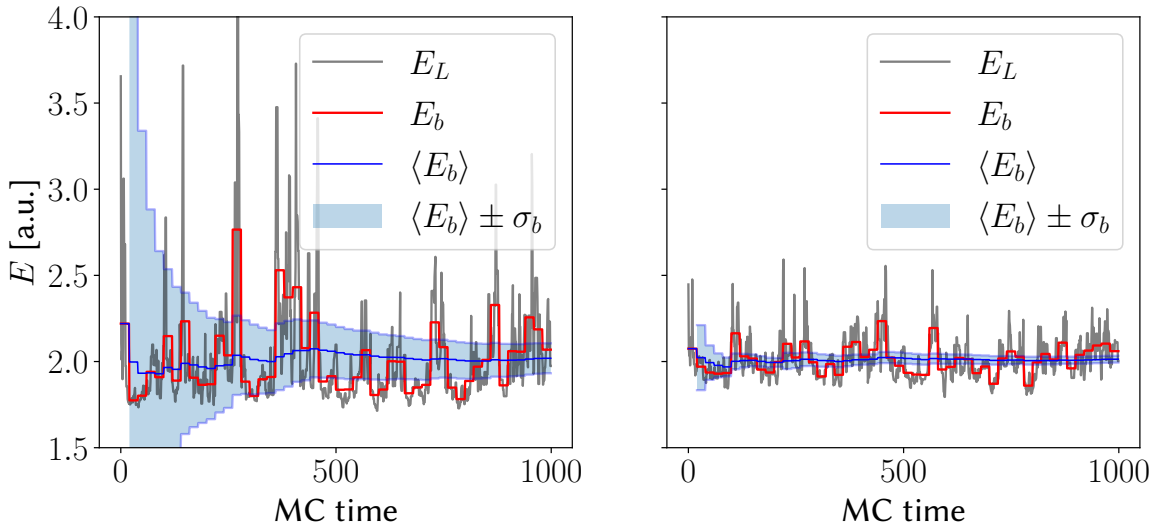


Fig. 2.1 **Ansatz quality in VMC**. Appropriateness of the variational wave function limits the quality of VMC. A poor choice of ansatz results in typical spikes of local energy and biased result (**left**), along with slower convergence compared to an accurate trial wave function (**right**). Figures show the local energy E_L, reblocked average energy $\langle E_b \rangle$ and variance σ_b of a VMC simulation of Hookium.

energy and second is the optimisation of the parameters. Any expectation of an operator $\hat{O}$ can be expressed in terms of the trial wave function as

$$\langle \hat{O} \rangle = \frac{\langle \Psi_T \mid \hat{O} \mid \Psi_T \rangle}{\langle \Psi_T \mid \Psi_T \rangle} = \frac{\sum_x \langle \Psi_T \mid x \rangle \langle x \mid \hat{O} \mid \Psi_T \rangle}{\sum_x \langle \Psi_T \mid x \rangle \langle x \mid \Psi_T \rangle}, \tag{2.17}$$

where $\mid x \rangle$ are orthogonal and normal states of the Hilbert space. If we rewrite the above expression as

$$\langle \hat{O} \rangle = \frac{\sum_x \mid \Psi_T(x) \mid^2 \hat{O}_L(x)}{\sum_x \mid \Psi_T(x) \mid^2}, \tag{2.18}$$

with $\hat{O}_L$ being the *local operator*

$$\hat{O}_L(x) = \frac{\langle x \mid \hat{O} \mid \Psi_T \rangle}{\langle x \mid \Psi_T \rangle}, \tag{2.19}$$

we can interpret $|\Psi(x)|^2/\sum_x |\Psi(x)|^2$ as a probability. Meaning that eq. (2.18) can be estimated as an average of the local operator $\hat{O}_L$

$$\langle \hat{O} \rangle \approx \frac{1}{M} \sum_{m=1}^{M} \hat{O}_L(x_m), \tag{2.20}$$

sampled from this probability distribution. The sampling can be performed using **Markov Chain Monte Carlo** (MCMC). The second step of the procedure is variational optimisation of the trial wave function, where the optimal parameters of the approximation are found by minimising the *cost function*. The straightforward choice of the variational energy E_V as a cost function turns out to be inferior to minimizing the *variance* of the energy σ_E [29]. This is because σ_E obeys the *zero-variance* property, meaning that if Ψ_T is an exact eigenvalue of the Hamiltonian

$$\hat{H}|\Psi_T\rangle = E_V|\Psi_T\rangle, \tag{2.21}$$

then the local energy E_L is constant and equal to E_V

$$E_L(x) = \Psi_T(x)^{-1}\hat{H}\Psi_T(x) = \Psi_T(x)^{-1}E_V\Psi_T(x) = E_V, \tag{2.22}$$

irrespective of the sampled configuration x and hence has zero variance. The zero-variance property has important consequences for numerical stability of optimisation., because energy variance minima are robust to finite sampling. Minimizing the variance of energy drives the trial wave function towards eigenstates of the Hamiltonian. Moreover, the statistical error of any expectation value $\langle \hat{O} \rangle$ is proportional to the variance of $\hat{O}$, making low variance doubly desirable. The parameters can be updated using several approaches e.g. gradient descent, stochastic reconfiguration [71], or the linear method [54]. It is crucial that the methods are robust to statistical noise and converge quickly as the MC step can be expensive to perform. Moreover they are only as good as the estimates of the energy (variance) gradients w.r.t the parameters.

Projector QMC (PMC) techniques

PMC is a class of QMC methods which are in essence nothing more than stochastic implementations of the power method to obtain the dominant eigenvector of a matrix or a kernel function [32]. Their distinct advantage over VMC is that they are not constrained by our parametrisation of the trial wave function, as they can describe arbitrary probability distributions. PMC methods are based on the imaginary Schrödinger equation

$$\partial_t|\Psi_t\rangle = -\hat{H}|\Psi_t\rangle. \tag{2.23}$$

Its formal solution, the time propagation of an initial wave function $|\Psi_0\rangle$ at $t = 0$, is written as

$$|\Psi_t\rangle = e^{-\hat{H}t}|\Psi_0\rangle. \tag{2.24}$$

From the spectral decomposition of the operator $e^{-\hat{H}t}$ in terms of eigenstates $|\Phi_n\rangle$ and eigen-energies E_n of the Hamiltonian $\hat{H}$

$$e^{-\hat{H}t} = \sum_n e^{-E_n t}|\Phi_n\rangle\langle\Phi_n|, \tag{2.25}$$

it follows that the term corresponding to the ground state of the system $|\Phi_0\rangle$ decays the slowest. Thus starting in some initial state and propagating for a long imaginary time $\tau = it$ leads into the ground state with the decay rate giving the ground state energy E_0 as

$$\lim_{t\to\infty}|\Psi_t\rangle \propto e^{-E_0 t}|\Phi_0\rangle, \tag{2.26}$$

where $|\Phi_0\rangle$ is the corresponding state of E_0. This of course holds if the eigenstates of $\hat{H}$ are all positive, which can be achieved by shifting the potential by a constant energy E_c, which does not change the ground state wave function. The basic step of a PMC simulation is the projection step, where an existing ensemble of configurations is projected into a new one. This projection $\hat{P}$ is done in such a way that eq. (2.26) is satisfied

$$|\Phi_0\rangle = \lim_{n\to\infty}\hat{P}^n|\Psi_0\rangle. \tag{2.27}$$

Flavours of PMC differ in the choice of $\hat{P}$, the most popular **Diffusion Monte Carlo** (DMC) [29, 64] works with the time-dependent Green's function $G(x', t'; x, t)$ of eq. (2.23)

$$\Psi(x,t) = \int \mathcal{G}\left(x,t;x',t'\right)\Psi\left(x',t'\right)\mathrm{d}x', \tag{2.28}$$

while **Green's function MC** (GFMC) [42, 43] uses the time integrated version of the Green's function

$$\Psi^{(n+1)}(x) = E\int \mathcal{G}\left(x,x'\right)\Psi^{(n)}\left(x'\right)\mathrm{d}x'. \tag{2.29}$$

Both formulations are exact, but need some additional approximations to be made practical, as Green's functions are not known for a general system. In DMC the Green's function

$$\mathcal{G}(x',t';x,t) = \langle x \mid e^{-(t-t')[\hat{T}+\hat{V}-E_c]} \mid x'\rangle, \tag{2.30}$$

is approximated for short times $\tau = t - t'$ using Trotter-Suzuki formula

$$\mathcal{G}(x' \longrightarrow x; \tau) = \underbrace{(2\pi\tau)^{-3N/2} e^{-\frac{(x-x')^2}{2\tau}}}_{\text{ordinary diffusion}} \cdot \underbrace{e^{-\tau\left[V(\mathbf{R})+V(\mathbf{R}')-2E_c\right]/2}}_{\text{reweighting} \equiv \text{birth/death}} + \mathcal{O}(\tau^3), \tag{2.31}$$

where the kinetic term is recognised to be ordinary diffusion. In practice eq. (2.31) is implemented as a simulation of a diffusion process, but instead of weighting the paths of the walkers, the potential contribution to $\mathcal{G}$ is interpreted as a probability of a walker to either branch or die, which is numerically more stable. This stochastic process converges to the ground state for sufficiently long times, see Fig. 2.2. **Reptation quantum Monte Carlo** [64] (RMC) is an alternative formulation which only uses a single walker, and instead of branching and dying, the MC moves mutate the path of the walker.

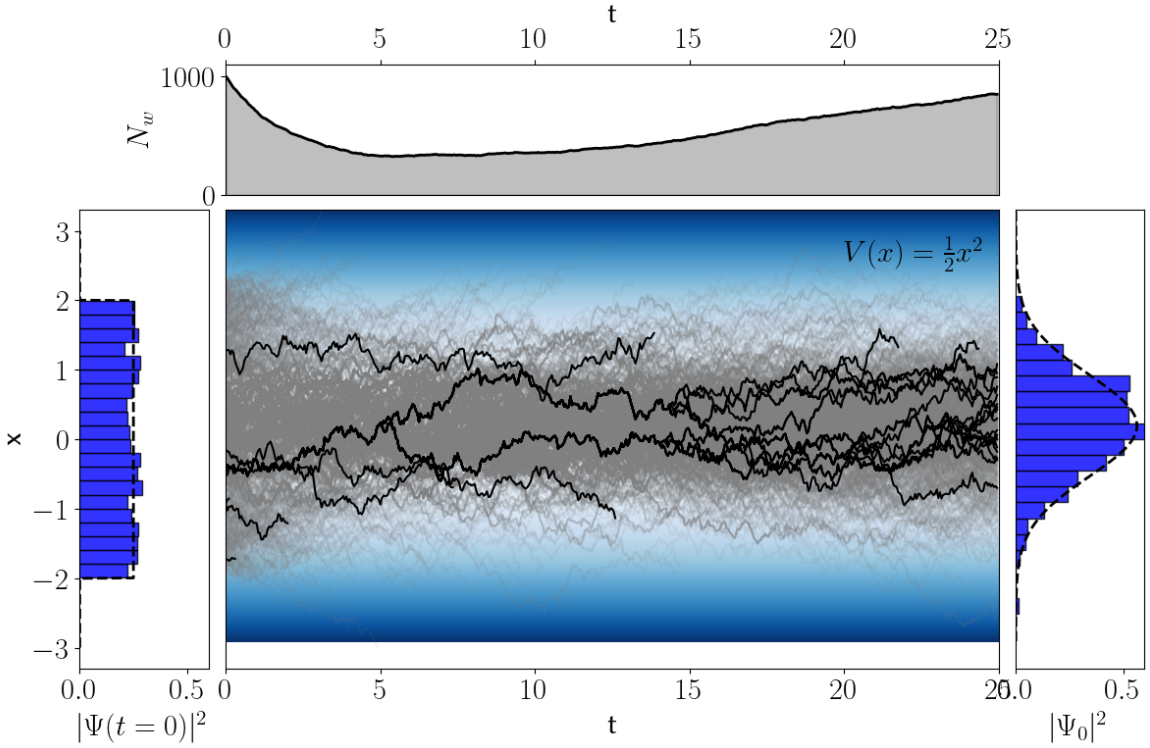


Fig. 2.2 **Diffusion Monte Carlo simulation of harmonic oscillator**, starting with $N_w = 1000$ walkers, $\tau = 0.05$, $E_c = 0.25$ and uniformly sampling their initial positions from $(-2, 2)$ (**left**). The number of walkers at each step decreases rapidly before slowly increasing (**top**). The number of walkers is controlled by adjusting E_c. Walker paths, with a few highlighted in black to emphasise birth/death process (**middle**), diffuse into the approximate ground state of the HO $u_0(x) = \frac{1}{\pi^{\frac{1}{4}}} e^{-\frac{1}{2}x^2}$ (**right**).

Using a trial wave function Ψ_T as a guiding function for importance sampling is an important improvement over vanilla DMC. This introduces a *drift* into the diffusion process, which leads the walkers into regions of large values of Ψ_T and greatly improves the statistical efficiency of the method. The guiding wave function is usually obtained by means of VMC. So far we have conveniently assumed that the wave function is positive everywhere in the domain which not generally true, e.g. in fermionic systems, and poses a problem for PMC methods.

The sign problem

Projector Monte Carlo methods can only operate with positive distributions, and as such they fall apart when applied to fermionic or frustrated systems [32]. A straightforward modification to the sampling scheme allows us to sample from a mixed-sign distribution. We sample from the distribution normally when it is positive, but sample from its absolute value and change the sign of the observable when it is negative. The issue with this approach is that the population of configurations is split between positive and negative regions, the averages over both are comparable in size and cancel out, leading to a large statistical error compared to the observable. We refer to the accompanying exponential decrease [32] in sampling efficiency with system size and temperature, **the sign problem**. Its general solution was shown to be NP-hard [82], and as it is believed that P $\neq$ NP, this implies that no *general* polynomial-time solutions exist. However, this does not mean that the problem cannot be avoided in special cases, and the search for solutions is still an area of active research [3, 5, 39]. In practice the sign problem is remedied either by the *fixed-node* [4] or *constrained-path* [91] approximation. Fixed-node imposes a boundary condition into the projection such that the projected state shares the nodal surface with the trial wave function. The projected state is now only exact when the nodal surface is exact.

2.4 Machine Learning and the quantum many-body problem

With recent growing interest in **Machine Learning** (ML) there came a wave of research that applies ML methods to the natural sciences. As it pertains to the quantum many-body problem, most of the work is focused on exploiting the expressive nature of ML models, such as **Restricted Boltzmann Machines** (RBM) [17] or **Deep Neural Networks** [14] (DNN), to efficiently represent quantum states. These approaches fall into the VMC framework, and have been used for lattice models [17], both fermionic [55] and bosonic [67]. Notably,

special **Neural Network** (NN) architectures have been used to achieve higher accuracies than coupled cluster calculations on a variety of atoms and small molecules [60, 73]. In lattice models **Convolutional Neural Networks** (CNN) have been shown to struggle to converge to nontrivial sign structures of frustrated systems, but modelling the phase and amplitude with separate networks helps in this regard [77].

The expressiveness of RBM has also been analysed in depth [16], and contrasted to Tensor Network States [19]. Very recently an application of the NN ansatz in DMC with fixed-node approximation [87] was used to improve earlier work results of the FermiNet [60].

Alternatively to above approaches, which all operate in the Schrödinger picture, reinforcement learning has been used to solve the many-body problem in the path integral representation [6, 30]. ML has also found place in mean field methods, perhaps most notably for learning the exchange and correlation functionals in DFT [25].

Chapter 3

Feynman-Kac: connecting Quantum Mechanics and Stochastic Processes

In this chapter we will provide a bridge between the quantum many-body problem discussed in the previous chapter and stochastic processes. This will entail introducing the **Feynman-Kac** (FK) formula and relating it to the **Fokker-Planck** (FP) equation and optimal control formulations of QM. Moreover, a probabilistic view of the cost function will lead us to proposals for loss functions that can be used to learn optimal transition rates and consequently sample the ground state.

The field of stochastic processes is a vast body of work, approached from different angles by mathematicians, physicists and engineers. A necessary consequence of this is that the literature ranges from extremely thorough and rigorous [65, 66] to more applied and intuitive [69]. For this reason, the mentioned discussion will be preceded by an overview of the mathematical notation, lemmas and results from stochastic processes and measure theory that underpin some core ideas of this thesis. Some concepts will not be rigorously defined, the text will point to relevant literature where this is the case.

3.1 Stochastic processes

3.1.1 Fundamentals

This brief, more formal, discussion of stochastic processes is based mostly upon classic texts [26, 65, 66] and borrows some intuitions from [69]. The most basic quantity that we will need is the **probability space**.

Definition 3.1.1 (Probability space)**.** *The probability space is a tuple* $(\Omega, \mathcal{F}, \mathbb{P})$*, where* Ω *is the* ***sample space****,* $\mathcal{F}$ *is a* σ*-****field****, and* $\mathbb{P}$ *is the* ***measure****.*

The sample space is simply the set of all possible outcomes. A canonical example would be the roll of a 6-sided dice, $\Omega = \{1,2,3,4,5,6\}$. Without measure $\mathbb{P}$, the tuple $(\Omega, \mathcal{F})$ is termed a **measurable space**.

Definition 3.1.2 (σ-field)**.** *A σ-field $\mathcal{F}$ on a set Ω, is a nonempty collection of subsets of Ω that includes Ω itself, is closed under complement, i.e. if $A \in \mathcal{F}$ then $A^c \in \mathcal{F}$, and is closed under countable unions, $\cup_i A_i \in \mathcal{F}$ if $A_i \in \mathcal{F}$ is a countable union of sets.*

The main utility of the σ-field to us is its use in defining measures. We want to be able to assign a non-negative real number to all subsets of Ω, as well as the size of the union of the disjoint sets to be the sum of their individual sizes. This is not always possible, a counterexample for the real line being Vitali sets. The collection $\mathcal{F}$, must thus only include *measurable* sets, which are precisely the ones that satisfy the constraints imposed by the σ-field.

Definition 3.1.3 (Measure)**.** *A non-negative countably additive set function $\mu : \mathcal{F} \longrightarrow \mathbb{R}$ that satisfies*

i) $\mu(A) \geq \mu(\emptyset) = 0$ for all $A \in \mathcal{F}$

ii) if $A_i \in \mathcal{F}$ is a countable sequence of disjoint sets, then $\mu(\cup_i A_i) = \sum_i \mu(A_i)$

*is a **measure**.*

If $\mu(\Omega) = 1$, then μ is a **probability measure** and will be denoted by $\mathbb{P}$. With this notion we are now able to define a random variable (r.v) and a stochastic process (s.p.).

Definition 3.1.4 (Random variable)**.** *A **random variable** X defined on Ω is a real-valued measurable function $X(\omega)$, $X : \Omega \longrightarrow \mathbb{R}^d$.*

For a function to be measurable, we require that its preimage X^{-1} is in the σ-field $\mathcal{F}$

$$X^{-1}(B) = \{\omega : X(\omega) \in B\} \in \mathcal{F}, \tag{3.1}$$

and that this holds for every Borel set B in the Borel σ-field [1] of $\mathbb{R}^d$, which is simply the smallest σ-field that contains all measurable sets in $\mathbb{R}^d$.

A random variable X induces a probability measure μ on $\mathbb{R}^d$ called its **distribution**, this is done by setting $\mu(A) = P(X \in A)$ for Borel sets A. Moreover, the distribution is usually given in terms of a **distribution function** $F(x)$

$$F(x) = \mathbb{P}(\{\omega \in \Omega : X(\omega) \leq x\}) = \mathbb{P}(X \leq x), \tag{3.2}$$

[1] For a proper definition of the Borell set see ch. 3 of [68].

and X is said to have a **density function** $f(x)$ if $F(x)$ can be written as

$$F(x) = \int_{-\infty}^{x} f(y)\mathrm{d}y. \tag{3.3}$$

In essence, the random variable provides a connection between the less familiar probability measure $\mathbb{P}$ and the cumulative distribution function (CDF).

3.1.2 Stochastic process

Definition 3.1.5 (Stochastic process)**.** *Given a probability space* $(\Omega, \mathcal{F}, \mathbb{P})$ *and a measurable (state) space* $(E, \mathcal{E})$*, we define the collection* $\{X_t : t \in T\}$ *of set* T *indexed and* $(E, \mathcal{E})$ *valued random variables a* ***stochastic process****.*

By far the most common case for the index set T is time $T = \mathbb{R}^+$. Such s.p's are called *temporal*, with examples including the model of velocity of a Brownian particle under influence of friction, in Fig. 3.1, or the Black-Scholes model. Nevertheless, the index set is not limited to time, as is often the case with Gaussian Process regression [63]. In this thesis we will mostly deal with temporal s.p's of the kind that do not "see into the future". This notion is formalized using **filtrations**. A filtration $\mathbb{F} = (\mathcal{F}_t)_{t\in T}$ on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ is just an increasing sequence or order of σ-fields

$$\mathcal{F}_s \subset \mathcal{F}_t \text{ if } 0 \leq s \leq t < \infty. \tag{3.4}$$

The filtration associated to a process that records its "past behaviour" at each time is called the **natural filtration**.

Definition 3.1.6 (Adapted process)**.** *A process* $\{X_t\}$ *is said to be* ***adapted to the filtration*** $(\mathcal{F}_t)_{t\in T}$ *if the random variable* $X_t : \Omega \longrightarrow E$ *is* $\mathcal{F}_t$*-measurable function for each* $t \in T$*.*

A process that is *non-anticipating*, i.e. depends only on the past and present, is adapted to the filtration $(\mathcal{F}_t)_{t\in T}$.

Definition 3.1.7 (Brownian motion)**.** ***Brownian motion*** *or a non-anticipating* ***Wiener process*** *is a stochastic process* W_t*, with the following properties:*

i) $W_0 = 0$

ii) W_t *is almost surely continuous in* t

iii) W_t *has independent increments*

iv) $W_t - W_s \sim \mathcal{N}(0, t-s)$ *for* $0 \leq s \leq t$

A realisation of Brownian motion can be found in Fig. 3.1.

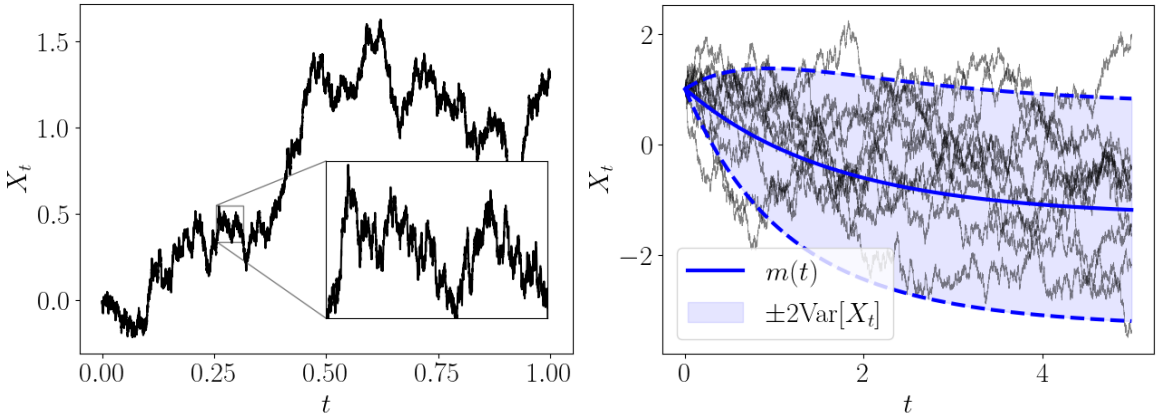


Fig. 3.1 **Brownian motion and Ornstein–Uhlenbeck process.** A single realisation of the Brownian process (**left**). The mean, variance and 10 samples of the Ohrnstein-Uhlenbeck process with $\theta = 0.6, \sigma = 1.1, X_0 = 1.0, \mu = -1.3$, integrated using Euler-Maruyama method (**right**).

3.1.3 Integrals

In order to proceed and define **stochastic differential equations** (SDE's) and the **Radon-Nikodyn** (RN) derivative, we must spend some time discussing various integrals we will use. In particular, alongside the usual Riemann integral, we will need three more types of integrals, which we will briefly describe without mathematical derivation. The simplest kind of integral we will introduce is the integral of a stochastic process

$$I = \int_0^t X_t \mathrm{d}t. \tag{3.5}$$

The simple appearance of the integral is deceiving as the integrand is a realisation of a $\mathcal{F}_t$-adapted stochastic process $\{X_t\} : \Omega \times T \longrightarrow \mathbb{R}^d$, meaning that I itself is a random variable. However, since each realisation of X_t is almost surely continuous, I can be expanded as a Riemann sum, which converges under mean-squared norm to I, so long as the mean $\mathbf{m}(t) = \mathbb{E}[X_t]$ and covariance $\mathbf{k}(t,s) = \mathrm{Cov}(X_t, X_s)$ are continuous. In practice, computing the mean and covariance of I is usually enough to understand the resulting stochastic process. Importantly, integrals of continuous functions of s.p's $h(X_t)$, $h : \mathbb{R} \longrightarrow \mathbb{R}$ can be computed in a similar manner.

