

Quantum Monte Carlo Simulations of Rydberg Atom Arrays

by

Ejaaz Merali

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Physics

Waterloo, Ontario, Canada, 2025

© Ejaaz Merali 2025

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner: Ying-Jer Kao
Professor, Department of Physics, National Taiwan University
Center Scientist, National Center for Theoretical Sciences

Supervisor(s): Roger G. Melko
Professor, Department of Physics and Astronomy,
University of Waterloo
Associate Faculty, Perimeter Institute for Theoretical Physics
Affiliate Member, Institute for Quantum Computing

Internal Member: Anton Burkov
Professor, Department of Physics and Astronomy,
University of Waterloo
Affiliate Member, Perimeter Institute for Theoretical Physics

Internal-External Member: Pierre-Nicholas Roy
Professor, Department of Chemistry,
University of Waterloo
Affiliate Member, Perimeter Institute for Theoretical Physics
Affiliate Member, Institute for Quantum Computing

Other Member(s): Juan Felipe Carrasquilla Álvarez
Associate Professor, Department of Physics, ETH Zürich
Adjunct Faculty, Department of Physics and Astronomy,
University of Waterloo

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

I am the sole author of Chapters 1,2,3,4 and 8. Chapters 5 through 7 are based on papers on which I collaborated.

Chapter 5 is based on research done with Isaac De Vlugt and Roger Melko published in Ref. [139]. I was responsible for the development of most of the code base on which the research is based, and performed simulations and analyses for the 2D square lattice in order to characterize the performance of the algorithm. Isaac performed many simulations and analyses for 1D systems, as well as deriving the groundstate energy estimator that I reference in that chapter. In the chapter itself, I focus on the parts of the paper that I was responsible for. As a result, I removed some of the sections that were covered in Isaac's thesis, such as the derivation of the groundstate energy estimator. Near the end of the chapter, I provide some ideas for a new update algorithm which I think could outperform the one detailed in the chapter. Roger Melko took an advisory role for the work detailed in this chapter.

Chapter 6 is based on research done with Mohamed Hibat Allah, Giacomo Torlai, Roger Melko, and Juan Carrasquilla which has been accepted for publication and is currently available as a preprint in Ref. [90]. Mohamed performed all of the VMC simulations and the related analyses, in addition to much of the writing. I performed all of the SSE QMC simulations and related analyses. Giacomo helped with some of the DMRG simulations as well as providing advice. Juan and Roger took advisory roles during the process. In the chapter I expand on the SSE simulation and the analyses of the simulation data.

Chapter 7 summarizes two papers that I contributed to. The first is a paper with Stefanie Czischek, M. Schuyler Moss, Matthew Radzihovsky, and Roger G. Melko, published in Ref. [37]. The second paper is with David Fitzek, Yi Hong Teoh, Hin Pok Fung, Gebremedhin A. Dagneu, M. Schuyler Moss, Benjamin MacLellan, Roger G. Melko, available as a preprint in Ref. [62]. I helped with the SSE simulations needed to generate the data used for the experiments as well as offering general advice about the QMC data.

I understand that my thesis may be made electronically available to the public.

Abstract

Rydberg atom arrays form a promising platform for quantum computation. Through their strong, long-range interaction, they are able to encode various difficult combinatorial problems, as well as hosting a plethora of intriguing physical phenomena. In this thesis, we develop and apply a Stochastic Series Expansion Quantum Monte Carlo method to simulate Rydberg systems at zero-temperature and above. We then apply this simulation method alongside variational models to verify correctness of both methods. The data produced from the simulations is also used to train Neural Network wavefunctions, which we find are effectively able to grasp some of the physics of the Rydberg atom array on a square lattice.

Acknowledgements

I could not have gotten this far without the help of several people. To my parents and my brother: thank you for your constant love and support. To my two best friends, Saoirse and Sophie: the last couple years would have been so much greyer without either of you, thank you for everything¹.

To everyone I've met at the PIQuIL since its beginning: Isaac De Vlugt, Giacomo Torlai, Anna Golubeva, Matt Beach, Bohdan Kulchytsky, Roger Luo, Stefanie Czischek, Shuyler Moss, Sebastian Wetzel, Estelle Inack, Dmitri Iouchtchenko, Mohamed Hibat Allah, and probably many more that I'm forgetting about. Thank you for all you've taught me in the past few years and for listening to my rants about random ideas I had.

To my PhD committee: Anton Burkov, Juan Carrasquilla, and P.N. Roy. Thank you for all of the advice you've given me over the years.

To my supervisor, Roger Melko: thank you for all of your support and guidance throughout the years, for putting up with my disorganization, for all the lunches and barbecues and Heartsmases, and—most importantly—for all the compute time.

¹Except maybe the chickenpox.

Dedication

This thesis is dedicated to whoever finds it useful.

Table of Contents

Examining Committee	ii
Author's Declaration	iii
Statement of Contributions	iv
Abstract	v
Acknowledgements	vi
Dedication	vii
List of Figures	xiv
List of Tables	xvi
List of Algorithms	xvii
1 Introduction	1
1.1 The Rydberg Hamiltonian	1
1.2 Outline	3

2	Monte Carlo Simulations	5
2.1	Direct Sampling	6
2.1.1	Inverse Transform Sampling	7
2.1.2	Rejection Sampling	10
2.1.3	The Alias Method	13
2.2	Markov Chain Monte Carlo	16
2.2.1	The Balance Equation	19
2.2.2	Ergodicity	22
2.2.3	Composition of Transition Kernels	27
2.2.4	Detailed Balance	28
2.2.5	The Metropolis-Hastings Algorithm	31
2.2.6	The Heat-Bath Algorithm	33
2.2.7	Choosing between the Metropolis-Hastings and the Heat-Bath Algorithms	35
2.2.8	Equilibration	39
2.3	Error Estimation	40
2.3.1	Dealing with independent samples	40
2.3.2	Dealing with autocorrelation	41
2.3.3	Dealing with non-linearity	43
2.3.4	Putting it all together	43
2.3.5	Multiple Markov Chains	44
3	Stochastic Series Expansion	45
3.1	Partition Function Decomposition	46
3.2	Hamiltonian Breakup and the SSE Configuration Space	50
3.2.1	The No-Branching Condition	51
3.2.2	Energy Shifts and the Diagonal Sign Problem	52
3.2.3	The Sign Problem	53

3.2.4	An Extended Representation	54
3.2.5	Operator Indices	55
3.3	Estimating Expectation Values	55
3.3.1	Diagonal Operators	56
3.3.2	Hamiltonian Sub-Operators	57
3.3.3	Products of Hamiltonian Sub-Operators	59
3.3.4	Energetic Quantities	60
3.3.5	More Complex Estimators	64
3.4	Groundstate Projector SSE	64
3.4.1	Normalizing Constant Decomposition	65
3.4.2	Estimating Expectation Values	66
4	SSE Updates	68
4.1	Diagonal Updates	69
4.1.1	A Simple Metropolis-Hastings Scheme	69
4.1.2	A Problematic Heat-Bath Scheme	71
4.1.3	Improving the Metropolis-Hastings Scheme	73
4.1.4	Can we try drawing samples until one is accepted?	75
4.1.5	Reducing Rejections	78
4.1.6	The Three-Step Scheme	78
4.1.7	The Two-Step Scheme	85
4.1.8	Full Diagonal Operator Replacement Sweep	90
4.1.9	A Brief Note on the Truncation Order M	90
4.1.10	Example: TFIM Diagonal Update	93
4.1.11	Summary	94
4.2	Off-Diagonal Updates	95
4.2.1	TFIM Site Operators	97
4.2.2	TFIM Bond Operators	98

4.2.3	The Multibranch Update for the TFIM	99
4.2.4	More General Transitions for Diagonal Bonds	101
4.2.5	The Line Update	102
4.2.6	More General Diagonal Bonds	102
4.2.7	High-Temperature Site Updates	103
4.2.8	Summary	103
4.3	Modifications for the Projector Method	104
4.3.1	Diagonal Update Modifications	104
4.3.2	Off-Diagonal Update Modifications	104
4.4	Performing an SSE simulation	105
4.4.1	Thermal SSE	105
4.4.2	Groundstate Projector SSE	106
5	Simulating Rydberg Atom Arrays	107
5.1	Abstract	107
5.2	Introduction	107
5.3	SSE Implementation for Rydberg atoms	109
5.3.1	Diagonal Update	111
5.3.2	Cluster Updates	112
5.3.3	Groundstate Energy Estimator	115
5.4	Results	115
5.4.1	51 atom 1D chain	116
5.4.2	256 atom 2D array	118
5.5	Future Directions	122
5.6	Conclusions	126

6	Simulating Rydberg Atom Arrays on a Kagome Lattice	128
6.1	Abstract	128
6.2	Introduction	129
6.3	Methods	130
6.3.1	Two dimensional RNNs	130
6.3.2	Variational monte carlo (VMC)	133
6.3.3	Supplementing RNN optimization with annealing	134
6.4	Results	135
6.4.1	Numerical comparisons	138
6.5	Conclusions and Outlooks	139
7	Machine Learning Applications of Rydberg Simulation Data	140
7.1	Data-Driven Training	141
7.2	Data-Enhanced Variational Monte Carlo Simulations for Rydberg Atom Ar- rays	142
7.2.1	Background	143
7.2.2	RNN training procedures	144
7.2.3	Data-enhanced VMC	148
7.3	RydbergGPT	149
7.3.1	Introduction	149
7.3.2	Rydberg atom array physics	149
7.3.3	Transformer architecture	150
7.3.4	Training dataset generation	151
7.3.5	Training transformer models	152
7.3.6	Results	152
7.4	Conclusions	154
8	Conclusions	155
8.1	Summary	155
8.2	Discussion and Future Directions	156

Letters of Copyright Permission	157
References	160
APPENDICES	184
A Super-Multiplicativity of the Metropolis Function	185

List of Figures

2.1	Example of an unnormalized probability distribution.	9
2.2	Example of a target distribution plotted alongside a proposal distribution.	11
2.3	Example of a discrete distribution.	14
2.4	Comparison of the Metropolis-Hastings and Heat-Bath algorithms for a binary random variable.	36
3.1	Example of an SSE simulation cell for a TFIM.	48
4.1	Examples of bond and site operators which may exist in an SSE simulation.	95
4.2	Cluster moves for the TFIM site operators.	97
4.3	Multibranch cluster construction for bond operators.	100
4.4	Examples of multibranch clusters in an SSE simulation cell.	101
4.5	Line cluster construction for bond operators.	102
4.6	Examples of line clusters in an SSE simulation cell.	103
5.1	A groundstate SSE simulation cell example for a Rydberg Hamiltonian.	111
5.2	Examples of multibranch and line clusters for a Rydberg Hamiltonian.	113
5.3	Multibranch cluster construction for Rydberg bond operators.	114
5.4	The energy density and the simulation cell size for a 51-site chain of Rydberg atoms as functions of temperature and detuning.	117
5.5	Energy density vs inverse projector length for a 51-site chain of Rydberg atoms.	118

5.6	Comparison of Line and Multibranch update performance across a Quantum Phase Transition for Rydberg atoms on a 51-site chain.	119
5.7	Comparison of Line and Multibranch update performance across a Quantum Phase Transition for Rydberg atoms on a 16×16 square lattice.	120
5.8	Cluster size histograms for the Line and Multibranch updates for a 16×16 square lattice array of Rydberg atoms.	122
5.9	Frequency heatmap of cluster counts vs interaction truncation for a 16×16 square lattice array of Rydberg atoms.	123
6.1	Illustrations of a positive RNN wavefunction and its sampling procedure on a Kagome lattice.	131
6.2	Various non-local quantities estimated using a 2DRNN in the purported “spin glass” phase	136
7.1	Energy density difference of data-driven trained RNN states	145
7.2	Quantum state reconstruction of an $N = 16 \times 16$ atom array	147
7.3	Overview of RydbergGPT	150
7.4	RydbergGPT predictions across the disordered-to-checkerboard transition .	153

List of Tables

6.1	Table of energy densities obtained by QMC and 2DRNN wavefunctions . . .	137
7.1	Table of QMC energy densities	145

List of Algorithms

1	Inverse Transform Sampling	8
2	Inverse Transform Sampling (Binary Search)	9
3	Rejection Sampling	13
4	Sampling from an Alias Table	15
5	Constructing an Alias Table	17
6	The Diagonal Operator Replacement Move	83
7	The Three-Step Diagonal Update Algorithm	84
8	The Diagonal Operator Replacement Move (Two-Step Scheme Variant)	88
9	The Two-Step Diagonal Update Algorithm	89
10	The Diagonal Operator Replacement Sweep	90
11	The Diagonal Update Algorithm (Projector Variant)	105

Chapter 1

Introduction

Rydberg atom arrays have emerged as a promising platform for quantum computing and simulation [181, 34, 122, 17]. Neutral atoms are arranged through the use of optical tweezers into arbitrary lattices [53, 9, 50], allowing experimentalists to probe extremely rich physics including many phases and phase transitions of theoretical interest [50, 187, 141, 233]. In order to verify the correctness of the experimental systems, as well as to probe system parameters which may currently be inaccessible to experiments, it is necessary to develop effective numerical simulation techniques for such systems.

This thesis aims to summarize some progress towards that goal, but first let us briefly review the physics of Rydberg atom arrays.

1.1 The Rydberg Hamiltonian

A Rydberg atom is a neutral hydrogen-like atom, such as Rubidium, in which the valence electron is excited by lasers into a high principle quantum number state. Labelling an excited state as $|r\rangle$ and an atomic groundstate as $|g\rangle$, we have the non-interacting Hamiltonian for each atom:

$$H = \frac{\Omega}{2} \sum_i (|r_i\rangle\langle g_i| + |g_i\rangle\langle r_i|) - \delta \sum_i |r_i\rangle\langle r_i| \quad (1.1)$$

where Ω and δ are laser parameters known as the *Rabi frequency* and *detuning*, respectively.

Although charge neutral, when in an excited state (which we often call the Rydberg state) the atom becomes a dipole. Two excited atoms will therefore interact via a dipole-dipole interaction:

$$V_{dd} = \frac{1}{4\pi\epsilon_0} \frac{\mathbf{d}_1 \cdot \mathbf{d}_2 - 3(\mathbf{d}_1 \cdot \mathbf{n})(\mathbf{d}_2 \cdot \mathbf{n})}{r_{12}^3} \quad (1.2)$$

where \mathbf{d}_i are the dipole moments of the atoms, r_{ij} is the distance between them, and \mathbf{n} is a unit vector pointing from one atom to the other.

This interaction can be treated formally as a perturbation to the atomic Hamiltonian, and we find that the first-order perturbation goes to zero, leaving the second-order term as the dominant interaction [232]. We then have a two-site interaction term which we can express in the following way:

$$H_{\text{int}} = \Omega \sum_{i < j} \left(\frac{R_b}{r_{ij}} \right)^6 n_i n_j \quad (1.3)$$

where $n_i = |r_i\rangle\langle r_i|$ is the Rydberg occupation, and R_b is the so-called *blockade radius*, a length-scale within which two simultaneously excited atoms are heavily penalized energetically. This blockade phenomenon forms the core feature of Rydberg atom arrays. We emphasize that the pre-factor of Ω does not indicate that the interaction is dependent on the lasers used to control the atoms; we include it in order to allow for the definition of a convenient energy scale.

Proposals to exist for both analog (annealer-based) and digital (circuit-based) quantum computation using Rydberg atoms. In the case of digital quantum computation, qubits are encoded in two hyperfine atomic groundstates. The Rydberg state is then used as an intermediate state which, when combined with the blockade effect, allows for the implementation of two-qubit gates [34, 232].

On the other hand, analog quantum computation with Rydberg atoms encodes qubits in the space spanned by the atomic groundstate and the Rydberg state. Putting together the two parts of the Hamiltonian, we get the full Rydberg Hamiltonian which will be the focus of this thesis:

$$H = \Omega \sum_{i < j} \left(\frac{R_b}{r_{ij}} \right)^6 n_i n_j - \delta \sum_i n_i + \frac{\Omega}{2} \sum_i (|r_i\rangle\langle g_i| + |g_i\rangle\langle r_i|) \quad (1.4)$$

By placing the atoms in specific locations, the blockade effect allows for the encoding of solutions to challenging combinatorial problems, such as the Maximum Independent Set

(MIS) problem [49, 233, 151]¹, into the Hamiltonian’s groundstate which may be found through quantum annealing, in which the . However, quantum annealing is not always guaranteed to converge to the correct state, which can occur when an improper annealing schedule is used. As such, exact simulations of both groundstate and thermal state properties at various system parameters and lattice geometries may be useful for validating experimental results. Furthermore, the groundstates are known to host extremely rich phase diagrams [12, 50]. Additionally, the Rydberg Hamiltonian’s long-range interaction makes it an interesting case study for the development of novel simulation schemes.

As we will see in Chapter 5, the Rydberg Hamiltonian in Eq. 1.4 can be transformed to have only non-positive matrix elements, meaning that it is amenable to simulation via Quantum Monte Carlo (QMC) methods. QMC methods are advantageous for this system as they are exact simulation methods whose time and space complexities scales polynomially in the inverse temperature β and the number of sites in the Rydberg atom array. It is, in principle, capable of simulating arbitrary lattice geometries unlike Tensor Network based methods like the Density-Matrix Renormalization Group (DMRG) algorithm [224, 225, 182] which scales poorly for two-dimensional lattices. We will develop a QMC algorithm for simulating groundstate and thermal properties of Rydberg Hamiltonians using the Stochastic Series Expansion (SSE) formalism, a standard formalism for QMC simulations of spin systems which boasts linear time and space scaling in the system size and inverse temperature.

1.2 Outline

This thesis is laid out as follows. We will begin with Chapter 2, where we will develop the elementary principles of Monte Carlo simulation, in particular the extremely powerful Markov Chain Monte Carlo (MCMC) method. The chapter will conclude with a discussion of error estimation for (MCMC) data.

Then, in Chapter 3, we will develop the Stochastic Series Expansion formalism, for both ground and thermal states, upon which we will eventually build our Rydberg Hamiltonian simulation. We will also discuss how to measure various properties of interest.

Next, Chapter 4 will construct the various updates necessary to perform an effective SSE simulation. We will begin these constructions from first principles as, to the author’s knowledge, no such detailed reference is easily available. The chapter will draw similarities

¹The variant of MIS that is typically implemented on Rydberg arrays is NP-complete [49].

between the groundstate and thermal state SSE updates, as well as discuss advantages and drawbacks of the different update styles derived.

The Rydberg SSE simulation is finally explained and tested in Chapter 5, along with some discussion of the presented scheme's limitations and suggestions for future improvements.

We then move on to applications for the SSE simulation, beginning with Chapter 6 in which we perform simulations of Rydberg arrays on a Kagome lattice using a variational wavefunction and comparing against SSE as a baseline. In particular, the chapter focuses on the purported existence of a spin glass phase claimed in Ref. [236].

Finally, Chapter 7 deals with some Machine Learning applications of the data produced by the SSE simulation. This begins with a proposal to pretrain variational wavefunctions with measurement data from the target groundstate in order to accelerate convergence of the variational training procedure. Following this, we briefly summarize the RydbergGPT model, a generative pretrained transformer model which is trained on measurement data from Rydberg arrays simulated at many different points in parameter space, and we find that in the cases where the target system was a groundstate, the RydbergGPT model is able to predict the correct expectation values of observables when evaluated at parameter points that were not in its training set.

Chapter 2

Monte Carlo Simulations

Many quantities of interest in Many-Body Physics can be expressed as integrals or sums (or ratios thereof). However, as the dimensionality of the system increases, so does the dimensionality of the integral or summation, and an exact computation quickly becomes intractable. Instead, we approximate the integral or summation by sampling terms to estimate the quantity of interest. This is the essence of the Monte Carlo method.

Consider the prototypical Ising Model, a classical many-body system meant to serve as a simple model of a magnet. We place M spins $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \dots, \sigma_M)$, where each σ_i may take the values ± 1 , onto the sites of a lattice. These spins interact with their nearest neighbours with coupling strength J . The energy of a given configuration of spins is given by:

$$E(\boldsymbol{\sigma}) = E(\sigma_1, \sigma_2, \dots, \sigma_M) = -J \sum_{\langle i,j \rangle} \sigma_i \sigma_j \quad (2.1)$$

Say we wish to compute the energy of the Ising Model at inverse temperature β . The average energy of the canonical ensemble is given by:

$$E(\beta) = \frac{1}{Z(\beta)} \sum_{\boldsymbol{\sigma}} E(\boldsymbol{\sigma}) e^{-\beta E(\boldsymbol{\sigma})} \quad (2.2)$$

where $Z(\beta) = \sum_{\boldsymbol{\sigma}} e^{-\beta E(\boldsymbol{\sigma})}$ is the partition function. The sum over configurations $\boldsymbol{\sigma}$ contains a very large number of terms (2^M , in fact), which is difficult to carry out exactly. We therefore estimate the ratio of these two intractable sums by randomly sampling terms

in both sums. As a first attempt, we may try to sample the configurations $\boldsymbol{\sigma}_\ell$ uniformly randomly from the configuration space, estimating the energy as:

$$E(\beta) \approx \frac{(1/N) \sum_{\ell=1}^N E(\boldsymbol{\sigma}_\ell) e^{-\beta E(\boldsymbol{\sigma}_\ell)}}{(1/N) \sum_{\ell=1}^N e^{-\beta E(\boldsymbol{\sigma}_\ell)}} \quad (2.3)$$

However, this could take quite a lot of samples in order to produce a sensible estimate of $E(\beta)$. Indeed, when sampling uniformly, it is possible that the configurations with the largest weight $e^{-\beta E(\boldsymbol{\sigma})}$ are never sampled and therefore never contribute to the sum! Additionally, since we are estimating a ratio, statistical noise will enter both the numerator and the denominator. Instead we attempt to sample configurations in proportion to their weights. Defining the probabilities $p(\boldsymbol{\sigma}) = e^{-\beta E(\boldsymbol{\sigma})}/Z(\beta)$, we then draw N samples $\boldsymbol{\sigma}_\ell \sim p(\boldsymbol{\sigma})$ and estimate the energy as

$$E(\beta) \approx \frac{1}{N} \sum_{\ell=1}^N E(\boldsymbol{\sigma}_\ell) \quad (2.4)$$

Notice that we managed to turn a ratio of two averages into just a single average by an appropriate choice of the probability function.

Devising methods to generate samples from the configuration space according to a specific weight function is usually non-trivial, and can become more difficult as the dimensionality of the configuration space grows. We will head down the path of developing a theory which will provide us with ways to do that, but first, we will begin by discussing the simpler cases of directly sampling from small, low-dimensional distributions in Sec. 2.1. Such methods will eventually form vital components of more complex sampling procedures over high-dimensional spaces. In Sec. 2.2, we will give an overview of the theory of Markov Chains and develop the key ideas behind the powerful Markov Chain Monte Carlo (MCMC) method. Finally, we will discuss how to properly estimate error bars of estimators computed through MCMC in Sec. 2.3.

2.1 Direct Sampling

Direct sampling methods—that is, methods which are able to directly sample from a given target distribution—although extremely powerful, are limited to distributions which are low-dimensional. We will begin by discussing Inverse Transform Sampling in Sec. 2.1.1, followed by Rejection Sampling in Sec. 2.1.2. These two methods are able to sample from

both discrete or continuous distributions identically. We then move on to a discussion of Alias Method in Sec. 2.1.3 which, although restricted to discrete distributions of finite size, allows us to generate samples extremely fast.

2.1.1 Inverse Transform Sampling

One of the simplest methods for drawing exact samples of a random variable is *Inverse Transform Sampling*. As the name implies, it draws samples from the target distribution by inverting some function. Consider a random variable X with a (normalized) probability distribution given by $f(x)$, the function we must invert is the cumulative distribution function (CDF) of the random variable:

$$F(x) = \int_{-\infty}^x f(x') dx' \tag{2.5}$$

As a result, this method can only be used for variables whose CDF can be inverted efficiently.

The method involves first drawing a uniform random variate U from the interval $[0, 1]$, and then computing $X = F^{-1}(U)$. The random variate X will have probability distribution $f(X)$. The proof of this fact is quite short:

$$P(X \leq x) = P(F^{-1}(U) \leq x) = P(U \leq F(x)) = F(x) \tag{2.6}$$

There are some subtleties in how to rigorously define the inverse CDF $F^{-1}(u)$, but these considerations are outside of the scope of this thesis.

2.1.1.1 A Simple Implementation for Discrete Distributions

If we have a (finite) discrete random variable with K events, having (normalized) probability vector $\mathbf{p} = (p_1, p_2, \dots, p_K)^\top \in \mathbb{R}^K$, then we can draw a sample from the distribution using Inverse Transform Sampling¹. We draw a random variate u from the uniform distribution $U[0, 1]$ and find the smallest index k such that $F(k) = \sum_{i=1}^k p_i \geq u$. We provide pseudocode for this method in Algorithm 1.

¹Though we label the events using the indices $1, 2, \dots, K$, the random variable can be any object we wish. We use the integer indices as convenient labels for these events.