The second type of integrals we need to consider, are integrals with respect to a s.p, known as **Itô integrals**

$$Y_t = \int_0^t H_s \mathrm{d}X_s, \tag{3.6}$$

where both H_s and X_s are stochastic processes. The result integral Y_t is itself a stochastic process which resides in the probability space $(\Omega, \mathcal{F}, \mathbb{P})$, filtered by $(\mathcal{F}_t)_{t\in T}$. The integral can be formalised by putting slight constraints on the sort of stochastic processes X_s and H_t can be, expanding Y_t as a Riemann sum and proving convergence. Details of this procedure can be found in [66].

Finally we must define the **Lebesgue-Stieltjes integral** [33], which is required to properly define expectations of stochastic processes.

Definition 3.1.8 (Lebesgue-Stieltjes Integral)**.** *Given probability space* $(\Omega, \mathcal{F}, \mathbb{P})$ *and measurable function* $f : \Omega \longrightarrow \mathbb{R}$*, the* ***Lebesgue-Stieltjes integral***

$$I = \int_A f(x) \mathrm{d}\mathbb{P}(x), \tag{3.7}$$

is the Lebesgue integral [2] *with respect to measure* $\mathbb{P}$*,* $A \in \mathcal{F}$*.*

With it we can define expectations in the probability space $(\Omega, \mathcal{F}, \mathbb{P})$ as

$$\underset{\mathbb{P}}{\mathbb{E}}[f(x)] = \int_\Omega f(x) d\mathbb{P}(\boldsymbol{x}). \tag{3.8}$$

For a newcomer to stochastic processes this formulation may seem redundant, can we not just calculate expectations using a Riemann integral and the pdf? We can, and when the distribution $\mathbb{P}$ can be expressed in terms of the pdf (3.3), the Lebesgue integral can be interpreted in this way. However, stochastic processes need not admit a pdf, that is when the Lebesgue-Stieltjes integral is necessary.

3.1.4 Stochastic Differential Equations

In this thesis we will refer to an SDE as an informal notation of an Itô integral equation or **Itô process**.

Definition 3.1.9 (Itô process)**.** *Given deterministic functions* $v : \mathbb{R}^d \times \mathbb{R}^+ \longrightarrow \mathbb{R}^d$ *and* $\sigma : \mathbb{R}^d \times \mathbb{R}^+ \longrightarrow \mathbb{R}^{d\times d}$*, we define the* ***Itô process*** X_t *as the sum of Itô and Lebesgue integrals*

$$X_{t+s} - X_t = \int_t^{t+s} \sigma(X_u, u)\, \mathrm{d}W_u + \int_t^{t+s} v(X_u, u)\, \mathrm{d}u, \tag{3.9}$$

[2] For proper definition of the Lebesgue integral see ch. 1 of [68].

where W_t is a Brownian motion.

In simplified notation eq. (3.9) can be written as

$$\mathrm{d}X_t = \sigma(X_t, t)\,\mathrm{d}W_t + v(X_t, t)\,\mathrm{d}t, \tag{3.10}$$

this is what we refer to as an SDE, an example can be found in Fig. 3.1. We will refer to functions v and σ, as the **drift** and **volatility** of an Itô process. The most intuitive interpretation of an SDE is in terms of the time evolution of the pdf of the process X_t. It is described by the **Fokker-Planck equation** [3]

$$\frac{\partial p(\mathbf{x}, t)}{\partial t} = -\sum_{i=1}^{N} \frac{\partial}{\partial x_i}\left[\mu_i(\mathbf{x}, t)p(\mathbf{x}, t)\right] + \sum_{i=1}^{N}\sum_{j=1}^{N} \frac{\partial^2}{\partial x_i \partial x_j}\left[D_{ij}(\mathbf{x}, t)p(\mathbf{x}, t)\right], \tag{3.11}$$

where $p(\mathbf{x}, t)$ is the pdf of the solution to the SDE and $\mathbf{D} = \frac{1}{2}\boldsymbol{\sigma}\boldsymbol{\sigma}^\top$ is the diffusion tensor. Finally we state without proof a consequence of Itô calculus, most commonly named **Itô's rule** or **lemma**. It is the stochastic calculus equivalent of the chain rule

Lemma 3.1.1 (Itô's lemma)**.** *Given an Itô process X_t as given by* (3.9) *and a twice differentiable scalar function $f(X_t, t)$, then the Itô process for f is*

$$\mathrm{d}f = \frac{\partial f}{\partial t}\mathrm{d}t + \sum_i \frac{\partial f}{\partial x_i}\mathrm{d}x_i + \frac{1}{2}\sum_{ij} \frac{\partial^2}{\partial x_i \partial x_j} f \mathrm{d}x_i \mathrm{d}x_j, \tag{3.12}$$

when compared to ordinary calculus we notice an additional quadratic term.

3.1.5 Radon-Nikodym Derivative and Girsanov theorem

To perform importance sampling we perform a change of measure in an integral

$$\int_A f(x)\mathrm{d}\mathbb{P}(x) = \int_A f(x)\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(x)\mathrm{d}\mathbb{Q}(x). \tag{3.13}$$

The function that measures the rate of change of density of one measure w.r.t another is the **Radon-Nikodym derivative** $\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(x)$.

Theorem 3.1.2 (Radon-Nikodym theorem)**.** *Let $\mathbb{P}$ and $\mathbb{Q}$ be probability measures on the measurable space $(\Omega, \mathcal{F})$, then the measurable function **Radon-Nikodym derivative** $\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(x) : \Omega \longrightarrow [0, \infty)$ exists and*

$$\mathbb{P}(A) = \int_A \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(x)\mathrm{d}\mathbb{Q}(x), \tag{3.14}$$

[3] Derivation in [69].

for set $A \subseteq \mathcal{F}$.

The RN derivative will also be useful in defining the KL divergence between two **path measures**. Properly defining the path measure would bring a lot of notational overhead. It is enough to think of it as a measure on the **path space**, i.e all possible paths of a SDE, for rigour see [47]. Finally, we state **Girsanov theorem** that is often used for transforming or removing drift functions of SDEs. It is the RN derivative between an Itô process and one with $v = 0$ and $\sigma = 1$, i.e. Brownian motion.

Theorem 3.1.3 (Girsanov Theorem)**.** *Given Itô process*

$$\mathrm{d}X_t = \mathrm{d}W_t + v(X_t, t)\mathrm{d}t \quad \textit{and} \quad X_0 = 0 \tag{3.15}$$

and Brownian motion $\mathrm{d}Y_t = \mathrm{d}W_t$*, the RN derivative of their respective path measures* $\mathbb{P}$ *and* $\mathbb{P}_0$ *is*

$$\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{P}_0} = \exp\left(-\frac{1}{2}\int_0^t |v(X_s, s)|^2 \,\mathrm{d}s + \int_0^t v(X_s, s)^\top \mathrm{d}W_s\right). \tag{3.16}$$

This *change in dynamics*, as we will call it, is true in the sense that expectations for an arbitrary functional $h(\cdot)$ of the path from 0 to t are

$$\mathop{\mathbb{E}}_{\mathbb{P}}[h(X_t)] = \mathop{\mathbb{E}}_{\mathbb{P}_0}\left[\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{P}_0}.h(Y_t)\right]. \tag{3.17}$$

For a more general case and proof see [69].

3.1.6 Markov processes

We now shift our view to a special kind of s.p's, ones that satisfy the **Markov property** called **Markov processes** or **Markovian**. The property is sometimes referred to as *memorylessness*, as the future of a Markov process depends only on the present state. We can classify the processes based on the system's **state-space** S, which can be either discrete (countable) or continuous, and **time indexing** of the system, either discrete-time $\{X_n\}_{n\geq 0}$ or continuous-time $\{X_t\}_{t\geq 0}$. A taxonomy is given in Table 3.1. We will not specifically discuss Markov processes in continuous state-space, but it is important to note that any Itô process with time-homogenous drift $v = v(X_t)$ and volatility $\sigma = \sigma(X_t)$ is Markovian.

From now on we refer to Markov processes in countable state-space as **Markov chains**. We base our discussion on [65] and [56].

Table 3.1 **Taxonomy of Markov processes**

	Countable state-space	Continuous state-space
Discrete time	**index:** $\{X_n\}_{n\geq 0}$, $n \in \mathbb{Z}^+$ **state-space:** countable set I **define:** stochastic $\{P\}_{ij}$ **example:** DTMC	**index:** $\{X_n\}_{n\geq 0}$, $n \in \mathbb{Z}^+$ **state-space:** general state-space Ω **define:** stochastic kernel K **example:** Harris Chain
Continuous time	**index:** $\{X_t\}_{t\geq 0}$, $t \in \mathbb{R}^+ = [0,\infty)$ **state-space:** countable set I **define:** rate $\{\Gamma\}_{ij}$ equiv. to jump chain $\{J_n\}_{n\geq 0}$ and hold times $\{S_n\}_{n\geq 1}$. **example:** CTMC	**index:** $\{X_t\}_{t\geq 0}$, $t \in \mathbb{R}^+ = [0,\infty)$ **state-space:** general state-space Ω **define:** stochastic kernel K **example:** Diffuson process

Discrete-time Markov Chains

The simplest and most common Markov process is a Markov chain in discrete time, see Fig. 3.2a. Its state-space is a countable set I and we call each $i \in I$ a **state**. We define a distribution λ in a familiar way

$$\lambda = \{\lambda_i : i \in I\} \quad \text{where} \quad \forall i : \ 0 \leq \lambda_i < \infty \quad \text{and} \quad \sum_{i\in I} \lambda_i = 1. \tag{3.18}$$

We can now set λ as a distribution of some random variable $X : \Omega \longrightarrow I$ as

$$\lambda_i = \mathbb{P}(X = i) = \mathbb{P}(\{\omega : X(\omega) = i\}), \tag{3.19}$$

where we are still working in the probability space $(\Omega, \mathcal{F}, \mathbb{P})$. A discrete-time Markov chain is defined in terms of its **transition matrix** $P = \{p_{ij} : i,j \in I\}$, which is a **stochastic matrix** meaning all of its rows $\{p_{ij} : j \in I\}$ are distributions.

Definition 3.1.10 (Discrete-time Markov chain)**.** *A discrete time stochastic process $\{X_n\}_{n\geq 0}$ is a **discrete-time Markov chain** with initial distribution λ and transition matrix P if for $i_1, \ldots, i_n + 1 \in I$ and $n \geq 0$*

i) $\mathbb{P}(X_0 = i_1) = \lambda_{i_1}$

ii) $\mathbb{P}(X_{n+1} = i_{n+1} \mid X_0 = i_0, \ldots, X_n = i_n) = p_{i_n i_{n+1}}$

After rewriting the second condition above, it is clear that the Markov chain is without memory as

$$\mathbb{P}(X_{n+1} = i_{n+1} \mid X_0 = i_1, \dots, X_n = i_n) = \mathbb{P}(X_{n+1} = i_{n+1} \mid X_n = i_n). \tag{3.20}$$

Intuitively we understand the discrete-time Markov chain as a system changing its state at discrete time intervals, each time choosing the next state according to the row of the Transition matrix corresponding to the current state.

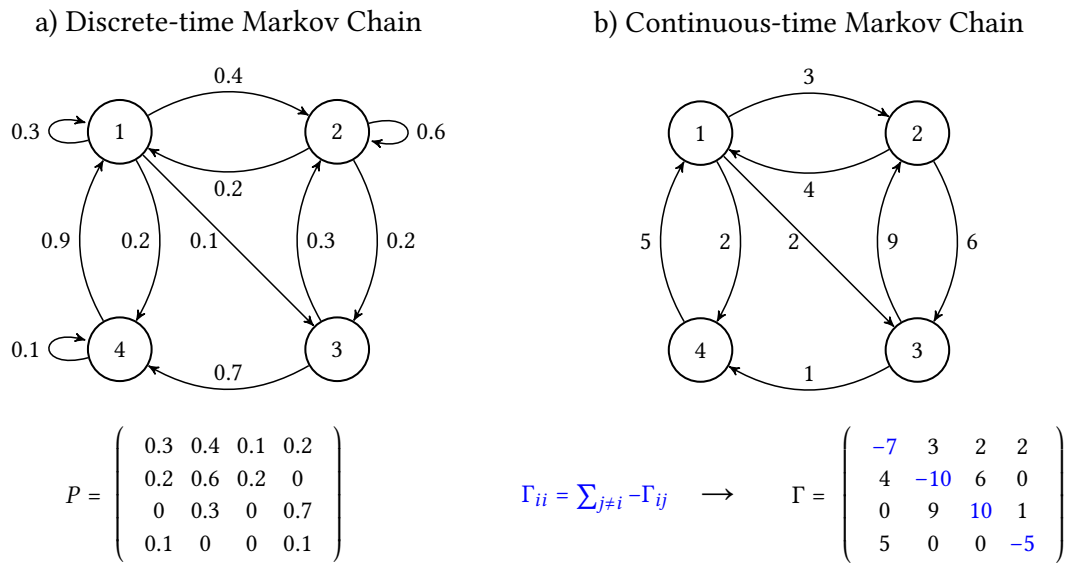


Fig. 3.2 **Discrete and continuous time Markov Chains.** Discrete-time Markov Chain defined by P (**left**). Continuous-time Markov Chain defined by Γ (**right**).

Continuous-time Markov Chains

Defining a Markov chain in continuous time is trickier, as describing the system with a stochastic matrix no longer suffices due to transition probabilities becoming zero considering an infinitesimal time. Instead a continuous-time Markov Chain (CTMC) is characterised with a **rate matrix** or **infinitesimal generator matrix** Γ defined on the set I, which has the following three properties

i) $0 \leq \Gamma_{ii} < \infty, \quad \forall i$

ii) $\Gamma_{ij} \geq 0, \quad \forall i \neq j$

iii) $\sum_{j \in I} \Gamma_{ij} = 0, \quad \forall i.$

While the CTMC can be interpreted in a number of ways, we shall use the so called **jump chain** and **holding times** representation, shown in Fig. 3.3.

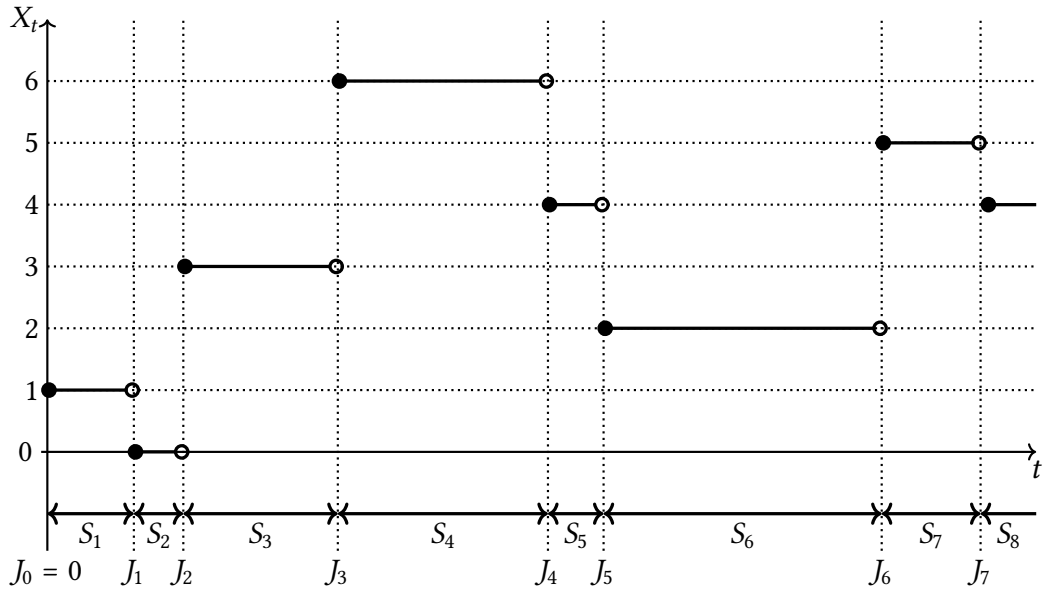


Fig. 3.3 **Jump chain and Holding times.** A discrete space Markov process $\{X_t\}_{t\geq 0}$ in continuous time. The holding times S_n are independent exponential random variables and the transition probabilities at jump times J_n are given with the jump matrix Π. Adapted from [56].

We can think of a CTMC as a series of discrete jumps, where the system remains in each state for a certain holding time. This suggests that we can construct the CTMC from a discrete-time chain with stochastic matrix Π, which we will call the **jump matrix**, and a set of independent random variables $\{S_n\}$ which determine the holding times. We construct matrix Π by rescaling rows of Γ so they add up to one and putting a 0 on the diagonal

$$\begin{aligned} \Pi_{ij} &= \begin{cases} \Gamma_{ij}/\Gamma_{ii} & \text{if } j \neq i \text{ and } \Gamma_{ii} \neq 0 \\ 0 & \text{if } j \neq i \text{ and } \Gamma_{ii} = 0 \end{cases} \\ \Pi_{ii} &= \begin{cases} 0 & \text{if } \Gamma_{ii} \neq 0 \\ 1 & \text{if } \Gamma_{ii} = 0. \end{cases} \end{aligned} \tag{3.21}$$

In order for the process to possess the Markov property, the distribution of holding times $\{S_n\}$ must be exponential [56],

$$S_{n+1} \sim \text{Exp}(-\Gamma_{ii}(X_n)), \tag{3.22}$$

with exponential parameters being $-\Gamma_{ii}$, where i is the current state. Processes with different holding time distributions are called **semi-Markov**. The jump times $\{J_n\}$ are simply

$$J_n = S_1 + \ldots + S_n. \tag{3.23}$$

Definition 3.1.11 (Continuous-time Markov chain)**.** *A stochastic process $\{X_t\}_{t\geq 0}$ on set I is a **continuous-time Markov chain** if its jump chain $\{Y_n\}_{n\geq 0}$ is a discrete-time Markov chain and its holding times $\{S_n\}_{n\geq 1}$ are independent exponential random variables $S_n \sim \text{Exp}(-\Gamma_{ii}(X_n))$.*

An equivalent formulation is in terms of **competing exponentials**. Transitions $\Gamma_{j\to k}$ from j to k are defined as independent exponential random variables $\tau_{j\to k}$

$$\tau_{j\to k} \sim \text{Exp}(\Gamma_{jk}), \quad j \neq k \tag{3.24}$$

the next state is then chosen as

$$Y_{n+1} = \text{argmin}_k \, \tau_{j\to k}. \tag{3.25}$$

The chain $\{Y_n\}_{n\geq 0}$ along with times

$$S_n = \min_k \tau_{j\to k}, \tag{3.26}$$

gives the full description of the CTMC. We now interpret Γ_{ii} as the rate ob *leaving* current state and Γ_{ij} as the rate of *going* from i to j.

3.2 The Feynman-Kac formula

The Feynman path integral formulation introduced in Chapter 2 was extensively used by physicists for decades, even in the absence of a formal mathematical formulation which is hard to define because of the difficulties with defining an appropriate measure on the path space. Kac [41] provided a rigorous formulation of the *real-valued* case of the Feynman path integral, and the resulting Feynman-Kac formula eq. (3.29) provides a bridge between *parabolic* partial differential equations and stochastic processes.

3.2.1 Feynman-Kac in continuous state space

To illustrate the Feynman-Kac formula, let us consider a single particle with Hamiltonian

$$\hat{H} = -\frac{\mathrm{d}^2}{\mathrm{d}x^2} + V(x) \tag{3.27}$$

and the Schrödinger equation in *imaginary time*, which is of the parabolic type,

$$\partial_t |\psi_t\rangle = -\hat{H}|\psi_t\rangle. \tag{3.28}$$

In close analogy to arguments presented in the DMC section 2.3.1, Kac noticed that the kinetic term of the Lagrangian in eq. (2.6) could be interpreted as a measure on Brownian walks, and a solution to the imaginary time Schrödinger equation can be written as

$$\psi(x,t) = \mathop{\mathbb{E}}_{X\sim \text{ Brownian with } X_t=x} \left[\exp\left(-\int_0^t V(X_\tau,\tau)\,\mathrm{d}\tau \right) \psi(X_0,0) \right], \tag{3.29}$$

where only the *endpoint* at time t of the Brownian process is fixed, whereas the starting point at time $t = 0$ is not. $\psi(x,0)$ encodes the initial condition into this representation. When there is no external potential $V(x) = 0$, the Schrödinger equation in imaginary time is the diffusion equation and the Feynman-Kac solution is simply

$$\begin{aligned} \psi(x,t) &= \mathop{\mathbb{E}}_{X\sim \text{ Brownian with } X_t=x} [\psi(X_0,0)] \\ &= \frac{1}{\sqrt{2\pi t}} \int e^{-(x-x')^2/2t} \psi_0\left(x'\right) \mathrm{d}x'. \end{aligned} \tag{3.30}$$

An illustration of the Feynman-Kac formula applied to a free particle in 1D is depicted in Fig. 3.4. The role of the potential $V(x)$ is to weight the Brownian paths, in turn defining the Feynman-Kac path measure $\mathbb{P}_{\mathrm{FK}}$ which is related to the Brownian measure $\mathbb{P}_0$ by the Radon-Nykodym derivative

$$\frac{\mathrm{d}\mathbb{P}_{\mathrm{FK}}}{\mathrm{d}\mathbb{P}_0} = \mathcal{N} \exp\left(-\int V(X_t)\,dt \right), \tag{3.31}$$

where $\mathcal{N}$ is a normalizing constant. Intuitively we can understand the measure as assigning more weight to Brownian paths that spend more time in the attractive region ($V(x) < 0$) than in repulsive regions ($V(x) > 0$). This is illustrated in Fig. 3.5.

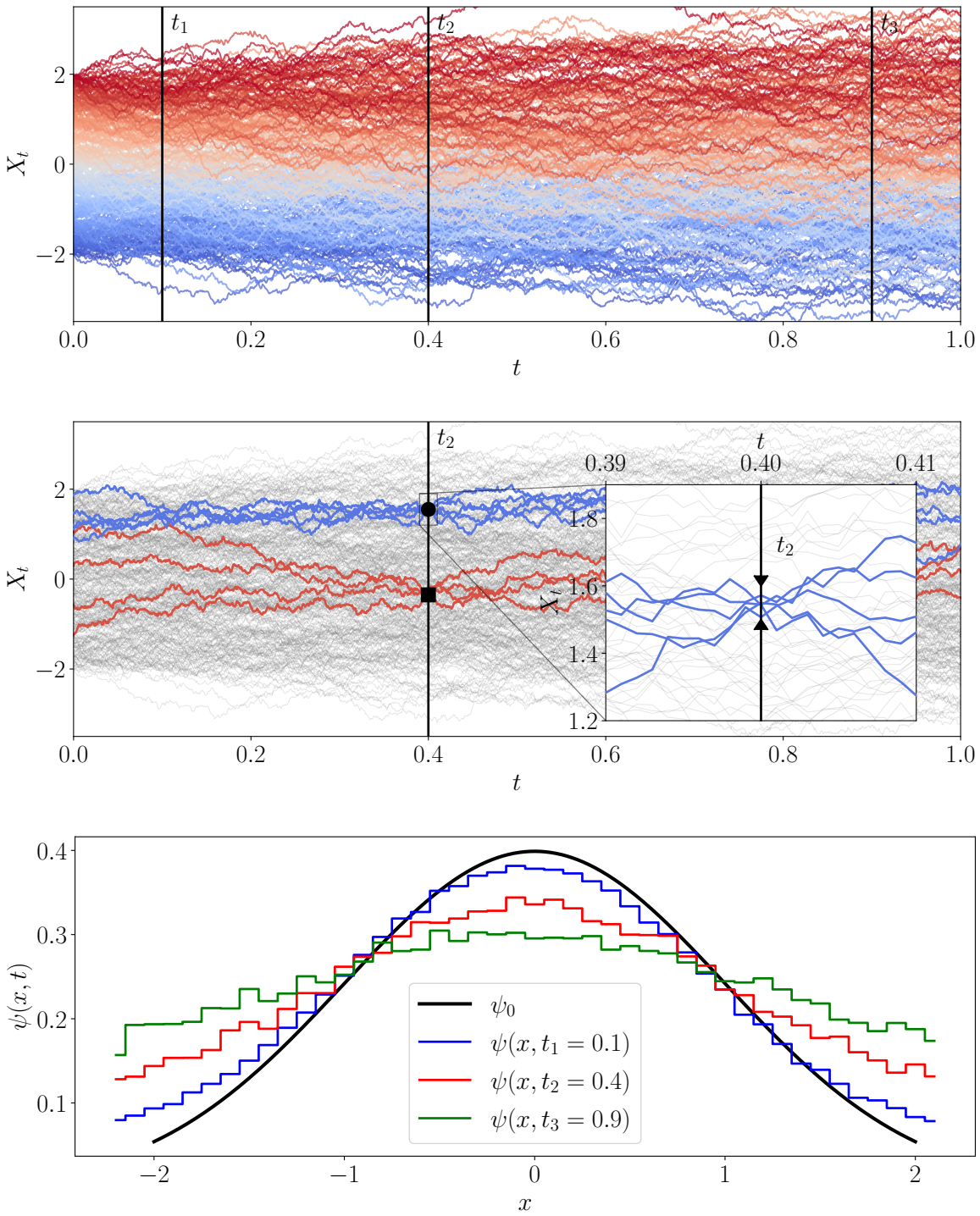


Fig. 3.4 **Feynman-Kac for a free particle in 1D.** $N = 400$ Brownian walks starting from different x_0, the colour signifies initial position (**top**). In order to evaluate ψ between $x - \frac{\delta x}{2}$ and $x + \frac{\delta x}{2}$ at some time t we must first find Brownian paths that end there. The paths that pass through $x \in (1.5, 1.6)$ (blue) and through $x \in (-0.4, -0.3)$ (red) at $t = 0.4$ are coloured, others are left in grey (**middle**). Time evolution of the initial condition $\psi_0 = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$, by estimating $\mathbb{E}[\psi(X_0, 0)]$ from the filtered paths at each timestep (**bottom**).