Algorithm 1 Inverse Transform Sampling

Require: $\mathbf{p} \in \mathbb{R}^K$ such that $\sum_{i=1}^K p_i = 1$, and $p_i \geq 0 \forall i = 1, 2, \dots, K$

```
1:  $u \sim U[0, 1]$ 
2:  $F \leftarrow 0$ 
3: for  $k = 1, \dots, K$  do
4:    $F \leftarrow F + p_k$ 
5:   if  $u \leq F$  then
6:     return  $k$ 
7:   end if
8: end for
```

2.1.1.2 Improving Sampling Time

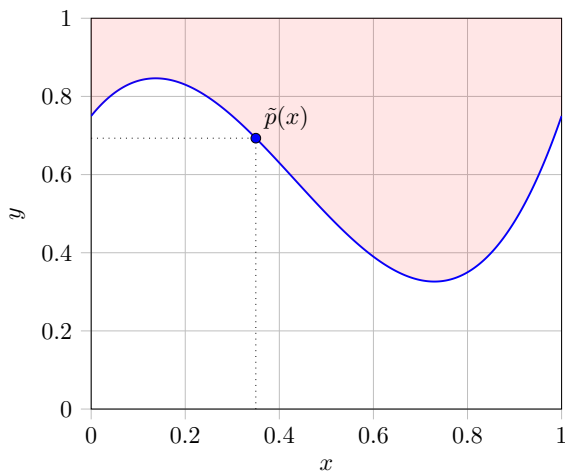
Although Algorithm 1 is perfectly capable of sampling from the correct distribution, it is not very efficient when we need to draw many samples from a distribution which does not change. Consider what exactly we are doing in that algorithm: we are computing the CDF $F(k)$ by summing up probabilities p_k until we find a k such that $u \leq F(k)$. If we have to generate many samples from the distribution then we should not be recomputing all the necessary CDF entries every time. We can pre-compute the CDF as a cumulative sum and store the result in a new vector $\mathbf{F} = (F(1), F(2), \dots, F(K))^T$. This requires a one-time cost of $O(K)$ time. While using this vector does save us quite a bit of computation, we are still not yet done. Considering the time complexity of this procedure, we see that in the worst case we will need to iterate through the entire vector \mathbf{F} , which requires $O(K)$ time. In the average case, we will need $O(\mu)$ time per sample, where $\mu = \sum_k k p_k$ is the mean of the distribution.

Noting that the vector \mathbf{F} is monotonically increasing²—which is to say it is sorted—the process of finding the smallest index k such that $u \leq F_k$ can be accomplished with a binary search. This involves successively halving the search space in which we know k must lie, giving time-complexity $O(\log K)$ per sample, which is on average much faster than the simple linear search given in Algorithm 1. Pseudocode for this sampling procedure is given in Algorithm 2.

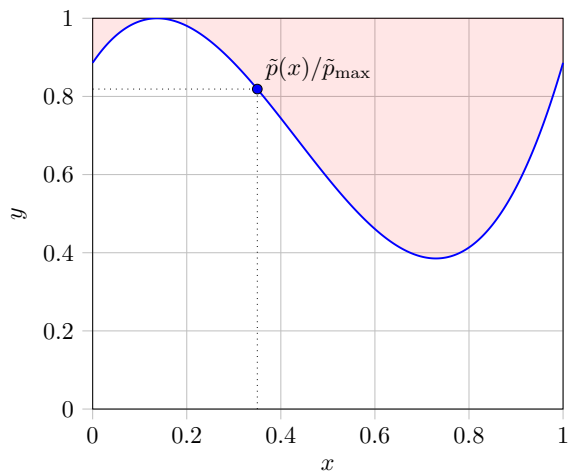
Algorithm 2 Inverse Transform Sampling (Binary Search)

Require: $F \in [0, 1]^K$

```
1:  $u \sim U[0, 1]$ 
2:  $l \leftarrow 1$ 
3:  $r \leftarrow K$ 
4: while  $l \neq r$  do
5:    $m \leftarrow \lfloor \frac{l+r}{2} \rfloor$ 
6:   if  $F_m < u$  then
7:      $l \leftarrow m + 1$ 
8:   else
9:      $r \leftarrow m$ 
10:  end if
11: end while
12: return  $l$ 
```



(a) An unnormalized probability distribution. The red region above the curve represents the proportion of samples which are rejected if we use a uniform proposal distribution.



(b) The unnormalized probability distribution rescaled by its maximum value. Note the reduction in the rejection area above the curve.

Figure 2.1: Example of an unnormalized probability distribution we may wish to draw samples from with rejection sampling.

2.1.2 Rejection Sampling

Let's say we wish to draw a random variable X (which could be continuous or discrete) from a (possibly unnormalized) distribution $\tilde{p}(x)$ that may be difficult to invert. For the sake of simplicity, and without loss of generality, we will assume (for now) that $\tilde{p}(x) \leq 1$ over the domain of interest \mathcal{D} . We will use the example function shown in Fig. 2.1a throughout this discussion to illustrate the rejection sampling procedure.

2.1.2.1 Rejection Sampling using Uniform Random Numbers

We may draw a sample from the target distribution \tilde{p} by first sampling uniformly from its domain: $x \sim U[\mathcal{D}]$. Next, we sample a uniform random number $y \sim U[0, 1]$. If $y \leq \tilde{p}(x)$, then we keep the sample x , otherwise we reject it and start over. In other words, we keep the sample x with probability $\tilde{p}(x)$. We can see that this process will keep samples in proportion to the weight assigned to them by the function \tilde{p} , hence we have devised a method to draw samples from the target distribution.

Now, we of course want to minimize the number of samples rejected by this procedure. We can think of our sampling procedure as drawing samples uniformly from the x - and y -axes of the graph shown in Fig. 2.1a, and then rejecting those which fall above the curve given by the function \tilde{p} —that is, the samples which fall in the red shaded area. In order to minimize the rejection rate, we want to make the red region as small as we can.

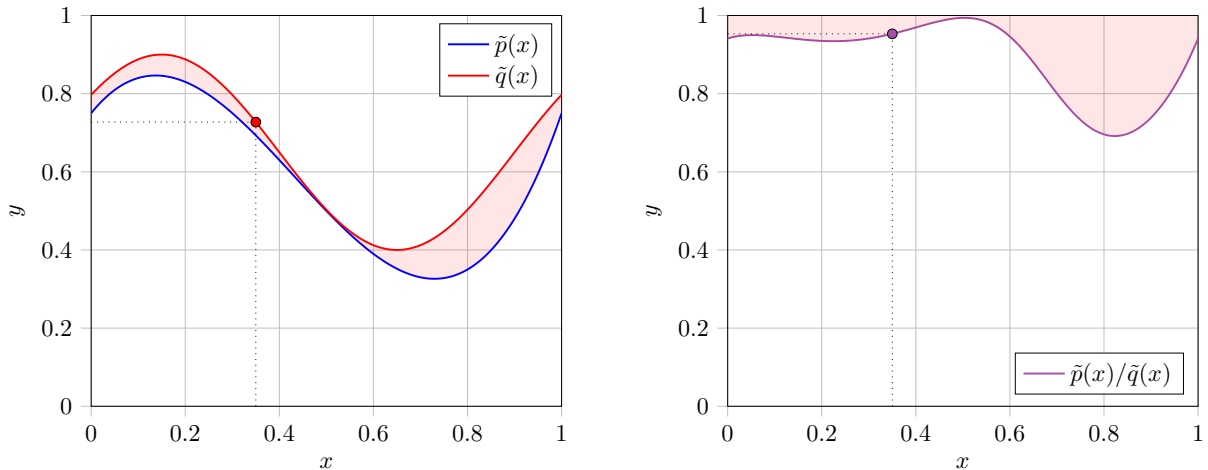
One obvious adjustment that can be made is to rescale the function \tilde{p} by dividing it by its maximum value³, \tilde{p}_{\max} , thus guaranteeing that there will be at least one point where the sample is accepted with unit probability, as shown in Fig. 2.1b. Equivalently, we may sample $y \sim U[0, 1]$, and then keep the sample x if $y \cdot \tilde{p}_{\max} \leq \tilde{p}$.

2.1.2.2 Using Non-uniform Proposal Distributions

In fact, we may consider a procedure where the value that we use to rescale y varies with x . Let's call this rescaling function \tilde{q} . It is clear that we need $\tilde{q}(x) \geq \tilde{p}(x)$ for all $x \in \mathcal{D}$; see Fig. 2.2a for an example. A consequence of this condition is that \tilde{q} can never be zero for an input x unless \tilde{p} is also zero for that same input. In other words, the support of \tilde{q}

²There may be cases where $F_k = F_{k+1}$, but this means that the event $k + 1$ has probability zero so it can be ignored.

³This of course assumes that the maximum value is easy to find or compute, otherwise we will need to make do with some other value which cannot exceed the maximum value.



(a) A target distribution $\tilde{p}(x)$ and proposal distribution $\tilde{q}(x)$. The red region between the curves represents the proportion of samples which are rejected.

(b) The ratio of the two unnormalized probability distributions. Note the reduction in the rejection area above the curve compared to Fig. 2.1a.

Figure 2.2: Example of a target distribution plotted alongside a proposal distribution.

must contain the support of \tilde{p} : $\text{supp}(\tilde{p}) \subseteq \text{supp}(\tilde{q})$. We should note that up until now, by using the uniform proposal distribution, we implicitly assumed that the domain of \tilde{p} , \mathcal{D} , was finite in volume. A nice consequence of the generalization to a non-uniform proposal \tilde{q} is that we can now simulate random variables which can take an infinite range of values, such as the Poisson or Gamma random variables.

If we draw x uniformly from \mathcal{D} , we can say that if $y \sim U[0, 1]$ satisfies $y > \tilde{q}(x)$ then x will obviously be rejected since $\tilde{q}(x) \geq \tilde{p}(x)$. Crucially, we will only need to check that $y \leq \tilde{p}(x)$ if $y \leq \tilde{q}(x)$, that is, if x is already a sample from the distribution \tilde{q} . This will be true by construction if we can directly draw samples from \tilde{q} cheaply, which will be our second condition on the distribution. Hence, we call \tilde{q} the *proposal distribution*. Just like with \tilde{p} , the proposal distribution need not be normalized.

So, given a sample $x \sim \tilde{q}$, we must then draw y uniformly from the range $[0, \tilde{q}(x)]$ and keep the sample x if $y \leq \tilde{p}(x)$. Drawing $y \sim U[0, \tilde{q}(x)]$ is equivalent to drawing $y \sim U[0, 1]$ and then multiplying y by $\tilde{q}(x)$, meaning we accept x if $y \cdot \tilde{q}(x) \leq \tilde{p}(x)$. Dividing both sides of this condition by the proposal distribution gives: $y \leq \tilde{p}(x)/\tilde{q}(x)$, meaning we accept the sample x with probability $\tilde{p}(x)/\tilde{q}(x)$. We then see that the total probability of our

sampling procedure is given by:

$$\tilde{q}(x)P(x \text{ is accepted}) = \tilde{q}(x)\frac{\tilde{p}(x)}{\tilde{q}(x)} = \tilde{p}(x) \quad (2.7)$$

exactly as we wanted. The acceptance probability if x is drawn from \tilde{q} is shown in Fig. 2.2b, where we see that overall the area above the curve has decreased significantly compared to Fig. 2.1a. We can improve this further again by finding a suitable constant $\tilde{M} \geq \tilde{p}(x)/\tilde{q}(x)$ for all $x \in \mathcal{D}$, and rescaling the ratio by dividing it by \tilde{M} . In practice, \tilde{M} may not be equal to $\max_{x \in \mathcal{D}}(\tilde{p}(x)/\tilde{q}(x))$ as this quantity may be time consuming to compute. Instead the best that we can do is find a value of \tilde{M} that is reasonably close and at least as large, but not smaller.

2.1.2.3 Average Acceptance Probability

We now turn to computing the average acceptance probability of the above procedure. It will be useful to now work with the normalized versions of the probability distributions we defined above. Let $p(x) = \tilde{p}(x)/\mathcal{P}$, and $q(x) = \tilde{q}(x)/\mathcal{Q}$, where the calligraphic letters are the normalizing constants of the respective distributions. Let M be the scaling factor for the normalized distributions, $M = \tilde{M}\mathcal{Q}/\mathcal{P}$, and M satisfies $M \geq p(x)/q(x)$ for all $x \in \mathcal{D}$.

$$\langle P(X \text{ is accepted}) \rangle_X = \int_{\text{supp}(q)} q(x)P(x \text{ is accepted}) dx \quad (2.8)$$

$$= \int_{\text{supp}(q)} q(x)P\left(U \leq \frac{p(x)}{Mq(x)}\right) dx \quad (2.9)$$

$$= \int_{\text{supp}(q)} q(x) \left(\frac{p(x)}{Mq(x)}\right) dx \quad (2.10)$$

$$= \int_{\text{supp}(q)} \frac{p(x)}{M} dx \quad (2.11)$$

$$= \frac{1}{M} \left(\int_{\text{supp}(p)} p(x) dx + \int_{\text{supp}(q) \setminus \text{supp}(p)} p(x) dx \right) \quad (2.12)$$

$$= \frac{1}{M}(1 + 0) \quad (2.13)$$

$$= \frac{1}{M} \quad (2.14)$$

Hence, the average acceptance rate is given by $1/M$. It is clear that this means that M must be at least 1, but is otherwise unbounded. We can think of M as the average number of iterations needed before a sample from the correct distribution is generated. Thus, we want M to be as close to 1 as is feasible, maximizing the average acceptance rate. In practice M will be larger than one, as $M = 1$ would imply that $p(x) = q(x)$ for all $x \in \text{supp}(q)$, which would mean we did not need to use rejection sampling at all!

We summarize the rejection sampling procedure in Algorithm 3.

Algorithm 3 Rejection Sampling

Require: p, q, M

```

1: repeat
2:    $x \sim q(x)$ 
3:    $y \sim U[0, 1]$ 
4: until  $y \leq \frac{p(x)}{Mq(x)}$ 
5: return  $x$ 

```

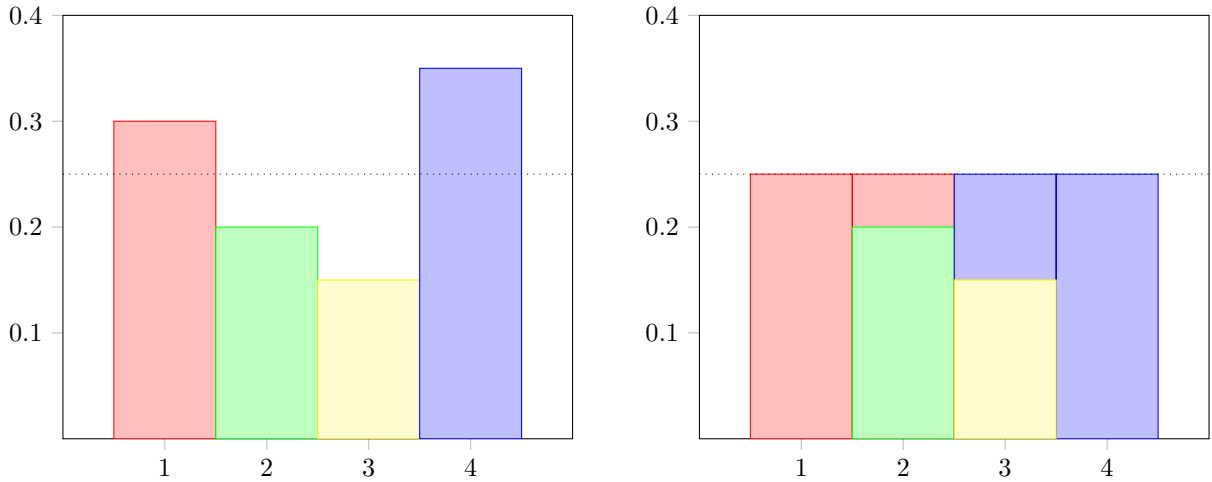
2.1.2.4 Example: Sampling from a Conditional Distribution

A basic example of rejection sampling, which we will make use of in Chapter 3, will be for conditional sampling. Say we have a joint distribution $p(x, y)$ that we can sample from easily. If we wish to draw samples (x, y) for which $y = y_0$ —that is, we wish to draw samples from the conditional distribution $p(x, y|y = y_0)$ ⁴—then we may use a rejection sampling procedure. In this case, we repeatedly sample from $p(x, y)$: if $y \neq y_0$, then $p(x, y \neq y_0|y = y_0) = 0$ and the acceptance probability is of course zero. Otherwise, if $y = y_0$, we keep the sample with probability 1, meaning $p(x|y_0) = Mp(x, y_0)$. Using the chain rule of probabilities on the joint distribution gives $p(x, y) = p(y)p(x|y)$, hence $M = 1/p(y)$; the proportion of accepted samples is $p(y_0)$, as we would expect.

2.1.3 The Alias Method

Although the method given in Algorithm 2 is quite fast when we need to sample from a discrete distribution, we can do better. In fact, it is possible to draw a sample from such

⁴For the sake of defining a sampling procedure and evaluating probabilities it makes sense to sometimes write the conditional distribution as a function over both x and y , however we will often skip writing the explicit y -dependence as it is superfluous.



(a) Example of a discrete distribution with four events. The dotted horizontal line is the average probability $p_{\text{avg}} = 1/4$.

(b) The discrete distribution from Fig. 2.3a represented as an alias table.

Figure 2.3: Example of a discrete distribution along with an equivalent alias table.

a distribution in $O(1)$ time. The Alias Method was invented by Walker in 1974, and is able to do just that [215, 216]. We will introduce this method by first considering a simple rejection sampling procedure for generating samples from a discrete distribution.

Consider a discrete distribution with K events, which has probability vector $\mathbf{p} = (p_1, p_2, \dots, p_K)^\top$. Refer to Fig. 2.3a for a visual example. We can draw samples from the distribution by first uniformly sampling an index $i \sim U\{1, 2, \dots, K\}$, and then accepting the index with probability p_i . This will of course result in many rejections, and although we could rescale by $\max_i p_i$, we will never be able to completely avoid the possibility of rejection.

However, let us consider drawing a horizontal line across the bar chart of probabilities at $p_{\text{avg}} = 1/K$. This value is the average probability in a probability vector of length K . Obviously there will be some probabilities smaller than p_{avg} and some larger. The key insight however, comes from realizing that we can fill up the entire area below the horizontal line by moving the excess probability from a high probability event to one with low probability. Then, if we reject the low probability event, we can instead return the high probability event. This is the main idea behind the Alias Method, and is illustrated in Fig. 2.3b. Given a probability vector, we can construct a vector of cut-off probabilities, $\mathbf{c} = (c_1, c_2, \dots, c_K)^\top$, and a vector of *aliases*, $\mathbf{a} = (a_1, a_2, \dots, a_K)^\top$, where each $a_i \in$

$\{1, 2, \dots, K\}$. We call this pair of vectors an *alias table*. The alias table for a given probability distribution is not unique, which can be seen from parameter counting. Drawing a sample from such an alias table then proceeds as follows: we first uniformly sample an index i from $\{1, 2, \dots, K\}$. Then we draw $u \sim U[0, 1]$ and if $u \leq c_i$ we return the index i as our sample, otherwise we return the index a_i . This process is written out more explicitly in Algorithm 4.

Algorithm 4 Sampling from an Alias Table

Require: $\mathbf{c} \in [0, 1]^K$, $\mathbf{a} \in \{1, 2, \dots, K\}^K$

1: $i \sim U\{1, 2, \dots, K\}$

2: $u \sim U[0, 1]$

3: **if** $u \leq c_i$ **then**

4: | **return** i

5: **else**

6: | **return** a_i

7: **end if**

2.1.3.1 Constructing the Alias Table

We of course need to be able to construct an alias table from a probability vector. Walker’s original algorithm for constructing the alias table was quite inefficient, requiring $O(K^2)$ time. This was eventually rectified by Vose in 1991 who proposed an algorithm requiring only $O(K)$ time [214], which we will discuss here. Although we can write the algorithm in such a way that it can take unnormalized probability distribution vectors as input, for the sake of simplicity we assume that the distribution is already normalized.

We start by rescaling all the probabilities by multiplying by K , $\tilde{p}_i = Kp_i$, and we think of each index $i = 1, 2, \dots, K$ as a “bin” which we want to fill up until it contains total probability weight equal to 1. Divide the scaled probabilities, \tilde{p}_i , into two groups: an “overfull” group for $\tilde{p}_i \geq 1$, and an “underfull” group for $\tilde{p}_i < 1$. Then, we remove one element from each group: call l the element from the underfull group and m the element from the overfull group. We then dump as much probability weight as possible from bin m into bin l , filling up bin l ⁵. Next, we set $c_l = \tilde{p}_l$ and $a_l = m$, and update the weight of m to reflect the probability weight that we just transferred: $\tilde{p}_m \leftarrow \tilde{p}_m - (1 - \tilde{p}_l) = (\tilde{p}_m + \tilde{p}_l) - 1$ ⁶. The element l is now exactly full so we can leave it alone. However, the element m isn’t

⁵We do not want to partially fill up bin l or else we will need to assign it multiple aliases.

⁶The second formula is more numerically stable than the first.

yet, so we sort it into either the underfull or overfull group as necessary. We repeat this process of taking an element from each group until one of the groups is completely empty, at which point we are just about done. The members of whichever group still remain will actually be exactly full (if an index m exists such that $\tilde{p}_m > 1$, there must be an underfull index to balance it out), and all that remains is to initialize their cutoffs and aliases. Since these remaining indices are exactly full and were not removed from the groups earlier, all we need to do is set $c_i = 1$, and $a_i = i$ and we're done! We summarize this procedure in Algorithm 5. The function POP removes an element from the given list and returns it, and the function PUSH adds an element to a list. These functions may add (remove) elements to (from) anywhere in the given list without compromising the correctness of the construction procedure as the alias table is not unique. For simplicity, we can implement the under- and overfull groups as stacks.

2.2 Markov Chain Monte Carlo

The Markov Chain Monte Carlo (MCMC) method is an extremely powerful sampling procedure which especially useful when we have a large set of states \mathcal{C} (which we will refer to as the *configuration space*) that we wish to sample from according to some probability distribution. Since the configuration space is so large, we do not expect to be able to store the entire distribution in memory, and hence we are only able to access the distribution indirectly through some weighting function that takes a configuration as input and outputs a non-negative number which is proportional to the target distribution.

In order to do this, we must first review some results regarding Markov Chains, which we will now define:

Definition 2.1 (Markov Chain). Given a set of states \mathcal{C} , a *Markov Chain* is a stochastic process in which an abstract system in a given state $\mu \in \mathcal{C}$ can transition to another state $\nu \in \mathcal{C}$ with probability $P(\mu \rightarrow \nu)$ which *only* depends on the states μ and ν .

The crucial point is that the transition probabilities $P(\mu \rightarrow \nu)$ do not depend on the history of the Markov Chain, only on the states μ and ν . The stochastic map $P(\mu \rightarrow \nu)$ is called a *transition kernel*. For configuration spaces which are discrete and finite⁷, the transition kernel can be written as a square matrix $P_{\mu\nu} = P(\mu \rightarrow \nu)$, which we call the

⁷We will only consider configuration spaces which are discrete and finite in this chapter. However, much of this theory can be extended to countably infinite configuration spaces, and then (with significantly more effort) to uncountably infinite configuration spaces; but these cases are outside of the scope of this thesis.

Algorithm 5 Constructing an Alias Table

Require: $\mathbf{p} \in \mathbb{R}^K$, $\sum_{i=1}^K p_i = 1$

```
1:  $\tilde{\mathbf{p}} \leftarrow K\mathbf{p}$ 
2:  $\mathbf{c} \leftarrow (0, 0, \dots)$ 
3:  $\mathbf{a} \leftarrow (0, 0, \dots)$   $\triangleright$  Zero vectors of length  $K$ 
4:  $O \leftarrow \{i | \tilde{p}_i \geq 1\}$   $\triangleright$  Overfull group
5:  $U \leftarrow \{i | \tilde{p}_i < 1\}$   $\triangleright$  Underfull group
6: while  $O$  is not empty AND  $U$  is not empty do
7:    $l \leftarrow \text{POP}(U)$ 
8:    $m \leftarrow \text{POP}(O)$ 
9:    $c_l \leftarrow \tilde{p}_l$ 
10:   $a_l \leftarrow m$ 
11:   $\tilde{p}_m \leftarrow (\tilde{p}_m + \tilde{p}_l) - 1$ 
12:  if  $\tilde{p}_m \geq 1$  then
13:     $\text{PUSH}(O, m)$ 
14:  else
15:     $\text{PUSH}(U, m)$ 
16:  end if
17: end while
18: while  $O$  is not empty do
19:    $\triangleright$  All remaining events will be exactly full.  $\triangleleft$ 
20:    $m \leftarrow \text{POP}(O)$ 
21:    $c_m \leftarrow 1$ 
22:    $a_m \leftarrow m$ 
23: end while
24: while  $U$  is not empty do
25:    $\triangleright$  This loop will only run if numerical errors cause an exactly full index to be
       $\text{misclassified as underfull.}$   $\triangleleft$ 
26:    $l \leftarrow \text{POP}(U)$ 
27:    $c_l \leftarrow 1$ 
28:    $a_l \leftarrow l$ 
29: end while
30: return  $\mathbf{c}, \mathbf{a}$ 
```

transition matrix. The only conditions on the transition kernel of a general Markov chain is for all of its values to be non-negative and to satisfy the following normalization condition:

$$\sum_{\nu} P(\mu \rightarrow \nu) = \sum_{\nu} P_{\mu\nu} = 1 \quad \forall \mu \in \mathcal{C} \quad (2.15)$$

The above relation is essentially saying that the transition kernel must map an arbitrary input state μ to *something* in the configuration space, and that for fixed μ , the transitions $P(\mu \rightarrow \nu)$ must form a valid probability distribution over ν . Indeed, $P(\mu \rightarrow \nu)$ is just the conditional probability distribution $P(\nu|\mu)$ written in a notation that emphasizes the fact that the Markov chain is transitioning from one state to another. A matrix which satisfies these conditions is called a *stochastic matrix*, which we now define:

Definition 2.2 (Stochastic Matrix). A square matrix is called *stochastic* if and only if all of its rows or columns form valid probability distributions. In other words, its entries must be non-negative and its rows or columns must each sum to one. A *row-stochastic* matrix is one whose rows each sum to one, while a *column-stochastic* matrix is one whose columns each sum to one.

Markov Chain Monte Carlo operates by building a Markov chain with transition probabilities chosen such that states are visited in proportion to the target distribution we wish to simulate. Taking the Markov chain to be in some state μ_0 , drawn from the target distribution π , we repeatedly apply the transition kernel to the current state, thereby allowing the chain to evolve and producing more configurations drawn from π . The number of applications of the transition kernel is analogous to time, hence we often refer to these time-steps as *Markov time*. Expectation values can then be estimated as:

$$\langle A \rangle = \frac{1}{T} \sum_{t=1}^T A(\mu_t) \quad (2.16)$$

where A is the quantity being estimated, μ_t is the state of the Markov chain at time-step t , and T is the number of time-steps for which we have allowed the chain to evolve. MCMC allows us to convert an integral over the configuration space into a sum over (Markov) time.

The rest of this section will be organized as follows. We will begin by deriving the *Balance Equation* a condition our transition probabilities must satisfy in order to allow sampling from the target distribution in Sec. 2.2.1, as well as discussing the concept of *ergodicity* in Sec. 2.2.2, which is the second condition necessary for our goal. Following this, we will move on to discussing some ideas which will be useful for actually constructing a practical

transition kernel, beginning with the composition of transition kernels in Sec. 2.2.3, and the simplifying constraint on the transition kernel known as the Detailed Balance condition in Sec. 2.2.4. Two general methods to build transition kernels that satisfy the detailed balance condition will then be introduced: the Metropolis-Hastings and Heat-Bath algorithms in Secs. 2.2.5 and 2.2.6, respectively. Finally, we will offer a simple comparison of the two algorithms in Sec. 2.2.7 and discuss when it is advantageous to use one over the other, before concluding with a brief discussion on *equilibration* in Sec. 2.2.8.

2.2.1 The Balance Equation

We wish to find a transition kernel $P(\mu \rightarrow \nu)$ that will produce a Markov chain which will visit states for times proportionate to the target distribution we wish to sample from: $\pi(\mu)$. In order to do this, the transition kernel must satisfy the following property:

Property 2.3 (Stationarity of the Target Distribution). The transition kernel P must leave the target distribution invariant, which is to say, the target distribution must be left stationary.

$$\sum_{\mu} \pi(\mu)P(\mu \rightarrow \nu) = \pi(\nu) \quad (2.17)$$

Remark. This is also known as the *Balance Equation*, or sometimes the *Global Balance* condition.

If the above condition is satisfied we say that P satisfies global balance with respect to π . What this condition means is that if the Markov chain is producing samples in accordance to the distribution π , it will continue to do so, and we do not need to worry about the distribution of samples deviating from π at some later time. Writing the condition slightly differently, we get

$$\pi_{\nu} = \sum_{\mu} \pi_{\mu}P_{\mu\nu} \quad (2.18)$$

$$\boldsymbol{\pi}^{\text{T}} = \boldsymbol{\pi}^{\text{T}}P \quad (2.19)$$

Meaning the stationary distribution is a left-eigenvector of the transition matrix P with eigenvalue one⁸. Note that the above relation does not require π to be normalized, which

⁸We could easily write the transition matrix with the indices reversed, in which case π would be a column vector and a right-eigenvector of the transition matrix, and P would need to be column-stochastic instead of row-stochastic. However, we prefer this notation as (a) it is consistent with the existing literature on Markov Chains and (b) makes it easy to read off the flow of “Markov time” as progressing from left to right: $P_{\mu\nu} = P(\mu \rightarrow \nu)$.

is important as the normalization constant of a high-dimensional distribution is difficult to compute and the entire reason we wish to use Monte Carlo methods to estimate observables was because we want to avoid performing large summations such as those. By the same token, we are often unable to explicitly construct the transition matrix, and must therefore avoid doing so when we devise our transition kernel.

The stationarity property is conceptually simple however it raises a large issue: how do we get the Markov chain to produce samples from π if we do not already have samples from π ? If we initialize the chain with a sample drawn from some initial distribution p_0 then, perhaps after an initial *equilibration* period (see Sec. 2.2.8), we would like the Markov chain to begin producing samples from π . However, stationarity on its own is insufficient to guarantee this for an arbitrary initial distribution. If we think of the sampling process as repeatedly applying the transition matrix to the initial probability distribution written as a row vector, we would then like to be able to show that by applying a large enough power of P to this initial distribution p_0 , we will get π . We define the instantaneous distribution at a given Markov time-step as:

Definition 2.4 (Instantaneous distribution of a Markov Chain). Given a Markov chain with transition kernel P and initial distribution p_0 , the probability distribution of the state of the chain at time-step t is given by:

$$\mathbf{p}_t^\top = \mathbf{p}_0^\top P^t \tag{2.20}$$

In order to analyze the long-time behaviour of p_t , we need to know more about the eigenvalues of the transition matrix P , particularly the eigenvalues of largest magnitude. This brings us to the following definition:

Definition 2.5 (Spectral Radius). The *spectral radius* of a $n \times n$ matrix, A , with eigenvalues $\{\lambda_1, \lambda_2, \dots, \lambda_n\}$ is the maximal absolute eigenvalue:

$$\rho(A) = \max\{|\lambda_1|, |\lambda_2|, \dots, |\lambda_n|\} \tag{2.21}$$

We now make the following claim for stochastic matrices:

Lemma 2.1. *The spectral radius of a stochastic matrix is 1.*

To show this, we will need to make use of a result from linear algebra known as Gelfand's formula [120], which relates the spectral radius to a limit involving a matrix norm:

Theorem 2.2 (Gelfand’s Formula). *Given a square matrix A , and a matrix norm $\|\cdot\|$ induced by a vector norm, the spectral radius of A can be computed as:*

$$\rho(A) = \lim_{n \rightarrow \infty} \|A^n\|^{1/n} \quad (2.22)$$

With this formula available to us, the proof of Lemma 2.1 is quite simple.

Proof of Lemma 2.1. For a row-stochastic matrix, S , we may compute the ∞ -norm:

$$\|S\|_\infty = \max_\mu \sum_\nu |S_{\mu\nu}| = \max_\mu \sum_\nu S_{\mu\nu} = 1 \quad (2.23)$$

where the second equality follows from the non-negativity of the matrix elements of S , and the final equality follows from the fact that S is row-stochastic. On the other hand, if S were a column-stochastic matrix, we would use the 1-norm:

$$\|S\|_1 = \max_\nu \sum_\mu |S_{\mu\nu}| = 1 \quad (2.24)$$

These hold for any power of S as well, as the sets of row- and column-stochastic matrices are each closed under matrix multiplication. Applying Gelfand’s formula we find that:

$$\rho(S) = \lim_{n \rightarrow \infty} \|S^n\|^{1/n} = \lim_{n \rightarrow \infty} 1^{1/n} = 1 \quad (2.25)$$

□

Much is still unknown about the transition matrix P : can we be sure that π is the *only* left-eigenvector of P with eigenvalue of magnitude 1? We also do not know how many eigenvalues of magnitude 1 there are, and whether they are positive, negative, or complex. These questions may be answered by the Perron-Frobenius theorem:

Theorem 2.3 (The Perron-Frobenius Theorem). *Given a matrix A whose entries are strictly positive, we have that A has a positive eigenvalue equal to the spectral radius. Additionally, the eigenvector associated with this eigenvalue is also unique (upto scalar multiplication) and has strictly positive entries.*

The eigenvector having only positive entries may seem problematic at first as the target distribution we wish to simulate may assign some states a weight of zero, however the transition kernel only needs to act on the support of the target distribution, not on its full domain.

As it stands the theorem is not quite applicable to transition matrices whose entries are not guaranteed to be positive, only non-negative. We therefore require an additional condition on P which we will discuss in the next section.

2.2.2 Ergodicity

By invoking the Perron-Frobenius theorem, we have neglected one important detail. The usual form of the theorem only applies to matrices with *strictly positive* entries, meaning we cannot apply it if the matrix has any zero entries. This is problematic as the entries of the transition matrix, P , are only constrained to be non-negative. The generalization of the Perron-Frobenius theorem to non-negative matrices like P requires the matrix to satisfy one additional property: *irreducibility*.

There are many equivalent definitions of irreducibility, but the most relevant one for our purposes is the following definition related to graph theory.

Definition 2.6 (Irreducibility). Given an $n \times n$ matrix A , let $G(A)$ be a directed graph with n vertices, and a directed edge from vertex i to j when $A_{ij} \neq 0$. The matrix A is called *irreducible* if and only if there exists a path (i.e. a sequence of directed edges) in $G(A)$ from i to j for every pair of vertices i and j [140].

In the context of Markov Chains, this means that the transition kernel must be able to take any initial state μ to any final state ν in a finite number of steps with non-zero probability. Thus, if we assume non-negativity, we get the following simpler condition:

Definition 2.7 (Irreducibility for non-negative matrices). An $n \times n$ non-negative matrix A is called *irreducible* if and only if for all $i, j = 1, 2, \dots, n$, there exists an integer $m > 0$ such that

$$(A^m)_{ij} > 0 \tag{2.26}$$

Remark. The power m is in general dependent on the indices i and j .

We call a Markov Chain irreducible if its transition matrix is irreducible. Physicists usually use the word *ergodicity* as a synonym for irreducibility, and we will do the same throughout this thesis.

The Perron-Frobenius theorem for matrices with positive entries guarantees that the eigenvalue with magnitude equal to the spectral radius is both positive and *unique*, meaning all other eigenvalues will have magnitude strictly less than that eigenvalue. Weaker hypotheses result in weaker conclusions, and we have the following weaker form of the theorem for non-negative matrices:

Theorem 2.4 (The Perron-Frobenius Theorem for non-negative matrices). *Given a non-negative irreducible matrix A , there exists a positive eigenvalue $\lambda^* = \rho(A)$, with an associated eigenvector which is unique (upto scalar multiplication) and has strictly positive*

entries. However, λ^* may not be the only eigenvalue of magnitude $\rho(A)$. The eigenvalues of magnitude $\rho(A)$ take the form $\rho(A)e^{2\pi ik/h}$, with $k = 0, 1, 2, \dots, h-1$ where $h = 1, 2, \dots$ is called the period of the irreducible matrix. These other eigenvalues each have an associated unique eigenvector, but its components are not guaranteed to be positive. In fact, the only non-negative eigenvector of P is the eigenvector associated with eigenvalue $\lambda^* = \rho(P)$.

For a transition kernel, we will soon see that the weaker form of the Perron-Frobenius theorem is still sufficient to guarantee convergence to the stationary distribution π . However, due to periodicity, we cannot guarantee that the *instantaneous* distribution p_t will converge to π for an arbitrary initial distribution p_0 .

Indeed, for an irreducible transition kernel with period $h > 1$, if we take the limit of p_t as $t \rightarrow \infty$, we find that it does not exist. A general irreducible transition matrix may not be diagonalizable, so we must restrict ourselves to using the Jordan Normal form $P = Q^{-1}JQ$. The initial distribution p_0 , written as vector, can be expressed in terms of the rows of Q , $\{\mathbf{q}_k^\top\}$:

$$\mathbf{p}_0^\top = \sum_k c_k \mathbf{q}_k^\top \quad (2.27)$$

Applying a large power of P to this probability vector, we get

$$\mathbf{p}_0^\top P^t = \sum_{k=1}^{\mathcal{D}} c_k \mathbf{q}_k^\top P^t \quad (2.28)$$

$$= \sum_{k=1}^{\mathcal{D}} c_k \mathbf{q}_k^\top Q^{-1} J^t Q \quad (2.29)$$

The product $\mathbf{q}_i^\top Q^{-1}$ evaluates to the standard basis (row) vector \mathbf{e}_i^\top which, when multiplying the matrix J^t from the left, simply picks out the matrix's k th row, which we denote as $(J^t)_{k,:}$. The first h rows of J^t are the row vectors $\{\mathbf{e}_1^\top, \mathbf{e}_2^\top, \dots, \mathbf{e}_h^\top\}$, which then select the first h rows of Q , giving:

$$\mathbf{p}_0^\top P^t = \sum_{k=1}^h c_k \lambda_k^t \mathbf{q}_k^\top + \sum_{k=h+1}^{\mathcal{D}} c_k (J^t)_{k,:} Q \quad (2.30)$$

where \mathbf{q}_k is the unique left-eigenvector of P with eigenvalue $\lambda_k = e^{2\pi i(k-1)/h}$. Since all of the Jordan Blocks for $k > h$ have associated eigenvalues with magnitude less than one, the

second term will go to zero as $t \rightarrow \infty$, leaving:

$$\mathbf{p}_t^\top = \mathbf{p}_0^\top P^t = \sum_{k=1}^h c_k e^{2\pi i t(k-1)/h} \mathbf{q}_k^\top \quad (2.31)$$

$$= c_1 \mathbf{q}_1^\top + \sum_{k=2}^h c_k e^{2\pi i t(k-1)/h} \mathbf{q}_k^\top \quad (2.32)$$

The summation contains many oscillating terms whose magnitudes remain constant with t . Hence, the limit $\lim_{t \rightarrow \infty} \mathbf{p}_t$ does not exist. This, however, turns out not to be an issue when constructing our sampling procedure. Instead, we note that it is only the *history* of the Markov chain that must follow the distribution π , not necessarily the instantaneous distribution of the chain at a given time-step $\mathbf{p}_t = \mathbf{p}_0 P^t$. Thus, we have:

Theorem 2.5 (Ergodic Theorem). *Given an initial probability distribution p_0 and an ergodic transition kernel P that leaves the distribution π stationary, the distribution of samples produced by the Markov chain will converge to π :*

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{t=0}^{n-1} \mathbf{p}_t^\top = \boldsymbol{\pi}^\top \quad (2.33)$$

Proof. We begin by taking the average over all time of the instantaneous probability distributions of the Markov chain:

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{t=0}^{n-1} \mathbf{p}_t^\top = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{t=0}^{n-1} \mathbf{p}_0^\top P^t = \mathbf{p}_0^\top \left[\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{t=0}^{n-1} P^t \right] \quad (2.34)$$

The term in the square brackets can be evaluated as a Cesàro average, and will turn out to be an outer product of the left- and right-eigenvectors of P associated with the eigenvalue $\lambda^* = 1$ (we will not prove this, but see Ref. [140] for more details). We already know that the left-eigenvector is $\boldsymbol{\pi}^\top$. From the row-normalization of P we see that the column-vector filled with ones—which we will denote as $\mathbf{1}$ —is the right-eigenvector associated with $\lambda^* = 1$. Hence, we have:

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{t=0}^{n-1} P^t = \frac{\mathbf{1} \boldsymbol{\pi}^\top}{\boldsymbol{\pi}^\top \mathbf{1}} = \mathbf{1} \boldsymbol{\pi}^\top \quad (2.35)$$

which is a matrix with all rows equal to $\boldsymbol{\pi}^\top$. Left multiplying by the initial probability distribution, we get:

$$\mathbf{p}_0^\top \mathbf{1} \boldsymbol{\pi}^\top = \boldsymbol{\pi}^\top \quad (2.36)$$

□

So long as P is irreducible, and has π as its stationary distribution, the distribution of samples produced by the Markov chain will converge to π . In other words, the conversion of a configuration space integral into a temporal one is well founded. This is a beautiful result, as it allows us to in principle construct a Markov chain to sample from the stationary distribution π with *very few* constraints on the transition kernel P and *no constraints* on the initial distribution p_0 .

Nevertheless, it is advantageous to be able to guarantee that the instantaneous distribution has converged to π as this makes it easier to analyze the autocorrelation between samples drawn from the Markov chain. It therefore behooves us to at least understand the phenomenon of periodicity, which will allow us to prevent it when we construct Markov chains.

2.2.2.1 Periodicity

For a general non-negative matrix, the period of an index is defined as:

Definition 2.8 (Period of a non-negative matrix). Given an $n \times n$ non-negative matrix A , and index $i = 1, 2, \dots, n$, the *period* of i is a positive integer defined as

$$d(i) = \gcd \{m > 0 \mid (A^m)_{ii} > 0\} \quad (2.37)$$

That is, the period is the greatest common divisor of the lengths of all paths that return i to itself. A state with a period of one is called *aperiodic*.

This again simplifies greatly if the matrix is also irreducible:

Theorem 2.6 (Period of an irreducible matrix). *All indices of a non-negative irreducible matrix A have the same period.*

Proof. Given two indices i and j of the matrix, such that $i \neq j$, their periods are $d(i)$ and $d(j)$. Let I_1, I_2, \dots be the lengths of all paths from i back to i , and similarly, let J_1, J_2, \dots be the lengths of all paths from j back to j . We have:

$$d(i) = \gcd\{I_1, I_2, \dots\} \quad (2.38)$$

$$d(j) = \gcd\{J_1, J_2, \dots\} \quad (2.39)$$

Since the matrix is irreducible, there is a path from i to j and one from j to i . Thus, $d(i)$ divides the length, L , of the path $i \rightarrow j \rightarrow i$. By definition, $d(i)$ must also divide every

$L + J_k$, which implies that $d(i)$ divides every J_k as well and, using the fact that duplicated arguments do not change the gcd, we have that:

$$d(i) = \gcd\{I_1, I_2, \dots, J_1, J_2, \dots\} \quad (2.40)$$

$$= \gcd\{I_1, I_2, \dots, \gcd\{J_1, J_2, \dots\}\} \quad (2.41)$$

$$= \gcd\{I_1, I_2, \dots, d(j)\} \quad (2.42)$$

where in the second line we applied the associativity of gcd, which shows us that $d(i)$ divides $d(j)$. Similarly, we can show that $d(j)$ divides $d(i)$ and hence, $d(i) = d(j)$. \square

Remark. A non-negative irreducible matrix with period one is called *aperiodic*.

Corollary 2.6.1. *If a non-negative irreducible matrix has at least one non-zero diagonal element, then it is aperiodic.*

Proof. By assumption, there is a path of length one from some state to itself, hence the period of that state is one. Then, by Theorem 2.6, we conclude that the matrix is aperiodic. \square

We therefore have a very simple protocol to construct a transition kernel that is guaranteed to be aperiodic given an irreducible kernel P . If we can prove that there exists some state $\mu \in \mathcal{C}$ such that $P(\mu \rightarrow \mu) > 0$, then we are done. Otherwise, we can define a new transition kernel P' as a convex combination of P and the identity map I :

$$P' = rP + (1 - r)I \quad (2.43)$$

where $r \in (0, 1)$ is some probability. This new kernel is very simple to implement: with probability $1 - r$ we simply do nothing for the current Markov time-step, otherwise we apply P as usual. Clearly, P' is irreducible and aperiodic, we just need to make sure it leaves π stationary:

$$\pi^\top P' = r\pi^\top P + (1 - r)\pi^\top I = r\pi^\top + (1 - r)\pi^\top = \pi^\top \quad (2.44)$$

which it does! We may prefer r to be close to unity, in order to make sure we are not spending too many time-steps doing nothing. This construction was used in simulations of the classical Ising model where it was found that the employed transition kernel was periodic at high temperatures, which resulted in a loss of ergodicity when multiple transition kernels were composed sequentially [133].

* * *

In practice, it is often difficult to formally prove irreducibility/ergodicity of a transition kernel without explicitly constructing the transition matrix. We often only provide rough intuition-based arguments for ergodicity.

2.2.3 Composition of Transition Kernels

Oftentimes in practical simulations, one applies several different types of updates which individually satisfy global balance one after another. This is often necessary when it is difficult to produce a single update which is fully ergodic, and it may be simpler to devise multiple updates which are each ergodic within smaller subspaces.

In order to understand why this makes sense, we will consider a simple (abstract) example. Let us denote the configuration space of the Markov Chain, that is, the set of possible states that the Markov Chain may occupy, as \mathcal{C} . We will partition this space into two (non-empty and non-overlapping) sets in K different ways: $\mathcal{C} = \mathcal{A}_i \cup \mathcal{B}_i$, where $\mathcal{A}_i \cap \mathcal{B}_i = \emptyset$ for $i = 1, 2, \dots, K$. Define the transition kernels P_1, P_2, \dots, P_K , and we will assume that P_i is ergodic only on sets \mathcal{A}_i and \mathcal{B}_i individually, but not together. What this means is that the transition kernel P_i can take a state $\mu \in \mathcal{A}_i$ (resp. \mathcal{B}_i) to any other state $\nu \in \mathcal{A}_i$ (resp. \mathcal{B}_i) in a finite number of steps, but no state in \mathcal{A}_i can ever be mapped to a state in \mathcal{B}_i (and vice-versa). However, if the transition kernel P_j , $j \neq i$, has $\mathcal{A}_j \cap \mathcal{A}_i \neq \emptyset$ and $\mathcal{A}_j \cap \mathcal{B}_i \neq \emptyset$, or $\mathcal{B}_j \cap \mathcal{A}_i \neq \emptyset$ and $\mathcal{B}_j \cap \mathcal{B}_i \neq \emptyset$, or both, then P_j will produce transitions between \mathcal{A}_i and \mathcal{B}_i , and applying P_i and P_j sequentially will achieve ergodicity on the full configuration space.

As a second, more concrete, example: consider a simulation of a many-body system with N sites in which we only update a single site at a time. We will denote the state of site i by σ_i , which for simplicity takes only two values: 0, 1. The configuration of the entire system is then given by the N -tuple: $(\sigma_1, \sigma_2, \dots, \sigma_N) \in \{0, 1\}^N$. As the full configuration space is exponentially large, we cannot easily build a transition kernel that acts on the entire configuration at once. Instead, for each site i , we define an associated transition kernel P_i , which can only update the state of that one site, σ_i . Each transition kernel P_i can therefore only map the system between the sets: $A_i^{(0)} = \{(\sigma_1, \sigma_2, \dots, \sigma_N) \in \{0, 1\}^N | \sigma_i = 0\}$, and $A_i^{(1)} = \{(\sigma_1, \sigma_2, \dots, \sigma_N) \in \{0, 1\}^N | \sigma_i = 1\}$, but is unable to have the system explore each of the individual sets. By randomly (or sequentially) selecting single-site transition kernels P_i to apply, we can in principle achieve ergodicity on the entire configuration space.

Although the transition kernels may individually satisfy global balance, we need to prove that their composition does as well.

Theorem 2.7 (Composition of transition kernels). *Consider two transition kernels P_1 and P_2 which both satisfy global balance (with respect to the same distribution π) individually. We then have the composite transition kernel*

$$P(\mu_0 \rightarrow \mu_2) = \sum_{\mu_1} P_1(\mu_0 \rightarrow \mu_1) P_2(\mu_1 \rightarrow \mu_2) \quad (2.45)$$

The composite kernel also satisfies global balance with respect to π .

Proof. To prove that global balance is still satisfied, we start with the left hand side of Eq. 2.17:

$$\sum_{\mu_0} \pi(\mu_0) P(\mu_0 \rightarrow \mu_2) = \sum_{\mu_0} \pi(\mu_0) \sum_{\mu_1} P_1(\mu_0 \rightarrow \mu_1) P_2(\mu_1 \rightarrow \mu_2) \quad (2.46)$$

$$= \sum_{\mu_1} \sum_{\mu_0} \pi(\mu_0) P_1(\mu_0 \rightarrow \mu_1) P_2(\mu_1 \rightarrow \mu_2) \quad (2.47)$$

$$= \sum_{\mu_1} \pi(\mu_1) P_2(\mu_1 \rightarrow \mu_2) \quad (2.48)$$

$$= \pi(\mu_2) \quad (2.49)$$

where in the second and third lines we applied the global balance condition on P_1 and P_2 . \square

Hence, transition kernels which satisfy global balance with respect to π may be composed without compromising global balance. Then, so long as the composite transition kernel is ergodic, the Markov chain will be able to produce samples from the target distribution π .

2.2.4 Detailed Balance

Ergodicity and global balance are the two primary conditions needed to ensure convergence of the transition kernel to its stationary distribution. How to build a chain that satisfies those two properties and whose stationary distribution is the target distribution of interest is still unclear. The presence of the summation over all states in the global balance condition makes it difficult to analyze and reason about. Indeed, global balance and the normalization condition (Eq. 2.15) provide only $O(\mathcal{D})$ constraints to the transition matrix which has \mathcal{D}^2 entries, leaving $O(\mathcal{D}^2 - \mathcal{D})$ free parameters. This is still quite a lot, so we seek to impose a stricter condition than global balance which will still allow us to converge to the stationary distribution but will make it easier to devise a practical implementation

of the necessary transition kernel. Consider again the global balance condition:

$$\sum_{\mu} \pi(\mu)P(\mu \rightarrow \nu) = \pi(\nu) \tag{2.50}$$

$$= \pi(\nu) \sum_{\mu} P(\nu \rightarrow \mu) \tag{2.51}$$

$$= \sum_{\mu} \pi(\nu)P(\nu \rightarrow \mu) \tag{2.52}$$

We now demand that not only should the two sums be equal to each other, but each term on the left hand side should be equal to the associated term on the right hand side. Put more simply, we drop the summations, which gives the famous *Detailed Balance* condition:

Property 2.9 (Detailed Balance). The transition kernel P satisfies detailed balance with respect to the distribution π if and only if

$$\pi(\mu)P(\mu \rightarrow \nu) = \pi(\nu)P(\nu \rightarrow \mu) \tag{2.53}$$

The detailed balance condition constrains roughly half the entries of the transition matrix, which still leaves us a lot of freedom in devising transition kernels, but is much simpler to manipulate than the global balance condition as we no longer need to deal with a summation over a large number of terms. It is clear that the detailed balance condition implies global balance therefore, so long as the transition kernel is ergodic, the Markov Chain will be able to converge to the desired stationary distribution, π .