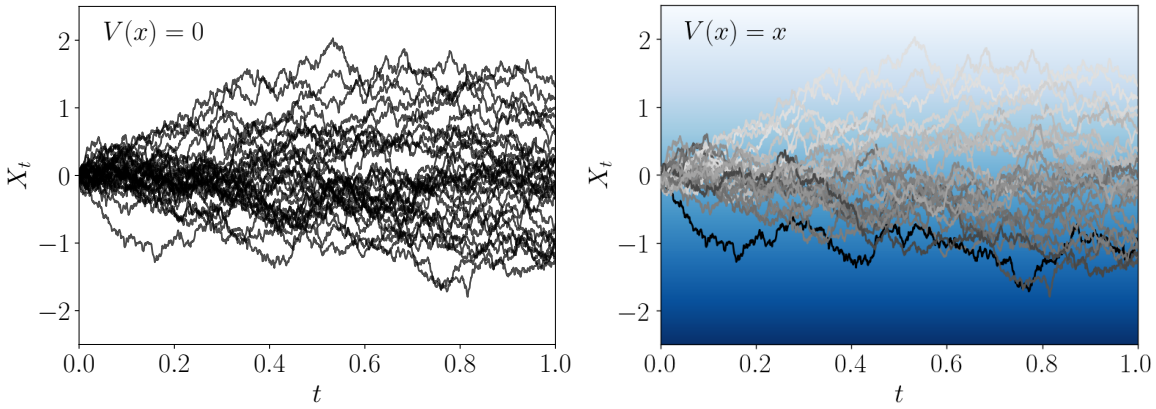


Fig. 3.5 **Feynman-Kac measure in a linear potential.** $N = 30$ Brownian paths (**left**). The paths coloured (P(black) = 1, P(white) = 0) by their likelihood under the Feynman-Kac measure with $V(x) = x$ (**right**).

Moreover, this new stochastic process is Markovian, meaning that a clear connection exists between the imaginary time Schrödinger equation and a SDE of form (3.10) with time-homogeneous σ and v. Indeed, in the continuous case the mapping between the Fokker-Planck equation (3.11) and the Schrödinger equation exists in the form of a similarity transform. Starting from the FP equation of a stochastic process with constant volatility $\sigma = 1$

$$\mathrm{d}X_t = \mathrm{d}W_t + v(X_t)\,\mathrm{d}t, \tag{3.32}$$

and drift $v(x) = -U'(x)$ given as a gradient of some potential function $U(x)$, the pdf $\rho(t,x)$ of the process is

$$\frac{\partial \rho}{\partial t} = \frac{\partial}{\partial x}\left[\frac{\partial \rho}{\partial x} + U'(x)\rho\right]. \tag{3.33}$$

We can define the function

$$\psi(x,t) = \frac{\rho(x,t)}{\sqrt{\rho_0(x)}}, \tag{3.34}$$

with ρ_0 being the stationary distribution of the FP equation

$$\frac{\partial}{\partial x}\left[\frac{\partial \rho}{\partial x} + U'(x)\rho\right] = 0 \quad \longrightarrow \quad \rho_0(x) \propto \exp(-U(x)), \tag{3.35}$$

which satisfies the imaginary time Schrödinger equation (3.28) with the Hamiltonian

$$\hat{H} = -\frac{\partial^2}{\partial x^2} \overbrace{-\frac{U''}{2} + \frac{U'^2}{4}}^{\equiv V(x)}. \tag{3.36}$$

The ground state of this Hamiltonian has zero energy and is

$$\psi_0(x) = \sqrt{\rho_0(x)}. \tag{3.37}$$

In other words, the quantum ground state probability distribution $|\psi_0|^2$ is the same as the classical stationary distribution ρ_0 of the stochastic process X_t which is in the literature known as the Nelson's ground state process [52, 2]. This connection is one that our computational method will exploit, as the ability to efficiently sample from the stochastic process with correct drift v is equivalent to sampling from the ground state of the quantum system. Even though the connection is simple, it comes with a caveat. Starting from the Schrödinger equation one needs to find the drift $v(x)$, and while the connection with $V(x)$ is clear-cut in this simple example, this is not the case in many-body systems. The *inverse* problem of finding the stochastic process of a given Hamiltonian is difficult, and is one of the core problems approached in this thesis.

3.2.2 Stoquastic Hamiltonians

Before we illustrate the connection between the imaginary time Schrödinger equation and CTMCs we must introduce **Stoquastic Hamiltonians** [11], a class of Hamiltonians which do not suffer from the sign problem.

Definition 3.2.1 (Stoquastic Hamiltonian)**.** *A k-local Hamiltonian $\hat{H} = \sum_i \hat{H}_i$ is **stoquastic** if there exists a local basis $\mathcal{B}$ in which off-diagonal matrix elements of terms $\hat{H}_i$ are zero or negative*

$$\langle x|\hat{H}|y\rangle \leq 0, \quad \forall x, y \in \mathcal{B} \quad \textit{with } x \neq y. \tag{3.38}$$

If we consider the matrix $e^{-\tau\hat{H}}$ for a non-positive $\hat{H}$, we see that every term in the expansion

$$e^{-\tau\hat{H}} = 1 - \tau\hat{H} + \frac{1}{2}(\tau\hat{H})^2 + \ldots \tag{3.39}$$

is a non-negative matrix, thus so is $e^{-\tau\hat{H}}$. In the infinite time limit, the ground state is projected out

$$\lim_{\tau\to\infty} e^{-\tau\hat{H}} = |\psi_0\rangle\langle\psi_0|, \tag{3.40}$$

and a global phase exists for which the ground state has non-negative amplitudes. Moreover, if $\hat{H}$ is irreducible then the ground state is node-less [Crosson]

$$\psi_0(x) > 0 \text{ for all } x \in \mathcal{B}. \tag{3.41}$$

We can decompose a stoquastic Hamiltonian into a rate matrix Γ, defined according to section 3.1.6, and a diagonal potential matrix V

$$H = -\Gamma + V. \tag{3.42}$$

The rates Γ are analogous to the Brownian motion in the continuum case, and can be interpreted as the kinetic contribution. They represent a CTMC which we will understand as *passive dynamics*. In terms of the Hamiltonian matrix,

$$\Gamma_{s\to s'} = \begin{cases} -H_{ss'} & \text{if } s \neq s \\ \sum_{s'\neq s} H_{ss'} & \text{if } s = s' \end{cases} \tag{3.43}$$

and the potential is

$$V(\boldsymbol{s}) = H_{ss} + \sum_{s'\neq s} H_{ss'}. \tag{3.44}$$

From now on we use $\longrightarrow$ notation to emphasise the transition between **adjacent states** $s \neq s'$, which satisfy $H_{ss'} \neq 0$.

3.2.3 Stoquastic representations of the lattice models

Let us now reinterpret lattice models introduced in section 2.2 in terms of the stoquastic Hamiltonian decomposition in eq. (3.42).

Transverse-field Ising model

In the z-spin basis a state is defined as $\mathbf{s} = \boldsymbol{\sigma}^z$, where $\boldsymbol{\sigma}^z = (\sigma_1^z, \sigma_2^z, \dots, \sigma_N^z) \equiv (s_1^z, s_2^z, \dots, s_N^z)$ and σ_i^N represent either spin-up or spin-down state at site i. The TFIM Hamiltonian eq. (2.9) in this representation becomes

$$\langle \mathbf{s}'|H_{\text{TFIM}}|\mathbf{s}\rangle = -J\sum_{\langle i,j\rangle} s_i s_j \langle \mathbf{s}'|\mathbf{s}\rangle - h\sum_i \langle \mathbf{s}'|\sigma_i^+ + \sigma_i^-|\mathbf{s}\rangle. \tag{3.45}$$

If we examine the second term for $\mathbf{s}' \neq \mathbf{s}$ we see that states will be adjacent, meaning there will be passive transitions between them, if the two configurations differ by only a single spin flip. In any such case the second term contributes $-h$ and the passive rates are

$$\Gamma_{\mathbf{s}\to\mathbf{s}'} = h. \tag{3.46}$$

The passive process in the TFIM is then just individual spin flips, see Fig. 3.6. The potential follows from its definition

$$V(\mathbf{s}) = -\sum_{\mathbf{s}\neq\mathbf{s}'} \Gamma_{\mathbf{s}\to\mathbf{s}'} + H_{ss} = -hN - J\sum_{\langle i,j\rangle} s_i s_j, \tag{3.47}$$

where N is the number of lattice sites.

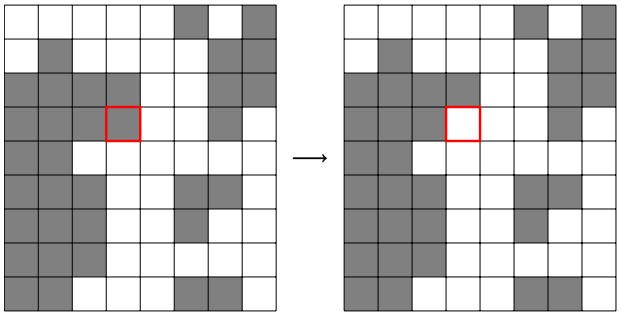


Fig. 3.6 **Ising passive process.** Adjacent states in the TFIM are ones that differ by a single spin flip. The transition rates for all adjacent states are constant h and the passive process flips individual spins according to the rates.

The XY model

In the z-spin basis the XY Hamiltonian (2.14) becomes

$$\langle \mathbf{s}'|H_{\mathrm{XY}}|\mathbf{s}\rangle = -\sum_{\langle i,j\rangle} \langle \mathbf{s}'|\hat{\sigma}_i^+\hat{\sigma}_j^- + \hat{\sigma}_i^-\hat{\sigma}_j^+|\mathbf{s}\rangle. \tag{3.48}$$

Here two states $\mathbf{s}\neq\mathbf{s}'$ are adjacent if we can find two neighbouring i, j with $s_i \neq s_j$ in $\mathbf{s}$, such that swapping the values s_i and s_j gives s'. The passive rates are then

$$\Gamma_{\mathbf{s}\to\mathbf{s}'} = 1, \tag{3.49}$$

and the corresponding passive process is the **symmetric exclusion process** (SEP), depicted in Fig. 3.7. The potential follows as

$$V(\mathbf{s}) = -\sum_{\mathbf{s}\neq\mathbf{s}'} \Gamma_{\mathbf{s}\to\mathbf{s}'} = -\sum_{\langle i,j\rangle} \left[n_i\left(1-n_j\right) - n_j\left(1-n_i\right)\right] \tag{3.50}$$

where n_i is the occupation number of site i.

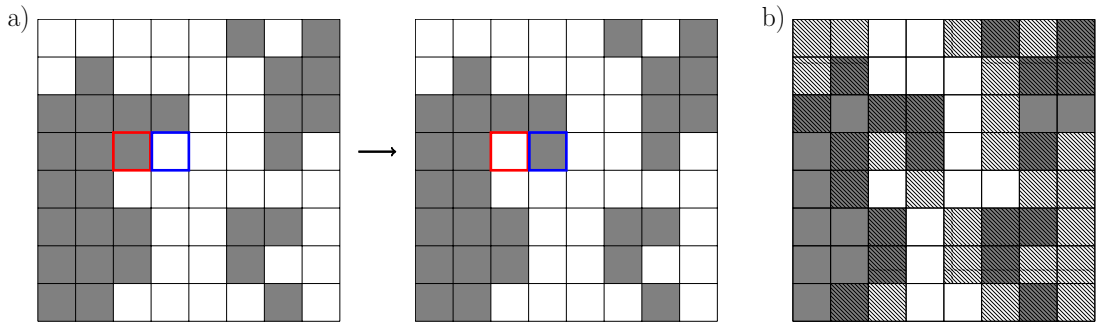


Fig. 3.7 **Symmetric exclusion process.** Adjacent states of the XY model are ones separated by a switch of two non-equal neighboring spins, the associated passive process is the SEP **a)**. On subfigure **b)** a mask of all sites that can make a transition in the next time step (either up or right) is shown, the mask is necessary for implementing the SEP, see section 4.4.

3.2.4 Feynman-Kac in discrete state space

The decomposition $H = -\Gamma + V$ allows us to define the Feynman-Kac formula in discrete state space [66] as

$$\psi(s_t, t) = \mathop{\mathbb{E}}_{\Sigma_{[0,t]\sim\Gamma}} \left[\exp\left(-\int_0^t V(\boldsymbol{s}_{t'})\,\mathrm{d}t' \right) \psi(s_0, 0) \right]. \tag{3.51}$$

The expectation is now taken over the process driven by Γ and weighted by the potential V. We denote the trajectory as $\Sigma_{[0,t]}$, where $\Sigma_{t'}$ is the state of the system at time $t' \in [0, t]$. Analogous as with the Feynman-Kac formula in continuous state space, this defines a new CTMC with measure $\mathbb{P}_{\mathrm{FK}}$, which is related to passive dynamics with measure $\mathbb{P}_0$ via the RN derivative, eq. (3.31).

How exactly is this CTMC related to the imaginary time Schrödinger equation? Again via a similarity transform. The difference being that instead of Fokker-Planck we use the master equation to describe the time evolution of the pdf P

$$\frac{\partial P(s)}{\partial t} = \sum_{s' \neq s} \left[\Gamma_{s' \to s} P(s') - \Gamma_{s \to s'} P(s) \right]. \tag{3.52}$$

The stationary state P_0 of the master equation satisfies detailed balance

$$\Gamma_{k \to j} = \exp\left(\frac{V_{s'} - V_s}{2} \right), \tag{3.53}$$

and is thus

$$P_0(s) \propto \exp(-V_s). \tag{3.54}$$

The wave function

$$\psi(s,t) = \frac{P_s(t)}{\sqrt{P_0(s)}}, \tag{3.55}$$

then satisfies the imaginary time Schrödinger equation with the Hamiltonian

$$\hat{H}_{s's} = \begin{cases} -P_0^{-\frac{1}{2}}(s)\Gamma_{s'\to s}P_0^{\frac{1}{2}}(s') = -1 & s' \neq s \\ \sum_{s'\neq s}\Gamma_{s\to s'} & s' = s. \end{cases} \tag{3.56}$$

Again this Hamiltonian has a zero-energy ground state

$$\psi_0(s) = \sqrt{P_0(s)} \propto \exp\left(-\frac{V_s}{2}\right). \tag{3.57}$$

The quantum probability in the ground state $|\psi_0(s)|^2$ coincides with the stationary distribution of the CTMC. The relation in the direction from Markov process to Hamiltonian is clear, but we are interested in the inverse, starting from the Hamiltonian and finding the corresponding stochastic process. We now turn our attention towards defining a suitable optimisation objective that, when optimised, will yield the correct Markov process.

3.3 Control theoretic approach to QM and loss functions

3.3.1 Holland Cost in continuous space

In section 3.2.1 we explored a connection between the ground state of some quantum system and Itô processes. We have seen that the ground state probability $|\psi_0|^2$ is also the stationary distribution π of some stochastic process

$$\mathrm{d}X_t = \mathrm{d}W_t + v'(X_t)\,\mathrm{d}t, \tag{3.58}$$

where v' is the optimal drift. Finding the correct drift in one-dimension was straightforward, but how to do it in higher dimensions ($W_t, X_t \in \mathbb{R}^n$ and $v : \mathbb{R}^n \longrightarrow \mathbb{R}^n$) remained unanswered. Holland [37] formulated the search for optimal drift as a stochastic control problem with cost function

$$C[v] = \lim_{T\to\infty}\frac{1}{T}\mathbb{E}\left[\int\left(\frac{1}{2}|v(X_t)|^2 + V(X_t)\right)\mathrm{d}t\right], \tag{3.59}$$

where the expectation is over the process in eq. (3.58). Its derivation and proof of unique solution can be found in Appendix A.1. To see that the minimum of $C[v]$ really corresponds to the ground state, we first rewrite the cost in terms of the stationary distribution $\pi(x|v)$

of Itô process generated by the drift v

$$C[v] = \int \left[\frac{1}{2}|v(X_t)|^2 + V(X_t)\right] \pi(x \mid v)\mathrm{d}x. \tag{3.60}$$

The distribution π must satisfy the stationary Fokker-Planck equation

$$\frac{1}{2}\nabla^2\pi - \nabla\cdot(v\pi) = 0,. \tag{3.61}$$

Since the equality holds for drift $v = \frac{\nabla\psi}{\psi}$ and distribution $\pi = \psi^2$, cost is

$$C[v] = \int \left[\frac{1}{2}(\nabla\psi)^2 + V(X_t)\,\psi^2\right] \mathrm{d}x. \tag{3.62}$$

For normalised ψ the integrand is the expected value of the quantum energy of the system, its minimum value λ is the ground state energy E_0 which is achieved for the ground state wave function ψ_0. We could directly use the Holland's cost to find the optimal rates v', but we will instead use the fact that trajectories sampled under the Feynman-Kac measure $\mathbb{P}_{\mathrm{FK}}$ coincide with ones sampled from the optimal drift, and base our variational approach on finding the FK measure.

3.3.2 Loss for continuous state space

To see that the path measures $\mathbb{P}_v$ of the process in eq. (3.32) and the Feynman-Kac measure $\mathbb{P}_{\mathrm{FK}}$ coincide for the optimal rates v', we first need a measure of similarity between probability distributions. For this purpose we will use the **Kullback–Leibler** (KL) divergence D_{KL}

$$D_{\mathrm{KL}}(p\|q) = \mathop{\mathbb{E}}_{p}\left[\log\left(\frac{p}{q}\right)\right]. \tag{3.63}$$

The quantity is a divergence because of the asymmetry $D_{\mathrm{KL}}(p\|q) \neq D_{\mathrm{KL}}(q\|p)$. It is a strictly positive quantity $D_{\mathrm{KL}}(p\|q) \geq 0$ except for $p = q$ when $D_{\mathrm{KL}}(p\|q) = 0$. In order to obtain the KL divergence between $\mathbb{P}_v$ and $\mathbb{P}_{\mathrm{KL}}$ we first express the RN derivative of each path measure w.r.t. Brownian motion using Girsanov theorem (3.16). Both RN derivatives can then be combined to express the logarithm Radon-Nikodym derivative or the *log-likelihood* ratio as

$$\log\left(\frac{d\mathbb{P}_v}{d\mathbb{P}_{\mathrm{FK}}}\right) = \tilde{\ell} - E_0\,T + \log\left(\frac{\varphi_0(r_0)}{\varphi_0(r_T)}\right), \tag{3.64}$$

where we have defined $\tilde{\ell}$ as

$$\tilde{\ell} \equiv \int v(r_t)\,\mathrm{d}W_t + \int \left(\frac{1}{2}|v(r_t)|^2 + V(r_t)\right)\mathrm{d}t. \tag{3.65}$$

The boundary term depends the ground state φ_0 at initial r_0 and final r_T points in the trajectory, and originates from the normalisation constant $\mathcal{N}$ in the Radon-Nikodym derivative. The KL divergence is

$$D_{\mathrm{KL}}(\mathbb{P}_v \| \mathbb{P}_{\mathrm{FK}}) = \mathop{\mathbb{E}}_{\mathbb{P}_v}\left[\ell' - E_0 T + \log\left(\frac{\varphi_0(r_0)}{\varphi_0(r_T)}\right)\right]. \tag{3.66}$$

If we consider the above Kullback-Leibler divergence in the long time limit $T \longrightarrow \infty$, we see that the expectation of the first term of $\tilde{\ell}$ is zero. Moreover, if the marginal distributions $\psi_0(r_0)$ and $\psi_0(r_T)$ coincide, the boundary term vanishes as well, and the KL divergence is equivalent to the Holland cost, thus vanishes for optimal drift. This means that finding the correct path measure is equivalent to finding the optimal drift, and in standard machine learning fashion, minimizing D_{KL} can be used to find the optimal rates. Sampling the trajectories becomes a matter of integrating a SDE and can be done with some standard approach, e.g. Euler-Mayurama method. Tje gradients of D_{KL} w.r.t rate parameters θ can be obtained using stochastic backpropagation where the reparameterisation trick is used for each increment in the discretised SDE. The gradient of the boundary term is non-zero, but can be estimated by expressing ψ_0 in terms of v_θ. Missing steps of the derivation can be found in Appendix A.2, and more details in [6].

3.3.3 Todorov Cost in discrete state space

We now turn our attention towards finding a variational approach in discrete state space and continuous time. We rely on foundational work done on linearly solvable Markov Decision Processes[4] (MDP) by Todorov [79, 80] and its applications to lattice problems [30]. The main idea is to reinterpret the dynamics of the imaginary time Schrödinger equation in the Todorov MDP framework by treating control as a modification of the passive dynamics of the system. The imaginary time Schrödinger equation can be written as

$$\frac{\partial \psi(s_j,t)}{\partial t} = \underbrace{-\sum_{s_k \neq s_j} \Gamma_{s_j \to s_k}\left[\psi(s_k,t) - \psi(s_j,t)\right]}_{\text{passive dynamics}} - V_{s_j}\psi(s_j,t) \tag{3.67}$$

[4]For a general discussion of MDP's see [76]

to emphasise the decomposition of the Hamiltonian eq. (3.42) into passive dynamics and the potential. Akin to the transformation in discrete space, we use $\psi(s_j, t) = \exp[-u(s_j, t)]$, to arrive at an alternative form

$$-\frac{\partial u(s_j, t)}{\partial t} = \min_{\Gamma^{(v)}} \left[\ell(s_j, v) + \sum_{s_k} \Gamma^{(v)}_{s_j \to s_k} (u(s_k, t) - u(s_j, t)) \right], \tag{3.68}$$

where $\Gamma^{(v)}$ are the rates of the modified CTMC, and $\ell(s_j, v)$ is the cost per time associated with state s_j and parameters v. This is a form of Bellman equation, a very general concept in control theory and reinforcement learning, related to the Hamilton-Jacobi equation in physics. It is used to find the optimal actions, i.e. how we should choose v at each moment w.r.t the cost associated with each state and parameters $\ell(s_j, v)$. The function $u(s_j, t)$ is the cost-to-go function, it is the cumulative cost obtained starting from state s_j at time t and acting optimally afterwards. The optimal parameters $v(t)$ are computed greedily with each step and guarantee minimum $u(s_j, t)$, this is most easily illustrated by looking Δt into the past. Here the cost-to-go is given by the cost of remaining in the same state $\ell(s_j, t)\Delta t$ and the changes in costs-to-go when transitioning to other states weighted by the probability of making the transition $\Gamma^{(v)}_{s_j \to s_k}$

$$u_j(t - \Delta t) = u_j(t) + \Delta t \min_{\Gamma^{(v)}} \left[\ell(j, v) + \sum_k \Gamma^{(v)}_{j \to k} (u(k, t) - u(j, t)) \right]. \tag{3.69}$$

The cost $\ell(s_j, v)$ introduced by Todorov is of the form

$$\ell(j, v) = q(j) + D_{\mathrm{KL}}(v(\cdot \mid j) \| p(\cdot \mid j)), \tag{3.70}$$

where the first term encodes how undesirable different states are, and the second measures how costly the deviation from passive dynamics is, i.e. the agent (controlled rates) pay a price for reshaping the environment (passive rates). A more rigorous discussion can be found in Appendix A.3. In the case of the imaginary time Schrödinger equation, these terms are

$$q(s_j) = V(s_j)\Delta t, \tag{3.71}$$

and

$$D_{\mathrm{KL}} = \mathop{\mathbb{E}}_{\Sigma_{[0,t]} = k_t \sim \Gamma^{(v)}} \left[\sum_n D_{\mathrm{IS}} \left(\Gamma^{(v)}_{k^{(n)} \to k^{(n+1)}}, \Gamma_{k^{(n)} \to k^{(n+1)}} \right) \right], \tag{3.72}$$

where $\Sigma_{[0,t]}$ is the trajectory of states sampled from controlled dynamics $\Gamma^{(v)}$ which visits states $k^{(n)}$ for $t_{n-1} < t < t_n$ and D_{IS} is the Itakura-Saito divergence, see Appendix A.3.

3.3.4 Loss for discrete state space

Analogous to the derivation in continuous space, we will compare the cost to the Kullback-Liebler divergence between path measures $\mathbb{P}_v$ and $\mathbb{P}_{\mathrm{FK}}$ to see that the optimal rates $\Gamma^{(v)}$ coincide with sampling from the Feynman-Kac measure. Full derivations in A.4.

The logarithm Radon-Nikodym between the controlled rates and Feynman-Kac measure is

$$\log\left(\frac{\mathrm{d}\mathbb{P}_v}{\mathrm{d}\mathbb{P}_{FK}}(k(t))\right) = \tilde{\ell} + \sum_n \log\left(\frac{\Gamma^{(v)}_{k^{(n)}\to k^{(n+1)}}}{\Gamma_{k^{(n)}\to k^{(n+1)}}}\right) - E_0\,T - \log\left(\frac{\varphi(k^{(N)})}{\varphi(k^{(0)})}\right), \tag{3.73}$$

where we have defined $\tilde{\ell}$ as

$$\tilde{\ell} = \int \left[V(k(t)) + \sum_{l\neq k(t)} \left(\Gamma_{k(t)\to l} - \Gamma^{(v)}_{k(t)\to l}\right)\right] \mathrm{d}t. \tag{3.74}$$

The KL divergence can be written as

$$\begin{aligned} D_{\mathrm{KL}}(\mathbb{P}_v \mid \mathbb{P}_{\mathrm{FK}}) = \mathop{\mathbb{E}}_{P_v}\Bigg[\int V(k(t)) + \sum_{l\neq k(t)} \left(\Gamma_{k(t)\to l} - \Gamma^{(v)}_{k(t)\to l}\right) \\ + \Gamma^{(v)}_{k(t)\to l} \log\left(\frac{\Gamma^{(v)}_{k(t)\to l}}{\Gamma_{k(t)\to l}}\right) dt - \log\left(\frac{\varphi(k^{(N)})}{\varphi(k^{(0)})}\right)\Bigg] - E_0\,T, \end{aligned} \tag{3.75}$$

which is the Todorov's cost and vanishes for optimal rates, meaning that finding them is equivalent to finding the Feynman-Kac measured process. One might be tempted to use D_{KL} as an optimisation objective, but this is not possible because its gradient is biased due to the boundary term $\log\left(\frac{\varphi(k^{(N)})}{\varphi(k^{(0)})}\right)$. Hence, we propose two alternatives.

Loss no. 1: Variance of $\tilde{\ell}$

The variance of quantity $\tilde{\ell}$

$$\tilde{\ell} = \log\left(\frac{\mathrm{d}\mathbb{P}_v}{\mathrm{d}\mathbb{P}_{FK}}(k(t))\right) - \sum_n \log\left(\frac{\Gamma^{(v)}_{k^{(n)}\to k^{(n+1)}}}{\Gamma_{k^{(n)}\to k^{(n+1)}}}\right) + E_0\,T + \log\left(\frac{\varphi\left(k^{(N)}\right)}{\varphi\left(k^{(0)}\right)}\right) \tag{3.76}$$

can be used as a loss function. Kolmogorov's criterion tells us that

$$\log\left(\frac{\Gamma^{(v)}_{k^{(n)}\to k^{(n+1)}}}{\Gamma_{k^{(n)}\to k^{(n+1)}}}\right) = \log\left(\frac{\varphi(k^{(n+1)})}{\varphi(k^{(n)})}\right), \tag{3.77}$$

combined with the fact that the log Radon-Nikodym derivative is 0 for correct transition rates $\Gamma^{(v)} = \Gamma'$ means that $\mathrm{Var}[\tilde{\ell}]$ vanishes in that case, moreover it is bounded below by zero making it a suitable objective function.

Loss no. 2: Variance of log Radon-Nikodym derivative with fixed endpoints

While the gradients of $D_{\mathrm{KL}}(\mathbb{P}_v \mid \mathbb{P}_{\mathrm{FK}})$ are not accessible, we can still make use of the logarithm RN derivative in eq. (3.73). It vanishes for optimal rates and while we can treat $E_0 T$ as constant, we have no easy way to approximate the wave function φ in the boundary term which in general does not have zero variance. We can exploit the fact that log RN must be constant on all trajectories and consider a batch of trajectories with fixed endpoints. The variance of the boundary term vanishes if this is the case

$$\operatorname*{Var}_{\substack{\Sigma_{[0,t]}\sim r,\\ \text{with fixed } k^{(0)} \text{ and } k^{(N)}}} \left[\tilde{\ell} + \sum_n \log\left(\frac{\Gamma^{(v)}_{k^{(n)}\to k^{(n+1)}}}{\Gamma_{k^{(n)}\to k^{(n+1)}}} \right) - E_0 T - \log\left(\frac{\varphi(k^{(N)})}{\varphi(k^{(0)})} \right) \right] \underset{\Gamma^{(v)}=\Gamma'}{\longrightarrow} 0. \tag{3.78}$$

The trajectories can be sampled using arbitrary rates r, but for convenience $\Gamma^{(v)}$ can be used. How to obtain a batch poses a technical challenge which is to be solved on a system by system basis, and is discussed in section 4.4. This is the loss we focus on, we refer to it as the logRN loss.

Chapter 4

Methodology

This chapter presents the computational method developed in this thesis. We start by introducing each component of the method separately. We discuss neural networks, automatic differentiation, gradient-based optimisation, and importance sampling. Finally, we present the actual implementation in *JAX*, and discuss computational considerations and intricacies of thereof.

4.1 Neural Networks

4.1.1 The Multilayer Perceptron

The **multilayer perceptron** (MLP), also referred to as **deep neural network** (DNN), is the paradigmatic model of deep learning and serves as a foundation for more advanced models. In essence it is nothing more than a mapping of inputs to outputs

$$f(\mathbf{x};\ \theta) : \mathbb{R}^{\text{in}} \longrightarrow \mathbb{R}^{\text{out}}, \tag{4.1}$$

which is structured in a certain way and depends on parameters θ. The mapping is a composition of vector-valued functions f which are called *layers* of the network, and with each layer we associate variational parameters $\mathbf{w}$, or simply the weights. The MLP can be described with a directed acyclic graph which details the compositions of the layers. The simplest and most common is a chain of compositions, in Fig 4.1b,

$$f_{\text{MLP}} = \left(f^{(n)} \circ f^{(n-1)} \circ \cdots \circ f^{(2)} \circ f^{(1)}\right)(\mathbf{x}), \tag{4.2}$$

where the input $\mathbf{x}$ passes through *hidden* layers before the *output* layer outputs the result. The length of this chain is the *depth* of the network, and the dimensionality of hidden layers

is the *width* of the network. We can interpret each transformation $f^{(i)}$ as consisting of a unit/node/neuron for each input dimension, which is a vector-to-scalar transformation, Fig 4.1a.

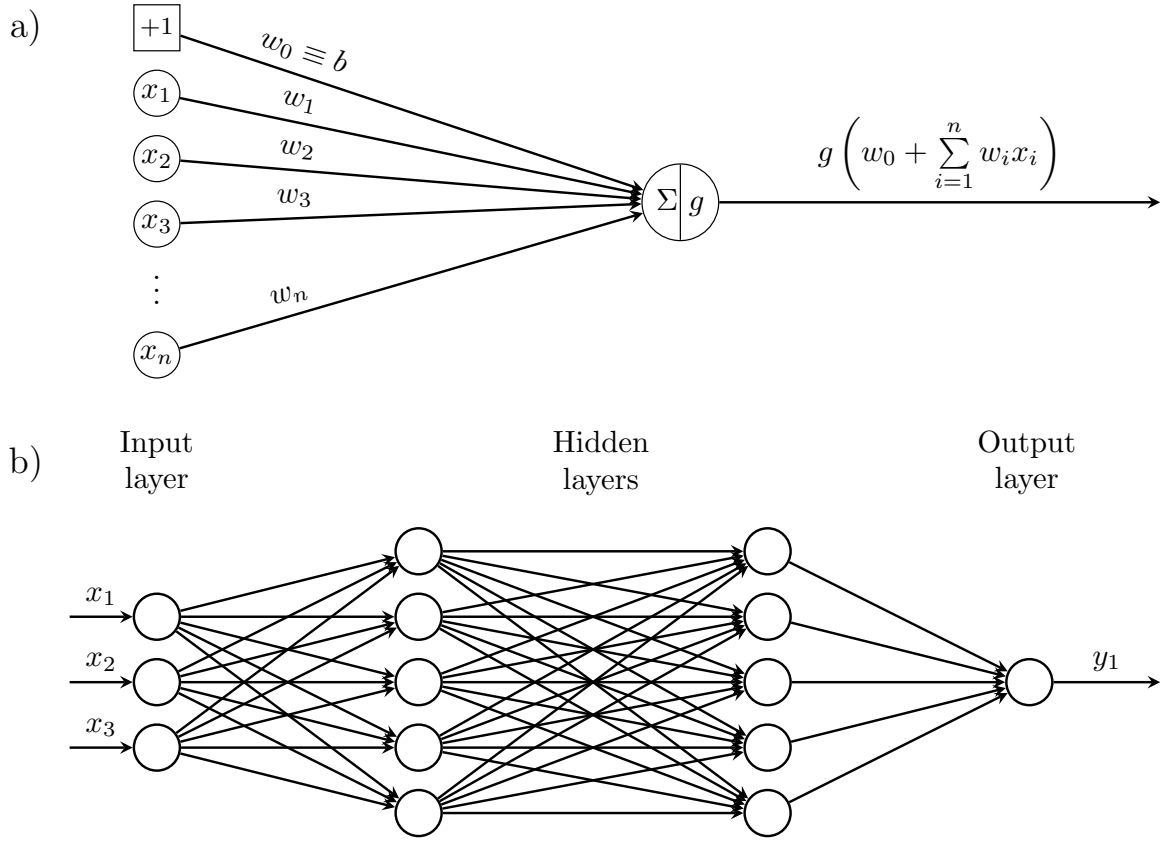


Fig. 4.1 **Multilayer Perceptron.** A node outputs the activation function σ evaluated at the weighted average of outputs from the previous layer and bias (**a**). A MLP with two hidden layers ($f^{(1)} : \mathbb{R}^3 \to \mathbb{R}^4$, $f^{(2)} : \mathbb{R}^4 \to \mathbb{R}^4$, $f^{(3)} : \mathbb{R}^4 \to \mathbb{R}$) in (**b**).

The simplest layer would be a linear one, composed of

$$f(\mathbf{x}; \mathbf{w}, b) = \mathbf{x}^\mathsf{T}\mathbf{w} + b. \tag{4.3}$$

However, a linear neural network famously cannot learn the XOR function [51], and in practice a nonlinearity or *activation* function $g(\cdot)$ is used in each node to bolster the network's representational power

$$f(\mathbf{x}; \mathbf{w}, b) = g\left(\mathbf{x}^\mathsf{T}\mathbf{w} + b\right). \tag{4.4}$$

A variety of activations have been used, namely tanh(·) and the logistic sigmoid $\sigma(\cdot)$, but have since been displaced by the use of the **rectified linear unit** or ReLu(·), which is advantageous for training. For the output layer we will use the *softplus*, which fulfils the requirement of being positive everywhere, Fig. 4.2. A multilayer perceptron is a **universal function approximator**, meaning that it can approximate any Borel measurable function mapping from a finite-dimensional space to another with desired accuracy, if it has at least one hidden layer and sufficient hidden units [48]. While this means that a large enough network will be able do represent the rates, it provides no guarantee that learning the correct rates will be efficient or possible.

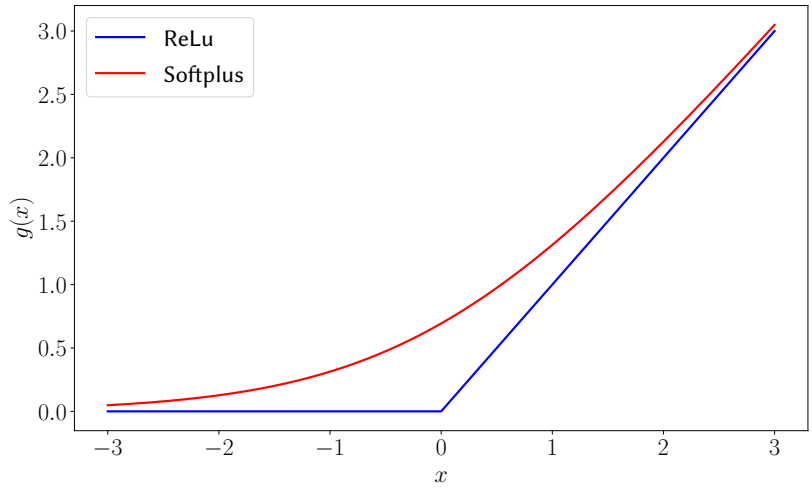


Fig. 4.2 ReLu **and Softmax nonlinearities.**

4.1.2 Convolutional Neural Networks

Convolutional neural networks (CNN) [46] make use of three powerful concepts to increase performance, **sparse interactions**, **parameter sharing** and **equivariant representations** [31]. At the core of a CNN is the convolutional layer, in which the input I is convolved with the **kernel** K as

$$S_{i,j} = (I * K)_{i,j} = \sum_m \sum_n I_{m,n} K_{i-m,j-n}, \tag{4.5}$$

in two dimensions, Fig. 4.3a. Outputs of convolutions are referred to as *feature maps*. In practice the kernel K is much smaller than the input, meaning that each neuron is connected only to a small fraction of neurons in the previous layer. Hence the layers are sparsely connected in contrast to the fully connected layer, see Fig. 4.3c. This decreases the number of weights and operations required. Moreover, the kernel is applied everywhere in the

input, meaning the weights are shared across all connections and need to be learned for the whole input as opposed to every single position in the input. This makes the convolutional layer much more memory efficient than the fully connected layer. Moreover, the nature of convolution in eq. (4.5) means that the convolutional layer is equivariant to translation, i.e. a translation of the input results in the same translation of the output. Altogether, a convolutional layer is a transformation acting on a batch of examples N_b with channels N_c

$$f : \mathbb{R}^{(N_b,N_w,N_h,N_c)} \longrightarrow \mathbb{R}^{(N_b,O_w,O_h,N_c)}. \tag{4.6}$$

The output dimensions also depend on the **stride** (how quick the kernel travels) N_S and **padding** N_P (how much the dimension of input is increased), Fig. 4.3a,

$$O_{\text{out}} = \frac{N_{\text{in}} - N_K + 2N_P}{N_S} + 1. \tag{4.7}$$

Alongside convolutional layers, CNNs often employ *pooling* layers which downsample the input by combining a cluster of neurons into a single one. *Max* and *average* pooling are in common use, the pooling output is the max or average of the cluster respectively. A pooling layer can act globally, on the whole feature map, or locally.

The models we are interested in will be image-to-image networks that preserve the input shape, as the outputs will represent the rates corresponding to adjacent states. In the Ising model, the output at (i,j) is the rate associated with transition $s \longrightarrow s'$ where the spin at (i,j) is flipped.

Periodic CNN

The simplest model we will use is a CNN in which all layers preserve the shape of the input $N_{\text{in}} = O_{\text{out}}$. This means that the input into each layer needs to be padded

$$N_P = \frac{N_K - 1}{2}, \tag{4.8}$$

for $N_S = 1$. Given that the underlying lattice is chosen to be periodic, we use periodic instead of zero padding, see Fig. 4.3b. Hidden units use ReLu and the last layer uses softmax.

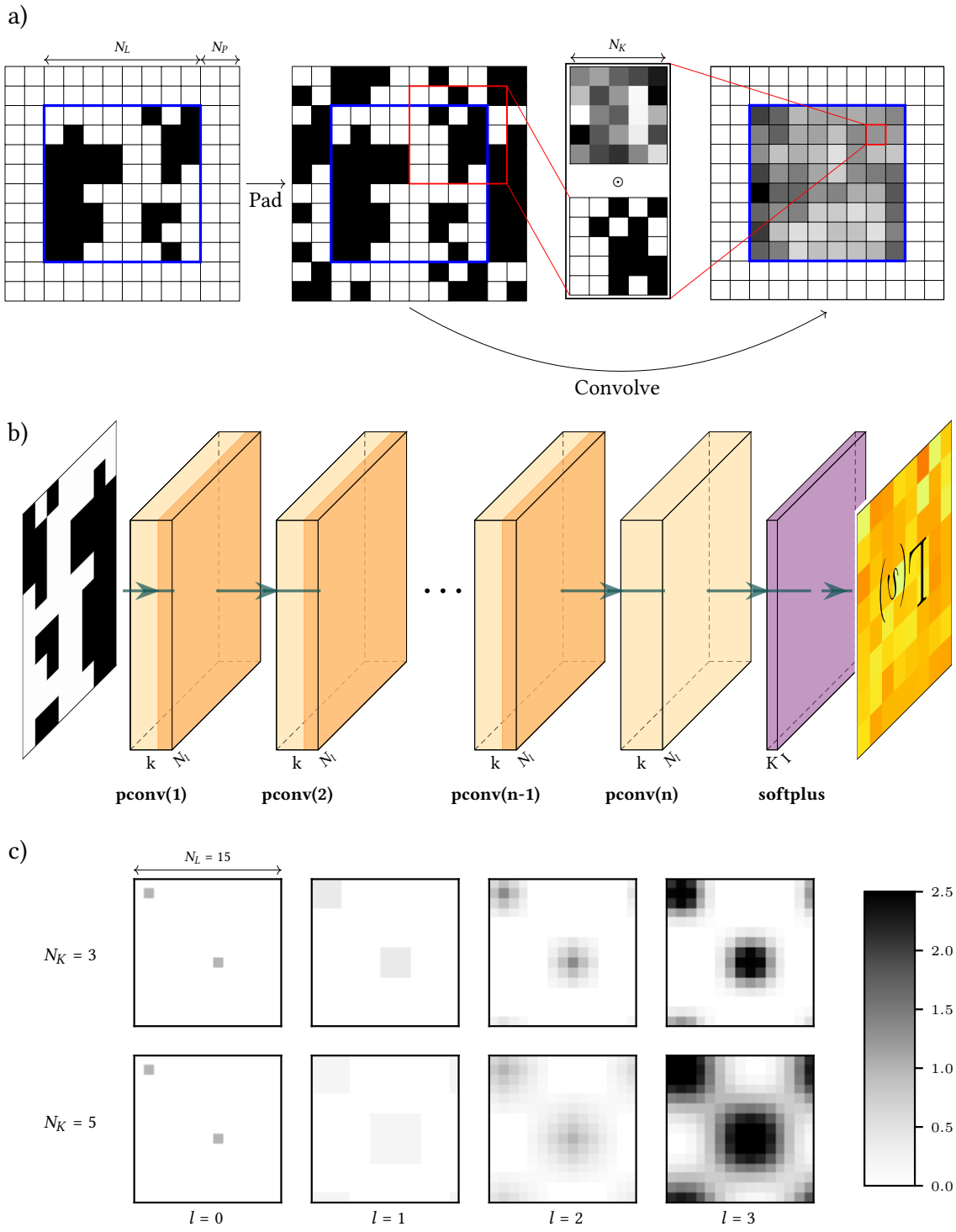


Fig. 4.3 **Periodic CNN.** Each layer consists of periodic padding, and then convolving. The output shape matches the input shape (**top**). The pCNN architecture, takes state s as input and outputs rates $\Gamma(s)$, similar to [30] (**middle**). While the receptive field of a single node is reduced in a CNN, faraway nodes are still connected indirectly after enough layers, how many layers depends on the kernel size and stride. Figure shows successive applications of the convolutional layers for $N_s = 1$ and $N_K = 3$ or $N_K = 5$, (**bottom**).

4.1.3 Group-Equivariant CNN

The basic convolutional layer is translation equivariant, providing an advantage when we expect the same equivariant behaviour between the input and output. Alongside translational symmetry we can take advantage of other symmetries present in the lattice model. Recent work [12, 20] in the field of **geometric** deep learning has shown how to construct layers equivariant to arbitrary group symmetry, in the case of discrete grid-like data referred to as **group-equivariant** convolutional layers. At the core of group convolution is moving the filter using group action (rotation, translation, etc.). Adopting notation of Bronstein [12], we write a group convolution as the inner product of the input x and a filter transformed by group element $\mathfrak{g} \in \mathfrak{G}$ via group representation $\rho(\mathfrak{g})$ as $\rho(\mathfrak{g})\theta_u = \theta_{\mathfrak{g}^{-1}u}$,

$$(x \star \theta)(\mathfrak{g}) = \sum_{u \in \Omega} x_u \rho(\mathfrak{g}) \theta_u. \tag{4.9}$$

We can separate the ordinary convolution from additional group transformations by writing

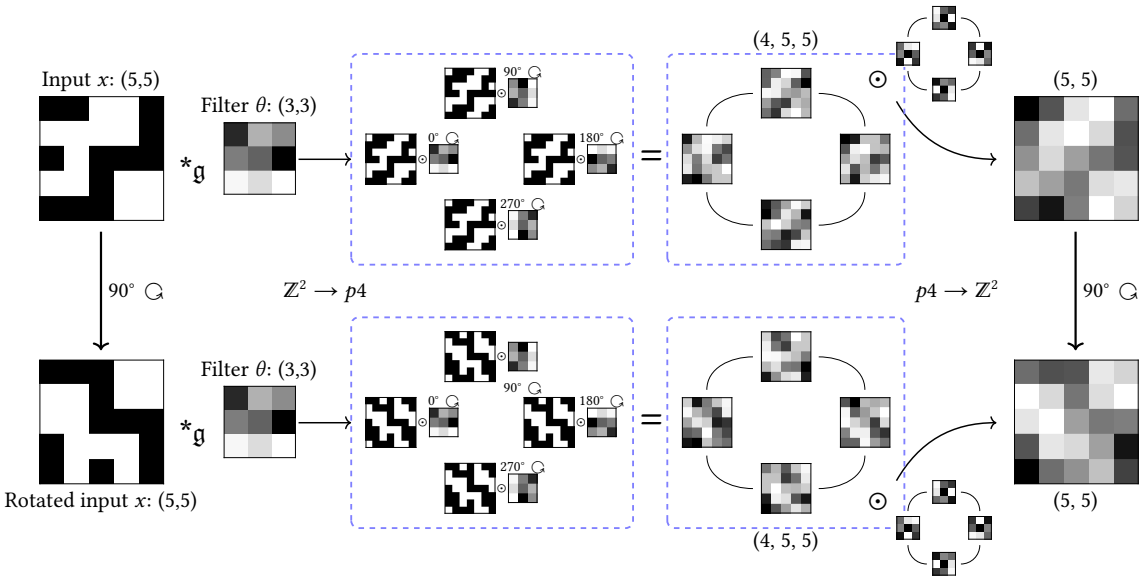


Fig. 4.4 **g-CNN layer.** Comparison of rotated inputs passing through two $p4$ equivariant layers. The first layer $\mathbb{Z}^2 \to p4$ produces a structured output, one convolution per each rotation of the filter θ, the second layer $p4 \to \mathbb{Z}^2$ is an elementwise convolution between the structured output and rotations of the second filter. The layers are equivariant to $\frac{\pi}{2}$ rotation.

a general group element $\mathfrak{g} \in \mathfrak{G}$ as a composition of a translation $\mathfrak{t}$ and rotation $\mathfrak{r}$, $\mathfrak{g} = \mathfrak{t}\mathfrak{r}$, and

using $\rho(\mathfrak{tr}) = \rho(\mathfrak{t})\rho(\mathfrak{r})$

$$(x \star \theta)(\mathfrak{tr}) = \sum_{u \in \Omega} x_u \rho(\mathfrak{t})\rho(\mathfrak{r})\theta_u = \sum_{u \in \Omega} x_u (\rho(\mathfrak{r})\theta)_{u-\mathfrak{t}}, \tag{4.10}$$

This yields a standard convolution with a transformed filter $\rho(\mathfrak{r})\theta$, meaning that we can implement the group-equivariant layer by first transforming the filter and performing convolution with each of these transformations, for details see Fig. 4.4 and [20].

4.2 Gradient-based optimisation

4.2.1 Automatic differentiation

The demand for automatic evaluation of derivatives is today greater than ever, and automatic differentiation has seen wide adoption within the scientific computing and machine learning communities [8, 50, 78], with the development of high profile libraries such as *PyTorch* [58], *TensorFlow* [1], or *JAX* [10]. Not to be confused with numerical or symbolic differentiation, **automatic differentiation** (AD) provides a way to mechanically find derivatives of functions expressed as a computer program, with certain complexity guarantees [59]. While modern implementations employ a variety of tricks, AD has two basic *modi operandi* of calculating partial derivatives of $f(x_1, x_2, \ldots, x_n) : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ at $\mathbf{a}$. We start by representing the computation $y = f(\mathbf{x})$ as an *evaluation trace* of elementary operations, composed of inputs, intermediate values, and outputs:

- **Forward accumulation mode** [84]: In forward mode, when evaluating the partial derivative w.r.t input x_j, we associate each intermediate value v_i with the partial derivative $\frac{\partial v_i}{\partial x_j}$. We then apply the chain rule to each operation in the evaluation trace, producing the derivative trace. A single forward pass, in tandem passing both primals v_i and their tangents $\frac{\partial v_i}{\partial x_j}$ through the trace, produces the output of the function, as well as the desired partial derivative. Forward mode AD produces one *column* of the Jacobian $\mathbf{J}_f$ at each pass, and is suited for functions with $n \ll m$.