As we will see in later chapters, the detailed balance condition is a powerful assumption that will allow us to easily devise transition kernels for extremely complex distributions. In fact, the almost all MCMC methods used in practice employ detailed balance to construct their basic transition kernels. However, there has been some work on relaxing the assumption in the hopes of building more efficient simulations [192, 183].

The detailed balance condition is also known as *reversibility*, in reference to the fact that each side of Eq. 2.53 is the time-reversed form of the other—the stochastic dynamics are *reversible*.

2.2.4.1 Composing Reversible Transition Kernels

A natural question to ask is whether compositions of reversible transition kernels give another reversible kernel. This is actually not true. Consider again the composite kernel

from Eq. 2.45. This time we will assume that P_1 and P_2 satisfy detailed balance with respect to π . We then have:

$$\pi(\mu_0)P(\mu_0 \rightarrow \mu_2) = \sum_{\mu_1} \pi(\mu_0)P_1(\mu_0 \rightarrow \mu_1)P_2(\mu_1 \rightarrow \mu_2) \quad (2.54)$$

$$= \sum_{\mu_1} P_1(\mu_1 \rightarrow \mu_0)\pi(\mu_1)P_2(\mu_1 \rightarrow \mu_2) \quad (2.55)$$

$$= \sum_{\mu_1} P_1(\mu_1 \rightarrow \mu_0)P_2(\mu_2 \rightarrow \mu_1)\pi(\mu_2) \quad (2.56)$$

$$= \pi(\mu_2) \sum_{\mu_1} P_2(\mu_2 \rightarrow \mu_1)P_1(\mu_1 \rightarrow \mu_0) \quad (2.57)$$

the kernel P_2P_1 is not necessarily equal to $P = P_1P_2$, hence detailed balance is broken. This is analogous to how the transpose of a product of two matrices is not necessarily equal to the product of transposes. As a consequence of this, detailed balance is often broken implicitly in practical simulations which apply multiple transition kernels in sequence. Of course, as we saw earlier in Sec. 2.2.3, global balance is still satisfied.

However, if we define a transition kernel as a convex combination of reversible transition kernels: $P_{cv}(\mu \rightarrow \nu) = \sum_i w_i P_i(\mu \rightarrow \nu)$, where $w_i \geq 0$ and $\sum_i w_i = 1$, then P_{cv} will be reversible:

$$\pi(\mu)P_{cv}(\mu \rightarrow \nu) = \sum_i w_i \pi(\mu)P_i(\mu \rightarrow \nu) \quad (2.58)$$

$$= \sum_i w_i \pi(\nu)P_i(\nu \rightarrow \mu) \quad (2.59)$$

$$= \pi(\nu)P_{cv}(\nu \rightarrow \mu) \quad (2.60)$$

We may implement P_{cv} by randomly sampling an index i from the probability distribution defined by the weights w_i , and then applying the transition kernel P_i . Crucially, the weights w_i must be independent of the state μ .

As a historical note, it was a source of confusion for a short time why simulations of the classical Ising model would converge to the target distribution when single-spin flip updates were performed in a predetermined sequence—which breaks detailed balance—instead of by randomly selecting the sites—which maintains detailed balance as it would correspond to a convex combination of transition kernels weighted uniformly [133, 159]. This is a testament to the hegemony of the detailed balance paradigm: practitioners almost forgot that it is *not* a necessary condition!

2.2.4.2 Spectral Properties of Reversible Kernels

The constraint of detailed balance turns out to have a profound effect on the eigenspectrum of the transition matrix.

Theorem 2.8. *A transition matrix P which is reversible with respect to a distribution π , is diagonalizable and its eigenvalues are all real.*

Proof. Letting Π be the square matrix with the entries of π on its main diagonal, we define the matrix Q as $Q = \sqrt{\Pi}P\sqrt{\Pi}^{-1}$. Detailed balance implies that this matrix is symmetric:

$$Q_{\mu\nu} = \sqrt{\frac{\pi_\mu}{\pi_\nu}} P_{\mu\nu} = \sqrt{\frac{\pi_\mu}{\pi_\nu} \frac{\pi_\nu}{\pi_\mu}} P_{\nu\mu} = \sqrt{\frac{\pi_\nu}{\pi_\mu}} P_{\nu\mu} = Q_{\nu\mu} \quad (2.61)$$

By the spectral theorem, Q 's eigenvalues must all be real, and this property carries over to P as the two matrices share eigenvalues due to being related by a similarity transformation. \square

Corollary 2.8.1. *An irreducible reversible kernel can have period at most 2.*

Proof. The Perron-Frobenius theorem for an irreducible non-negative matrix A gives us that the eigenvalues of largest magnitude take the form $\lambda_k = \rho(A)e^{2\pi i(k-1)/h}$ where $k = 1, 2, \dots, h$ and $h > 0$ is the period of the matrix. By reversibility, A cannot have complex eigenvalues. Hence, if $h > 1$ the eigenvalues of largest magnitude may only take the values $\pm\rho(A)$, and therefore $h = 2$. \square

Imposing detailed balance does not completely rule out the possibility of periodic behaviour of the Markov chain—as the transition matrix may still have -1 as an eigenvalue—but it has greatly reduced the complexity of this behaviour.

2.2.5 The Metropolis-Hastings Algorithm

We are now prepared to construct a concrete transition kernel that converges to a desired target distribution. Let $\Phi(c) \geq 0$ be a function which assigns weights to configurations c . The weight function Φ defines a probability distribution:

$$\pi_\Phi(c) = \frac{\Phi(c)}{\sum_{c'} \Phi(c')} \quad (2.62)$$

where the normalizing constant in the denominator is too difficult to compute exactly. We wish to construct a transition kernel whose stationary distribution is $\pi_\Phi(c)$.

To do this, we substitute π_Φ into the detailed balance equation, noting that the normalizing constant cancels out, giving:

$$\Phi(\mu)P(\mu \rightarrow \nu) = \Phi(\nu)P(\nu \rightarrow \mu) \quad (2.63)$$

Inspired by the rejection sampling algorithm (Sec. 2.1.2), the Metropolis-Hastings algorithm works by splitting the transition kernel into two steps: a *selection* or *proposal* step, and an *acceptance* step. The first step employs a probability distribution $g(\mu \rightarrow \nu)$ which proposes a new state ν given the current state μ . This selection process may be deterministic or probabilistic; if probabilistic, the distribution should ideally be easy to sample from (perhaps through the use of one of the methods discussed in Sec. 2.1). The second step then accepts the newly proposed state ν with some probability $A(\mu \rightarrow \nu)$ which depends on both μ and ν . If the new state ν is rejected then the Markov Chain will remain in state μ . Mathematically, we split the transition probability into two factors, giving the following detailed balance equation:

$$\Phi(\mu)A(\mu \rightarrow \nu)g(\mu \rightarrow \nu) = \Phi(\nu)A(\nu \rightarrow \mu)g(\nu \rightarrow \mu) \quad (2.64)$$

By splitting up the transition probability, we allow ourselves to use a proposal distribution $g(\mu \rightarrow \nu)$ which is not quite a perfect transition kernel in the sense that it does not satisfy detailed balance with respect to Φ on its own. The acceptance probability then corrects for the imperfections in g .

We do not yet know how to set the acceptance probabilities, so we solve for their ratio:

$$\frac{A(\mu \rightarrow \nu)}{A(\nu \rightarrow \mu)} = \frac{\Phi(\nu) g(\nu \rightarrow \mu)}{\Phi(\mu) g(\mu \rightarrow \nu)} \quad (2.65)$$

On the right hand side are all the quantities which we know how to compute. The ratio above still leaves a lot of freedom in how to set the acceptance probabilities. Intuitively, we want the Markov Chain to move through the configuration space as quickly as possible, so we try to maximize the acceptance probability.

The Metropolis-Hastings acceptance formula does exactly that, taking the form:

$$A(\mu \rightarrow \nu) = \min \left(1, \frac{\Phi(\nu) g(\nu \rightarrow \mu)}{\Phi(\mu) g(\mu \rightarrow \nu)} \right) \quad (2.66)$$

We see that the above probability is guaranteed to be one for at least one of the two processes $\mu \rightarrow \nu$ or $\nu \rightarrow \mu$. Although the above acceptance probability is quite good, we

emphasize that a judicious choice of the proposal distribution $g(\mu \rightarrow \nu)$ is also important for performing an efficient simulation. In cases where g is non-deterministic, we try to construct $g(\mu \rightarrow \nu)$ to have a form that is somewhat similar to the weight function $\Phi(\nu)$, giving an acceptance probability closer to one. For example, we could construct $g(\mu \rightarrow \nu)$ to have a similar algebraic form to $\Phi(\nu)$, with many common factors shared between the two cancelling out.

In practical implementations we are often dealing with large configuration spaces, hence the proposal distribution can only update a small portion of the configuration at a time, and the selection probability $g(\mu \rightarrow \nu)$ is only non-zero for a small fraction of configurations ν . The ergodicity of the Metropolis-Hastings kernel is highly dependent on the quality of the proposals, and one must make sure that the distribution g is able to propose a reasonably diverse set of new configurations. This is of course not completely sufficient to ensure ergodicity as $A(\mu \rightarrow \nu)$ is also a component of the transition probability. As such, we may sometimes find that even with a distribution g that is itself ergodic, if the acceptance probability is too frequently zero (or close to zero), the overall transition kernel will not be ergodic.

Due to how frequently we will be using it throughout this thesis, we will sometimes refer to the function $m(p) = \min(1, p)$ as the Metropolis function.

2.2.6 The Heat-Bath Algorithm

An alternative method to the Metropolis-Hastings algorithm is called the Heat-Bath or Gibbs algorithm.

Let's assume that the configuration μ can be partitioned into L (non-overlapping) sub-configurations. We may then write μ as a list of length L , $\mu = [\mu_1, \mu_2, \dots, \mu_L]$, where each μ_i is a subset of the full configuration which we could imagine to be single sites or even groups thereof on a lattice. We wish to update a small portion of the configuration, say only one variable in the list μ . The Gibbs algorithm proceeds by exactly sampling from the probability distribution of that one subset of the configuration *conditioned* on the state of the remainder of the configuration. More precisely, we update the sub-configuration μ_i by sampling from $p(\mu_i | \mu_1, \mu_2, \dots, \mu_{i-1}, \mu_{i+1}, \dots, \mu_L) = p(\mu_i | \mu_{-i})$, where we've defined μ_{-i} as the list μ with entry μ_i excluded. A full update then consists of performing L Gibbs updates, either by sequentially updating all sub-configurations one by one, or by choosing L sub-configurations uniformly at random (with replacement).

The conditional distribution can be more explicitly written as:

$$p(\mu_i|\mu_{-i}) = \frac{p(\mu_1, \mu_2, \dots, \mu_i, \dots, \mu_L)}{p(\mu_{-i})} = \frac{p(\mu_1, \mu_2, \dots, \mu_i, \dots, \mu_L)}{\sum_{\mu'_i} p(\mu_1, \mu_2, \dots, \mu'_i, \dots, \mu_L)} \quad (2.67)$$

which can also be written in terms of the unnormalized configuration weight Φ since the normalizing constant cancels out:

$$p(\mu_i|\mu_{-i}) = \frac{\Phi(\mu_1, \mu_2, \dots, \mu_i, \dots, \mu_L)}{\sum_{\mu'_i} \Phi(\mu_1, \mu_2, \dots, \mu'_i, \dots, \mu_L)} \quad (2.68)$$

We now prove that the detailed balance condition is satisfied for the Heat-Bath transition from state μ to ν , where $\nu = [\nu_1, \nu_2, \dots, \nu_L]$ only differs from μ at i . The transition probability for the forward process is $P(\mu \rightarrow \nu) = p(\nu_i|\mu_{-i})$, and multiplying this by $\Phi(\mu)$ gives:

$$\Phi(\mu)P(\mu \rightarrow \nu) = \Phi(\mu)p(\nu_i|\mu_{-i}) = \Phi(\mu) \frac{\Phi(\mu_1, \mu_2, \dots, \nu_i, \dots, \mu_L)}{\sum_{\mu'_i} \Phi(\mu_1, \mu_2, \dots, \mu'_i, \dots, \mu_L)} \quad (2.69)$$

$$= \frac{\Phi(\mu)\Phi(\nu)}{\sum_{\mu'_i} \Phi(\mu_1, \mu_2, \dots, \mu'_i, \dots, \mu_L)} \quad (2.70)$$

$$= \Phi(\nu) \frac{\Phi(\mu_1, \mu_2, \dots, \mu_i, \dots, \mu_L)}{\sum_{\mu'_i} \Phi(\mu_1, \mu_2, \dots, \mu'_i, \dots, \mu_L)} \quad (2.71)$$

$$= \Phi(\nu) \frac{\Phi(\nu_1, \nu_2, \dots, \mu_i, \dots, \nu_L)}{\sum_{\nu'_i} \Phi(\nu_1, \nu_2, \dots, \nu'_i, \dots, \nu_L)} \quad (2.72)$$

$$= \Phi(\nu)p(\mu_i|\nu_{-i}) \quad (2.73)$$

$$\Phi(\mu)P(\mu \rightarrow \nu) = \Phi(\nu)P(\nu \rightarrow \mu) \quad (2.74)$$

and hence, detailed balance is satisfied.

The efficacy of the Heat-Bath algorithm is contingent on being able to efficiently sample from the conditional distribution of Eq. 2.68. The inversion sampling and alias methods require the conditional distribution to be normalized, hence the sub-configuration we wish to update cannot be too large, or else the number of possible states it can take will grow very quickly, making the normalizing constant to compute and the distribution too large to fit in memory. Additionally, since we will have to perform many updates sequentially, it is important that the conditional distribution not depend on too many of the variables that are not being updated. Ideally, the contributions of most of the non-updated variables

in the conditional distribution should factor out of both the numerator and denominator and cancel. Otherwise, the construction of the conditional distributions will become quite costly, especially if the variable to be updated can take on a large number of different states.

Alternatively, one may be able to implement a Heat-Bath algorithm using a rejection sampler to draw from the conditional distribution. In this case, the distribution does not necessarily have to be normalized, and we don't need to worry about the frequent construction cost. The downside is of course the need to perform multiple iterations before being able to output a sample.

Nevertheless, in cases where the Heat-Bath sampler can be used efficiently, it can be very effective.

As for ergodicity, the Heat-Bath kernel is able to—in principle—generate all possible states of a given sub-configuration so long as they are assigned a non-zero probability by the conditional distribution.

2.2.7 Choosing between the Metropolis-Hastings and the Heat-Bath Algorithms

As we saw above, the Heat-Bath algorithm has no formal “rejection” of proposals in the sense that the Metropolis-Hastings algorithm does. Nevertheless, the Heat-Bath algorithm can still propose a move into the same state the random variable was already in. As such, in order to avoid confusion with the nomenclature of “acceptance” and “rejection”, we will define two new slightly different terms: “leaving”, meaning the random variable *leaves* the current state for a new one, and “staying”, meaning the random variable *stays* in the same state it was already in. It is clear that for the case of Metropolis-Hastings style updates, “leaving” and “accepting” are equivalent, and similarly for “staying” and “rejecting”. Of course, by construction they mean slightly different things for Heat-Bath style updates.

Say we perform an update to the configuration (possibly a small portion of it, say a single spin) where we are deciding between two choices. The current configuration has weight W_0 , while the opposite configuration has weight W_1 . Let's define the weight ratio $\lambda = W_1/W_0$.

For a Metropolis style update, assuming the simple proposal distribution $g(0 \rightarrow 1) = g(1 \rightarrow 0) = 1$ (that is, we always try to flip the variable), we have the following staying

and leaving probabilities:

$$p_{\text{stay}}^{\text{MH}} = 1 - \min\left(1, \frac{W_1}{W_0}\right) = 1 - \min(1, \lambda) \quad (2.75)$$

$$p_{\text{leave}}^{\text{MH}} = \min\left(1, \frac{W_1}{W_0}\right) = \min(1, \lambda) \quad (2.76)$$

Meanwhile for Heat-Bath style updates, we instead have:

$$p_{\text{stay}}^{\text{HB}} = \frac{W_0}{W_1 + W_0} = \frac{1}{1 + \frac{W_1}{W_0}} = \frac{1}{1 + \lambda} \quad (2.77)$$

$$p_{\text{leave}}^{\text{HB}} = \frac{W_1}{W_1 + W_0} = 1 - \frac{1}{1 + \lambda} \quad (2.78)$$

If our goal is to explore as much of the configuration space as quickly as possible, then we will want to choose an update with a larger p_{leave} . Since $p_{\text{stay}}^{\text{HB}} = 1/(1 + \lambda) > 0$, the leaving probability for the Heat-Bath scheme will never reach 1. On the other hand, the Metropolis scheme can easily give a unit leaving probability. This is due in part to the fact that the Metropolis scheme conditions *on the current state of the random variable*; that is, it uses the weight of the current state to inform the decision to move to the proposed state. The Heat-Bath scheme, meanwhile, forgets the current state immediately.

In Fig. 2.4 we plot the leaving (in blue) and staying (in red) probabilities defined above for the Metropolis-Hastings (solid lines) and Heat-Bath (dashed lines) schemes in the case where we are updating a random variable with only two possible states.

The above analysis becomes trickier once we move to the case of a random variable with more than two states. Labelling $n > 2$ states with weights W_1, W_2, \dots, W_n , which sum to

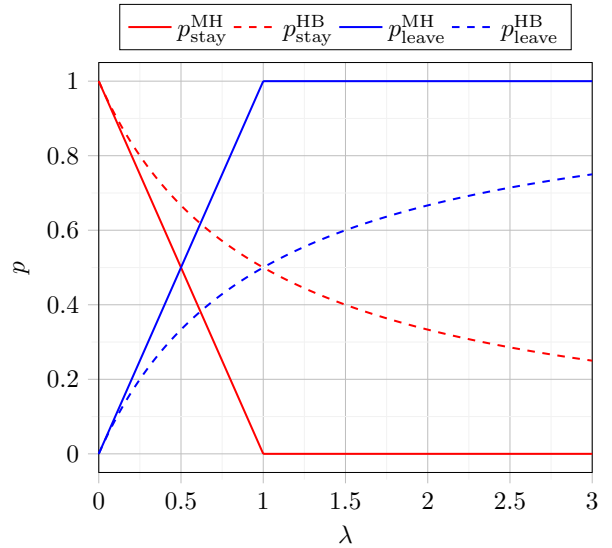


Figure 2.4: Comparison of staying and leaving probabilities for the Metropolis-Hastings and Heat-Bath algorithms in the case of a binary choice.

$\mathcal{S} = \sum_{i=1}^n W_i$, assume the random variable is currently in the state i . For the Heat-Bath scheme, the staying and leaving probabilities become:

$$p_{\text{stay}}^{\text{HB}} = \frac{W_i}{\mathcal{S}} \quad (2.79)$$

$$p_{\text{leave}}^{\text{HB}} = 1 - \frac{W_i}{\mathcal{S}} = \frac{\mathcal{S} - W_i}{\mathcal{S}} \quad (2.80)$$

and we see that the leaving probability can easily approach 1 if $\mathcal{S} \gg W_i$, which may occur if there are many different states to choose from.

Meanwhile, the Metropolis scheme with the uniform proposal distribution $g(i \rightarrow j) = (1 - \delta_{ij})/(n - 1)$, gives:

$$p_{\text{stay}}^{\text{MH}} = 1 - \frac{1}{n - 1} \sum_{j \neq i} \min \left(1, \frac{W_j}{W_i} \right) \quad (2.81)$$

$$p_{\text{leave}}^{\text{MH}} = \frac{1}{n - 1} \sum_{j \neq i} \min \left(1, \frac{W_j}{W_i} \right) \quad (2.82)$$

which is admittedly a bit impenetrable. However, we can provide an upper-bound for the leaving probability:

$$p_{\text{leave}}^{\text{MH}} \leq \frac{1}{n - 1} \sum_{j \neq i} \frac{W_j}{W_i} = \frac{(\mathcal{S} - W_i)/(n - 1)}{W_i} \quad (2.83)$$

with equality when *all* $W_j \leq W_i$. When the average weight of states $j \neq i$ is less than W_i , the total probability of leaving state i will be less than one, even if there are some states $j \neq i$ for which $W_j \geq W_i$.

If we were lucky enough to have g propose a state j with $W_j \geq W_i$, the random variable would be guaranteed to update, otherwise it would remain in state i with probability $1 - W_j/W_i < 1$. This isn't too different from the Heat-Bath case which always has some non-zero staying probability. For a distribution of weights that is relatively flat, this is quite alright. However, given a weight distribution which has only a few peaks among an otherwise flat weight landscape, the Markov Chain may take some time to both find each of those peaks and transition between them. For example, consider a distribution where the sub-configuration we wish to update can take $n = 301$ different states, with weights $W_i = 1$ for all $i > 3$, and $W_i = 1000$ for $i \leq 3$. If the Markov Chain begins with the variable in any of the states $i > 3$, transitioning to one of the higher weight states will

be extremely difficult as there is only a 1% chance of even *proposing* a move to one of these higher weight states, meaning the chain will spend most of its time exploring states of low probability. Once in one of these high weight states, transitioning to a proposed lower weight state will only occur with probability 1/1000, which is fine since in order to generate samples with probability proportional to the state's weight, the Markov Chain *should* spend proportionately more time in the higher weight state. The issue however, is that transitions *between* higher weight states are also difficult due to the small proposal probability. The Metropolis scheme is at the mercy of the proposal distribution.

To emphasize this point further, consider the staying probability, for which we get the lower-bound using the bound on the leaving probability:

$$p_{\text{stay}}^{\text{MH}} \geq 1 - \frac{\mathcal{S} - W_i}{W_i(n-1)} = \frac{n - \frac{\mathcal{S}}{W_i}}{n-1} \quad (2.84)$$

again with equality when all states have weight less than or equal to W_i . Defining the average weight as $\bar{W} = \sum_i W_i/n = \mathcal{S}/n$, we get:

$$p_{\text{stay}}^{\text{MH}} \geq \frac{1 - \bar{W}/W_i}{1 - 1/n} = \left(1 - \frac{\bar{W}}{W_i}\right) \left(1 + \frac{1}{n} + \frac{1}{n^2} + \dots\right) > 1 - \frac{\bar{W}}{W_i} \quad (2.85)$$

meaning that the staying probability for the Metropolis-Hastings algorithm will remain positive when the W_i is larger than the average weight, regardless of the size of the distribution. When $\bar{W} \geq W_i$, the lower-bound on $p_{\text{leave}}^{\text{MH}}$ is no longer positive and therefore useless.

We can remedy this issue by instead proposing states with probabilities according to their weight, bringing us back to the Heat-Bath scheme. Here the Markov chain is able to move to (and between) the higher weight states easily, as it is no longer being slowed down by the uniform proposal distribution in a large sample space. Indeed, for the Heat-Bath algorithm we have:

$$p_{\text{stay}}^{\text{HB}} = \frac{1}{n} \frac{W_i}{\bar{W}} \quad (2.86)$$

which approaches zero as the size of the distribution n grows, so long as the ratio W_i/\bar{W} does not grow faster than n . Of course, as we discussed earlier, the Heat-Bath algorithm is only practical if the distribution that we are sampling over in the Heat-Bath step is small enough to fit in memory, such as when performing updates to pieces of a larger configuration. Thus, we may instead need to use a proposal distribution for the Metropolis-Hastings algorithm which is structurally similar to the conditional distribution of the sub-configuration used in the Heat-Bath algorithm, but is simpler to compute. In this way we

will avoid the undesirable wandering behaviour of a uniform proposal distribution, without the potential computational burden of the Heat-Bath algorithm. Although the example given above was fairly contrived, one could imagine that similar (though likely less extreme) scenarios of sample spaces with a few high weight peaks separated by lower weight “basins” could occur when simulating physical systems at low temperatures.

2.2.8 Equilibration

While we have shown that it is possible to devise transition kernels which are able to converge to a desired stationary distribution, the above discussion does not tell us anything about how long it may take to converge. Indeed, convergence was only actually guaranteed in the infinite time limit. Although an MCMC simulation will not quite be sampling from the stationary distribution (that is, the simulation is slightly biased) after a finite amount of time from its initialization, we find that for the simulations discussed in this thesis the simulations are able to converge to the stationary distribution fairly quickly. In fact, we can show that the convergence to the stationary distribution is exponential, with the rate of convergence being related to the eigenvalue of second largest magnitude of the transition matrix.

Recall the definition of the instantaneous distribution of the Markov chain at time-step t : $\mathbf{p}_t^\top = \mathbf{p}_0^\top P^t$. We will assume for simplicity that P is reversible and therefore diagonalizable, $P = Q^{-1}DQ$, as well as aperiodic. Let the eigenvalues of P be: $\lambda_1, \lambda_2, \dots, \lambda_{\mathcal{D}}$, where we’ve sorted them in decreasing order by absolute value: $1 = \lambda_1 > |\lambda_2| \geq \dots \geq |\lambda_{\mathcal{D}}|$. We expand the distribution \mathbf{p}_t in terms of the rows of Q :

$$\mathbf{p}_0^\top P^t = c_1 \mathbf{q}_1^\top + c_2 \lambda_2^t \mathbf{q}_2^\top + c_3 \lambda_3^t \mathbf{q}_3^\top + \dots \quad (2.87)$$

and \mathbf{q}_1 is of course the stationary distribution $\boldsymbol{\pi}$. From this we see that we need the coefficient c_1 to be non-zero, meaning the initial distribution must have non-zero overlap with the stationary distribution. This is actually a fairly trivial condition to satisfy as, by the Perron-Frobenius theorem, $\boldsymbol{\pi}$ has strictly positive entries. Thus, any initial probability distribution should work, even a constant distribution⁹.