- **Reverse accumulation mode** [72]: In reverse mode, we associate each intermediate value v_i with adjoint $\bar{v}_i = \frac{\partial y_i}{\partial v_i}$, partial derivative of the output y_i. Derivatives w.r.t inputs are then evaluated in a two step process. First, a forward pass populates intermediate values v_i and stores their dependencies before the adjoints $\bar{v}_i$ are propagated in reverse, from outputs to inputs. Reverse mode AD produces one *row* of the Jacobian $\mathbf{J}_f$ at each pass, and is suited for functions with $n \gg m$.

AD is most commonly implemented in one of two ways, either by *operator overloading*, abstracting away the derivative part of the calculation, e.g. in *TensorFlow* of *JAX* [1, 10]. Or alternatively via *source code transformation*, where a new function is constructed by altering the source code, see *Zygote* [40].

4.2.2 Optimisation algorithms

Variants of the **stochastic gradient descent** (SGD) are by far the most used optimisation algorithms in ML. Instead of calculating the true gradient of the parameters w.r.t loss, it is estimated using an unbiased estimator which takes the average gradient on a minibatch of examples. SGD introduces only a single hyperparameter, the learning rate α, which determines the size of correction $\theta \longleftarrow \theta - \alpha \nabla_\theta L$ at each iteration. The main advantage of this approach is that the time complexity of an iteration does not depend on the number of examples. This is useful for cases with large, or even infinite in our case, training sets. The first improvement over vanilla SGD is the use of momentum [62], where an exponentially decaying average of past gradients determines the update direction. Second improvement is using an adaptive learning rate, *ADAM* [44] is perhaps the most popular such algorithm.

4.3 Importance Sampling

Ultimately we want to use the optimal learned rates $\Gamma^{(v)}$ to evaluate observables $\hat{O}$ in our model, be it the ground state energy or something else. Recall from section 2.3.1, that we can express observables as a Monte Carlo average of the local operator $\hat{O}_L$. But not all operators are made equal and some are more convenient to evaluate than others. In our implementation it will be helpful to make a distinction between *diagonal* and *off-diagonal* observables [81]. Since the wave functions of stoquastic Hamiltonians are real and positive, we denote $\psi(\boldsymbol{s}) = \sqrt{\rho(\boldsymbol{s})/Z}$, where Z is the normalisation constant. If the observable

$$\hat{O} = \sum_{\boldsymbol{s},\boldsymbol{s}'} O_{\boldsymbol{s}\boldsymbol{s}'} |\boldsymbol{s}\rangle\langle \boldsymbol{s}'|, \tag{4.11}$$

is diagonal $\hat{O}^D_{\boldsymbol{s}\boldsymbol{s}'} = O(\boldsymbol{s})\delta_{\boldsymbol{s}\boldsymbol{s}'}$ in our chosen basis $\{\boldsymbol{s}\}$, then its expectation simplifies to

$$\langle \hat{O^D} \rangle = \sum_{\boldsymbol{s},\boldsymbol{s}'} \psi^*(\boldsymbol{s})\psi(\boldsymbol{s}')O_{\boldsymbol{s}\boldsymbol{s}'} = \frac{1}{Z}\sum_{\boldsymbol{s}} \rho(\boldsymbol{s})O(\boldsymbol{s}) \approx \frac{1}{M}\sum_{k=1}^{M} O(\boldsymbol{s}_k), \tag{4.12}$$

which is very easy to evaluate given the MC samples. Alternatively for off-diagonal observables the expectation becomes

$$\left\langle \hat{O}^{OD} \right\rangle = \sum_{\boldsymbol{s},\boldsymbol{s}'} \psi^*(\boldsymbol{s})\psi(\boldsymbol{s}')O_{\boldsymbol{s}\boldsymbol{s}'} = \frac{1}{\sum_{\boldsymbol{s}} |\psi_{\boldsymbol{\lambda}}(\boldsymbol{s})|^2} \sum_{\boldsymbol{s}\boldsymbol{s}'} \psi^*_{\boldsymbol{\lambda}}(\boldsymbol{s})\psi_{\boldsymbol{\lambda}}(\boldsymbol{s}')O_{\boldsymbol{s}\boldsymbol{s}'} \approx \frac{1}{M}\sum_{k=1}^{M} O^L(\boldsymbol{s}_k), \tag{4.13}$$

where the local operator has the form

$$O^L_{\boldsymbol{s}_k} = \sum_{\boldsymbol{s}'} \frac{\psi(\boldsymbol{s}')}{\psi(\boldsymbol{s}_k)} O_{\boldsymbol{s}_k\boldsymbol{s}'} = \sum_{\boldsymbol{s}'} \sqrt{\frac{\rho(\boldsymbol{s}')}{\rho(\boldsymbol{s}_k)}} O_{\boldsymbol{s}_k\boldsymbol{s}'}. \tag{4.14}$$

The local estimate of $\hat{O}$ is efficient to evaluate if the matrix representation $O_{\boldsymbol{s}_k\boldsymbol{s}'}$ is sparse in the reference basis. When working in the z-spin basis, an example of a diagonal observable would be the average longitudinal magnetisation per spin

$$\langle \hat{\sigma}_z \rangle = \sum_i \frac{\langle \hat{\sigma}^z_i \rangle}{N}. \tag{4.15}$$

On the other hand, transverse magnetisation $\langle \hat{\sigma}^x_j \rangle$ for spin j is off-diagonal[1]

$$\left\langle \boldsymbol{s} \middle| \hat{\sigma}^x_j \middle| \boldsymbol{s}' \right\rangle = \delta_{s'_j, 1-s_j} \prod_{i\neq j} \delta_{s'_i, s_j}, \tag{4.16}$$

thus the local operator for $\langle \hat{\sigma}^x_j \rangle$ is sparse with only one term in the sum

$$\left(\sigma^x_j\right)^L = \frac{\psi\left(s_1,\ldots,1-s_j,\ldots,s_N\right)}{\psi\left(s_1,\ldots,s_j,\ldots,s_N\right)} = \sqrt{\frac{\rho\left(s_1,\ldots,1-s_j,\ldots,s_N\right)}{\rho\left(s_1,\ldots,s_j,\ldots,s_N\right)}}. \tag{4.17}$$

4.3.1 MCMC and the Metropolis-Hastings Algorithm

The integration technique from the previous section relies on our ability to obtain samples from an unnormalised probability distribution ρ. **Markov Chain Monte Carlo** (MCMC) avoids sampling directly from ρ by constructing a Markov chain, whose stationary distribution π is the same as ρ. Propagating this chain produces a sequence of samples from the desired distribution. For the process to have a unique stationary distribution, it must be *ergodic* and it must obey *detailed balance*

$$P_{\mathbf{s}\to\mathbf{s}'}\rho(\mathbf{s}) = P_{\mathbf{s}'\to\mathbf{s}}\rho(\mathbf{s}'), \tag{4.18}$$

[1] $\boldsymbol{s}_i = (s_1, s_2, \ldots, s_i, \ldots, s_n)$, s_i is either 1 or 0.

where P are transition probabilities. The equivalent condition in continuous time uses the rate matrix Γ

$$\Gamma_{\mathbf{s}\to\mathbf{s}'}\rho(\mathbf{s}) = \Gamma_{\mathbf{s}'\to\mathbf{s}}\rho(\mathbf{s}'). \tag{4.19}$$

The *correct* transition probabilities P (or rates Γ) are not known, instead each move is proposed using a trial move probability $T_{\mathbf{s}\to\mathbf{s}'}$ and accepted using the Metropolis-Hastings acceptance probability $A_{\mathbf{s}\to\mathbf{s}'}$ which guarantees detailed balance is met

$$A_{\mathbf{s}\to\mathbf{s}'} = \min\left(1, \frac{T_{\mathbf{s}'\to\mathbf{s}}\rho(\mathbf{s}')}{T_{\mathbf{s}\to\mathbf{s}'}\rho(\mathbf{s})}\right). \tag{4.20}$$

Thus to sample from any probability distribution we only need the ability to calculate ratios $\frac{\rho(s')}{\rho(s)}$ and to sample from a trial transition probability $\mathrm{T}_{\mathbf{s}\to\mathbf{s}'}$. The efficiency of the algorithm depends on the amount of trial moves that we reject and accepting every trial move would mean **optimal importance sampling**. All moves are accepted only when $T = P$, which would just mean sampling from P directly.

In practice we are interested in sampling from the ground state ψ_0, and we can improve upon the standard method of using uniform transition probabilities for all adjacent states, by recalling that sampling from the Feynman-Kac $\mathbb{P}_{\mathrm{FK}}$ measure is equivalent to sampling from the ground state. Thus it is possible to achieve optimal importance sampling by finding the rates Γ' that correspond to the Feynman-Kac measure, which is precisely what we discussed in section 3.3. This is possible in theory, but our variational approximation of the rates $\Gamma^\theta \approx \Gamma'$ is not exact in the same way that a trial wave function is not, and we are really sampling from the stationary distribution of Γ^θ. The actual sampling of the chain can be done in discrete time, using

$$T_{s\to s'} = \frac{\Gamma^\theta_{s\to s'}}{\sum_{s'\neq s}\Gamma^\theta_{s\to s'}} \tag{4.21}$$

and a Metropolis correction step[2] with $\frac{\rho(s')}{\rho(s)} = \frac{\Gamma_{s\to s'}}{\Gamma_{s'\to s}}$. Alternatively it can be done in continuous time, where the stationary distribution $\rho = [\tilde{\rho}_1, \tilde{\rho}_2, \tilde{\rho}_3, \ldots]$, where $\tilde{\rho}_i$ is the fraction of time spent in state i.

4.4 Software implementation details

The main developing principles for the implementation are flexibility in the choice of variational model, hyperparameters, or stoquastic Hamiltonian itself, and extendability,

[2] Throwing away holding times τ_i biases the chain.

the ability to add new Hamiltonians and models with relative ease. Additionally, the implementation needs to be computationally efficient with its most demanding parts running on the GPU. This sort of dichotomy between flexibility and speed is aided by modern ML frameworks. We work in *JAX* [10] which extends *numpy* with *function transformations*, provides automatic differentiation capabilities and uses just-in-time[3] (jit) compilation to improve performance. The design of the implemented software is shown in Fig. 4.4. Since the workflow of the method is split between training and sampling, so is the software.

The **input layer** includes the implementation of a Stoquastic Hamiltonian, which must include methods for calculating the passive rates, potential, adjacent states, and for transforming one adjacent state into another. The variational module must be a subclass of *flax.linen.Module* and returns $\Gamma_{s\to s'}$ for input s. All parameters are taken as an *.ml_collections* dictionary.

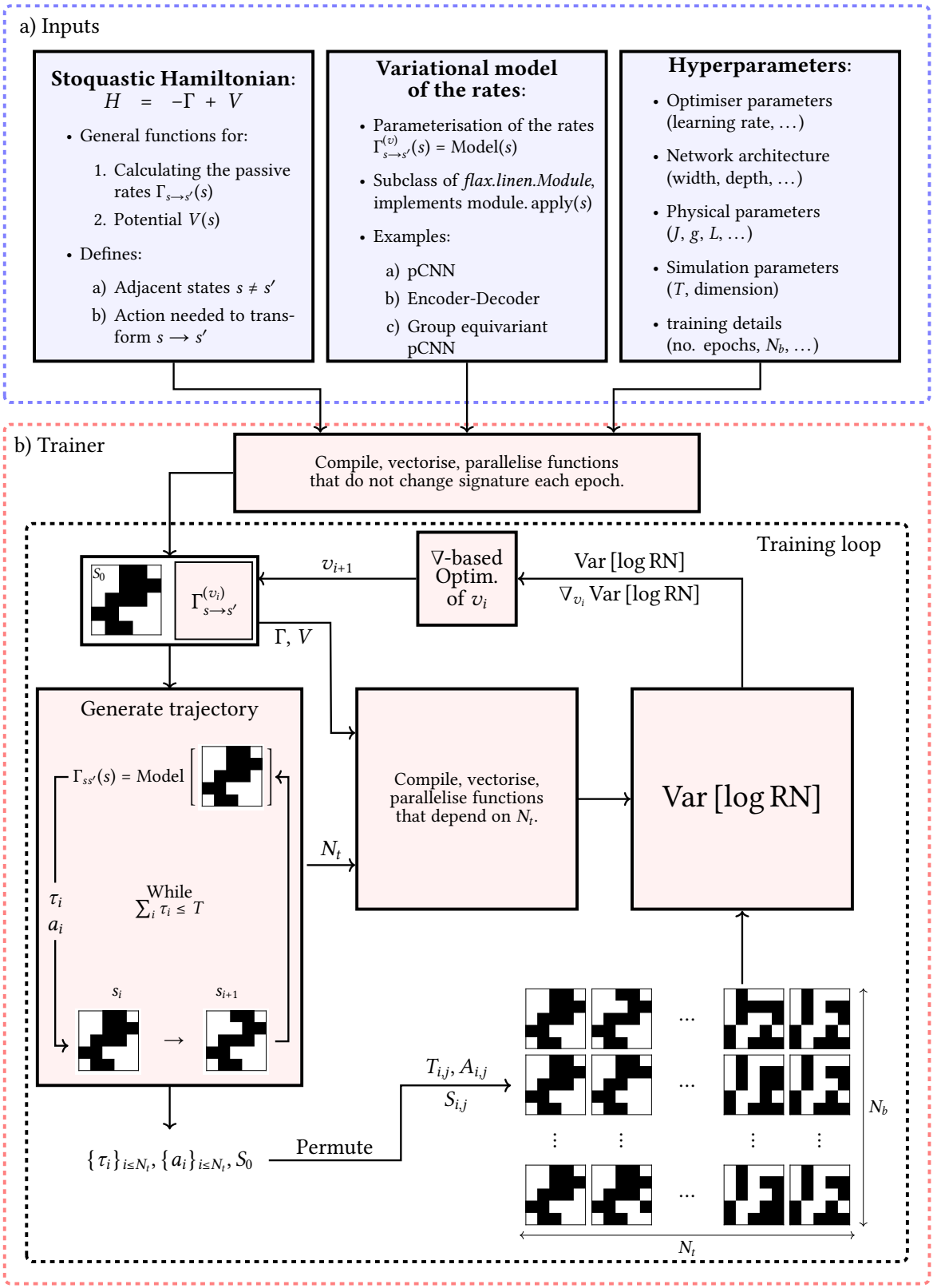
The **training layer**, takes the input functions and partially compiles them for the input parameters before performing training. The most important part of the training procedure is the generation of a batch of trajectories. This can be performed in a number of ways, so long as all trajectories have constant T and fixed endpoints. The generation is system dependent, and we propose the following three approaches for the TFIM model, where spin flips are independent.

i) **Permute:** This is the simplest batch generation technique. First a single trajectory is obtained and is stored as holding times τ_i and actions a_i, along with the initial state S_0. Then the times and actions are permuted into a batch of N_b permutations. The choice of initial trajectory is arbitrary, current implementation supports sampling it from the latest rates $\Gamma^{v_i}_{s\to s'}$, passive rates, or the initial rates $\Gamma^{v_0}_{s\to s'}$. The problem with *permute* batch generation is that the number of actions is constant, and this may pose problems as discussed in 5.1.1.

ii) **Split:** Batch generation using the *split* method also relies on sampling an initial trajectory before modifying it. Instead of only permuting the original, we now append two flips of a randomly chosen spin to the trajectory before shuffling. This has no bearing on the endpoints of the trajectory but changes its length[4]. The times can be resampled or a random holding time can be split among the new visited states, so that all trajectories in the batch have constant T.

[3] It uses the *Accelerated Linear Algebra* XLA compiler.

[4] Length refers to number of spin flips not T.


a) Inputs
Stoquastic Hamiltonian:
$H = -\Gamma + V$
• General functions for:
1. Calculating the passive rates $\Gamma_{s\to s'}(s)$
2. Potential $V(s)$
• Defines:
a) Adjacent states $s \neq s'$
b) Action needed to transform $s \to s'$
Variational model of the rates:
• Parameterisation of the rates $\Gamma^{(v)}_{s\to s'}(s)$ = Model(s)
• Subclass of *flax.linen.Module*, implements module. apply(s)
• Examples:
a) pCNN
b) Encoder-Decoder
c) Group equivariant pCNN
Hyperparameters:
• Optimiser parameters (learning rate, …)
• Network architecture (width, depth, …)
• Physical parameters (J, g, L, …)
• Simulation parameters (T, dimension)
• training details (no. epochs, N_b, …)
b) Trainer
Compile, vectorise, parallelise functions that do not change signature each epoch.
Training loop
S_0
$\Gamma^{(v_i)}_{s\to s'}$
v_{i+1}
∇-based Optim. of v_i
Var [log RN]
∇_{v_i} Var [log RN]
Γ, V
Generate trajectory
$\Gamma_{ss'}(s)$ = Model
τ_i
a_i
While $\sum_i \tau_i \leq T$
s_i
s_{i+1}
N_t
Compile, vectorise, parallelise functions that depend on N_t.
Var [log RN]
$T_{i,j}, A_{i,j}$
$S_{i,j}$
$\{\tau_i\}_{i\leq N_t}, \{a_i\}_{i\leq N_t}, S_0$
Permute
N_b
N_t

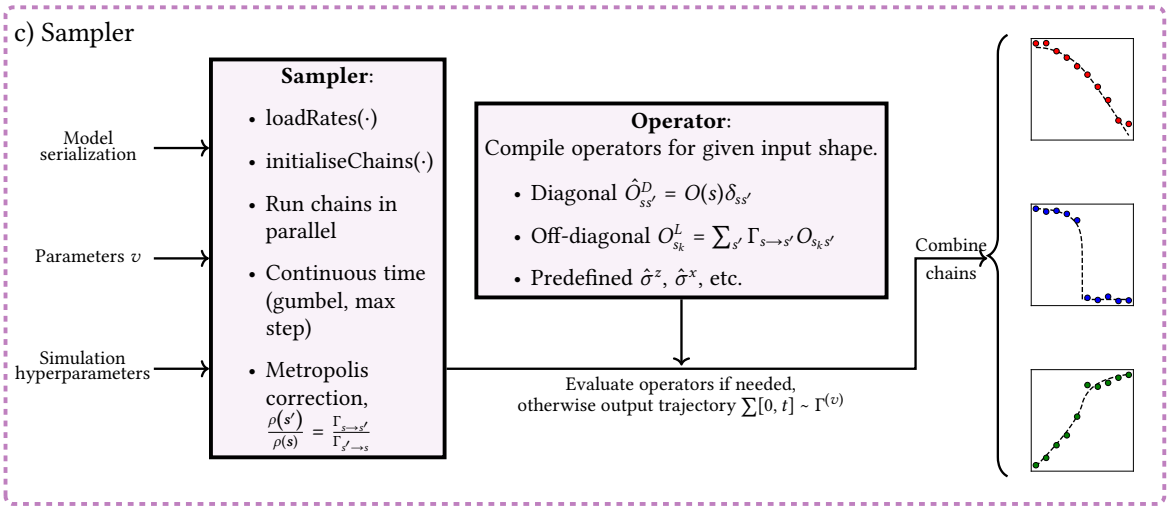


Fig. 4.4 **Software implementation details.** The software implementation in JAX is divided into three layers. The input layer (**a**) specifies the Hamiltonian for which we are learning the rates, the model used to represent the rates and training hyperparameters. The trainer layer (**b**) specialises the general functions for the system at hand and training details (batch size, time, etc.), before training the rates and outputing a sampler object. The sampler object (**c**) can then be used to perform importance sampling using the trained rates in either discrete or continuous time, if operators are provided observables can be evaluated while sampling a trajectory.

iii) **Construct:** The *construct* method does not sample any trajectory. It exploits the fact that if we assign each spin an odd or even number of flips, we obtain a family of trajectories with fixed endpoints. We first randomly choose N_a spins, which will be flipped during the trajectory. Of these spins, we then mark N_e spins to be even and the rest to be odd. To obtain a batch of trajectories with variable length, we assign different odd numbers of flips to odd spins, and analogous to even flips. The times are sampled independently from an exponential distribution and normalised to add up to desired T. The benefit of this method is that we can have trajectories of very different length in the same batch.

The gradient of the $\log RN$ loss is then evaluated using an MC estimate and the parameters updated accordingly, the implementation only supports the *ADAM* optimiser.

The **sampling layer** is essentially a wrapper that provides a friendly way of working with the learned rates. It implements functionality for sampling with rates in parallel, either in discrete or continuous time, as well as evaluation diagonal and off-diagonal observables. The sampling from a CTMC can be done with one of two implemented methods

i) **Max step (competing exponentials):** For all adjacent states sample a holding time $\tau_i \sim \text{Exp}(\lambda_i)$, then move into the state with the minimum time τ_i.

ii) **Gumbel step:** The minimum of n independent exponential distributions $\{\lambda_1, \lambda_2, \cdots, \lambda_n\}$ is also exponentially distributed with rate $\lambda = \sum_i \lambda_i$, its minimizer is distributed according to a multinomial distribution with $\pi_i = \frac{\lambda_i}{\lambda_1 + \cdots + \lambda_n}$ and independent of time. We first sample a holding time τ from the exponential distribution and then the index from the multinomial.

Chapter 5

Experiments and Results

In the penultimate chapter we present results obtained from the methodology and software presented in Chapter 4. We begin by discussing the training of rates $\Gamma^{(v)}$ and problems that we run into, before turning our attention to sampling with the obtained rates.

5.1 Training the rates

5.1.1 The elusive timescale λ

An intuitive explanation of the Todorov loss introduced in section 3.3.3, is in terms of penalisation for deviations away from the passive dynamics of the system, i.e. an agent has full control over the system but pays for straying far from the passive dynamics. Our method in its most basic form, using the *latest* variational rates $\Gamma^{(v)}_{s\to s'}$ to sample and using *permute* batch generation, seems to completely disregard this penalty for straying far away from the passive dynamics when applied to the TFIM. Optimised rates $\Gamma^{(v)}_{*}$ obtained with this method vary depending on the initialisation, but the training itself is not sensitive to it. It is robust in the sense that low loss is obtained irrespective of the initialisation, barring vary bad hyperparameter settings, but the rates $\Gamma^{(v_i)}_{*}$ corresponding to these low losses are not the same. At first glance it appears that there are many *local optima* for the rates, or alternatively, that there is an invariance in the loss and many rates are optimal. The latter turns out to be true. We refer to this problem as the inability to learn the correct **timescale** λ. This issue is illustrated in Fig. 5.1 which shows the number of spin flips in the sampled trajectory throughout training for three initialisations, along with the distributions of holding times $P(\tau) \sim \lambda e^{-\lambda x}$ of the final learned rates. If we assume that the holding times are approximately exponentially distributed[1] and compare the rate constants λ for different

[1] In each state the holding time is exponentially distributed, see section 3.1.6.

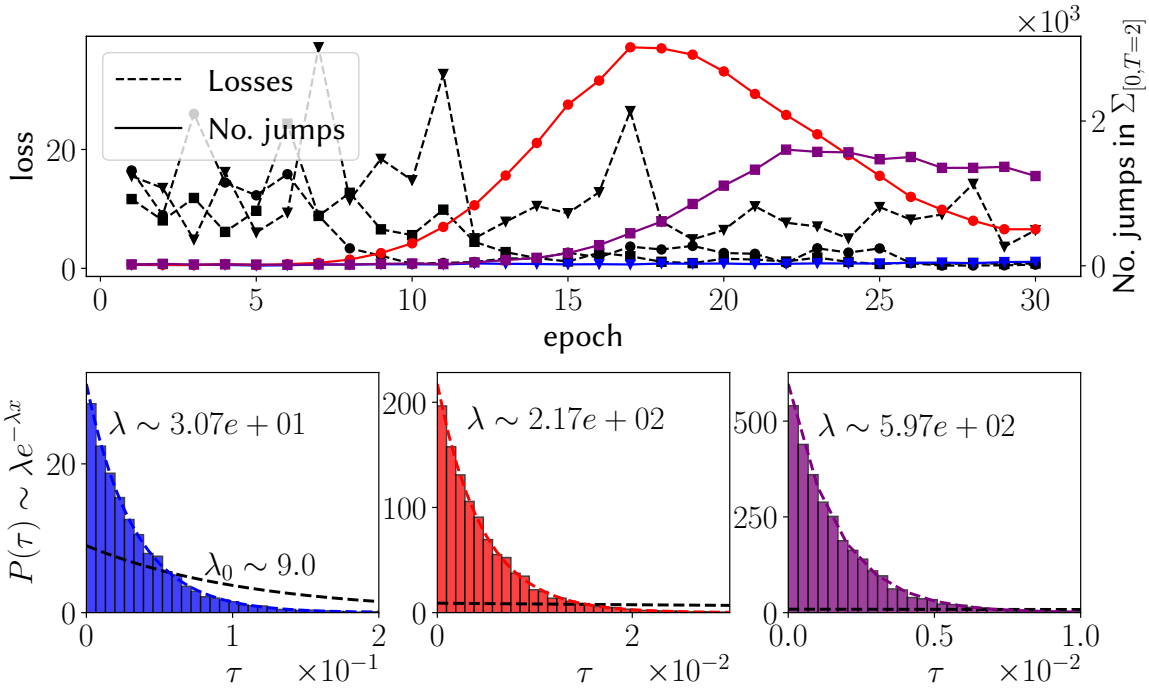


Fig. 5.1 **The inability of the method to learn the correct timescale λ.** The logRN + *permute* batch, fails to learn the correct timescale of the TFIM model. Figure shows the discrepancy between three learned rates in 1D ($L = 8, J = 1.0, g = 1.0$). We see that the number of flips in the original sampled trajectory varies wildly throughout training, and is not related to the loss (**top**). All three initialisations reach low loss, but corresponding learned rates vary in scale, as exemplified by the distribution of holding times τ_i of each CTMC (**bottom**). The rate parameter λ can be orders of magnitude different from the passive rates λ_0 (black dashed line).

optimal rates, we see that they can be orders of magnitude different. Yet the obtained rates do appear to have something in common, the similarity can be seen in Fig. 5.2, which depicts a pair of optimized rates for 1d-TFIM ($L = 6$) and the absolute difference of their *jump chains*. The obtained rates seem to share jump chains $T^{(v)}_{s_i \to -s_i}$ and differ in holding time distributions. It is important to note, that this jump chain is not necessarily the correct jump chain, but it merely smooths out the logRN values in this type of batch to achieve low loss. Why is this the case?