Next, we take t sufficiently large so all the contributions to the sum corresponding to eigenvalues of magnitude smaller than $|\lambda_2|$ become negligible.

$$\mathbf{p}_0^\top P^t = c_1 \boldsymbol{\pi}^\top + c_2 \lambda_2^t \mathbf{q}_2^\top \quad (2.88)$$

⁹For a constant distribution, it would be beneficial to choose one of the modes of the target, if possible, in order to maximize the overlap.

From this, it is clear that the convergence to the stationary distribution is exponential, with the rate of convergence given by λ_2 . Hence, a transition kernel with $|\lambda_2|$ close to one will equilibrate slowly, whereas one with $|\lambda_2|$ much less than one will equilibrate very fast.

Although this is a nice property of reversible aperiodic Markov chains, for finite simulations, the samples taken before convergence will introduce some bias to whatever expectation values we try to estimate. Therefore, to reduce this bias we throw out some fraction of samples from the beginning of our simulation. How many samples to discard is difficult to determine *a priori*. Retaining all generated samples (including the samples from the equilibration period) for later manual inspection is a common, though still quite heuristic method of determining how long it took the Markov chain to converge. Additionally, some estimators may converge quicker than others, which makes manual analyses even more cumbersome. Oftentimes we simply define some heuristic fraction (say, 10%) of our total computational budget as the “burn-in” or “equilibration” period. Of course, this does not always guarantee that the instantaneous distribution has converged to π , especially for short simulations, but it was found to be sufficient for the simulations discussed in this thesis. The general recommendation is that it is better to discard too many samples than to risk keeping biased samples [5].

Before we conclude this section, we mention that there has been some intriguing recent work in developing rigorous methods to assess convergence of Markov chains, see Refs. [52, 97].

2.3 Error Estimation

In order to make useful conclusions based on Monte Carlo data, we need to be able to quantify how trustworthy an estimate of a given quantity is. That is, we need to be able to account for statistical noise in the estimates that we report.

2.3.1 Dealing with independent samples

In the simplest case, we wish to estimate the expectation value of some function of the configuration: $A(C)$. Assuming we have N independent samples C_1, C_2, \dots, C_N , we estimate the expectation value with the sample mean:

$$\langle A \rangle \approx \bar{A} = \frac{1}{N} \sum_{i=1}^N A(C_i) \quad (2.89)$$

and sample standard deviation:

$$\sigma_A \approx \sqrt{\frac{1}{N-1} \sum_{i=1}^N (A(C_i) - \bar{A})^2} \quad (2.90)$$

The sample mean can be thought of as a random variable in its own right which, by the Central Limit Theorem, will be normally distributed with mean $\langle A \rangle$ and standard deviation $\sigma_{\bar{A}} = \sigma_A / \sqrt{N}$. This standard deviation is often called the *standard error of the mean*, and it is the quantity that we will use to construct the error bars for our estimate of the mean \bar{A} . We note that as we increase the number of samples used to compute \bar{A} , the error bars shrink.

2.3.2 Dealing with autocorrelation

Of course, when we're dealing with data coming from a Markov Chain Monte Carlo simulation, the samples drawn are rarely independent. The samples, considered as a time-series, exhibit some amount of *autocorrelation*. In order to quantify the autocorrelation, we first define the autocovariance function for a given estimator A as:

$$\mathcal{C}_A(t) = \frac{1}{N-t} \sum_{\tau=1}^{N-t} (A_{\tau} - \bar{A})(A_{\tau+t} - \bar{A}) \quad (2.91)$$

from which we define the autocorrelation function as:

$$\rho_A(t) = \frac{\mathcal{C}_A(t)}{\mathcal{C}_A(0)} \quad (2.92)$$

For irreducible aperiodic Markov chains, we expect the autocorrelation to decay exponentially as

$$\rho_A(t) \sim e^{-t/\tau_A} \quad (2.93)$$

with time-scale τ_A , which we call the *autocorrelation time*.

When performing error analysis, we must account for the autocorrelation of our data and rescale our error bars accordingly. A standard method of handling autocorrelated data is called *binning* [217, 5]. Given a sequence of N MCMC samples C_1, C_2, \dots, C_N , we divide them into N_b sequential “bins”. Assuming for simplicity that N_b divides N , the bins all

have size $B = N/N_b$, and the b th bin is: $\{C_{(b-1)B+1}, C_{(b-1)B+2}, \dots, C_{bB}\}$. We compute the estimator on each bin separately:

$$\bar{A}_b = \frac{1}{B} \sum_{j=1}^B A(C_{(b-1)B+j}) \quad (2.94)$$

Then, the estimator for the mean is given by:

$$\bar{A} = \frac{1}{N_b} \sum_{b=1}^{N_b} \bar{A}_b = \frac{1}{N} \sum_{i=1}^N A(C_i) \quad (2.95)$$

which is the same as the un-binned case. The sample standard deviation changes:

$$\sigma_A = \sqrt{\frac{1}{N_b - 1} \sum_{b=1}^{N_b} (\bar{A}_b - \bar{A})^2} \quad (2.96)$$

For large enough bin-sizes, the estimators computed on individual bins are effectively uncorrelated, hence we can compute the standard error of the mean as: $\sigma_{\bar{A}} = \sigma_A / \sqrt{N_b}$.

In practice, we often start by taking a bin-size of two and compute the binned standard error. We then re-bin the binned data, again taking a bin-size of two (effectively giving a total bin-size of four), and compute the standard error again on this re-binned data. This is done repeatedly until we see a convergence in the standard error, meaning we've reached a bin-size large enough that the individual bins are effectively uncorrelated. This variety of binning is often called *logarithmic binning* [217, 5] and it very useful for estimating the autocorrelation time τ . The autocorrelation time of A can be written as:

$$\tau_A = \frac{\sum_{t=1}^{\infty} \langle A_1 A_{1+t} \rangle - \langle A \rangle^2}{\langle A^2 \rangle - \langle A \rangle^2} \quad (2.97)$$

where $A_{1+t} = A(C_{1+t})$ is the value of A computed at time-step $1+t$. The standard error of an autocorrelated estimator is related to the sample standard deviation by a scaling factor: $\sigma_{\bar{A}} = \sigma_A \sqrt{(1 + 2\tau_A)/N}$. The quantity $N/(1 + 2\tau_A)$ is often called the *effective sample size*. Solving this for τ_A gives:

$$\tau_A = \frac{1}{2} \left(\frac{\sigma_A^2}{\sigma_A^2/N} - 1 \right) \quad (2.98)$$

We can compute this quantity once we've observed convergence of the standard error in our logarithmic binning procedure.

This binning procedure must be done separately for each estimated quantity as different estimators will have different autocorrelation times.

2.3.3 Dealing with non-linearity

Now that we have an estimate of the expectation of A , suppose that we want to compute some function of the expectation value: $f(\langle A \rangle)$. We can estimate this quantity as $f(\bar{A})$, but what about the error bars? If f is a linear function $f(x) = mx + b$, then the error bars are simply multiplied by m , but if f is non-linear this doesn't work.

We can make use of the *Jackknife* procedure to estimate the error bars for an arbitrary function f [239]. For now we will again assume that we have N independent samples C_1, C_2, \dots, C_N . The Jackknife procedure proceeds as follows: we compute N estimates of $\langle A \rangle$ over the dataset, leaving out a single sample each time.

$$A_i^{(J)} = \frac{1}{N-1} \sum_{j \neq i} A(C_j) \quad (2.99)$$

We then compute N estimates of $f(\langle A \rangle)$ as $f_i^{(J)} = f(A_i^{(J)})$. Intuitively, what we are doing here is manually simulating the noise present in the estimate of $f(\langle A \rangle)$, given by $f(\bar{A})$, by creating many new artificial datasets over which we compute the estimate. The error in the estimate is then given by:

$$\sigma_f = \sqrt{\frac{N-1}{N} \sum_{i=1}^N (f_i^{(J)} - f^{(J)})^2} \quad (2.100)$$

where $f^{(J)} = (1/N) \sum_{i=1}^N f_i^{(J)}$.

At first glance, it may appear that the error bars provided by the jackknife procedure do not decrease with increasing N . However, consider the summation in the square root: as we increase N , the jackknife estimates $f_i^{(J)}$ will get less and less noisy, and hence the variance computed within the square root will decrease.

2.3.4 Putting it all together

If we want to perform a jackknife procedure using autocorrelated data, we will need to combine the binning procedure with the jackknife.

The jackknife estimates of $\langle A \rangle$ are now taken over the sequentially binned averages \bar{A}_b .

$$A_b^{(J)} = \frac{1}{N_b - 1} \sum_{b' \neq b} \bar{A}_{b'} \quad (2.101)$$

We have N_b estimates of $f(\langle A \rangle)$ computed as $f_b^{(J)} = f(A_b^{(J)})$. The estimate $f(\langle A \rangle) \approx f(\bar{A})$ has error given by:

$$\sigma_f = \sqrt{\frac{N_b - 1}{N_b} \sum_{b=1}^{N_b} \left(f_b^{(J)} - f^{(J)} \right)^2} \quad (2.102)$$

where $f^{(J)} = (1/N_b) \sum_{b=1}^{N_b} f_b^{(J)}$. Just like in Sec. 2.3.2, we can perform successive re-binning of the data until we observe convergence of the error estimate.

2.3.5 Multiple Markov Chains

When performing MCMC simulations, we often run multiple independent Markov Chains in parallel and then aggregate the results together. Since the different chains are independent, we do not need to worry about correlations between them. However, the data from each chain is still autocorrelated, so we perform binning in each chain separately until we create bins large enough to be approximately independent¹⁰. Once the bins are independent enough, we pool all bins from all chains together and estimate the quantities we are interested in along with associated error bars (which may be done using jackknife).

¹⁰It is important to note that the optimal bin-sizes—and hence the autocorrelation times—may vary from chain to chain. Conveniently, the use of multiple chains thereby allows for the estimation of error bars on the autocorrelation times themselves.

Chapter 3

Stochastic Series Expansion

In this chapter we will introduce and develop the formalism for the Stochastic Series Expansion (SSE) Quantum Monte Carlo (QMC) method. The Stochastic Series Expansion was developed as an improvement upon a method originally developed by Handscomb [77, 78] which was limited in applicability due to its dependence on being able to analytically perform the traces of terms in the partition function. It is also an exact alternative to QMC methods based on path integrals, which suffer from systematic errors due to their (flawed) discretization of imaginary-time [193, 91].

This chapter will begin with Sec. 3.1, in which we will develop the mathematical formalism for the SSE by deriving an expression for the partition function of a thermal state as a sum. Sec. 3.2 will discuss how to represent the many-body Hamiltonian as a sum of terms amenable to simulation with SSE, as well as some discussion of the SSE configuration space. In Sec. 3.3 we will derive Monte Carlo estimators for various observables of interest, and finally, in Sec. 3.4 we will introduce an alternative SSE formalism specialized to the simulation of Hamiltonian groundstates.

Throughout this chapter, we will use various forms of the TFIM as a recurring example throughout this section. The general form of this Hamiltonian is

$$H = \sum_{i < j} J_{ij} \sigma_i^z \sigma_j^z + \sum_i h_i \sigma_i^x \quad (3.1)$$

where J_{ij} is the strength of the interaction between sites i and j , with $J_{ij} < 0$ ($J_{ij} > 0$) giving a ferromagnetic (antiferromagnetic) interaction; and h_i is the strength of the transverse field at site i . The double-sum for the interaction term is restricted to $i < j$ in order to avoid double counting; it can be turned into a simple double-sum over all pairs

(i, j) by either restricting the matrix J_{ij} to be upper- or lower-triangular, or by allowing J_{ij} to be a symmetric matrix and simply dividing the interaction term by 2. The σ operators are the usual Pauli operators:

$$\sigma^x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \sigma^y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad \sigma^z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad (3.2)$$

The most common and simplest form of the TFIM Hamiltonian, however, is the following:

$$H = -J \sum_{\langle i, j \rangle} \sigma_i^z \sigma_j^z - h \sum_i \sigma_i^x \quad (3.3)$$

where $J, h > 0$, and the notation $\langle i, j \rangle$ restricts the summation to pairs of sites (i, j) which are nearest-neighbours on the lattice.

3.1 Partition Function Decomposition

As discussed in Chapter 2, in order to develop a Monte Carlo simulation for a probability distribution, we must first be able to write down the distribution's normalizing constant. Begin with the partition function for a quantum Gibbs state at inverse temperature β

$$Z = \text{Tr}[e^{-\beta H}] \quad (3.4)$$

and expand the matrix exponential with a Taylor Series

$$Z = \sum_{n=0}^{\infty} \text{Tr} \left[\frac{(-\beta H)^n}{n!} \right] = \sum_{n=0}^{\infty} \frac{\beta^n}{n!} \text{Tr}[(-H)^n] \quad (3.5)$$

Next we expand the trace,

$$Z = \sum_{\alpha \in A} \sum_{n=0}^{\infty} \frac{\beta^n}{n!} \langle \alpha | (-H)^n | \alpha \rangle \quad (3.6)$$

where the basis $|\alpha\rangle$ is typically chosen to be a tensor-product basis because they admit simple computational representations as strings of classical states. The specific tensor-product basis chosen is usually taken to be one where some specific terms in the Hamiltonian are diagonal. For example, in the case of the TFIM, the basis is often taken to be the eigenbasis of the interaction term, i.e. the eigenbasis of σ^z . We must now insert resolutions

of the identity between each factor of $(-H)$. As the resolutions of identity will all be in the same basis, we will label them all as $|\alpha_p\rangle$, with p being the index which counts the factors of $-H$. We will relabel the states which originated from the expansion of the trace as $|\alpha_0\rangle = |\alpha_n\rangle$.

$$Z = \sum_{n=0}^{\infty} \frac{\beta^n}{n!} \sum_{\alpha_0} \langle \alpha_0 | (-H) \sum_{\alpha_1} |\alpha_1\rangle \langle \alpha_1 | (-H) \cdots \sum_{\alpha_{n-1}} |\alpha_{n-1}\rangle \langle \alpha_{n-1} | (-H) | \alpha_0 \rangle \quad (3.7)$$

$$Z = \sum_{n=0}^{\infty} \frac{\beta^n}{n!} \sum_{\alpha_0} \sum_{\alpha_1} \cdots \sum_{\alpha_{n-1}} \langle \alpha_0 | (-H) | \alpha_1 \rangle \langle \alpha_1 | (-H) \cdots | \alpha_{n-1} \rangle \langle \alpha_{n-1} | (-H) | \alpha_0 \rangle \quad (3.8)$$

It is useful to take a moment to define some notation. Given a set X with generic elements labelled as $x \in X$, we define the set X^n as the n -fold Cartesian product of the set X , and we will denote generic elements of X^n using an underline: $\underline{x} \in X^n$. That is, $X^n = \{\underline{x} = (x_1, x_2, \dots, x_n) \mid x_1 \in X, x_2 \in X, \dots, x_n \in X\}$. Now, denoting the (orthonormal) set of basis states as A , we will apply this notation to simplify the iterated sums over basis states:

$$Z = \sum_{n=0}^{\infty} \frac{\beta^n}{n!} \sum_{\underline{\alpha} \in A^n} \prod_{p=1}^n \langle \alpha_{p-1} | (-H) | \alpha_p \rangle \quad (3.9)$$

$$Z = \sum_{n=0}^{\infty} \sum_{\underline{\alpha} \in A^n} \frac{\beta^n}{n!} \prod_{p=1}^n \langle \alpha_{p-1} | (-H) | \alpha_p \rangle \quad (3.10)$$

Before we proceed further, let's take a moment to understand the product in this expression. The state $|\alpha_{p-1}\rangle$ is propagated by the negative Hamiltonian to the state $|\alpha_p\rangle$ and this process is repeated n times. This gives us a notion of “time” in the SSE formalism, and is in fact related to imaginary time in the Path Integral formalism [170, 176]. Although it is not quite a temporal dimension (it is not continuous), the analogy is quite strong, and we often refer to it colloquially as the imaginary time dimension, and we refer to each step in this dimension as a *time-slice*. At the end of the propagation we end up with the state $|\alpha_n\rangle$ which is equal to the initial state $|\alpha_0\rangle$ due to the cyclic property of the trace. Hence, we find that the trace itself imposes periodic boundary conditions along the imaginary time dimension. We thus see how, just like in the Path Integral case, a d -dimensional quantum partition function can be mapped to a $(d + 1)$ -dimensional classical partition function. Oftentimes the $(d + 1)$ -dimensional configuration is referred to as a *simulation cell*, an example of which can be seen in Fig. 3.1.

Now, we could stop here with the partition function in Eq. 3.10, and we could imagine performing a Monte Carlo sampling of the matrix elements of the full Hamiltonian. The

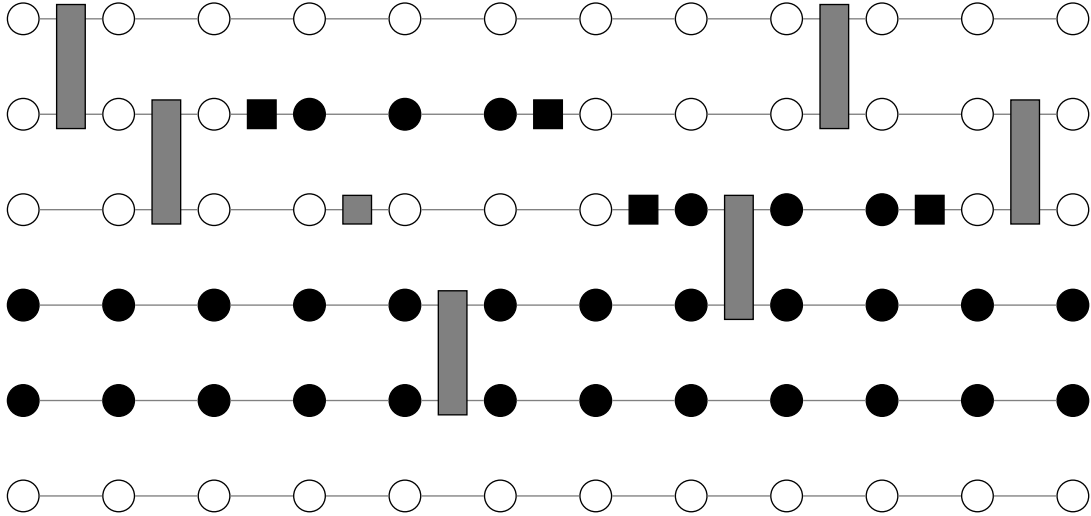


Figure 3.1: Example of an SSE simulation cell for a TFIM with 6 sites. The imaginary time dimension is along the horizontal; note the periodic boundary conditions.

issue with this approach is that it may not be possible to quickly compute a given matrix element of H at each step, and storing all of these matrix elements would also be impractical due to the exponential growth of the Hilbert space as a function of the system volume. Instead we must break the Hamiltonian into pieces. Hamiltonians for systems of interest can typically be written as sums of operators which act only on a subset of the system. For example, in the case of the transverse-field Ising model, the Hamiltonian is a sum of $\sigma_i^z \sigma_j^z$ and σ_i^x terms, which act on at most 2 sites at a time. We can therefore write a general Hamiltonian as:

$$H = \sum_{s \in S} (-H_s) \quad (3.11)$$

where s is indexing the summands of the Hamiltonian, S is the set of all summand indices, and the extra minus sign is included in order to simplify the algebra. The way in which the Hamiltonian is broken into summands is important, as the proper Hamiltonian breakup will allow us to perform efficient Monte Carlo sampling of the terms of the partition function; we will discuss this further in Sec. 3.2. Plugging the above expression into the partition

function we get

$$Z = \sum_{n=0}^{\infty} \sum_{\underline{\alpha} \in A^n} \frac{\beta^n}{n!} \prod_{p=1}^n \left\langle \alpha_{p-1} \left| \sum_{s_p \in S} H_{s_p} \right| \alpha_p \right\rangle \quad (3.12)$$

$$Z = \sum_{n=0}^{\infty} \sum_{\underline{\alpha} \in A^n} \sum_{\underline{s} \in S^n} \frac{\beta^n}{n!} \prod_{p=1}^n \langle \alpha_{p-1} | H_{s_p} | \alpha_p \rangle \quad (3.13)$$

which means we are now also summing over all possible products of Hamiltonian summands which propagate the state $|\alpha_0\rangle$ back to itself. We typically refer to the string of Hamiltonian summands in the product as an *operator string* or an *operator list*. Before we finish, let's clean up the expression a little bit. Let Φ denote the weight function:

$$\Phi(n, \underline{\alpha}, \underline{s}; \beta) = \frac{\beta^n}{n!} \prod_{p=1}^n \langle \alpha_{p-1} | H_{s_p} | \alpha_p \rangle \quad (3.14)$$

We have separated the inverse temperature β from the other variables as it is a constant of the system and is not being summed over. We are now done with the (formal) partition function decomposition:

$$Z = \sum_{n=0}^{\infty} \sum_{\underline{\alpha} \in A^n} \sum_{\underline{s} \in S^n} \Phi(n, \underline{\alpha}, \underline{s}; \beta) \quad (3.15)$$

A Monte Carlo simulation of the above expression would thus require sampling the random variables $(n, \underline{\alpha}, \underline{s})$. The variables $\underline{\alpha}$ and \underline{s} are of length n , a quantity which is also varying randomly. This presents a small issue, as performing a computer simulation of the partition function this way would require us to dynamically re-allocate memory at each step, and will therefore dramatically slow down the simulation. To bypass this, we first truncate the Taylor series to some maximum order M and pad the operator string with $M - n$ trivial identity operators $H_0 = \mathbf{1}$ which will be randomly distributed uniformly throughout the operator string. The constant M is grown during the equilibration phase of the simulation. We typically set $M = \lceil cn_{\max} \rceil$ where c is some (heuristic) constant between 1 and 2 and n_{\max} is the maximum number of non-trivial operators encountered during equilibration. By introducing trivial operators we are now over-counting operator strings which we must compensate for. The total number of possible placements of trivial identity operators in the operator string can be computed using the binomial coefficient:

$$\binom{M}{n} = \frac{M!}{n!(M-n)!} \quad (3.16)$$

Dividing the weights by this quantity gives:

$$Z \approx \sum_{\alpha \in A^M} \sum_{\underline{s} \in S^M} \frac{\beta^n (M-n)!}{M!} \prod_{p=1}^M \langle \alpha_{p-1} | H_{s_p} | \alpha_p \rangle \quad (3.17)$$

where the set S has been amended to include the trivial operator and the explicit sum over n has been dropped as it is now included in the sum over the set S^M . Due to the truncation of the Taylor series, the above summation is only an approximation of the true partition function, though in practice the truncation error is negligible due to the way we select M , and the fact that statistical errors will be dominant when we perform the Monte Carlo simulations. As a result, we will simply use $=$ when expressing the partition function from now on, trusting that the constant M has been taken sufficiently large. Lastly, we redefine the weight function

$$\Phi(n, \underline{\alpha}, \underline{s}; \beta, M) = \frac{\beta^n (M-n)!}{M!} \prod_{p=1}^M \langle \alpha_{p-1} | H_{s_p} | \alpha_p \rangle \quad (3.18)$$

giving the final form of our SSE partition function:

$$Z = \sum_{\alpha \in A^M} \sum_{\underline{s} \in S^M} \Phi(n, \underline{\alpha}, \underline{s}; \beta, M) \quad (3.19)$$

3.2 Hamiltonian Breakup and the SSE Configuration Space

We will develop a Monte Carlo simulation to sample the terms in Eq. 3.19, but first we must discuss the Hamiltonian breakup which will allow us to construct the concrete SSE configuration space. In general, we can imagine that there would be many different ways to breakup or group together terms of the Hamiltonian. We are of course most interested in those which will be conceptually simple for us to reason about, hence we almost always try to stay close to the original form of the Hamiltonian. This brings us to our first condition for the Hamiltonian breakup: *The No-Branching Condition* (Sec. 3.2.1). The issue of negative configuration weights is a serious problem which may arise when performing Monte Carlo simulations of quantum Hamiltonians, and constitutes the second condition that must be satisfied by the Hamiltonian breakup; this will be discussed further in Secs. 3.2.2 and 3.2.3. Finally, Secs. 3.2.4 and 3.2.5 will briefly discuss some details of the SSE configuration space.

3.2.1 The No-Branching Condition

All this condition requires is that every Hamiltonian term H_s must map the basis state $|\alpha\rangle$ to a *single* basis state $|\alpha'\rangle$, not a superposition of basis states.

$$H_s |\alpha\rangle \propto |\alpha'\rangle \in A \quad \forall s \in S, \alpha \in A \quad (3.20)$$

Crucially, this does *not* limit the Hamiltonians we are able to simulate; it is a condition which exists solely to simplify bookkeeping. For example, an arbitrary single-qubit Hamiltonian will take the form:

$$-H = \begin{bmatrix} H_{00} & H_{01} \\ H_{10} & H_{11} \end{bmatrix} \quad (3.21)$$

and would map the states $|0\rangle = (1, 0)^\top$ and $|1\rangle = (0, 1)^\top$ to $H_{00}|0\rangle + H_{01}|1\rangle$ and $H_{10}|0\rangle + H_{11}|1\rangle$, respectively. The no-branching condition demands that we split this Hamiltonian into multiple terms $-H = H_a + H_b$, which could look like:

$$H_a = \begin{bmatrix} H_{00} & 0 \\ 0 & H_{11} \end{bmatrix} \quad H_b = \begin{bmatrix} 0 & H_{01} \\ H_{10} & 0 \end{bmatrix} \quad (3.22)$$

where we can easily see that neither piece maps a basis state to a superposition state. Imposing this condition allows us to simplify the partition function decomposition. Recall,

$$Z = \sum_{\alpha \in A^M} \sum_{s \in S^M} \frac{\beta^n (M-n)!}{M!} \prod_{p=1}^M \langle \alpha_{p-1} | H_{s_p} | \alpha_p \rangle \quad (3.23)$$

Explicitly writing out the multiple summations over basis states we have:

$$Z = \sum_{s \in S^M} \frac{\beta^n (M-n)!}{M!} \sum_{\alpha_0 \in A} \langle \alpha_0 | H_{s_0} \sum_{\alpha_1 \in A} |\alpha_1\rangle \langle \alpha_1 | H_{s_1} \cdots \sum_{\alpha_{M-1} \in A} |\alpha_{M-1}\rangle \langle \alpha_{M-1} | H_{s_{M-1}} | \alpha_M \rangle \quad (3.24)$$

Due to the no-branching condition, the Hamiltonian term H_{s_0} maps $|\alpha_0\rangle$ to a single basis state, therefore the sum over α_1 reduces to a single term. Repeating this simplification for every summation over basis states gives

$$Z = \sum_{s \in S^M} \frac{\beta^n (M-n)!}{M!} \sum_{\alpha_0 \in A} \langle \alpha_0 | H_{s_0} | \alpha_1 \rangle \langle \alpha_1 | H_{s_1} | \alpha_2 \rangle \cdots \langle \alpha_{M-1} | H_{s_{M-1}} | \alpha_M \rangle \quad (3.25)$$

$$Z = \sum_{\alpha_0 \in A} \sum_{s \in S^M} \frac{\beta^n (M-n)!}{M!} \prod_{p=1}^M \langle \alpha_{p-1} | H_{s_p} | \alpha_p \rangle \quad (3.26)$$

which looks quite similar to what we had before! By imposing the no-branching condition, we've managed to reduce M summations over basis states into a single summation. Additionally, this allows us to simplify our Monte Carlo configuration from $(n, \underline{\alpha}, \underline{s})$ to (n, α, s) , meaning we no longer need to keep track of M different basis states. We can think of the no-branching condition as reducing the summations over basis states at the cost of growing the number of Hamiltonian terms H_s . Since the Hamiltonian term index s can be expressed as a single integer while the basis states $|\alpha\rangle$ (being tensor product states) need an integer per site, even though we've increased the range of s , we have tremendously reduced the total memory requirements of the basis states that need to be stored.

3.2.2 Energy Shifts and the Diagonal Sign Problem

Up until now we've swept one large problem under the rug. What happens if a Hamiltonian term H_s has a negative (or even complex) matrix element? In order to Monte Carlo sample the terms of the partition function we require them to be positive (or zero). If H_s is diagonal with respect to the basis states $|\alpha\rangle$, we can easily fix this problem. Since the Hamiltonian is Hermitian, its diagonal elements are guaranteed to be real. We can "cure" any negative diagonal elements by introducing an energy shift. Consider the interaction term of the general TFIM Hamiltonian (3.1):

$$H_s = -J_{ij}\sigma_i^z\sigma_j^z = \begin{bmatrix} -J_{ij} & & & \\ & J_{ij} & & \\ & & J_{ij} & \\ & & & -J_{ij} \end{bmatrix} \quad (3.27)$$

By adding the diagonal energy shift $|J_{ij}|\mathbf{1}$, we get

$$|J_{ij}|\mathbf{1} - J_{ij}\sigma_i^z\sigma_j^z = \begin{bmatrix} |J_{ij}| - J_{ij} & & & \\ & |J_{ij}| + J_{ij} & & \\ & & |J_{ij}| + J_{ij} & \\ & & & |J_{ij}| - J_{ij} \end{bmatrix} \quad (3.28)$$

If $J_{ij} < 0$, we have a ferromagnetic interaction, and the only non-zero elements of Eq. 3.28 will be the first and last diagonal elements, corresponding to aligned spins. Alternatively, $J_{ij} > 0$ corresponds to an antiferromagnetic interaction, which results in only the second and third diagonal elements of Eq. 3.28 being non-zero, corresponding to anti-aligned spins.

In general, the energy shift C_s for a diagonal operator H_s will be given by

$$C_s = \left| \min \left(0, \min_{\alpha} \langle \alpha | H_s | \alpha \rangle \right) \right| \quad (3.29)$$

That is, the energy shift is given by the absolute value of the *most negative* matrix element, or zero if all matrix elements are positive. Oftentimes, a small constant $\varepsilon > 0$ is also added to C_s in order to aid numerics [179, 194]. It is important to keep track of all shifts C_s which have been added to the Hamiltonian, as they will later need to be subtracted from the appropriate observable estimators; we will discuss this further in Sec. 3.3.

3.2.3 The Sign Problem

If a Hamiltonian sub-operator has negative or complex off-diagonal elements, there is no immediately obvious way to modify the Hamiltonian in order to allow sampling of operator strings. Negative matrix elements can still be sampled by weighting them according to their absolute values and rewriting expectation values as ratios of expectations over this new modified distribution:

$$\langle O \rangle_{\Phi} = \sum_C O(C) \Phi(C) = \frac{\sum_C O(C) \frac{\Phi(C)}{|\Phi(C)|} |\Phi(C)|}{\sum_C \frac{\Phi(C)}{|\Phi(C)|} |\Phi(C)|} = \frac{\langle O(C) \text{sgn}(\Phi(C)) \rangle_{|\Phi(C)|}}{\langle \text{sgn}(\Phi(C)) \rangle_{|\Phi(C)|}} \quad (3.30)$$

However, the signal-to-noise ratio of both expectations is extremely poor: their relative errors increase exponentially with the system size. This is known as *The Sign Problem* [155, 203], and it greatly limits the class of Hamiltonians which can be simulated using QMC methods.

For certain systems, unitary transformations of the Hamiltonian have been devised which can eliminate the sign problem. An example of one such system is the simple TFIM (3.3) with a reversed transverse field:

$$H = -J \sum_{\langle i,j \rangle} \sigma_i^z \sigma_j^z + h \sum_i \sigma_i^x \quad (3.31)$$

The negated Hamiltonian which we would use in an SSE simulation will obviously have negative off-diagonal matrix elements in this case. However, by applying the unitary transformation $U = \bigotimes_i \sigma_i^z$ to the Hamiltonian, we can flip the sign associated with the transverse field. We have thus cured the sign problem in this case.

In principle, there must always exist a unitary which can cure the sign problem. Whether this unitary can be found and applied efficiently is another question entirely. Indeed, the unitary which diagonalizes the Hamiltonian would obviously cure the sign problem.

We may expect that complex off-diagonal matrix elements may pose an even greater challenge. However, due to Hermiticity of the Hamiltonian, complex off-diagonal matrix elements turn out to only be as challenging as their real parts. Consider an SSE configuration $C = (n, \alpha, (s_1, s_2, \dots, s_M))$, as well as the time-reversed configuration $C_r = (n, \alpha, (s_M, \dots, s_2, s_1))$. It is clear that the weights of these two configurations are complex conjugates: $\Phi(C) = \Phi^*(C_r)$. Hence, when performing the summation over the configuration space, the imaginary parts of the weights will cancel, leaving only the real part. As a result, we only need to worry about negative matrix elements.

3.2.4 An Extended Representation

Here we will briefly introduce an extended representation of the SSE partition function which highlights the constraints imposed on valid SSE configurations. Consider Eq. 3.17 again, and let us insert further resolutions of identity after each Hamiltonian operator:

$$Z = \sum_{\alpha \in A^M} \sum_{\underline{s} \in S^M} \frac{\beta^n (M-n)!}{M!} \prod_{p=1}^M \langle \alpha_{p-1} | H_{s_p} | \alpha_p \rangle \quad (3.32)$$

$$= \sum_{\alpha \in A^M} \sum_{\alpha' \in A^M} \sum_{\underline{s} \in S^M} \frac{\beta^n (M-n)!}{M!} \prod_{p=1}^M \langle \alpha_{p-1} | H_{s_p} | \alpha'_p \rangle \langle \alpha'_p | \alpha_p \rangle \quad (3.33)$$

$$Z = \sum_{\alpha \in A^M} \sum_{\alpha' \in A^M} \sum_{\underline{s} \in S^M} \frac{\beta^n (M-n)!}{M!} \prod_{p=1}^M \langle \alpha_{p-1} | H_{s_p} | \alpha'_p \rangle \delta_{\alpha'_p, \alpha_p} \quad (3.34)$$

This extended representation explicitly shows the consistency requirements of an SSE configuration. Adjacent matrix elements in the operator string need to have consistent basis states, meaning that the ket acting to its left on the operator at time-slice p must be dual to the bra acting to its right on the operator at time-slice $p+1$. The Kronecker deltas in Eq. 3.34 enforce this consistency. An *inconsistent configuration* is therefore a configuration for which at least one $\alpha'_p \neq \alpha_p$. We cannot naively sample random strings of matrix elements or else the Kronecker deltas may assign the resulting configuration a weight of zero. It is easy to convince oneself that *most* strings of matrix elements will in fact result in configurations of zero weight.

3.2.5 Operator Indices

The Hamiltonian sub-operator indices s are the main random variables which will be modified by the Monte Carlo update. As such, their explicit construction merits some discussion.

The index s is meant to track which Hamiltonian sub-operator is active in a given time-slice. Hence, it needs to contain information on both the type of operator t that it represents as well as which physical sites (i, j, \dots) that the operator is acting on. Therefore there exists a bijection which can map the index s to and from the tuple of integers (t, i, j, \dots) . It is usually this tuple that we work with in practical implementations as we need to be able to quickly access the elements of the tuple without computing them from s .

Returning to the example of the general TFIM,

$$H = \sum_{i < j} J_{ij} \sigma_i^z \sigma_j^z + \sum_i h_i \sigma_i^x \quad (3.1)$$

the operator tuple will have only three indices as the Hamiltonian terms act on at most two sites. For this Hamiltonian we will therefore begin with three different types of operators: $(0, 0, 0)$ for trivial identity operators, (b, i, j) for bonds, and $(f, i, 0)$ for transverse field terms. The integers b and f are constants and can be set as the user pleases so long as $b \neq f$. Operator tuples are kept at a uniform length for the same reason we truncated the partition function to a maximum order: M tuples of varying length will require dynamic allocation of memory which can slow down the simulation. In cases where M is large, we may prefer to instead use the single index s in the operator list in order to reduce memory consumption. We then keep a pair of lookup tables which can quickly map $s \mapsto (t, i, j, \dots)$ and $(t, i, j, \dots) \mapsto s$.

3.3 Estimating Expectation Values

The end goal of any Monte Carlo simulation is to estimate expectation values of various quantities of interest. In this section, we will discuss how to estimate expectation values of physical observables using an SSE simulation. There are three basic types of estimators that we will discuss: estimators for expectations of diagonal operators, Hamiltonian sub-operators, and products of such sub-operators. Afterwards, we will show how to estimate energetic quantities; that is, how to estimate expectation values of the full Hamiltonian H along with powers thereof. We will conclude by mentioning some richer physical quantities

that may also be estimated using SSE simulations but which are too complex to discuss here.

3.3.1 Diagonal Operators

Estimators for expectation values of diagonal operators are extremely simple to construct. Given an operator D which is diagonal with respect to our chosen basis α , (that is, $D|\alpha\rangle = D(\alpha)|\alpha\rangle$) we can simply compute D for our sampled basis state $|\alpha_0\rangle$ as $D(\alpha_0)$ and average this over the course of our Monte Carlo simulation:

$$\langle D \rangle = \frac{1}{Z} \text{Tr}(De^{-\beta H}) \quad (3.35)$$

$$= \frac{1}{Z} \sum_{\alpha_0 \in \mathcal{A}} \sum_{\underline{s} \in \mathcal{S}^M} \frac{\beta^n (M-n)!}{M!} \left\langle \alpha_0 \left| D \prod_{p=1}^M H_{s_p} \right| \alpha_0 \right\rangle \quad (3.36)$$

$$= \frac{1}{Z} \sum_{\alpha_0 \in \mathcal{A}} \sum_{\underline{s} \in \mathcal{S}^M} \frac{\beta^n (M-n)!}{M!} \left\langle \alpha_0 \left| D(\alpha_0) \prod_{p=1}^M H_{s_p} \right| \alpha_0 \right\rangle \quad (3.37)$$

$$= \frac{1}{Z} \sum_{\alpha_0 \in \mathcal{A}} \sum_{\underline{s} \in \mathcal{S}^M} D(\alpha_0) \frac{\beta^n (M-n)!}{M!} \left\langle \alpha_0 \left| \prod_{p=1}^M H_{s_p} \right| \alpha_0 \right\rangle \quad (3.38)$$

$$= \frac{1}{Z} \sum_{\alpha_0 \in \mathcal{A}} \sum_{\underline{s} \in \mathcal{S}^M} D(\alpha_0) \Phi(n, \alpha, \underline{s}; \beta, M) \quad (3.39)$$

$$\langle D \rangle = \langle D(\alpha_0) \rangle_{\text{MC}} \quad (3.40)$$

Additionally, due to the cyclic property of the trace, we can even average over all α_p in the simulation cell.

$$\langle D \rangle = \frac{1}{M} \left\langle \sum_{p=1}^M D(\alpha_p) \right\rangle_{\text{MC}} \quad (3.41)$$

While this estimator will produce a lower variance, its autocorrelation time will increase due to its dependence on a larger portion of the simulation cell. This means that after accounting for autocorrelation, the rescaled variance may not be much lower than that of the simple estimator 3.40, meaning our error bars may not have decreased as much as we had hoped. In addition, the increased computation required may render the estimator 3.41 too costly to use in practice. One could instead only average over every K time-slices in order to reduce the autocorrelation and computational cost, or even select time-slices at

random. Which method ends up being more efficient in practice is extremely dependent on both the simulation and the observable being estimated.

A straightforward alternative method to reduce the error bars with very little computational overhead is to compute the estimator 3.40 by taking α from a single random time-slice. While the estimator 3.41 aimed to give a smaller error bars by increasing the number of samples used, this alternative method shrinks the error bars by reducing the autocorrelation between samples.

3.3.2 Hamiltonian Sub-Operators

In order to estimate the expectation value of a Hamiltonian sub-operator H_t (which could be diagonal or off-diagonal), we begin by writing out the expectation value explicitly using the non-truncated representation:

$$\langle H_t \rangle = \frac{1}{Z} \text{Tr}(H_t e^{-\beta H}) \quad (3.42)$$

$$= \frac{1}{Z} \sum_{n=0}^{\infty} \sum_{\alpha \in A} \sum_{\underline{s} \in S^n} \frac{\beta^n}{n!} \left\langle \alpha \left| H_t \prod_{p=1}^n H_{s_p} \right| \alpha \right\rangle \quad (3.43)$$

We insert a resolution of identity α' after H_t , noting that due to the No-Branching Condition (see Sec. 3.2.1), the summation over α' only has a single term. Additionally, we shift the index n up by 1.

$$\langle H_t \rangle = \frac{1}{Z} \sum_{n=1}^{\infty} \sum_{\alpha \in A} \sum_{\underline{s} \in S^{n-1}} \frac{\beta^{n-1}}{(n-1)!} \langle \alpha | H_t | \alpha' \rangle \left\langle \alpha' \left| \prod_{p=1}^{n-1} H_{s_p} \right| \alpha \right\rangle \quad (3.44)$$

$$= \frac{1}{Z} \sum_{n=1}^{\infty} \sum_{\alpha \in A} \sum_{\underline{s} \in S^{n-1}} \frac{\beta^n}{n!} \frac{n}{\beta} \langle \alpha | H_t | \alpha' \rangle \left\langle \alpha' \left| \prod_{p=2}^n H_{s_{p-1}} \right| \alpha \right\rangle \quad (3.45)$$

The only terms which give a non-zero weight are those for which H_t can map α to α' . Then, the product of matrix elements in the numerator becomes a consistent configuration; we'll shift the operator indices up by one position so that we may refer to t as s_1 . We extend

the summation over n to include zero again, as the n/β factor will zero it out anyway.

$$\langle H_t \rangle = \frac{1}{Z} \sum_{n=0}^{\infty} \sum_{\alpha \in A} \sum_{\underline{s} \in S^{n-1}} \frac{\beta^n n}{n! \beta} \left\langle \alpha \left| \left(H_t \prod_{p=2}^n H_{s_p} \right) \right| \alpha \right\rangle \quad (3.46)$$

$$= \frac{1}{Z} \sum_{n=0}^{\infty} \sum_{\alpha \in A} \sum_{\underline{s} \in S^n} \frac{\beta^n}{n!} \left\langle \alpha \left| \prod_{p=1}^n H_{s_p} \right| \alpha \right\rangle \left(\frac{n}{\beta} \delta_{s_1, t} \right) \quad (3.47)$$

$$= \frac{1}{Z} \sum_{n=0}^{\infty} \sum_{\alpha \in A} \sum_{\underline{s} \in S^n} \Phi(n, \alpha, \underline{s}; \beta) \left(\frac{n}{\beta} \delta_{s_1, t} \right) \quad (3.48)$$

$$= \left\langle \frac{n}{\beta} \delta_{s_1, t} \right\rangle_{\text{MC}} \quad (3.49)$$

This estimator simply checks whether the operator H_t is present at the first time-slice of the simulation cell. One can imagine that, since the estimator is a Kronecker delta, looking at only a single time-slice may not be sufficient to achieve good statistics. In order to improve the estimator, we use the same trick that we employed in Sec. 3.3.1: averaging over the full simulation cell.

$$\langle H_t \rangle = \left\langle \frac{1}{n} \frac{n}{\beta} \sum_{p=1}^n \delta_{s_p, t} \right\rangle_{\text{MC}} = \frac{\langle N(t) \rangle_{\text{MC}}}{\beta} \quad (3.50)$$

where $N(t)$ counts the number of instances of the operator H_t in the operator string. One may worry that the autocorrelation of this estimator will render it inefficient as it is now dependent on a much larger portion of the configuration. It turns out that estimators based on counting single operators tend to de-correlate quite fast in SSE due to the structure of the Monte Carlo updates which perform many changes to the operator string at each step.

It should be noted that the expectation value of H_t may not directly correspond to the expectation value of the physical operator on which it is based. First, recall that when defining the Hamiltonian breakup, we included a minus sign in front of the sub-operators in order to simplify the algebra:

$$H = \sum_{s \in S} (-H_s) \quad (3.11)$$

Second, as discussed in Sec. 3.2.2, energy shifts may have been added to diagonal Hamiltonian sub-operators. These adjustments must be taken into account before reporting data on Hamiltonian expectation values.

As an example, consider again the simple nearest neighbour ferromagnetic TFIM with N sites,

$$H = -J \sum_{\langle i,j \rangle} \sigma_i^z \sigma_j^z - h \sum_i \sigma_i^x \quad (3.3)$$

where $J, h > 0$. There are two types of operators in this Hamiltonian: the bond operators,

$$H_{b_{ij}} = J\mathbf{1} + J\sigma_i^z \sigma_j^z = \begin{bmatrix} 2J & & & \\ & 0 & & \\ & & 0 & \\ & & & 2J \end{bmatrix} \quad (3.51)$$

and the single-site field operators,

$$H_{f_i} = h\sigma_i^x = \begin{bmatrix} 0 & h \\ h & 0 \end{bmatrix} \quad (3.52)$$

The expectation value of the transverse magnetization for site i will be given as:

$$\langle \sigma_i^x \rangle = \frac{1}{h} \langle H_{f_i} \rangle = \frac{\langle N(f_i) \rangle_{\text{MC}}}{\beta h} \quad (3.53)$$

Meanwhile, the expectation values of the two-point correlator $\sigma_i^z \sigma_j^z$ will be given by

$$\langle \sigma_i^z \sigma_j^z \rangle = \frac{\langle H_{b_{ij}} \rangle - J}{J} = \frac{\langle N(b_{ij}) \rangle_{\text{MC}} - \beta J}{\beta J} = \frac{\langle N(b_{ij}) \rangle_{\text{MC}}}{\beta J} - 1 \quad (3.54)$$

3.3.3 Products of Hamiltonian Sub-Operators

Deriving estimators for arbitrary off-diagonal operators tends to be quite difficult. In the case where such operators can be written as products of Hamiltonian sub-operators, or sums thereof, one can use the estimator:

$$\left\langle \prod_{k=1}^m H_{t_k} \right\rangle = \frac{1}{\beta^m} \left\langle \frac{(n-1)!}{(n-m)!} N(t_1, t_2, \dots, t_m) \right\rangle_{\text{MC}} \quad (3.55)$$

where $N(t_1, t_2, \dots, t_m)$ counts the number of instances in which the ordered sequence (t_1, t_2, \dots, t_m) occurs in the operator string [170]. It is clear that as m increases, the ordered sequence of interest will become less likely to manifest, making it difficult to compute good estimates.

3.3.4 Energetic Quantities

In Sec. 3.3.2, we saw how to estimate expectation values of Hamiltonian sub-operators. Constructing an estimator for the system's energy is now as simple as adding together the sub-operator estimators.

$$-\sum_s \langle H_s \rangle = -\sum_s \frac{\langle N(s) \rangle_{\text{MC}}}{\beta} = -\frac{\langle n \rangle_{\text{MC}}}{\beta} \quad (3.56)$$

So the energy is given simply by the number of non-trivial operators in the operator string. Of course, we still need to remove the energy shifts we've added to this Hamiltonian.

$$\langle H_{\text{unshifted}} \rangle = \langle H_{\text{shifted}} \rangle + \sum_s C_s = \sum_s (C_s - \langle H_s \rangle) = C_{\text{total}} - \frac{\langle n \rangle_{\text{MC}}}{\beta} \quad (3.57)$$

In fact, there is an even simpler way to derive the energy estimator. Since we do not need to keep track of the individual Hamiltonian sub-operators, we can (ignoring the energy shift) derive the energy estimator using the Taylor Series representation of the partition function, Eq. 3.5:

$$Z = \sum_{n=0}^{\infty} \frac{(-\beta)^n}{n!} \text{Tr}(H^n) \quad (3.5)$$

The energy is given by differentiating $-\ln Z$ with respect to beta:

$$\langle H \rangle = -\frac{\partial \ln Z}{\partial \beta} = \frac{-1}{Z} \sum_{n=0}^{\infty} \frac{\text{Tr}(H^n)}{n!} \frac{\partial (-\beta)^n}{\partial n} \quad (3.58)$$

$$= \frac{-1}{Z} \sum_{n=0}^{\infty} \frac{\text{Tr}(H^n)}{n!} (-n)(-\beta)^{n-1} \quad (3.59)$$

$$= \frac{-1}{Z} \sum_{n=0}^{\infty} \frac{\text{Tr}(H^n)}{n!} (-\beta)^n \frac{n}{\beta} \quad (3.60)$$

$$= -\frac{\langle n \rangle_{\text{MC}}}{\beta} \quad (3.61)$$

as expected.

By inverting the above relation we can see that the average expansion order of the SSE simulation is given by $\langle n \rangle_{\text{MC}} = \beta \langle H \rangle$. This gives us a rough estimate on the expected runtime of the SSE algorithm as well. Assuming the cutoff M is on the order of $\langle n \rangle_{\text{MC}}$,

then if our updates take time linear in M^1 , the expected time complexity of SSE is $O(M) \sim O(\beta \langle H \rangle)$, where H is the shifted Hamiltonian we use in our simulation. Since the energy is an extensive quantity, we see that the algorithm scales both linearly in the inverse temperature as well as the volume of the system.

3.3.4.1 Higher-Order Moments of the Hamiltonian

Deriving higher-order cumulants or moments of the energy can be done by simply taking appropriate higher-order derivatives of $\ln Z$ or Z .

For the case of the moments, we can also derive the estimators by directly manipulating the Taylor Expansion of the exponential:

$$\langle H^k \rangle = \frac{1}{Z} \sum_{n=0}^{\infty} \frac{(-\beta)^n}{n!} \text{Tr}(H^k H^n) \quad (3.62)$$

$$= \frac{1}{Z} \sum_{n=0}^{\infty} \frac{(n+k)(n+k-1)\cdots(n+1)(-\beta)^{n+k}}{(-\beta)^k (n+k)!} \text{Tr}(H^{n+k}) \quad (3.63)$$

$$= \frac{1}{Z} \sum_{n=0}^{\infty} \frac{n(n-1)\cdots(n-k+1)(-\beta)^n}{(-\beta)^k n!} \text{Tr}(H^n) \quad (3.64)$$

$$= \frac{1}{(-\beta)^k} \langle n(n-1)\cdots(n-k+1) \rangle_{\text{MC}} \quad (3.65)$$

$$= \frac{1}{(-\beta)^k} \left\langle \frac{n!}{(n-k)!} \right\rangle_{\text{MC}} \quad (3.66)$$

where in the third line we shifted the index n down by k , and we used the fact that the product $n(n-1)(n-2)\cdots(n-k+1)$ evaluates to zero when $n = -k, -k+1, \dots, -1, 0$, allowing us to leave the lower limit of the summation at zero. As expected, the above estimator is very similar to Eq. 3.55, with two key differences. The first difference is the minus sign included with beta which had cancelled out with the extra minus sign we included in the Hamiltonian sub-operators in Eq. 3.55. The second difference is within the Monte Carlo expectation: the extra factor of n in Eq. 3.66 comes from the fact that the counting function in Eq. 3.55 is now including *every* Hamiltonian sub-operator, meaning it will simply evaluate to n . Unlike the estimator in Eq. 3.55, the estimator for the Hamiltonian moment has statistics which are much better behaved as it is not tracking potentially rare events. Although the standard error will grow with k , the moments themselves will also

¹As we will see in Chapter 4, they will.

be growing, and hence the relative error will be fairly well behaved. There may be some issues with statistics when k is on the order of n , meaning we will sometimes have $k > n$ and the estimator will give a value of zero. In this case the estimator is indeed tracking a potentially rare event, and when combined with the fact that the typical (non-zero) values of the estimator will be quite large (for large k), it is clear that the zeroes will tremendously widen the range of possible values the estimator can take, thus giving very large standard errors. Fortunately, we are not usually interested in such large values of k where this could pose an issue.