The issue in learning the timescale is not in the logarithm Radon-Nikodym loss, but in the way we generate batches. Moreover, this issue applies to a wider range of models than just the TFIM. For any model which has the following properties:

1. The passive rates are constant and the same for all adjacent states $\Gamma_{s \to s'} = g$
2. The number of adjacent states N_a is independent for any state s

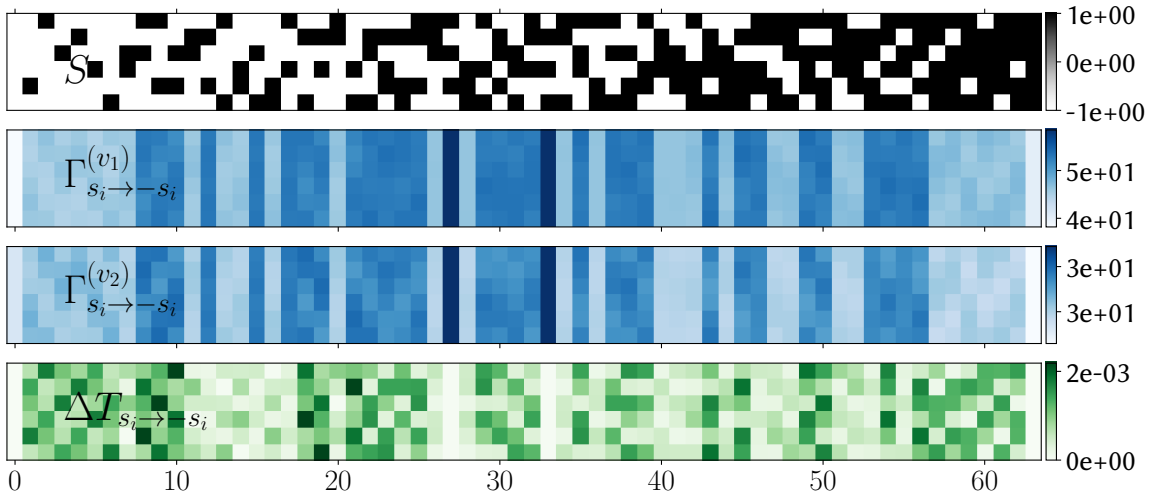


Fig. 5.2 **Structure of rates in 1d-TFIM.** Figure displays the underlying *structure* learned in the 1d-TFIM ($L = 6, J = 1.0, h = 1.0$) using the simple permutation batch. The optimal rates $\Gamma^{(v_1)}_{s_i \to -s_i}$, $\Gamma^{(v_2)}_{s_i \to -s_i}$ obtained with two initialisations (**middle**) for each configuration (**top**), look very similar. The jump matrices $T^{(v_1)}_{s_i \to -s_i}$, $T^{(v_2)}_{s_i \to -s_i}$ corresponding to the learned CTMCs are nearly the same (**bottom**), with maximum absolute difference being $\Delta_{\max} T_{s_i \to -s_i} \approx 2 \cdot 10^{-3}$.

it the logRN loss is **detached** from the passive rates g, if the variance is calculated from a batch of trajectories with same time T and number of actions N_{actions}. Detached in this context means that the passive rates g have no effect on the variance and hence on the optimisation of the rates. This can be seen by inserting

$$V(\mathrm{s}) = -\sum_{s \neq s'} \Gamma_{s \to s'} + H_{ss} = -gN_a + H_{ss} \quad \text{and} \quad \Gamma_{s \to s'} = g \tag{5.1}$$

into

$$\begin{aligned} \mathrm{Var}\Bigg[\int \Bigg(V(k(t)) + \sum_{l \neq k(t)} \left(\Gamma_{k(t) \to l} - \Gamma^{(v)}_{k(t) \to l} \right) \Bigg) \mathrm{d}t \\ + \sum_n \log \left(\frac{\Gamma^{(v)}_{k^{(n)} \to k^{(n+1)}}}{\Gamma_{k^{(n)} \to k^{(n+1)}}} \right) - E_0 T - \log \left(\frac{\varphi(k^{(N)})}{\varphi(k^{(0)})} \right) \Bigg], \end{aligned} \tag{5.2}$$

to obtain

$$\begin{aligned} \mathrm{Var}\left[\int \left(H_{ss} - \sum_{l \neq k(t)} \Gamma^{(v)}_{k(t) \to l} \right) \mathrm{d}t + \sum_n \log \left(\Gamma^{(v)}_{k^{(n)} \to k^{(n+1)}} \right) \right] \\ - E_0 T - \log \left(\frac{\varphi(k^{(N)})}{\varphi(k^{(0)})} \right) - gN_a T + gN_a T - N_{\text{actions}} \log(g). \end{aligned} \tag{5.3}$$

The terms in red contain the passive rates, and can be taken out of Var[·]. The TFIM obeys both listed properties, and a *permute* batch has constant T and N_{actions}, meaning that we find ourselves in a regime where the correct timescale λ **cannot** be learned.

5.1.2 Alternative Batch generation

Unlearnable timescale for the *permute* batch puts us in a precarious situation. The only way forward in models with the timescale learning issue is to work with trajectories of different lengths N_{actions}, fixed endpoints, and fixed time[2] T. This is computationally disadvantageous, because we can no longer evaluate log RN on the whole batch in parallel. Moreover, the generation of the batch becomes difficult to implement and may become a bottleneck itself.

We test the *construct* batch, an alternative batch generation technique described in section 4.4. Fig. 5.3 displays the training loss for the TFIM with different values of h when

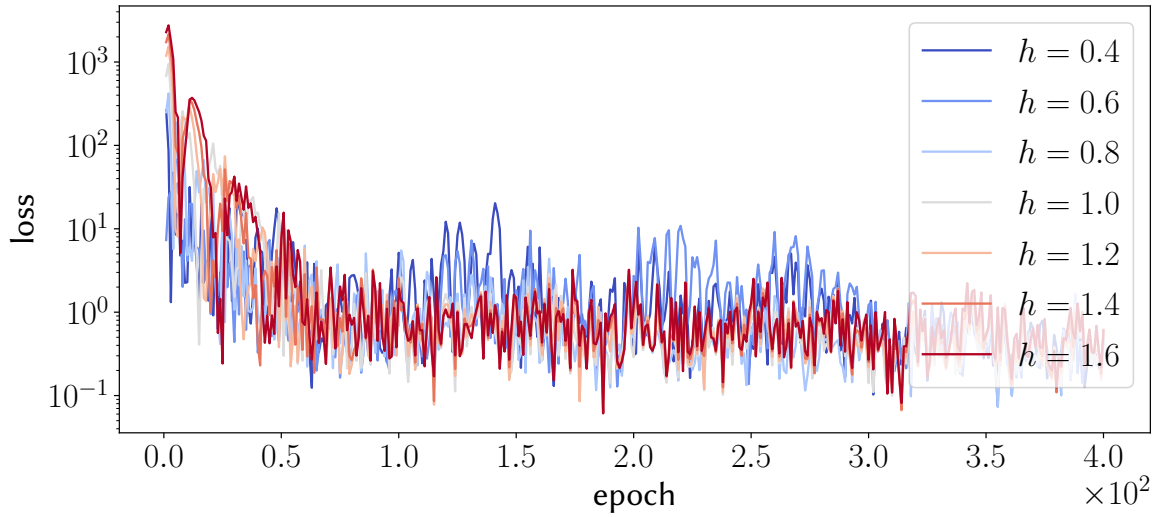


Fig. 5.3 **Rate training in 1d-TFIM using the construct method**. Optimisation with the *construct* batch method, converges for different values of parameter h, moreover the obtained rates are not independent of h. Solved with $L = 6, N_a = 4, N_e = 2, T = 1$.

using the *construct* method for batch generation. Two things are apparent when optimising this new loss. First, it is even more volatile and increasing the batch size helps less than with the *permute* batch. Second, the optimisation becomes more difficult when h is smaller. It is also clear, that this loss now has the capability to scale the outputs of the model throughout optimisation, and at least for small times T the timescale of the model is close to the passive

[2] Otherwise we cannot neglect $E_0 T$ term.

timescale λ_0. This is clearly seen in Fig. 5.4, where distributions of holding times are shown for a range of optimisation settings.

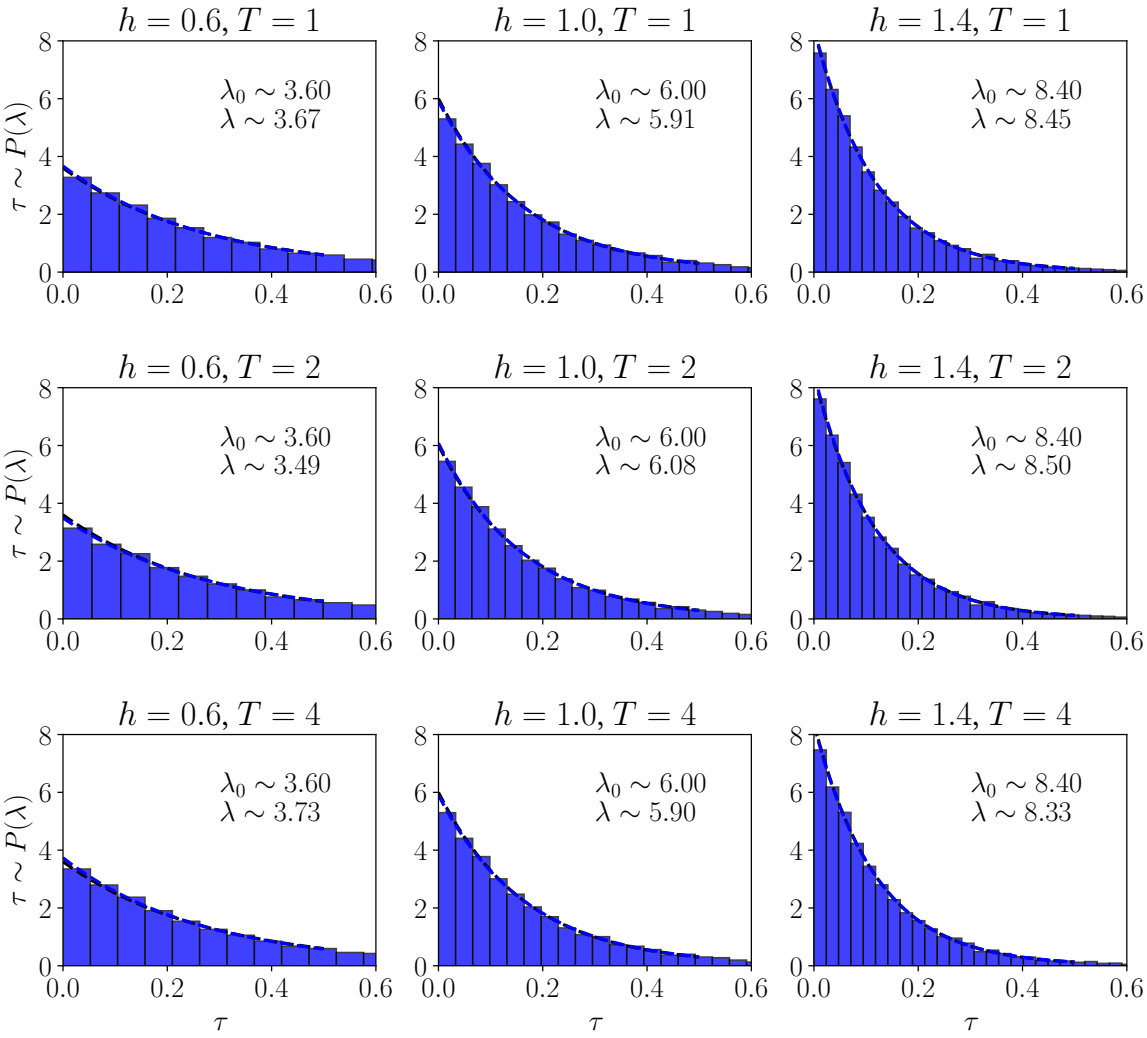


Fig. 5.4 **Distributions of holding times τ in the 1d-TFIM with *construct* batch.** The logRN loss is no longer decoupled from the rates, we see that the timescale λ is learned by the model and is comparable to the passive rates $\lambda_0 = hL$ for range of setups (T, h, at $L = 6$).

However, taking a look at the optimised rates for a range of values of h is worrying, shown in Fig. 5.5. The structure of the rates does not change much when varying parameter h. Even though the system undergoes a phase transition between $h = 0.4$ and $h = 1.6$ this is not evident from the obtained transition probabilities. Moreover, while the optimal rates scale correctly with h, they are nearly uniform, meaning that the distribution has minimal preference about which spin to flip, regardless of h and state. This is also exemplified when

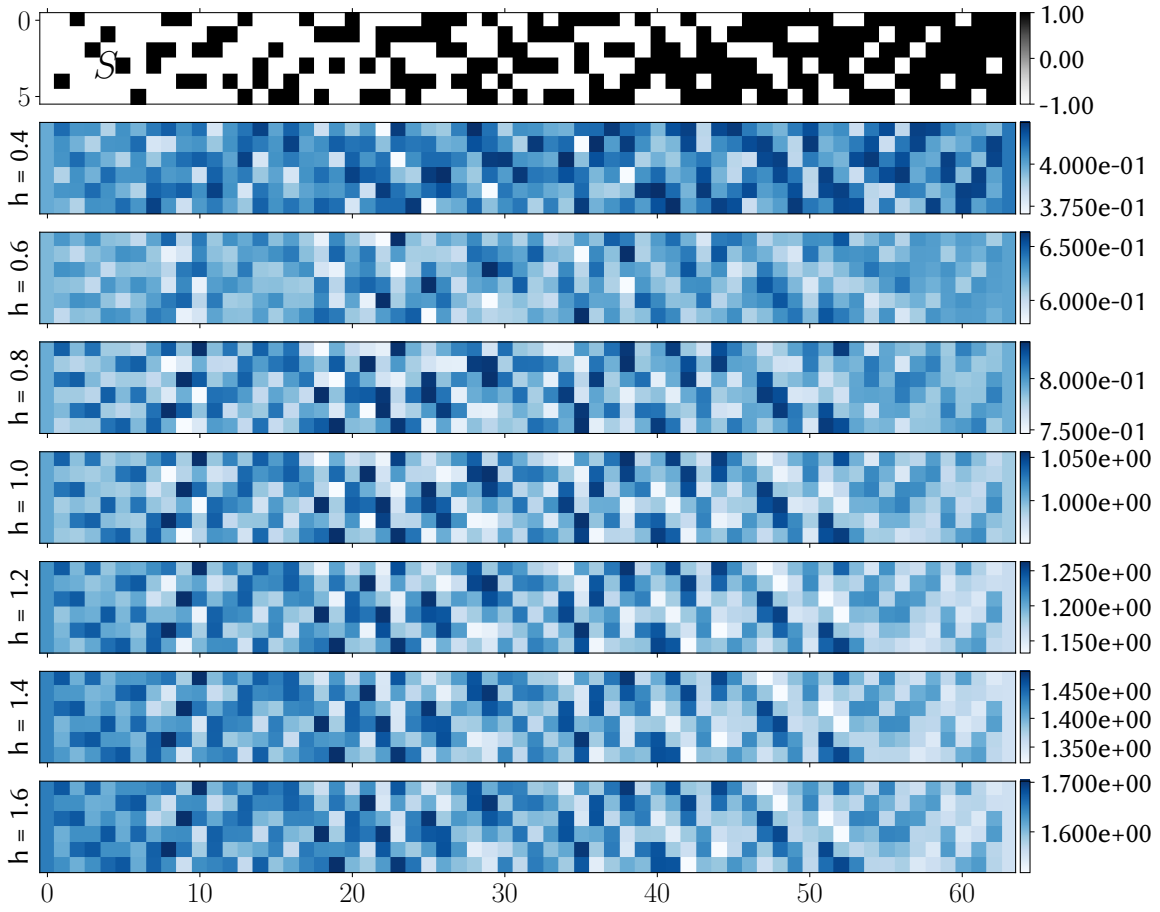


Fig. 5.5 **Learned rates of the 1-dTFIM with the *construct* batch.** The optimised rates for the 1d-TFIM scale with h, but are nearly uniform.

using the rates to perform sampling, the results are remarkably similar to using passive rates, see 5.2.

For a comparison of jump chains T that belong to the optimised rates at different h, refer to Fig. 5.8. And for a comparison between jump chains obtained with different batch construction methods, see Fig. 5.7. While the absolute difference between jump matrices in either of the two cases may seem small, this does not speak to any hidden structure uncovered by the loss, as it is merely a consequence of the jump matrices being nearly uniform. Most jump probabilities fall in the range $T_{i\to j} \in (0.16, 0.17)$ which is remarkably close to $\frac{1}{6} \approx 1.66$.

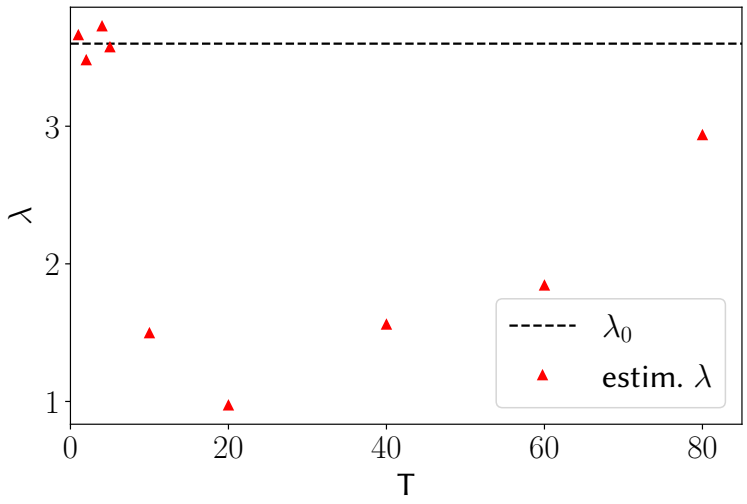


Fig. 5.6 **Timescale gap for the *construct* batch.** When optimising the rates $\Gamma^{(v)}$ using the logRN loss and *construct* batch, we find ourselves in one of two learning regimes depending on the simulation time T at constant h and constant range $(N_a^{(\min)}, N_a^{(\max)})$. For small times T the rates are forced towards the passive rates but after a certain T_{treshold} the timescale has a hard time converging. Figure shows example for $h = 0.6$ and $N_a^{(\max-\min)} = 60$.

While training the rates using the *construct* batch method with different parameters T, h and N_b, it is qualitatively evident that we may find ourselves in one of two training regimes. We have either quick convergence with optimised rates very close to the passive dynamics, such as in Fig. 5.3, or we have trouble converging at all and the optimised rates stray far away from the passive dynamics. Neither of the two regimes leads to the right solution. The "phase transition" from one learning regime to another occurs when we increase the simulation time T and do not change N_b or h, see Fig. 5.6. Let us examine the loss again,

$$\text{Var}\left[\int\left(H_{ss} - \sum_{l\neq k(t)}\Gamma^{(v)}_{k(t)\to l}\right)\mathrm{d}t + \sum_n \log\left(\Gamma^{(v)}_{k^{(n)}\to k^{(n+1)}}\right) - N_{\text{actions}}\log(g)\right], \tag{5.4}$$

where we have purposely left out the terms that cancel out or are constant over all trajectories in the batch. The third term, is the only term tying the variance to the passive rates. In any batch there will be two trajectories with the maximum difference in length $N_a^{(\max-\min)}$. This puts a bound on how much the third term can vary across trajectories in a batch. If we increase the time T, the variability of first term grows proportionally, while it stays constant for the other two terms. This means that at a certain time T the majority contribution will be from the first term. We find ourselves in a similar situation to using the *permute* batch, where the passive terms do not matter at all and the timescale is impossible

to learn[3]. Unfortunately this same argument can be applied to the *split* batch, which may be limited in this same way.

A way around this may be to have $N_a^{(\max-\min)}$ vary from epoch to epoch, but this introduces additional computational cost, as well as additional stochasticity into the loss. Varying the time T for each epoch would be easier to implement, since the holding times are arbitrarily assigned to states anyway.

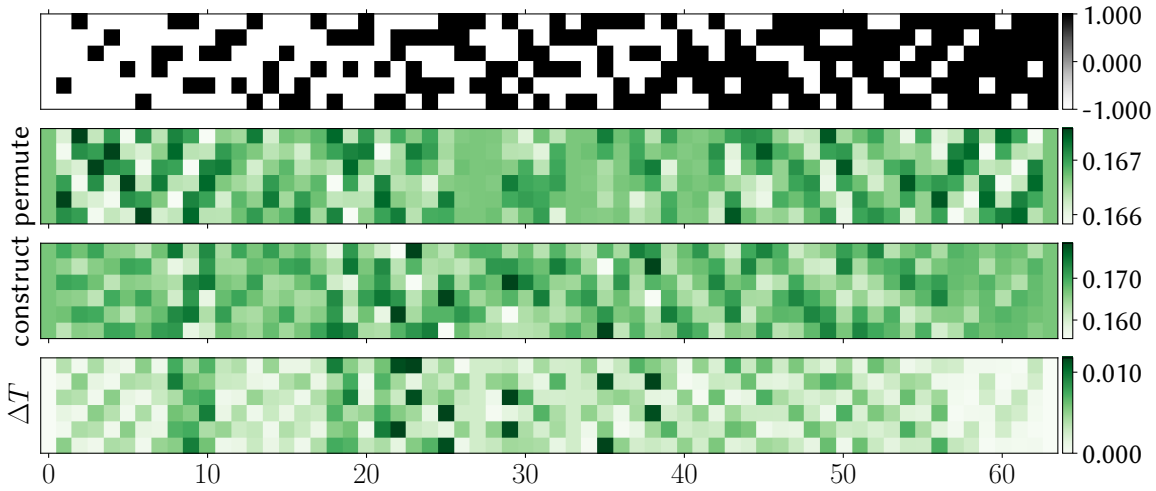


Fig. 5.7 **Comparison of *jump matrices* T obtained with different batch types.**

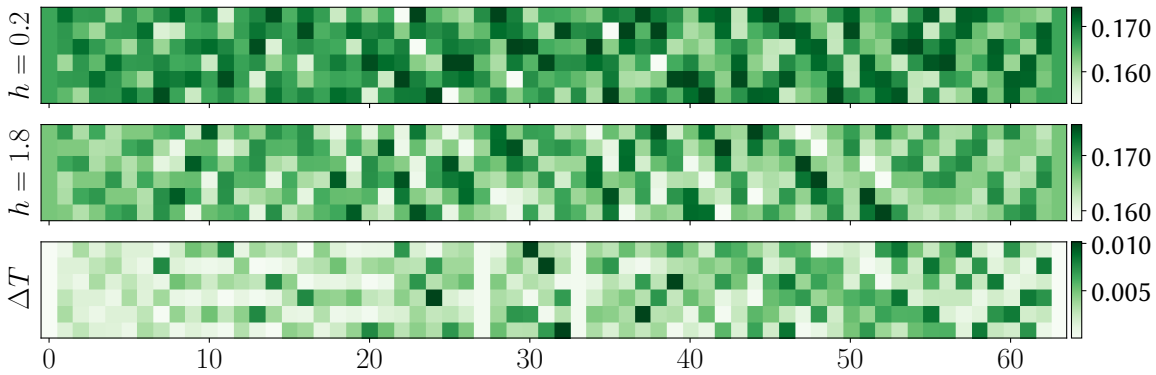


Fig. 5.8 **Comparison of *jump matrices* T for different h.**

[3]Although this case is not as extreme.