It should be kept in mind that the moments of a random variable are not shift-invariant, and one must account for the energy-shift that was added to the Hamiltonian in order to perform the SSE simulation. Letting $H_{\text{unshifted}} = H_{\text{shifted}} + E$, we can express the moments of $H_{\text{unshifted}}$ in terms of the moments of H_{shifted} using the transformation of centre formula²:

$$\langle H_{\text{unshifted}}^k \rangle = \langle (H_{\text{shifted}} + E)^k \rangle = \sum_{l=0}^k \binom{k}{l} \langle H_{\text{shifted}}^l \rangle E^{k-l} \quad (3.67)$$

Computing $\langle H_{\text{shifted}}^l \rangle$ with Eq. 3.66, the full estimator then becomes:

$$\langle H_{\text{unshifted}}^k \rangle = \left\langle \sum_{l=0}^k \binom{k}{l} \frac{E^{k-l}}{(-\beta)^l} \frac{n!}{(n-l)!} \right\rangle_{\text{MC}} \quad (3.68)$$

We may prefer to work with cumulants as opposed moments, as they are manifestly shift-invariant (besides the first cumulant) and are all additive for independent random variables. However, cumulants are polynomial functions of the moments, and the exact expressions for higher-order cumulants tend to be fairly complicated and tedious to derive. Fortunately, we usually only care about lower-order cumulants, such as the second-order cumulant (the variance) which for the energy is directly related to the heat capacity of the system.

3.3.4.2 The Heat Capacity

The heat capacity is given by the derivative of the energy with respect to β , or the variance of the energy:

$$C = -\beta^2 \frac{\partial \langle H \rangle}{\partial \beta} = \beta^2 (\langle H^2 \rangle - \langle H \rangle^2) \quad (3.69)$$

²This is essentially the binomial theorem.

Using the estimator for energetic moments derived above, the SSE estimator takes the form:

$$C = \langle n^2 \rangle_{\text{MC}} - \langle n \rangle_{\text{MC}}^2 - \langle n \rangle_{\text{MC}} \quad (3.70)$$

Since the heat capacity is proportional to the variance of the energy, it is clear that any energy shifts added to the Hamiltonian will automatically cancel out, hence the above expression directly estimates the physical quantity.

3.3.4.3 Some Practical Considerations

When computing moments of the Hamiltonian, or expectations of products of Hamiltonian sub-operators, care must be taken to avoid numerical stability issues when computing the estimators. The ratios of factorials should be computed as falling factorials, instead of computing each factorial separately. For large enough k it may even be necessary to compute the estimators in log-space, as the falling factorial can become quite large.

Expressing each factor in the falling factorial as $(n - i)/(-\beta)$ before multiplying may help somewhat when β (and therefore n) is large. However, β may not always be large enough in magnitude to keep the estimator under control. Indeed, if we recall that the expansion order is roughly $n \sim \beta \langle H \rangle$, if $\langle H \rangle > 1$ then the falling factorial will grow rapidly with k even if we divide each factor by β .

On that note, we may wish to instead divide each factor in the falling factorial by the average expansion order $\langle n \rangle$, hence each $(n - i)/\langle n \rangle$ will be on the order of one. This is actually equivalent to computing the dimensionless ratio $\langle H^k \rangle / \langle H \rangle^k$, whose estimator takes the form:

$$\frac{\langle H^k \rangle}{\langle H \rangle^k} = \left\langle \frac{n(n-1)\cdots(n-k+1)}{(-\beta)^k} \right\rangle_{\text{MC}} \left(\frac{-\beta}{\langle n \rangle_{\text{MC}}} \right)^k \quad (3.71)$$

$$= \left\langle \frac{n(n-1)\cdots(n-k+1)}{(\langle n \rangle_{\text{MC}})^k} \right\rangle_{\text{MC}} \quad (3.72)$$

$$= \left\langle \prod_{i=0}^{k-1} \frac{n-i}{\langle n \rangle_{\text{MC}}} \right\rangle_{\text{MC}} \quad (3.73)$$

where we can see that each factor in the product will be close to one for large n . The disadvantage with this method is the need to evaluate $\langle n \rangle_{\text{MC}}$ before one can evaluate the estimator for the ratio, thereby requiring two passes over the data instead of just one.

If we are interested in estimating the moments for a range of different values of k , say $k = 1, 2, \dots, k_{\text{max}}$, then it would be best to compute the estimators all together as

a cumulative product. In this way we avoid performing the same multiplications several times.

The error analysis of the above quantity differs depending on what actual quantity we wish to report. If we do indeed wish to report an estimate of the ratio $\langle H^k \rangle / \langle H \rangle^k$, then we must perform a jackknife procedure (potentially combined with binning to handle autocorrelation) in order to estimate the error bars correctly, as the jackknife will account for fluctuations of the denominator as well as correlations between the two expectation values. In this case, the fact that we must perform two passes over the data is less of an issue, as we must perform multiple passes over the data anyway for the jackknife procedure. The jackknife procedure is also necessary when estimating error bars for cumulants (such as the heat capacity) or shifted moments as in Eq. 3.67 as they are functions of correlated estimators³.

On the other hand, if one is only interested in the moment $\langle H^k \rangle$, then things become simpler as we can simply estimate the dimensionless ratio using Eq. 3.73, treating $\langle H \rangle^k$ as a scaling constant which does not affect the error analysis, and then multiply the resulting estimate (along with its error bars) by $\langle H \rangle^k$. In this way we avoid the need to perform a jackknife procedure.

3.3.5 More Complex Estimators

Estimators for more complex quantities of interest can also be derived, such as for the Fidelity Susceptibility [184, 4, 220] and imaginary-time correlation functions [46]. Additionally, the amount of entanglement present in a quantum state can be quantified through Rényi versions of more commonly known quantities. These include the Rényi Entanglement Entropies [116, 115], Rényi Mutual Information [138], and Rényi Negativities [230, 45]. These more complex estimators are outside of the scope of this thesis and the reader is encouraged to consult the literature.

3.4 Groundstate Projector SSE

Thus far, we have been working with a QMC method which simulates a thermal state. If we were interested specifically in groundstate properties, we could proceed in one of two

³Estimating the shifted moment using Eq. 3.68 does *not* require jackknifing, as we are computing the entire estimator at once, thereby accounting for the correlations directly.

ways: either we perform simulations at larger and larger β until we observe convergence, or we develop a simulation which directly models the groundstate. The latter case is what we will discuss in this section.

Beginning with a trial state $|\psi\rangle$, we can find the (unnormalized) groundstate using power-iteration. Let us denote the energy eigenstates of the Hamiltonian H as $\{|E_i\rangle\}_{i=0}^{\mathcal{D}-1}$, where \mathcal{D} is the dimensionality of the Hilbert space. The trial state can be expressed in this basis as $|\psi\rangle = \sum_{i=0}^{\mathcal{D}-1} c_i |E_i\rangle$. Applying a large power of $(-H)$ to the trial state gives:

$$(-H)^M |\psi\rangle = c_0 |E_0\rangle^M \left[|E_0\rangle + \sum_{m=i}^{\mathcal{D}-1} \frac{c_i}{c_0} \left(\frac{E_i}{E_0} \right)^M |E_m\rangle \right] \stackrel{M \rightarrow \infty}{\equiv} c_0 |E_0\rangle^M |E_0\rangle \quad (3.74)$$

where we have implicitly assumed that we've added a large enough energy shift to the Hamiltonian that E_0 is the eigenenergy of largest magnitude. By applying large powers of $(-H)$, we are “projecting out” the higher energy eigenstates present in trial state $|\psi\rangle$, hence the name of this technique. Before we move on, we must highlight that Eq. 3.74 will only work if $c_0 \neq 0$, meaning the trial state must have some non-zero overlap with the groundstate.

In this section we will briefly develop the modified SSE formalism for projector simulations. We will begin with defining the normalizing constant decomposition in Sec. 3.4.1, after which we will move on to discussing estimation of observables in Sec. 3.4.2.

3.4.1 Normalizing Constant Decomposition

The normalizing constant will be taken to be the norm of the projected wavefunction:

$$Z = \langle E_0 | E_0 \rangle = \langle \psi_\ell | (-H)^M (-H)^M | \psi_r \rangle \quad (3.75)$$

where M has been taken sufficiently large so as to converge the simulation to the groundstate, and $|\psi_\ell\rangle$ and $|\psi_r\rangle$ are the trial states which in principle do not need to be identical but usually are for simplicity. One will typically need to run multiple short simulations of increasing M until convergence of estimators has been observed, after which the full simulation can be performed using the value of M which successfully produces the groundstate.

Inserting resolutions of identity between each factor of $(-H)$, and breaking up the Hamiltonian into sub-operators, we get:

$$Z = \sum_{\alpha \in \mathbb{A}^{2M+1}} \sum_{s \in \mathbb{S}^{2M}} \langle \psi_\ell | \alpha_0 \rangle \langle \alpha_{2M} | \psi_r \rangle \prod_{p=1}^{2M} \langle \alpha_{p-1} | H_{s_p} | \alpha_p \rangle \quad (3.76)$$

We sometimes denote the basis states at the edges of the simulation cell as $|\alpha_\ell\rangle = |\alpha_0\rangle$ and $|\alpha_r\rangle = |\alpha_{2M}\rangle$. The factors $\langle\psi_\ell|\alpha_\ell\rangle$ and $\langle\alpha_r|\psi_r\rangle$ can be simplified by choosing a trial state which has uniform overlap with all computational basis states: $|\psi\rangle = \mathcal{D}^{-1/2} \sum_\alpha |\alpha\rangle$. We can then eliminate these constants as they are no longer relevant to the simulation. In the case of the TFIM, the basis states are taken to be the σ_z eigenstates, hence the trial state is tensor-product of the σ_x eigenstate $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. The simplified normalizing constant is therefore:

$$Z = \sum_{\alpha \in \mathcal{A}^{2M+1}} \sum_{s \in \mathcal{S}^{2M}} \prod_{p=1}^{2M} \langle\alpha_{p-1}|H_{s_p}|\alpha_p\rangle \quad (3.77)$$

We note the presence of an extra sum over basis states compared to sums over sub-operators, in contrast to the thermal SSE case (Eq. 3.15) where there were an equal number of sums of each. This is a consequence of the lack of an operator trace, and we sometimes say—as a flagrant abuse of terminology⁴—that translational invariance of imaginary-time has been broken. Another difference is the lack of identity operators in the projector simulation, which are no longer necessary due to the projection length M being a constant that is set before starting the simulation.

In spite of these differences, the normalizing constant decompositions and associated configuration spaces in the thermal and projector cases are quite similar, which will allow us to make use of our existing update algorithms with only minor modifications as we will see in Sec. 4.3.

3.4.2 Estimating Expectation Values

Due to the fundamentally different starting point of the projector method, the estimators for expectation values of observables derived in Sec. 3.3 can no longer be used, except of course for diagonal operators. However, we must note that only basis states occurring in the middle of the simulation cell (between the two factors of $(-H)^M$) can be considered samples from the groundstate wavefunction. Hence, our estimator for the expectation value of a diagonal operator D will be:

$$\langle D \rangle = \langle D(\alpha_M) \rangle_{MC} \quad (3.78)$$

where α_M is the basis state taken from the centre time-slice. Of course, time-slices near the centre will often be sufficiently converged that basis states within K time-slices of the

⁴In spite of there being no concept of imaginary-time in the projector case, we nevertheless continue to refer to each individual factor of $(-H)$ as a time-slice, just like in the thermal case.

centre can be used to estimate diagonal observables.

$$\langle D \rangle = \frac{1}{2K + 1} \left\langle \sum_{p=M-K}^{M+K} D(\alpha_p) \right\rangle_{MC} \quad (3.79)$$

The same caveat from Sec. 3.3.1 should be noted here: by using a larger portion of the simulation cell for our diagonal estimator, we risk increasing the autocorrelation in our estimates, which could negate much of the reduction of error bars we had hoped to gain. Since the number of time-slices we would use in Eq. 3.79 will be relatively small compared to the entire simulation cell, the autocorrelation will most likely not increase too much, though this is again a question whose answer will depend greatly on the Hamiltonian being studied, the observable being estimated, and the specific update schemes being employed for the simulation. In the hopes of decreasing autocorrelations, we could select α by randomly sampling a single time-slice within the window defined in Eq. 3.79.

Unfortunately, off-diagonal operators (including the energy) become much more difficult to derive estimators for. We will give an example of an energy estimator for the case of the Rydberg Hamiltonian (which also works for the TFIM) in Chapter 5.

Chapter 4

SSE Updates

In the SSE formalism there are many different types of update schemes used to evolve the Markov chain. The updates can be broken down into two main types: diagonal updates, which insert both diagonal Hamiltonian operators and identity operators; and off-diagonal updates, which insert off-diagonal Hamiltonian operators. Both update types are able to change the operator string \underline{s} . However, diagonal updates can change the expansion order n , while off-diagonal updates can change the basis state α . Typically, off-diagonal updates work by proposing that a diagonal operator be replaced by an off-diagonal operator of a similar spatial structure and vice-versa. As a result, by inserting diagonal operators acting on different spatial locations, the diagonal updates are responsible for modifying the topology of the simulation cell, a task which most types of off-diagonal updates are unable to do. In isolation, each of these updates is therefore not ergodic, but together they are, and we will see that our SSE simulations will proceed by first performing a diagonal update step followed by an off-diagonal update. Off-diagonal updates tend to be the most complicated, as producing configurations which have non-zero weight requires the update to be temporally non-local, whereas the diagonal update only acts on one time-slice at a time. Nevertheless, there are many subtleties involved in developing an efficient and general diagonal update scheme.

This chapter will be laid out as follows: in Sec. 4.1, we will construct a few different diagonal update schemes from first principles, each improving in some way upon the last. Next, in Sec. 4.2, we will develop two off-diagonal updates, focusing specifically on models like the transverse-field Ising model, which feature diagonal two-site interactions and single-site off-diagonal fields. Then, Sec. 4.3 will discuss the necessary changes we must make to the diagonal and off-diagonal updates in order to apply them to the groundstate projector

formalism. We will conclude the chapter with Sec. 4.4, where we will briefly discuss how to actually perform an SSE simulation from start to finish.

4.1 Diagonal Updates

At a high-level, the diagonal update is performed by sequentially scanning through the simulation cell from the first imaginary time-slice to the last. As we go, we keep track of the propagated basis state, which is initialized to the state $|\alpha_0\rangle$. If at a given time-slice p we encounter an off-diagonal operator, we simply update the propagated basis state $|\alpha_p\rangle \propto H_{s_p} |\alpha_{p-1}\rangle$, which is simple due to the No-Branching Condition from Sec. 3.2.1. Since $|\alpha_p\rangle$ is a normalized basis state, the matrix element of H_{s_p} is ignored in this step; only the action of the operator upon $|\alpha_{p-1}\rangle$ is relevant. If instead we find a diagonal operator, then we must decide whether to replace it with an identity operator and vice-versa; the propagated basis state remains unchanged: $|\alpha_p\rangle = |\alpha_{p-1}\rangle$. In this section, we will consider some ways to perform updates which replace identity operators with diagonal operators or replace diagonal operators with identities. We will start from the simplest such update scheme and refine upon it until we reach the state-of-the-art. Following this, we will briefly discuss the relation between the diagonal update and the truncation order M , before moving on to derive the special-case of the diagonal update algorithm for the TFIM. Finally, we will summarize the general algorithm in pseudo-code.

4.1.1 A Simple Metropolis-Hastings Scheme

Consider a configuration $(n, \alpha, \underline{s})$, we wish to update the operator string $\underline{s} = (s_1, s_2, \dots, s_M)$ by either converting a diagonal operator into an identity operator or vice-versa. Assume we wish to replace an identity operator at time-slice t with the diagonal operator labelled s :

$$\underline{s} = (s_1, s_2, \dots, s_t = 0, \dots, s_M) \rightarrow (s_1, s_2, \dots, s'_t = s, \dots, s_M) = \underline{s}'$$

We call this an “operator insertion” move. The detailed balance condition is

$$T(\underline{s} \rightarrow \underline{s}')\Phi(n, \alpha, \underline{s}) = T(\underline{s}' \rightarrow \underline{s})\Phi(n+1, \alpha, \underline{s}') \quad (4.1)$$

$$\frac{T(\underline{s} \rightarrow \underline{s}')}{T(\underline{s}' \rightarrow \underline{s})} = \frac{\Phi(n+1, \alpha, \underline{s}')}{\Phi(n, \alpha, \underline{s})} \quad (4.2)$$

$$= \frac{\beta^{n+1}(M-n-1)!M! \prod_{p=1}^M \langle \alpha_{p-1} | H_{s'_p} | \alpha_p \rangle}{M! \beta^n (M-n)! \prod_{p=1}^M \langle \alpha_{p-1} | H_{s_p} | \alpha_p \rangle} \quad (4.3)$$

$$= \frac{\beta}{M-n} \frac{\langle \alpha_{t-1} | H_{s'_t} | \alpha_t \rangle}{\langle \alpha_{t-1} | H_{s_t} | \alpha_t \rangle} \quad (4.4)$$

where all matrix elements besides the one at step t divided out. Since $H_{s_t} = H_0 = \mathbf{1}$, the matrix element in the denominator must be $\langle \alpha_{t-1} | \alpha_t \rangle = \delta_{\alpha_{t-1}, \alpha_t}$. The current configuration’s weight cannot be zero, therefore we have $\alpha_{t-1} = \alpha_t$. Hence,

$$\frac{T(\underline{s} \rightarrow \underline{s}')}{T(\underline{s}' \rightarrow \underline{s})} = \frac{\beta \langle \alpha_t | H_{s'_t} | \alpha_t \rangle}{M-n} \quad (4.5)$$

Now for the reverse process, which we call an “operator removal”, we have

$$\underline{s}' = (s_1, s_2, \dots, s'_t = s, \dots, s_M) \rightarrow (s_1, s_2, \dots, s_t = 0, \dots, s_M) = \underline{s}$$

and similarly,

$$\frac{T(\underline{s}' \rightarrow \underline{s})}{T(\underline{s} \rightarrow \underline{s}')} = \frac{\Phi(n-1, \alpha, \underline{s})}{\Phi(n, \alpha, \underline{s}')} \quad (4.6)$$

$$= \frac{\beta^{n-1}(M-n+1)!M! \prod_{p=1}^M \langle \alpha_{p-1} | H_{s_p} | \alpha_p \rangle}{M! \beta^n (M-n)! \prod_{p=1}^M \langle \alpha_{p-1} | H_{s'_p} | \alpha_p \rangle} \quad (4.7)$$

$$= \frac{M-n+1}{\beta} \frac{\langle \alpha_{t-1} | H_{s_t} | \alpha_t \rangle}{\langle \alpha_{t-1} | H_{s'_t} | \alpha_t \rangle} \quad (4.8)$$

$$= \frac{M-n+1}{\beta \langle \alpha_t | H_{s'_t} | \alpha_t \rangle} \quad (4.9)$$

In anticipation of applying the Metropolis-Hastings principle, we split the transition probabilities into selection (g) and acceptance (A) probabilities.

$$T(\underline{s} \rightarrow \underline{s}') = A(\underline{s} \rightarrow \underline{s}')g(\underline{s} \rightarrow \underline{s}') \quad (4.10)$$

If we sample our diagonal operators uniformly, the selection probabilities for the forward process $\underline{s} \rightarrow \underline{s}'$ will be $1/N_d$, where N_d is the number of diagonal operators, while $g(\underline{s}' \rightarrow \underline{s}) = 1$. Then, applying the Metropolis-Hastings acceptance formula gives

$$A(\underline{s} \rightarrow \underline{s}') = \min \left(1, \frac{\beta N_d \langle \alpha_t | H_{s_t} | \alpha_t \rangle}{M - n} \right) \quad (4.11)$$

when inserting the operator H_{s_t} at time-step t , and

$$A(\underline{s}' \rightarrow \underline{s}) = \min \left(1, \frac{M - n + 1}{\beta N_d \langle \alpha_t | H_{s_t} | \alpha_t \rangle} \right) \quad (4.12)$$

when removing the operator. It should be noted that when performing the simulation, the matrix elements in the above acceptance probabilities will need to be computed or retrieved memory as the operator H_{s_t} may assign different weights to different states, and in some cases may even assign a weight of zero. However, the Metropolis-Hastings algorithm with uniform selection probability is inefficient (meaning there will be many rejections) when there are more than two options which can be chosen, as is often the case. Indeed, even a simple nearest-neighbour interacting Hamiltonian with multiple sites will have a number of operators scaling as the number of sites. By switching to a different type of sampler, we can devise a more efficient diagonal update scheme.

4.1.2 A Problematic Heat-Bath Scheme

The Heat-Bath, or Gibbs algorithm, proceeds by selecting one random variable and updating it by sampling from its conditional distribution. Before we can do that, however, we'll need to define some new notation.

Let $H_d^\alpha = \langle \alpha | H_d | \alpha \rangle$, and let C be a compact notation for an SSE configuration which can be represented equivalently as $C = (n, \underline{\alpha}, \underline{s})$ or as

$$C = (\langle \alpha_0 | H_{s_1} | \alpha_1 \rangle, \dots, \langle \alpha_{M-1} | H_{s_M} | \alpha_M \rangle)$$

Then, we will say that the configuration $C \odot_p H_d^\alpha$ is the result of replacing the operator at time-slice p in C with H_d^α ¹.

Let us also extend our weight function Φ to be able to take inconsistent SSE configurations as input. We'll define an additional weight function which does not care about

¹We will often abuse this notation slightly when discussing updates which replace the operator at time-slice p with the identity operator, a process which we will write as $C \odot_p H_d^\alpha \rightarrow C$

configuration consistency, called Φ^* . The weight $\Phi^*(C)$ is computed as a simple product of matrix elements in C multiplied by $\frac{\beta^n(M-n)!}{M!}$. Thus, given an arbitrary (possibly inconsistent) configuration $C = (\langle \alpha_0 | H_{s_1} | \alpha'_1 \rangle, \langle \alpha_1 | H_{s_2} | \alpha'_2 \rangle, \dots, \langle \alpha_{M-1} | H_{s_M} | \alpha'_M \rangle)$, we have:

$$\Phi^*(C) = \frac{\beta^n(M-n)!}{M!} \prod_{p=1}^M \langle \alpha_{p-1} | H_{s_p} | \alpha'_p \rangle \quad (4.13)$$

On the other hand, recalling the extended representation of the SSE partition function (Eq. 3.34), $\Phi(C)$ is given by:

$$\Phi(C) = \frac{\beta^n(M-n)!}{M!} \prod_{p=1}^M \langle \alpha_{p-1} | H_{s_p} | \alpha'_p \rangle \delta_{\alpha'_p, \alpha_p} = \Phi^*(C) \delta_{\alpha', \alpha} \quad (4.14)$$

So Φ will map inconsistent configurations to zero as it should, Obviously, $\Phi(C) = \Phi^*(C)$ for all consistent configurations C .

Now, let's continue, we will begin by building a probability distribution which will consist of all diagonal matrix elements present in the Hamiltonian breakup. The probability distribution for diagonal matrix elements at time-slice p takes the form:

$$T(C \rightarrow C \odot_p H_{d_p}^{\alpha_p}) = \frac{\Phi^*(C \odot_p H_{d_p}^{\alpha_p})}{\Phi(C \odot_p H_0) + \sum_{d, \alpha} \Phi^*(C \odot_p H_d^\alpha)} \quad (4.15)$$

where the sum in the denominator runs over all matrix elements of all diagonal Hamiltonian sub-operators. In the sum, we are ignoring whether the matrix element H_d^α can even be placed at time-slice p ; configuration consistency can be enforced later, and this is how we can turn the distribution into a conditional distribution. Simplifying, we get

$$T(C \rightarrow C \odot_p H_{d_p}^{\alpha_p}) = \begin{cases} \frac{\beta H_{d_p}^{\alpha_p}}{(M-n) + \beta \sum_{d, \alpha} H_d^\alpha} & \text{if } d_p \neq 0 \\ \frac{M-n}{(M-n) + \beta \sum_{d, \alpha} H_d^\alpha} & \text{if } d_p = 0 \end{cases} \quad (4.16)$$

in the case that the original operator in C at time-slice p was the identity operator. Otherwise, the above expression needs to be modified by substituting $n \rightarrow n-1$. The dependence on n renders this method difficult to use in practice, as every update would require us to rebuild the probability distribution from scratch. Even with very efficient sampling methods such as the Alias Method [185, 214, 216], initializing the probability distribution will still

take $O(K)$ time, where K is the number of elements in the distribution. As a result, we must give up on the Heat-Bath scheme as it is currently formulated. We can note however, that the problem stems from the fact that we are including the identity element in the probability distribution. If we separated our sampling into two steps: decide to insert a Hamiltonian or identity operator and then decide which Hamiltonian operator to insert, we can keep the matrix element distribution constant.