5.1.3 Architectural choices

The pCNN

As expected architecture choices and hyperparameters play an important role in learning the rates $\Gamma^{(v)}$, however how each separate parameter impacts the optimisation process is not very surprising. More interesting is perhaps the relative importance we can assign to each hyperparameter when compared to others. As we will see hyperparameters of the pCNN play one of two roles. The CTMC simulation time T and batch size N_b stabilise the estimation of variance loss, thus improving the stability of the learning process and convergence, while the width N_w of the pCNN and its depth N_l increase the representational power of the NN.

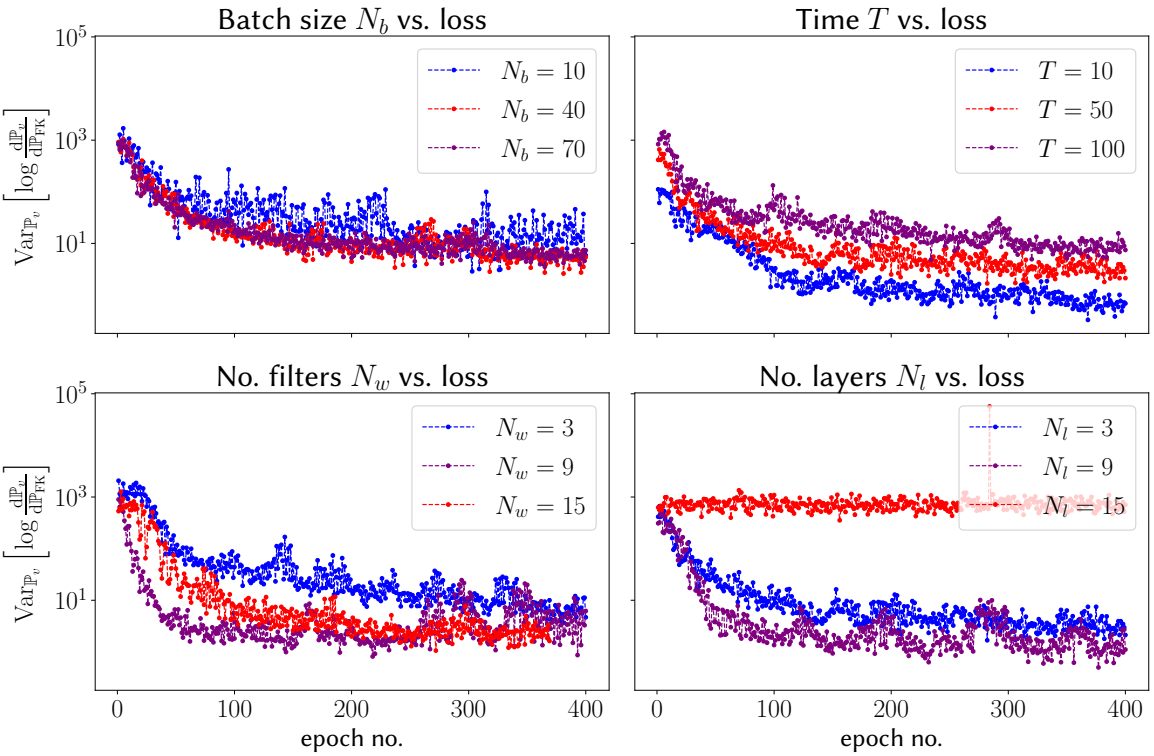


Fig. 5.9 **Initial rate training experiments**. Learning experiments using *initial* sampling mode and different simulation setups for 2d-TFIM ($L = 3, J = 1.0, h = 1.0$). **top left:** $T = 75, N_w = 5, N_l = 3$, **top right:** $N_b = 50, N_w = 5, N_l = 3$, **bottom left:** $N_b = 50, T = 50, N_l = 3$, **bottom right:** $N_b = 50, T = 50, N_w = 5$.

Initial experiments are shown in Fig. 5.9, the top row displays the aforementioned stability. Increasing the batch size decreases the "volatility" of the loss, and same is true for T, but only if we consider loss normalized per time unit. As a rule of thumb N_w and N_l do not have a significant impact on the efficiency and speed of training, so long as both

hyperparameters are chosen to be somewhere in the goldilocks zone, deep and narrow networks sometimes struggle to converge, (bottom right) Fig. 5.9.

Going beyond the training itself, we need to evaluate the *generalisation* of the rates. The worry is that training the rates at certain T and N_b could overfit and negatively impact sampling down the line, where the T will be very different. To test this the rates were trained for various settings of T, N_b, N_w, N_l and tested them by evaluating the loss at $T = 50$, $N_b = 50$, Fig. 5.10. What we learn is that the rates do not overfit, average test loss is in the same order of magnitude than the train loss, and that N_b is more important for stability than T. Finally, as we want to construct the most effective ansatz for eventual sampling experiments, we also investigate the relative time and size cost of increasing hyperparameters, the results are in Fig. 5.11.

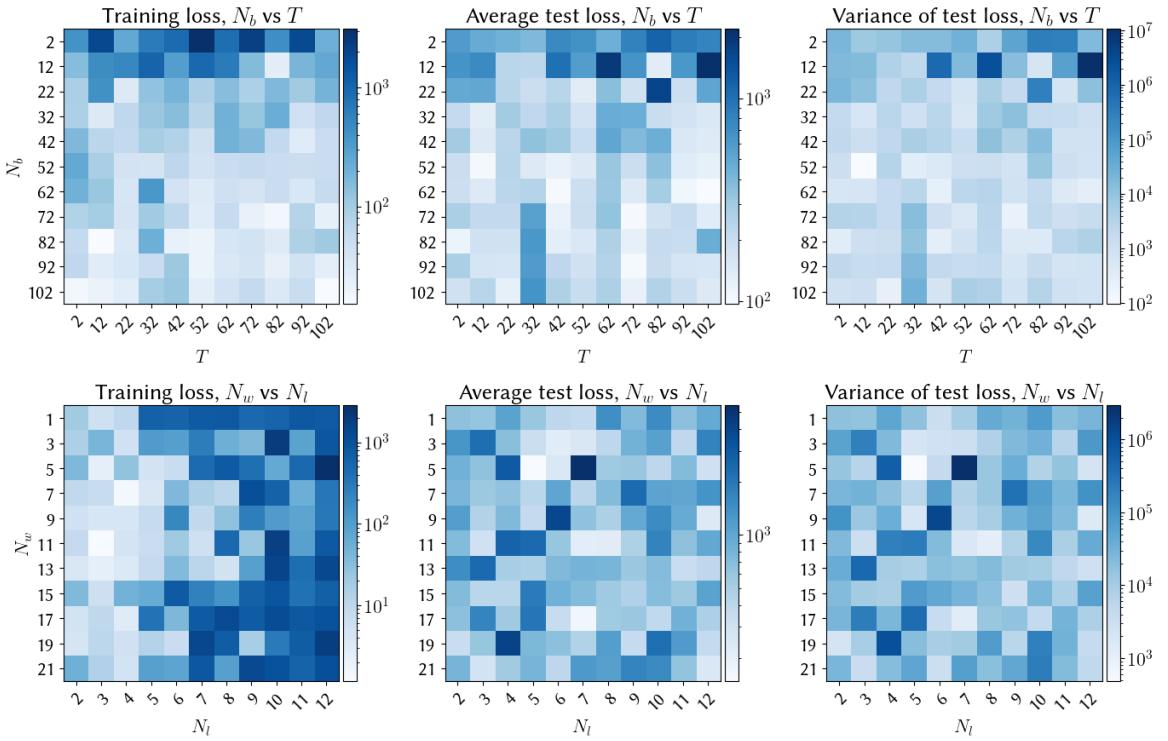


Fig. 5.10 **Generalisation capabilities of the pCNN.** Figure displays the effect of T, N_b and N_w, N_l on transferability of the learned rates using initial sampling, 2d-TFIM (L=3, J=1.0, h=1.0). Data was obtained by training the rates for different setups, sampling $N = 15$ trajectories with $T = 50$ from the learned rates for each setup, permuting the trajectories to obtain batches of $N_b = 50$ and finally evaluating the average and sample variance of the log RN loss. Subfigures show (left to right) the loss after final epoch of training, average test loss, and variance of test loss. Setups: varying N_b and T at $N_w = 3$, $N_l = 3$ (**top**), varying N_w and N_l at $N_b = 32$ and $T = 30$ (**bottom**). All loss values are T normalised.

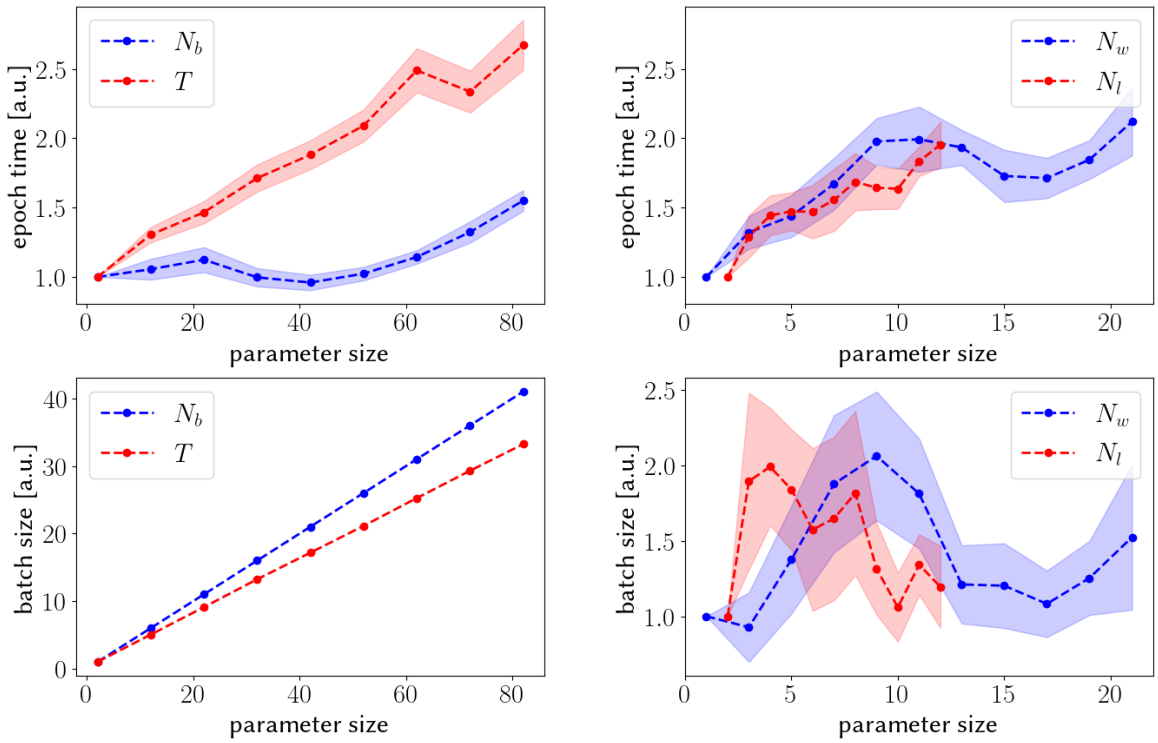


Fig. 5.11 **Time and space complexity of the pCNN.** Comparison of normalised time needed per epoch averaged over $N = 15$ samples, for different T, N_b (**top left**) and N_w, N_l (**top right**). Comparison of normalised batch size averaged over $N = 15$ samples, for different T, N_b (**bottom left**) and N_w, N_l (**bottom right**). All values obtained from the same simulations as in Fig. 5.10.

Comparison with G-pCNN

A strong inductive bias, e.g. equivariance, to translation or other appropriate symmetries should provide an advantage when learning a mapping equivariant to these same symmetries. Moreover, we expect this advantage to be even more significant in the case of learning with the log RN loss. This is because variational rates $\Gamma^{(v)}$ are evaluated multiple times in succession to evaluate the logarithm Radon-Nikodym derivative on a single trajectory, and this is repeated for a whole batch of trajectories.

Initial training experiments were conducted on the TFIM, comparing the learning of a fully connected network, and the pCNN in 1D, as well as both compared to the group-equivariant CNN in two dimensions. Qualitatively, there is little difference in the performance of the pCNN and the MLP in 1d, whereas the difference becomes more notable in 2d, where the gCNN seems to learn quickest, when comparing networks with comparable numbers of variational parameters. These experiments are limited in two ways. First, due to the stochastic nature of the loss it is hard to tell which architecture is performing better

and statistical metrics are needed. Second, because of a suboptimal implementation of the gCNN, runs with many epochs could not have been carried out and the results are thus inconclusive. Further investigation is needed to confirm these qualitative results, and is suggested as future work.

5.2 Importance sampling

Ultimately we were unsuccessful in learning the optimal rates Γ' in the TFIM model with the logRN loss. The restrictions that the fixed endpoints and variable length of trajectories put onto the batch construction process are too severe, and the learning algorithm is unable to efficiently learn from the batches constructed in such an artificial way. This section shows that the sampler part of the code was indeed implemented, that sampling can be performed with either the discrete or continuous time formulation of the rates, see Fig 5.12 left, and that the rates in Fig. 5.5 resemble sampling from the passive rates of the TFIM, see Fig. 5.12 right.

5.2.1 Ising model

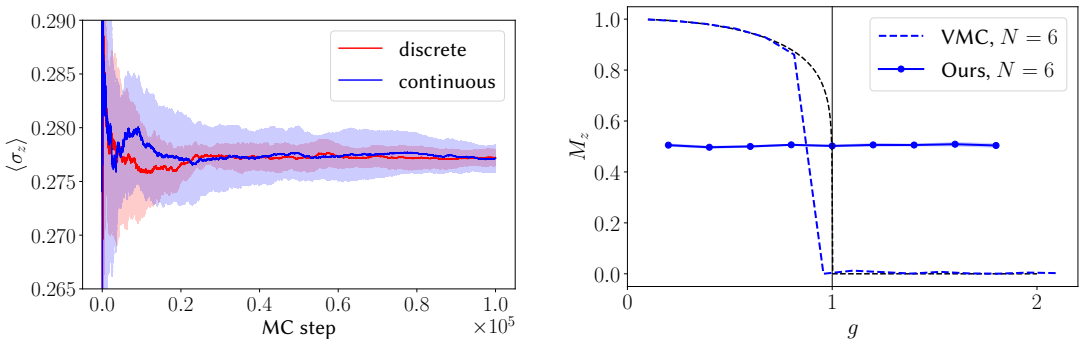


Fig. 5.12 **Sampling in the TFIM model.** Convergence of discrete and continuous formulations of the rates towards the expected σ_z value, taken as an average of $N = 10$ chains (**left**). In the limited sampling experiments performed, neither formulation provides a considerable advantage, although the continuous chain is more elegant. Estimation of the modulus per spin magnetisation $M_z = \frac{2}{N}\left|\sum_i s_i\right|$ using the rates from Fig. 5.5. The analytical solution for $N = \infty$ spins, and a VMC result with NetKet [15] are shown for comparison. (**right**).

Chapter 6

Discussion

6.1 Summary and contributions

In this work, we provided an overview of common quantum many-body methods, with a special focus on Monte Carlo and Machine Learning methods, as well as an introduction to stochastic processes. We then derived a novel loss function termed the variance log RN loss with fixed endpoints, which when optimised yields a trajectory distribution of a stoquastic lattice model. This new method is closely related to previous work of Barr et al. [6], which uses a similar loss function but for continuous systems where there is no need for fixed endpoint trajectories. The method is also closely related to work of Gispen et al. [30], which is based on much of the same theoretical foundation, but uses reinforcement learning to find ground states of lattice models. The main disadvantage of our method over standard methods is its limited applicability. It can only be applied to stoquastic Hamiltonians, and the batch generation process must be tailored to the system at hand. The main advantage of our method is at inference time, where it is much more efficient at sampling from the ground state than vanilla Metropolis-Hastings as no transitions are ever rejected.

Based on this, the main contributions of this thesis are twofold:

1. We provide a clear and thorough introduction to the many very diverse topics needed to understand, derive, and implement the log RN loss.

2. We implement the log RN loss and analyse its properties on the TFIM in both one and two dimensions. Most notably, we show that while the loss has nice convergence properties in theory, the restriction to *fixed endpoint* trajectories is very impractical. Since such trajectories cannot be sampled, they need to be constructed in some way. We propose three batch generation methods for the TFIM, identify a family

of Hamiltonians which pose problems for the logRN loss, and discuss their failure modes.

6.2 Direction for further work

As with any scientific work, this thesis leaves many questions unanswered and many more new questions uncovered. Future work should focus on:

1. **Questions regarding the loss and trajectories:**
    - Answering the most pressing question the thesis leaves unanswered. Why does the *construct* method not work?
    - Is there a batch construction method that will behave nicely for TFIM like models?
    - How do we construct batches in models where the actions are not independent (e.g. XY), and we cannot just permute them. Can we devise a general way to construct fixed endpoint trajectories under certain constraints?
    - More general questions regarding the generalisation properties of the loss. Since we artificially construct trajectories to feed into the loss, does this hinder its learning? Can the loss overfit to certain trajectories? Do trajectories that are very *instructive* and expedite the training process exist? Can we specifically engineer trajectories to facilitate learning?
2. **Technical improvements:**
    - Improve the implementation of the gCNN and conduct more extensive tests.
    - Reduce the memory requirement of the method to facilitate larger computations.
    - Can our method be used effectively in tandem with some other standard method?
    - Further exploration of the effects of equivariance on learning.
    - Randomly choose the time for each epoch, perhaps from the Erlang distribution. Does this stabilise learning?
3. **Theoretical considerations:**
    - How can we extend the method to a broader range of Hamiltonians? Perhaps using the fixed-node Feynman-Kac formula? [13].
    - Does a lattice model with an analytical solution for the rates exist?

References

[1] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283.

[2] Albeverio, S., Ho/egh-Krohn, R., and Streit, L. (1977). Energy forms, hamiltonians, and distorted brownian paths. *Journal of Mathematical Physics*, 18(5):907–917.

[3] Alexandru, A., Basar, G., Bedaque, P. F., and Warrington, N. C. (2020). Complex paths around the sign problem. *arXiv preprint arXiv:2007.05436*.

[4] Anderson, J. B. (1975). A random-walk simulation of the schrödinger equation: H+ 3. *The Journal of Chemical Physics*, 63(4):1499–1503.

[5] Assaraf, R., Caffarel, M., and Khelif, A. (2007). The fermion monte carlo revisited. *Journal of Physics A: Mathematical and Theoretical*, 40(6):1181.

[6] Barr, A., Gispen, W., and Lamacraft, A. (2020). Quantum ground states from reinforcement learning. In *Mathematical and Scientific Machine Learning*, pages 635–653. PMLR.

[7] Baxter, R. J. (1972). One-dimensional anisotropic heisenberg chain. *Annals of Physics*, 70(2):323–337.

[8] Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. (2018). Automatic differentiation in machine learning: a survey. *Journal of machine learning research*, 18.

[9] Bethe, H. (1931). Zur theorie der metalle. *Zeitschrift für Physik*, 71(3):205–226.

[10] Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. (2018). JAX: composable transformations of Python+NumPy programs.

[11] Bravyi, S., Divincenzo, D. P., Oliveira, R. I., and Terhal, B. M. (2006). The complexity of stoquastic local hamiltonian problems. *arXiv preprint quant-ph/0606140*.

[12] Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. (2021). Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*.

[13] Caffarel, M. (2015). Talking across fields: A physicist's presentation of some mathematical aspects of quantum monte carlo methods. In *Annales de la Faculté des sciences de Toulouse: Mathématiques*, volume 24, pages 949–972.

[14] Cai, Z. and Liu, J. (2018). Approximating quantum many-body wave functions using artificial neural networks. *Physical Review B*, 97(3):035116.

[15] Carleo, G., Choo, K., Hofmann, D., Smith, J. E. T., Westerhout, T., Alet, F., Davis, E. J., Efthymiou, S., Glasser, I., Lin, S.-H., Mauri, M., Mazzola, G., Mendl, C. B., van Nieuwenburg, E., O'Reilly, O., Théveniaut, H., Torlai, G., Vicentini, F., and Wietek, A. (2019). Netket: A machine learning toolkit for many-body quantum systems. *SoftwareX*, page 100311.

[16] Carleo, G., Nomura, Y., and Imada, M. (2018). Constructing exact representations of quantum many-body systems with deep neural networks. *Nature communications*, 9(1):1–11.

[17] Carleo, G. and Troyer, M. (2017). Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606.

[18] Childs, A. M. (2010). On the relationship between continuous-and discrete-time quantum walk. *Communications in Mathematical Physics*, 294(2):581–603.

[19] Clark, S. R. (2018). Unifying neural-network quantum states and correlator product states via tensor networks. *Journal of Physics A: Mathematical and Theoretical*, 51(13):135301.

[20] Cohen, T. and Welling, M. (2016). Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR.

[Crosson] Crosson, E. Discussion on stoquastic hamiltonians.

[22] Dai Pra, P. and Pavon, M. (1990). On the markov processes of schroedinger, the feynman-kac formula and stochastic control. In *Realization and Modelling in System Theory*, pages 497–504. Springer.

[23] De Gennes, P. (1963). Collective motions of hydrogen bonds. *Solid State Communications*, 1(6):132–137.

[24] Des Cloizeaux, J. and Pearson, J. (1962). Spin-wave spectrum of the antiferromagnetic linear chain. *Physical Review*, 128(5):2131.

[25] Dick, S. and Fernandez-Serra, M. (2020). Machine learning accurate exchange and correlation functionals of the electronic density. *Nature communications*, 11(1):1–10.

[26] Durrett, R. (2019). *Probability: theory and examples*, volume 49. Cambridge university press.

[27] Feiguin, A. E. and White, S. R. (2005). Time-step targeting methods for real-time dynamics using the density matrix renormalization group. *Physical Review B*, 72(2):020404.

[28] Feynman, R. P. (2018). Simulating physics with computers. In *Feynman and computation*, pages 133–153. CRC Press.

[29] Foulkes, W., Mitas, L., Needs, R., and Rajagopal, G. (2001). Quantum monte carlo simulations of solids. *Reviews of Modern Physics*, 73(1):33.

[30] Gispen, W. and Lamacraft, A. (2020). Ground states of quantum many body lattice models via reinforcement learning. *arXiv preprint arXiv:2012.07063*.

[31] Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*. MIT press Cambridge.

[32] Gubernatis, J., Kawashima, N., and Werner, P. (2016). *Quantum Monte Carlo Methods: Algorithms for Lattice Models*. Cambridge University Press.

[33] Halmos, P. R. (2013). *Measure theory*, volume 18. Springer.

[34] Heisenberg, W. (1985). Zur theorie des ferromagnetismus. In *Original Scientific Papers Wissenschaftliche Originalarbeiten*, pages 580–597. Springer.

[35] Held, K. (2007). Electronic structure calculations using dynamical mean field theory. *Advances in physics*, 56(6):829–926.

[36] Hohenberg, P. and Kohn, W. (1964). Inhomogeneous electron gas. *Physical review*, 136(3B):B864.

[37] Holland, C. J. (1977). A new energy characterization of the smallest eigenvalue of the schrödinger equation. *Communications in Pure Applied Mathematics*, 30:755–765.

[38] Hulthén, L. (1938). *Über das austauschproblem eines kristalles*. PhD thesis, Almqvist & Wiksell.

[39] Hutcheon, M. (2020). Stochastic nodal surfaces in quantum monte carlo calculations. *Physical Review E*, 102(4):042105.

[40] Innes, M. (2018). Don't unroll adjoint: Differentiating ssa-form programs. *arXiv preprint arXiv:1810.07951*.

[41] Kac, M. (1949). On distributions of certain wiener functionals. *Transactions of the American Mathematical Society*, 65(1):1–13.

[42] Kalos, M. (1962). Monte carlo calculations of the ground state of three-and four-body nuclei. *Physical Review*, 128(4):1791.

[43] Kalos, M. (1966). Stochastic wave function for atomic helium. *Journal of Computational Physics*, 1(2):257–276.

[44] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[45] Kipnis, C. and Landim, C. (1998). *Scaling limits of interacting particle systems*, volume 320. Springer Science & Business Media.

[46] LeCun, Y. et al. (1989). Generalization and network design strategies. *Connectionism in perspective*, 19:143–155.

[47] Léonard, C. (2014). Some properties of path measures. In *Séminaire de Probabilités XLVI*, pages 207–230. Springer.

[48] Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S. (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867.

[49] Lieb, E., Schultz, T., and Mattis, D. (1961). Two soluble models of an antiferromagnetic chain. *Annals of Physics*, 16(3):407–466.

[50] Lyu, Z., Kenway, G. K., Paige, C., and Martins, J. R. (2013). Automatic differentiation adjoint of the reynolds-averaged navier-stokes equations with a turbulence model. In *21st AIAA Computational Fluid Dynamics Conference*, page 2581.

[51] Minsky, M. and Papert, S. A. (2017). *Perceptrons: An introduction to computational geometry*. MIT press.

[52] Nelson, E. (1967). Dynamical theories of brownian motion, princeton univ. *Press, Princeton, NJ*.

[53] Niemeijer, T. (1967). Some exact calculations on a chain of spins 12. *Physica*, 36(3):377–419.

[54] Nightingale, M. and Melik-Alaverdian, V. (2001). Optimization of ground-and excited-state wave functions and van der waals clusters. *Physical review letters*, 87(4):043401.

[55] Nomura, Y., Darmawan, A. S., Yamaji, Y., and Imada, M. (2017). Restricted boltzmann machine learning for solving strongly correlated quantum systems. *Physical Review B*, 96(20):205152.

[56] Norris, J. R. and Norris, J. R. (1998). *Markov chains*. Number 2. Cambridge university press.

[57] Parkinson, J. B. and Farnell, D. J. (2010). *An introduction to quantum spin systems*, volume 816. Springer.

[58] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch.

[59] Pearlmutter, B. A. (2016). Automatic differentiation: History and headroom. Workshop on the future of gradient-based machine learning software, NIPS.

[60] Pfau, D., Spencer, J. S., Matthews, A. G., and Foulkes, W. M. C. (2020). Ab initio solution of the many-electron schrödinger equation with deep neural networks. *Physical Review Research*, 2(3):033429.

[61] Pfeuty, P. (1970). The one-dimensional ising model with a transverse field. *ANNALS of Physics*, 57(1):79–90.

[62] Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17.

[63] Rasmussen, C., Williams, C., Press, M., Bach, F., and (Firm), P. (2006). *Gaussian Processes for Machine Learning*. Adaptive computation and machine learning. MIT Press.

[64] Reynolds, P. J., Tobochnik, J., and Gould, H. (1990). Diffusion quantum monte carlo. *Computers in Physics*, 4(6):662–668.

[65] Rogers, L. C. G. and Williams, D. (1994). Diffusions, markov processes and martingales, volume 1: Foundations. *John Wiley & Sons, Ltd., Chichester*, 7.

[66] Rogers, L. C. G. and Williams, D. (2000). *Diffusions, Markov processes and martingales: Volume 2, Itô calculus*, volume 2. Cambridge university press.

[67] Saito, H. (2017). Solving the bose–hubbard model with machine learning. *Journal of the Physical Society of Japan*, 86(9):093001.

[68] Salamon, D. (2016). *Measure and integration*. European Mathematical Society.

[69] Särkkä, S. and Solin, A. (2019). *Applied stochastic differential equations*, volume 10. Cambridge University Press.

[70] Skylaris, C.-K., Haynes, P. D., Mostofi, A. A., and Payne, M. C. (2005). Introducing onetep: Linear-scaling density functional simulations on parallel computers. *The Journal of chemical physics*, 122(8):084119.

[71] Sorella, S. (1998). Green function monte carlo with stochastic reconfiguration. *Physical review letters*, 80(20):4558.

[72] Speelpenning, B. (1980). *Compiling fast partial derivatives of functions given by algorithms*. PhD thesis, University of Illinois at Urbana-Champaign.

[73] Spencer, J. S., Pfau, D., Botev, A., and Foulkes, W. M. C. (2020). Better, faster fermionic neural networks. *arXiv preprint arXiv:2011.07125*.

[74] Stinchcombe, R. (1973a). Ising model in a transverse field. i. basic theory. *Journal of Physics C: Solid State Physics*, 6(15):2459.

[75] Stinchcombe, R. (1973b). Ising model in a transverse field. ii. spectral functions and damping. *Journal of Physics C: Solid State Physics*, 6(15):2484.

[76] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

[77] Szabó, A. and Castelnovo, C. (2020). Neural network wave functions and the sign problem. *Physical Review Research*, 2(3):033075.

[78] Tamayo-Mendoza, T., Kreisbeck, C., Lindh, R., and Aspuru-Guzik, A. (2018). Automatic differentiation in quantum chemistry with applications to fully variational hartree–fock. *ACS central science*, 4(5):559–566.

[79] Todorov, E. (2007). Linearly-solvable markov decision problems. In *Advances in neural information processing systems*, pages 1369–1376.

[80] Todorov, E. (2009). Efficient computation of optimal actions. *Proceedings of the national academy of sciences*, 106(28):11478–11483.

[81] Torlai, G., Mazzola, G., Carrasquilla, J., Troyer, M., Melko, R., and Carleo, G. (2018). Neural-network quantum state tomography. *Nature Physics*, 14(5):447–450.

[82] Troyer, M. and Wiese, U.-J. (2005). Computational complexity and fundamental limitations to fermionic quantum monte carlo simulations. *Physical review letters*, 94(17):170201.

[83] Verstraete, F. and Cirac, J. I. (2004). Renormalization algorithms for quantum-many body systems in two and higher dimensions. *arXiv preprint cond-mat/0407066*.

[84] Wengert, R. E. (1964). A simple automatic derivative evaluation program. *Communications of the ACM*, 7(8):463–464.

[85] White, S. R. (1992). Density matrix formulation for quantum renormalization groups. *Physical review letters*, 69(19):2863.

[86] Wilson, K. G. (1975). The renormalization group: Critical phenomena and the kondo problem. *Reviews of modern physics*, 47(4):773.

[87] Wilson, M., Gao, N., Wudarski, F., Rieffel, E., and Tubman, N. M. (2021). Simulations of state-of-the-art fermionic neural network wave functions with diffusion monte carlo. *arXiv preprint arXiv:2103.12570*.

[88] Yang, C. and Yang, C. (1966a). One-dimensional chain of anisotropic spin-spin interactions. iii. applications. *Physical Review*, 151(1):258.

[89] Yang, C.-N. and Yang, C.-P. (1966b). One-dimensional chain of anisotropic spin-spin interactions. i. proof of bethe's hypothesis for ground state in a finite system. *Physical Review*, 150(1):321.

[90] Yang, C.-N. and Yang, C.-P. (1966c). One-dimensional chain of anisotropic spin-spin interactions. ii. properties of the ground-state energy per lattice site for an infinite system. *Physical Review*, 150(1):327.

[91] Zhang, S., Carlson, J., and Gubernatis, J. E. (1997). Constrained path monte carlo method for fermion ground states. *Physical Review B*, 55(12):7464.

Appendix A

Additional Derivations

A.1 Holland cost

Following Holland's [37] derivation of a stochastic control problem for the Schrödinger equation. We start by introducing the exponential transform

$$\psi(x) = \exp[-U(x)], \tag{A.1}$$

into the Schroödinger equation for $x \in \mathbb{R}^n$

$$H\psi = \left[-\frac{1}{2}\nabla^2 + V(x)\right]\psi = \lambda\psi, \tag{A.2}$$

to obtain

$$\frac{1}{2}\nabla^2 U - \frac{1}{2}(\nabla U)^2 + V(x) = \lambda. \tag{A.3}$$

We can reinterpret the second term in eq. (A.3) as a minimum over all vectors v for every $x \in \mathbb{R}^n$

$$\frac{1}{2}\nabla^2 U + \min_v \left[v \cdot \nabla U + \frac{1}{2}|v|^2 + V(x)\right] = \lambda. \tag{A.4}$$

We define v to be a Lipshitz continuous function, i.e. the drift $v(x)$. Each drift now generates an Itô process

$$\mathrm{d}X_t = \mathrm{d}W_t + v(X_t)\,\mathrm{d}t \quad \text{and} \quad X_0 = x, \tag{A.5}$$

and we can define the cost function $C[v]$ for each v as

$$C[v] = \lim_{T\to\infty} \frac{1}{T}\mathbb{E}\left[\int \left(\frac{1}{2}|v(X_t)|^2 + V(X_t)\right)\mathrm{d}x\right]. \tag{A.6}$$

Holland [37] proves the following theorem.

Theorem A.1.1 (Holland cost function)**.** *The minimum* $\lambda = \min_v C[v]$*, where the minimum is taken over drift functions* $v : \Omega \longrightarrow \mathbb{R}^n$*,* $\Omega \subset \mathbb{R}^n$ *with Neumann boundary conditions* $\frac{\partial \psi}{\partial n} = 0$ *on* $\partial\Omega$*, is obtained only for* $v = \frac{\nabla\psi}{\psi} = -\nabla U$*.*

Proof. From eq. (A.4) follows the inequality

$$\frac{1}{2}\nabla^2 U + v \cdot \nabla U + \frac{1}{2}|v(x)|^2 + V(x) \geq \lambda, \tag{A.7}$$

we notice that the first two terms are an infinitesimal generator $[\mathcal{L}_G U](x)$ of the process in eq. (A.5), which is defined as

$$[\mathcal{L}_G \psi](x) \equiv \lim_{t \longrightarrow 0} \frac{\mathbb{E}_x[\psi(X_t)] - \psi(x)}{t}, \tag{A.8}$$

where $\mathbb{E}_x$ is over processes with initial condition $X_0 = x$. With this in mind, we evaluate eq. (A.7) on the process X_t and integrate over time to obtain

$$\frac{\mathbb{E}[U(X_T) - U(x)]}{T} + \mathbb{E}\left[\frac{1}{T}\int_0^T \left(\frac{1}{2}|v(X_t)|^2 + V(X_t)\right) \mathrm{d}t\right] \geq \lambda. \tag{A.9}$$

In the $T \longrightarrow \infty$ limit, the first terms disappears, so long as U is bounded, and we are left with $C[v]$, meaning that we have proven the bound

$$C[v] \geq \lambda. \tag{A.10}$$

We see that the minimum λ is achieved for control function

$$v = \frac{\nabla\psi}{\psi} = -\nabla U, \tag{A.11}$$

in eq. (A.4), the uniqueness of this optimal v is due to the fact that if $v \neq \frac{\nabla\psi}{\psi}$ there cannot be a full equality in eq. (A.7). □

A.2 Probabilistic interpretation of Holland's cost function

This section follows Barr et al. [6] which is closely related to previous work [22]. To express the RN derivative we use the Girsanov theorem for both $\mathbb{P}_v$

$$\frac{\mathrm{d}\mathbb{P}_v}{\mathrm{d}\mathbb{P}_0} = \exp\left(\int v(X_t)\,\mathrm{d}X_t - \frac{1}{2}\int |v(X_t)|^2\,\mathrm{d}t\right), \tag{A.12}$$

and $\mathbb{P}_{\mathrm{FK}}$

$$\frac{\mathrm{d}\mathbb{P}_{\mathrm{FK}}}{\mathrm{d}\mathbb{P}_0} = \mathcal{N} \exp\left(-\int V(X_t)\,\mathrm{d}t\right). \tag{A.13}$$

We combine both, and we have up to exponential accuracy $\mathcal{N} \sim e^{\lambda T}$

$$\log\left(\frac{\mathrm{d}\mathbb{P}_v}{\mathrm{d}\mathbb{P}_{\mathrm{FK}}}\right) = \log\left(\frac{\mathrm{d}\mathbb{P}_v}{\mathrm{d}\mathbb{P}_0}\frac{\mathrm{d}\mathbb{P}_0}{\mathrm{d}\mathbb{P}_{\mathrm{FK}}}\right) = \int v(X_t)\,\mathrm{d}X_t + \int\left(-\frac{1}{2}|v(X_t)|^2 + V(X_t)\right)\mathrm{d}t - \lambda T, \tag{A.14}$$

finally we substitute $\mathrm{d}X_t = \mathrm{d}W_t + v(X_t)\mathrm{d}t$ to get

$$\begin{aligned}\log\left(\frac{\mathrm{d}\mathbb{P}_v}{\mathrm{d}\mathbb{P}_{\mathrm{FK}}}\right) &= \int v(X_t)\,\mathrm{d}W_t + \int |v(X_t)|^2\mathrm{d}t + \int\left(-\frac{1}{2}|v(X_t)|^2 + V(X_t)\right)\mathrm{d}t \\ &= \int v(X_t)\,dW_t + \int\left(\frac{1}{2}|v(X_t)|^2 + V(X_t)\right)\mathrm{d}t - \lambda T.\end{aligned} \tag{A.15}$$

A closer look at the normalisation constant $\mathcal{N}$ in eq. (A.13) gives rise to the boundary term. The normalisation is given by

$$\mathcal{N} = \frac{\tilde{\psi}(r_T, T)}{\bar{\psi}(r_0, 0)}, \tag{A.16}$$

where $\tilde{\psi}(r_t, t)$ is the solution to the backwards imaginary time Schrödinger equation [6], and is related to the distribution of the stochastic process in eq. (A.5) as

$$\pi(r, t) = \tilde{\psi}(r, t)\,\psi(r, t). \tag{A.17}$$

If we now take both distributions at initial time $\pi(r, 0)$ and terminal time $\pi(r, T)$ to be the ground state the normalisation constant becomes

$$\frac{\tilde{\psi}(r_T, T)}{\bar{\psi}(r_0, 0)} = e^{E_0 T}\frac{\varphi_0(r_T)}{\varphi_0(r_0)}, \tag{A.18}$$

and accounts for the boundary term and $E_0 T$ term in the Kullback-Leibler divergence.

A.3 Todorov cost

Optimal decision processes are formalised using Markov Decision processes, here we follow work of Todorov [79, 80] to fit the imaginary Schrödinger equation in this mould. An MDP is a 4-tuple (S, A, P_a, R_a):

- S: Is the *state space*, i.e. all possible configurations of the system s_k

- U: Is the *control space*, i.e. all possible single-spin flips in each configuration
- P_u: Is the probability $p(s^{(t+1)} = s'|s^{(t)} = s, u^{(t)} = u)$ of control u in state s leading to state s' in the next time step
- g: Is the cost received when moving from state s to s' due to u

The optimal decision/control problem is then given by the Bellman equation for the optimal cost-to-go function $v(s)$

$$v(s) = \min_u \left\{ \underbrace{\ell(s,u)}_{\text{immediate cost}} + \underbrace{\mathbb{E}_{s'\sim p(\cdot|s,u)}\left[v\left(x'\right)\right]}_{\text{expected cost of next state}} \right\}. \tag{A.19}$$

Todorov introduces a formalism where the agent does not perform specific symbolic actions (e.g. flips a certain spin) but is instead allowed to specify transition probabilities $u(s'|s)$. Formally this means that

$$p\left(s' \mid s, u\right) = u\left(s' \mid s\right), \tag{A.20}$$

and the agent reshapes the dynamics of the system as it wishes, but for this it pays a price depending on how much it changes the dynamics. In absence of controls u the system follows *passive dynamics* $p(s'|s)$ which correspond to the first term in eq. (3.42) of the stoquastic Hamiltonian. The cost is thus

$$\ell(s,u) = \underbrace{q(s)}_{\text{state cost}} + \underbrace{D_{\mathrm{KL}}((\cdot \mid s)\|p(\cdot \mid s))}_{\text{control cost}}. \tag{A.21}$$

Optimal control problem in this form can be linearised in terms of the *desirability* function $z(s,t) = \exp(-v(s,t))$, yielding optimal dynamics v'

$$v'(s_j \mid s_k) = \frac{p(s_j \mid s_k) z(s_j)}{\sum_{s_l} p(s_l \mid s_k) z(s_l)}, \tag{A.22}$$

with

$$z(s_k,t) = e^{-q(s_k)} \sum_{s_j} p(s_j \mid s_k) z(s_j, t+1). \tag{A.23}$$

It is this linear equation, that we can connect to the imaginary time Schrödinger equation. We start by transforming the MDP into continuous time (transition probabilities to rates

$p, u \longrightarrow \Gamma, \Gamma^{(v)}$) as

$$p(s_j \mid s_k) = \begin{cases} 1 - \Delta t \sum_{s_l} \Gamma_{s_k \to s_l} & s_j = s_k \\ \Delta t \Gamma_{s_k \to s_j} & s_j \neq s_k \end{cases}, \quad \text{and} \quad u(s_j \mid s_k) = \begin{cases} 1 - \Delta t \sum_{s_l} \Gamma^{(v)}_{s_k \to s_l} & s_j = s_k \\ \Delta t \Gamma^{(v)}_{s_k \to s_j} & s_j \neq s_k \end{cases} \tag{A.24}$$

and setting $q(s_j) = \Delta t V(s_j)$ eq. (A.23) becomes

$$\begin{aligned} z(s_k, t) &= e^{-\Delta t V(s_k)} \left[\underbrace{z(s_k, t + \Delta t) - \Delta t \sum_{s_l \neq s_k} \Gamma_{s_k \to s_l} z(s_k, t + \Delta t)}_{\text{from } s_j = s_k} + \underbrace{\Delta t \sum_{s_j \neq s_k} \Gamma_{s_k \to s_j} z(s_j, t + \Delta t)}_{\text{from } s_j \neq s_k} \right] \\ &= \left[1 - \Delta t V(s_k) + \ldots \right] \cdot \left[z(s_k, t + \Delta t) - \sum_{s_j \neq s_k} \Gamma_{s_k \to s_j} \left[z(s_j, t + \Delta t) - z(s_k, t + \Delta t) \right] \right] \end{aligned} \tag{A.25}$$

keeping only the first order in Δt, dropping the unnecessary $\neq$ in the sum

$$\frac{z(s_k, t) - z(s_k, t + \Delta t)}{\Delta t} = V(s_k) z(s_k, t + \Delta t) - \sum_{s_j} \Gamma_{s_k \to s_j} \left[z(s_j, t + \Delta t) - z(s_k, t + \Delta t) \right] \tag{A.26}$$

and taking the limit $\Delta t \longrightarrow 0$ finally gives

$$-\frac{\mathrm{d} z(s_k, t)}{\mathrm{d} t} = V(s_k) z(s_k, t) - \sum_{s_j} \Gamma_{s_k \to s_j} \left[z(s_j, t) - z(s_k, t) \right], \tag{A.27}$$

which is the imaginary time Schrödinger equation (3.67).

The D_{KL} in the loss $\ell(s, v)$ is expressed in the same manner

$$\begin{aligned} D_{\mathrm{KL}}(v(\cdot \mid s_k) \| p(\cdot \mid s_k)) &= \left[1 - \Delta t \sum_{s_l} \Gamma^{(v)}_{s_k \to s_l} \right] \log \left[\frac{1 - \Delta t \sum_{s_l} \Gamma^{(v)}_{s_k \to s_l}}{1 - \Delta t \sum_{s_l} \Gamma_{s_k \to s_l}} \right] + \Delta t \sum_{s_j \neq s_k} \Gamma^{(v)}_{s_k \to s_j} \log \left[\frac{\Gamma^{(v)}_{s_k \to s_j}}{\Gamma_{s_k \to s_j}} \right] \\ &= \Delta t \sum_{s_j \neq s_k} \Gamma^{(v)}_{s_k \to s_j} \underbrace{\left(\log \left[\frac{\Gamma^{(v)}_{s_k \to s_j}}{\Gamma_{s_k \to s_j}} \right] + \frac{\Gamma_{s_k \to s_j}}{\Gamma^{(v)}_{s_k \to s_j}} - 1 \right)}_{\text{Itakura–Saito divergence } D_{\mathrm{IS}}(\Gamma^{(v)}_{s_k \to s_j}, \Gamma_{s_k \to s_j})}. \end{aligned} \tag{A.28}$$

If we consider an ensemble of systems in state s_k there is a $\Delta t \Gamma^{(v)}_{s_k \to s_j}$ probability of transitioning $s_k \longrightarrow s_j$ in the next time increment, meaning that the ensemble contribution to the D_{KL} each time increment is $\sum_{s_j} \Delta t \Gamma^{(v)}_{s_k \to s_j} D_{\mathrm{IS}} \left(\Gamma^{(v)}_{s_k \to s_j}, \Gamma_{s_k \to s_j} \right)$, thus we can express the

Kullback-Liebler divergence as

$$D_{\mathrm{KL}} = \mathop{\mathbb{E}}_{\Sigma_{[0,t]}=k_t\sim\Gamma^{(v)}}\left[\sum_n D_{\mathrm{IS}}\left(\Gamma^{(v)}_{k^{(n)}\to k^{(n+1)}}, \Gamma_{k^{(n)}\to k^{(n+1)}}\right)\right]. \tag{A.29}$$

A.4 Probabilistic interpretation of Todorov's cost function

To find the Radon-Nikodym derivative between $\mathbb{P}_v$ and $\mathbb{P}_{\mathrm{FK}}$ we proceed analogous to the continuous case, by first finding the respective RN derivatives with the passive process. From the discrete space Feynman-Kac formula (3.51) follows

$$\log\left(\frac{\mathrm{d}\mathbb{P}_0}{\mathrm{d}\mathbb{P}_{FK}}(k(t))\right) = \int V(k(t))dt - E_0 T - \log\left(\frac{\varphi(k^{(N)})}{\varphi(k^{(0)})}\right), \tag{A.30}$$

and by using the Girsanov theorem equivalent for discrete space[1] we obtain

$$\log\left(\frac{\mathrm{d}\mathbb{P}_v}{\mathrm{d}\mathbb{P}_0}(k(t))\right) = \int \sum_{l\neq k(t)}\left(\Gamma_{k(t)\to l} - \Gamma^{(v)}_{k(t)\to l}\right)dt + \sum_n \log\left(\frac{\Gamma^{(v)}_{k^{(n)}\to k^{(n+1)}}}{\Gamma_{k^{(n)}\to k^{(n+1)}}}\right). \tag{A.31}$$

We combine both to get

$$\log\left(\frac{\mathrm{d}\mathbb{P}_v}{\mathrm{d}\mathbb{P}_{\mathrm{FK}}}\right) = \log\left(\frac{\mathrm{d}\mathbb{P}_v}{\mathrm{d}\mathbb{P}_0}\frac{\mathrm{d}\mathbb{P}_0}{\mathrm{d}\mathbb{P}_{\mathrm{FK}}}\right) = \tilde{\ell} + \sum_n \log\left(\frac{\Gamma^{(v)}_{k^{(n)}\to k^{(n+1)}}}{\Gamma_{k^{(n)}\to k^{(n+1)}}}\right) - E_0 T - \log\left(\frac{\varphi(k^{(N)})}{\varphi(k^{(0)})}\right), \tag{A.32}$$

with

$$\tilde{\ell} = \int\left[V(k(t)) + \sum_{l\neq k(t)}\left(\Gamma_{k(t)\to l} - \Gamma^{(v)}_{k(t)\to l}\right)\right]\mathrm{d}t. \tag{A.33}$$

To see that zero $D_{KL}(\mathbb{P}_v\|\mathbb{P}_{\mathrm{FK}})$ coincides with rates that minimize Todorov's cost, we need

$$\mathop{\mathbb{E}}_{\mathbb{P}_v}\left[\sum_n \log\left(\frac{\Gamma^{(v)}_{k^{(n)}\to k^{(n+1)}}}{\Gamma_{k^{(n)}\to k^{(n+1)}}}\right)\right] = \mathop{\mathbb{E}}_{\mathbb{P}_v}\left[\int\sum_{l\neq k(t)}\Gamma^{(v)}_{k(t)\to l}\log\left(\frac{\Gamma^{(v)}_{k(t)\to l}}{\Gamma_{k(t)\to l}}\right)\mathrm{d}t\right], \tag{A.34}$$

which holds because the expectation of a contribution of a single step of the trajectory $k^{(n)}\to k^{(n+1)}$ is equivalent to an ensemble average starting from the same state weighted by the probability of jump $\Gamma^{(v)}_{k(t)\to l}\Delta t$, this holds separately for each step in the trajectory and

[1] see proposition 2.6 in Appendix 1 of [45]

by writing $\sum_{t_i} \cdots \Delta t_i \longrightarrow \int \cdots \mathrm{d}t$ we obtain above equality. The KL divergence then becomes

$$D_{\mathrm{KL}}(\mathbb{P}_v \| \mathbb{P}_{\mathrm{FK}}) = \mathop{\mathbb{E}}_{\mathbb{P}_v}\Bigg[\int V(k(t)) + \sum_{l \neq k(t)} \left(\Gamma_{k(t)\to l} - \Gamma^{(v)}_{k(t)\to l} \right) \mathrm{d}t + \sum_n \log\left(\frac{\Gamma^{(v)}_{k^{(n)}\to k^{(n+1)}}}{\Gamma_{k^{(n)}\to k^{(n+1)}}} \right) - \log\left(\frac{\varphi(k^{(N)})}{\varphi(k^{(0)})} \right) \Bigg] - E_0 T \tag{A.35}$$

$$D_{\mathrm{KL}}(\mathbb{P}_v \| \mathbb{P}_{\mathrm{FK}}) = \mathop{\mathbb{E}}_{\mathbb{P}_v}\Bigg[\int V(k(t)) + \sum_{l \neq k(t)} \left(\Gamma_{k(t)\to l} - \Gamma^{(v)}_{k(t)\to l} \right) + \Gamma^{(v)}_{k(t)\to l} \log\left(\frac{\Gamma^{(v)}_{k(t)\to l}}{\Gamma_{k(t)\to l}} \right) \mathrm{d}t - \log\left(\frac{\varphi(k^{(N)})}{\varphi(k^{(0)})} \right) \Bigg] - E_0 T, \tag{A.36}$$

which agrees with Todorov.