4.1.3 Improving the Metropolis-Hastings Scheme

Instead of trying to Heat-Bath sample over all possible sub-operators including the identity, we split our sampling into two steps by using a non-uniform proposal distribution to choose the operator matrix elements and then accept the insertion using the Metropolis-Hastings Rule.

We begin by defining the probability distribution over all diagonal matrix elements:

$$q(d, \alpha) = \frac{1}{\mathcal{L}} \langle \alpha | H_d | \alpha \rangle \quad (4.17)$$

where it is understood that q only has support over pairs (d, α) which correspond to non-zero matrix elements of diagonal Hamiltonian sub-operators; $\mathcal{L} = \sum_{d, \alpha} H_d^\alpha$ is the normalizing constant. This probability distribution will be the selection probability for this Metropolis-Hastings scheme. Since there aren't too many diagonal matrix elements of Hamiltonian sub-operators (which is to say, the number of diagonal matrix elements grows only polynomially with the system size), we can store the entire vector of probabilities q in memory. As a result, drawing samples from q can be performed efficiently using the Alias Method [185, 214, 216]. In contrast to Sec. 4.1.2, the matrix element distribution only needs to be initialized once at the beginning of the QMC simulation.

If we encounter an identity operator at time-slice p , we will attempt to replace it with a diagonal matrix element, H_d^α , drawn from $q(d, \alpha)$. The detailed balance condition for such a process is then:

$$\frac{\Phi(C \odot_p H_d^\alpha)}{\Phi(C)} = \frac{T(C \rightarrow C \odot_p H_d^\alpha)}{T(C \odot_p H_d^\alpha \rightarrow C)} = \frac{A(C \rightarrow C \odot_p H_d^\alpha)g(C \rightarrow C \odot_p H_d^\alpha)}{A(C \odot_p H_d^\alpha \rightarrow C)g(C \odot_p H_d^\alpha \rightarrow C)} \quad (4.18)$$

giving the acceptance ratio:

$$\frac{A(C \rightarrow C \odot_p H_d^\alpha)}{A(C \odot_p H_d^\alpha \rightarrow C)} = \frac{\Phi(C \odot_p H_d^\alpha)g(C \odot_p H_d^\alpha \rightarrow C)}{\Phi(C)g(C \rightarrow C \odot_p H_d^\alpha)} \quad (4.19)$$

We now substitute $g(C \rightarrow C \odot_p H_d^\alpha) = q(d, \alpha)$ and $g(C \odot_p H_d^\alpha \rightarrow C) = 1$, giving:

$$\frac{A(C \rightarrow C \odot_p H_d^\alpha)}{A(C \odot_p H_d^\alpha \rightarrow C)} = \frac{\Phi(C \odot_p H_d^\alpha)}{\Phi(C)} \frac{\mathcal{L}}{\langle \alpha | H_d | \alpha \rangle} \quad (4.20)$$

The ratio of configuration weights will be slightly different that it was for the algorithm shown in Sec. 4.1.1. In the earlier section, we sampled the full diagonal operator, hence inserting it into the operator list posed no risk of breaking the consistency of the configuration. Here, on the other hand, we are sampling *matrix elements*, and as a result we must make sure that the resulting configuration is a valid one. When computing the ratio of configuration weights we therefore get some Kronecker delta factors:

$$\frac{\Phi(C \odot_p H_d^\alpha)}{\Phi(C)} = \frac{\beta \delta_{\alpha_{p-1}, \alpha} \langle \alpha | H_d | \alpha \rangle \delta_{\alpha, \alpha_p}}{M - n} = \frac{\beta \langle \alpha | H_d | \alpha \rangle \delta_{\alpha, \alpha_p}}{M - n} \quad (4.21)$$

where we used the fact that there is currently an identity operator in time-slice p , thus $\alpha_{p-1} = \alpha_p$ and we can drop one of the Kronecker deltas. The acceptance probability for the operator insertion move then becomes:

$$A(C \rightarrow C \odot_p H_d^\alpha) = \min \left(1, \frac{\beta \langle \alpha | H_d | \alpha \rangle \delta_{\alpha, \alpha_p}}{M - n} \frac{\mathcal{L}}{\langle \alpha | H_d | \alpha \rangle} \right) = \min \left(1, \frac{\beta \mathcal{L}}{M - n} \delta_{\alpha, \alpha_p} \right) \quad (4.22)$$

which can only be non-zero if $\alpha = \alpha_p$.

The acceptance probability for the reverse process, the operator removal move, where the diagonal operator at time-slice p is replaced with an identity operator, can similarly be shown to take the form:

$$A(C \odot_p H_d^\alpha \rightarrow C) = \min \left(1, \frac{M - n + 1}{\beta \mathcal{L}} \right) \quad (4.23)$$

We note that the Kronecker delta does not appear in this equation as the current configuration is already assumed to be consistent, and the identity operator assigns the unit weight to every basis state.

A nice property of the operator insertion probability (Eq. 4.22) is that the Kronecker delta can be moved outside of the min function, as it only takes the values zero and one.

$$A(C \rightarrow C \odot_p H_d^\alpha) = \min \left(1, \frac{\beta \mathcal{L}}{M - n} \right) \delta_{\alpha, \alpha_p} \quad (4.24)$$

Consequently, we can actually check whether operator insertion is possible *before* even drawing a sample from $q(d, \alpha)$, which is useful in cases where drawing samples from the

proposal distribution is expensive. Once the matrix element is drawn from q , we check whether $\alpha = \alpha_p$. If so, we insert the operator matrix element, otherwise we reject the insertion move and go on to the next time-slice.

The presence of the Kronecker delta presents an issue, as most proposed matrix elements will be rejected. Indeed, consider even the ferromagnetic TFIM again: say we propose to insert the matrix element $\langle 00|H_b|00\rangle$, where H_b is an Ising bond operator (see Eq. 3.28). If the basis state encountered is $|11\rangle$, the update will be rejected, in spite of the two matrix elements having equal weight. Thus, we need to devise a method to reduce rejections.

4.1.4 Can we try drawing samples until one is accepted?

It is reasonable to ask whether we can simply re-attempt the matrix element sampling step one which is compatible with the propagated basis state $|\alpha_p\rangle$ is drawn. As we shall soon see, this approach turns out to be extremely impractical.

If we repeatedly sample from $q(d, \alpha)$ until we draw a sample (d, α) such that $\alpha = \alpha_p$, we have in fact constructed a rejection sampling process (as discussed in Sec. 2.1.2) which produces a sample from the conditional distribution $q(d|\alpha_p)$. Effectively, we are drawing samples from an unnormalized distribution which we have expressed as:

$$q(d|\alpha_p) \sim \sum_{\alpha} q(d, \alpha) \delta_{\alpha, \alpha_p} \quad (4.25)$$

We can think of the rejection sampling process in which we enforce that $\alpha = \alpha_p$ as “masking” appropriate entries of the distribution $q(d, \alpha)$. Explicitly, the conditional distribution may take the forms:

$$q(d|\alpha_p) = \frac{q(d, \alpha_p)}{q(\alpha_p)} = \frac{\langle \alpha_p | H_d | \alpha_p \rangle}{q(\alpha_p) \mathcal{L}} = \frac{\langle \alpha_p | H_d | \alpha_p \rangle}{\mathcal{L}(\alpha_p)} \quad (4.26)$$

where $q(\alpha_p) = \mathcal{L}(\alpha_p)/\mathcal{L}$ and $\mathcal{L}(\alpha_p) = \sum_d \langle \alpha_p | H_d | \alpha_p \rangle$. Constructing the probability vector $q(d|\alpha_p)$ is, in principle, easy to do as $q(d, \alpha)$ is small enough to fit in memory. However, doing so would require iterating through every entry of $q(d, \alpha)$. Though the number of entries in $q(d, \alpha)$ scales polynomially in the system size N , this can still be quite costly. For example, a 2-local Hamiltonian with all-to-all interactions would have $O(N^2)$ bond operators, hence the number of entries in $q(d, \alpha)$ will scale quadratically with the system size. Since we will need to do this for *every time-slice*, in the end the procedure ends up being quite impractical, which is why we perform the simpler process of rejection sampling.

Now, let's analyze what happens to the acceptance probabilities for the insertion and removal moves if we use $q(d|\alpha_p)$ as our selection probability. Following the same steps as the previous section, the acceptance probability for the operator insertion move becomes:

$$A(C \rightarrow C \odot_p H_d^{\alpha_p}) = \min \left(1, \frac{\beta \langle \alpha_p | H_d | \alpha_p \rangle \delta_{\alpha_p, \alpha_p} \mathcal{L}(\alpha_p)}{M - n \langle \alpha_p | H_d | \alpha_p \rangle} \right) \quad (4.27)$$

$$= \min \left(1, \frac{\beta \mathcal{L}(\alpha_p)}{M - n} \right) \quad (4.28)$$

where we see that the Kronecker delta was trivially satisfied due to our sampling directly from the subset of matrix elements compatible with α_p .

The operator removal move then has acceptance probability:

$$A(C \odot_p H_d^{\alpha_p} \rightarrow C) = \min \left(1, \frac{M - n + 1}{\beta \mathcal{L}(\alpha_p)} \right) \quad (4.29)$$

The independence of the insertion probability from the actual sampled matrix element in this case allows us to delay sampling the matrix element until after the operator insertion move has been accepted, which is useful as rejection sampling (or explicitly constructing $q(d|\alpha_p)$) may generate significant computational overhead. However, we will still need to compute $\mathcal{L}(\alpha_p)$ no matter what, which will require just as much time (asymptotically) as explicitly constructing $q(d|\alpha_p)$. Hence, the entire advantage of using rejection sampling has been lost.

We must unfortunately conclude that employing rejection sampling for the sake of performing diagonal operator insertions is therefore impractical.

4.1.4.1 Diagonal Operator Replacement

Now we should note that there is one case in which one may repeatedly sample diagonal operators until one is accepted without needing to do any expensive computations. If time-slice p already contains a diagonal operator, we may replace the existing diagonal operator by drawing a new one from $q(d, \alpha)$. Indeed we can even repeatedly sample until a diagonal operator has been accepted (thereby sampling from $q(d|\alpha_p)$). This is possible because the proposal distribution $q(d|\alpha_p)$ is being used for both the forward and reverse processes, resulting in the cancellation of the normalizing constants. The acceptance ratio

for switching from the diagonal operator d to d' becomes:

$$\frac{A(C \odot_p H_d^{\alpha_p} \rightarrow C \odot_p H_{d'}^{\alpha_p})}{A(C \odot_p H_{d'}^{\alpha_p} \rightarrow C \odot_p H_d^{\alpha_p})} = \frac{\Phi(C \odot_p H_{d'}^{\alpha_p}) q(d|\alpha_p)}{\Phi(C \odot_p H_d^{\alpha_p}) q(d'|\alpha_p)} \quad (4.30)$$

$$= \frac{\langle \alpha_p | H_{d'} | \alpha_p \rangle}{\langle \alpha_p | H_d | \alpha_p \rangle} \frac{\mathcal{L}(\alpha_p)}{\langle \alpha_p | H_{d'} | \alpha_p \rangle} \frac{\langle \alpha_p | H_d | \alpha_p \rangle}{\mathcal{L}(\alpha_p)} \quad (4.31)$$

$$= 1 \quad (4.32)$$

The acceptance probability is unity, which means we do not need to compute $\mathcal{L}(\alpha_p)$ at all! Although in some cases rejection sampling may not be worth the computational overhead, one could easily impose a maximum number of iterations before aborting the process and leaving the current diagonal operator in place. As we will see in Sec. 4.3, this type of update will actually form the basis of the diagonal update for the Groundstate Projector variant of SSE.

As an aside, we may note that in the acceptance probability ratio above, had we used a proposal distribution which could generate samples inconsistent with α_p —for example, the joint distribution $q(d, \alpha)$ —then the resulting configuration would be assigned a weight of zero by Φ , resulting in an acceptance probability of zero as well. Indeed, in this case we find that the acceptance ratio is simply the Kronecker delta:

$$\frac{A(C \odot_p H_d^{\alpha_p} \rightarrow C \odot_p H_{d'}^{\alpha})}{A(C \odot_p H_{d'}^{\alpha} \rightarrow C \odot_p H_d^{\alpha_p})} = \frac{\Phi(C \odot_p H_{d'}^{\alpha}) q(d, \alpha_p)}{\Phi(C \odot_p H_d^{\alpha_p}) q(d', \alpha)} \quad (4.33)$$

$$= \frac{\langle \alpha | H_{d'} | \alpha \rangle \delta_{\alpha, \alpha_p}}{\langle \alpha_p | H_d | \alpha_p \rangle} \frac{\mathcal{L}}{\langle \alpha | H_{d'} | \alpha \rangle} \frac{\langle \alpha_p | H_d | \alpha_p \rangle}{\mathcal{L}} \quad (4.34)$$

$$= \delta_{\alpha, \alpha_p} \quad (4.35)$$

Although we can use this type of update independently of the insertion and removal moves, it is especially useful to attempt a replacement move upon failure of a removal move, thereby guaranteeing a change to the graphical structure defined by the operator matrix elements in the simulation cell. Indeed, this is intuitively the best way to employ the replacement move. If we perform a replacement move immediately after an insertion move, we have wasted the configuration that was produced by the insertion move. Similarly, performing a replacement move just before a removal move may result in the new diagonal operator being immediately removed, again wasting computation.

In order to employ the replacement move conditionally upon the failure of an operator removal, we must modify our selection probability to account for the failed removal as

$g(C \odot_p H_d^{\alpha_p} \rightarrow C \odot_p H_{d'}^{\alpha_p}) = q(d|\alpha_p) [1 - A(C \odot_p H_d^{\alpha_p} \rightarrow C)]$. The acceptance ratio is then:

$$\frac{A(C \odot_p H_d^{\alpha_p} \rightarrow C \odot_p H_{d'}^{\alpha_p})}{A(C \odot_p H_{d'}^{\alpha_p} \rightarrow C \odot_p H_d^{\alpha_p})} = \frac{\Phi(C \odot_p H_{d'}^{\alpha_p}) q(d|\alpha_p) [1 - A(C \odot_p H_d^{\alpha_p} \rightarrow C)]}{\Phi(C \odot_p H_d^{\alpha_p}) q(d'|\alpha_p) [1 - A(C \odot_p H_{d'}^{\alpha_p} \rightarrow C)]} \quad (4.36)$$

$$= \frac{[1 - A(C \odot_p H_d^{\alpha_p} \rightarrow C)]}{[1 - A(C \odot_p H_{d'}^{\alpha_p} \rightarrow C)]} \quad (4.37)$$

Noting that the removal probability in Eq. 4.23 is in fact independent of the diagonal operator, the acceptance ratio again simplifies to unity.

4.1.5 Reducing Rejections

Before we can proceed, recall that in Sec. 4.1.3 we noted that in some cases it is possible to delay sampling the diagonal operator matrix element until *after* the operator insertion move has been accepted. This allows us to develop two different strategies for reducing rejections depending on when we wish to sample the diagonal operator.

In the first strategy, we will develop a diagonal update scheme which begins by probabilistically deciding whether a diagonal operator should be inserted and if so, samples a diagonal operator which is then accepted with some probability. We will call this the *Three-Step Scheme*.

The second strategy will be simpler: we sample a diagonal operator, and then accept the proposed matrix element with probability given by the Metropolis-Hastings formula. This will be called the *Two-Step Scheme*.

4.1.6 The Three-Step Scheme

Inspecting the matrix element distribution in Eq. 4.17, we note that we can decompose the distribution into a conditional distribution over matrix elements and a marginal distribution over diagonal operators.

$$q(d, \alpha) = q(\alpha|d)q(d) = \frac{\langle \alpha | H_d | \alpha \rangle \text{Tr}(H_d)}{\text{Tr}(H_d) \mathcal{L}} \quad (4.38)$$

The above form of the distribution shows us that we may first draw an operator index from $q(d)$, and then accept it with probability $q(\alpha_p|d)$. This is equivalent to sampling directly from $q(d, \alpha)$, and then performing an accept-reject step based on whether

the propagated basis state α_p matches the drawn state α . A small source of confusion may arise here: since α_p is fixed, if we construct our selection probability distribution as $g(C \rightarrow C \odot_p H_d^{\alpha_p}) = q(d, \alpha_p)$ then it does not appear to actually be normalized properly. Indeed, if we try to sum over the index d , we find:

$$\sum_d q(d, \alpha_p) = \sum_d q(\alpha_p|d)q(d) \leq \sum_d q(d) = 1 \quad (4.39)$$

So where did the rest of the probability go? When the matrix element insertion is rejected we simply leave the current identity operator as is. This means that the remaining probability actually goes back to the identity operator! We may think of the matrix element rejection due to the basis state α_p as actually proposing to insert the identity operator. The selection probability, in its entirety, is then:

$$g(C \rightarrow C \odot_p H_d^{\alpha_p}) = q(\alpha_p|d)q(d) \quad (4.40)$$

$$g(C \rightarrow C) = \sum_{d'} (1 - q(\alpha_p|d'))q(d') \quad (4.41)$$

which is clearly normalized. When an identity operator is proposed it is then accepted with unit probability as the configuration weight does not change and the selection probability $g(C \rightarrow C)$ will appear for both the forward and reverse processes, resulting in a cancellation.

The goal then, is to minimize the chance of proposing to insert an identity operator.

In Eq. 4.38, we've defined $q(d)$ such that it assigns weight to each diagonal Hamiltonian sub-operator according to the operator's trace. Alternatively, we could use some other function of the matrix elements which would give a larger acceptance probability on average. What should such a function look like?

Let's begin by defining our new proposal distribution over diagonal operator matrix elements. We will call this distribution r and express it as a product of two factors, similar to how q was broken into two factors in Eq. 4.38,

$$g(C \rightarrow C \odot_p H_d^\alpha) = r(H_d)r(H_d \rightarrow H_d^\alpha) \quad (4.42)$$

The distribution over diagonal operators, $r(H_d)$, will be given by some function of the matrix elements of H_d , which we will call $w(H_d)$. Hence, $r(H_d) = w(H_d)/\mathcal{N}$ where $\mathcal{N} = \sum_d w(H_d)$. The probability $r(H_d \rightarrow H_d^\alpha)$ then becomes:

$$r(H_d \rightarrow H_d^\alpha) = \frac{\langle \alpha | H_d | \alpha \rangle}{w(H_d)} \quad (4.43)$$

The probability of proposing an identity operator is then:

$$g(C \rightarrow C) = \sum_d (1 - r(H_d \rightarrow H_d^{\alpha_p})) r(H_d) \quad (4.44)$$

$$= \sum_d \left(1 - \frac{\langle \alpha_p | H_d | \alpha_p \rangle}{w(H_d)} \right) \frac{w(H_d)}{\mathcal{N}} \quad (4.45)$$

In order to minimize the above probability, we would like the probability $r(H_d \rightarrow H_d^{\alpha_p})$ to be 1 (or close to it) as often as possible. The matrix element $\langle \alpha_p | H_d | \alpha_p \rangle$ is a fixed quantity that we have no control over, so we must instead try to minimize $w(H_d)$. Hence, setting $w(H_d) = \max_{\alpha} \langle \alpha | H_d | \alpha \rangle$ is the best choice as it is the smallest weight that we can assign to H_d without causing $r(H_d \rightarrow H_d^{\alpha_p})$ to exceed 1.

The acceptance probability of the diagonal operator insertion move will then be given by:

$$A(C \rightarrow C \odot_p H_d^{\alpha_p}) = \min \left(1, \frac{\Phi(C \odot_p H_d^{\alpha_p})}{\Phi(C)} \frac{g(C \odot_p H_d^{\alpha_p} \rightarrow C)}{g(C \rightarrow C \odot_p H_d^{\alpha_p})} \right) \quad (4.46)$$

$$= \min \left(1, \frac{\beta \langle \alpha_p | H_d | \alpha_p \rangle}{M - n} \frac{1}{r(H_d) r(H_d \rightarrow H_d^{\alpha_p})} \right) \quad (4.47)$$

$$= \min \left(1, \frac{\beta \langle \alpha_p | H_d | \alpha_p \rangle}{M - n} \frac{\mathcal{N}}{w(H_d)} \frac{w(H_d)}{\langle \alpha_p | H_d | \alpha_p \rangle} \right) \quad (4.48)$$

$$= \min \left(1, \frac{\beta \mathcal{N}}{M - n} \right) \quad (4.49)$$

which is independent of the proposed (non-identity) matrix element $H_d^{\alpha_p}$, meaning we can check to see whether a diagonal operator may be inserted even before we sample the operator from $r(H_d)$. If an identity operator was proposed (because a matrix element was rejected due to α_p), then the acceptance probability is of course unity as the configuration does not change; in this case we computed the probability in Eq. 4.49 unnecessarily, but this isn't a big issue as the computation is quite cheap.

The operator insertion move for this scheme therefore requires three steps. We first decide whether to replace an identity operator with a diagonal operator with probability given by Eq. 4.49. If so, we then propose a diagonal operator to insert by drawing from the distribution:

$$r(H_d) = \frac{1}{\mathcal{N}} \max_{\alpha} \langle \alpha | H_d | \alpha \rangle \quad (4.50)$$

The proposed diagonal operator matrix element is then accepted with probability

$$r(H_d \rightarrow H_d^{\alpha_p}) = \frac{\langle \alpha_p | H_d | \alpha_p \rangle}{\max_{\alpha} \langle \alpha | H_d | \alpha \rangle} \quad (4.51)$$

and the matrix element $\langle \alpha_p | H_d | \alpha_p \rangle$ is added to the configuration upon success. If the insertion is rejected, we simply move on to the next time-slice.

For the operator removal move, we can find the acceptance probability to be:

$$A(C \odot_p H_d^{\alpha_p} \rightarrow C) = \min \left(1, \frac{\Phi(C)}{\Phi(C \odot_p H_d^{\alpha_p})} \frac{g(C \rightarrow C \odot_p H_d^{\alpha_p})}{g(C \odot_p H_d^{\alpha_p} \rightarrow C)} \right) \quad (4.52)$$

$$= \min \left(1, \frac{M - n + 1}{\beta \langle \alpha_p | H_d | \alpha_p \rangle} \frac{r(H_d) r(H_d \rightarrow H_d^{\alpha_p})}{1} \right) \quad (4.53)$$

$$= \min \left(1, \frac{M - n + 1}{\beta \langle \alpha_p | H_d | \alpha_p \rangle} \frac{\langle \alpha_p | H_d | \alpha_p \rangle}{\mathcal{N}} \right) \quad (4.54)$$

$$= \min \left(1, \frac{M - n + 1}{\beta \mathcal{N}} \right) \quad (4.55)$$

which is also independent of $H_d^{\alpha_p}$, just like Eq. 4.49.

4.1.6.1 Diagonal Operator Replacement

We will briefly return to the optional move (introduced in Sec. 4.1.4.1) where we replace a diagonal operator with another, perhaps by repeated sampling until acceptance.

Using the proposal distribution $g(C \rightarrow C \odot_p H_d^{\alpha_p}) = r(H_d)$, the acceptance ratio for replacing the diagonal operator then becomes:

$$\frac{A(H_d^{\alpha_p} \rightarrow H_{d'}^{\alpha_p})}{A(H_{d'}^{\alpha_p} \rightarrow H_d^{\alpha_p})} = \frac{\Phi(C \odot_p H_{d'}^{\alpha_p}) g(H_{d'}^{\alpha_p} \rightarrow H_d^{\alpha_p})}{\Phi(C \odot_p H_d^{\alpha_p}) g(H_d^{\alpha_p} \rightarrow H_{d'}^{\alpha_p})} \quad (4.56)$$

$$= \frac{\langle \alpha_p | H_{d'} | \alpha_p \rangle (\max_{\alpha} \langle \alpha | H_d | \alpha \rangle) / \mathcal{N}}{\langle \alpha_p | H_d | \alpha_p \rangle (\max_{\alpha} \langle \alpha_p | H_{d'} | \alpha_p \rangle) / \mathcal{N}} \quad (4.57)$$

$$= \frac{\langle \alpha_p | H_{d'} | \alpha_p \rangle}{\max_{\alpha} \langle \alpha_p | H_{d'} | \alpha_p \rangle} \frac{\max_{\alpha} \langle \alpha | H_d | \alpha \rangle}{\langle \alpha_p | H_d | \alpha_p \rangle} \quad (4.58)$$

Inspecting the acceptance ratio we notice that if we set

$$A(H_d^{\alpha_p} \rightarrow H_{d'}^{\alpha_p}) = \frac{\langle \alpha_p | H_{d'} | \alpha_p \rangle}{\max_{\alpha} \langle \alpha | H_{d'} | \alpha \rangle} \quad (4.59)$$

then we are simply accepting the matrix element with probability given by $r(H_{d'} \rightarrow H_d^{\alpha_p})$, exactly as we do in the operator insertion move.

In order to reduce the probability of rejecting a proposed operator, we can use a rejection sampling procedure as part of our proposal distribution. To do so we would sample repeatedly from $r(H_d)$ until an operator is accepted with probability $r(H_d \rightarrow H_d^{\alpha_p})$. The resulting matrix element can then be accepted with unit probability, as we saw in Sec. 4.1.4.1. In a sense, we've moved the probability given by Eq. 4.59 into the sampling procedure, leaving $A(H_d^{\alpha_p} \rightarrow H_d^{\alpha_p}) = 1$.

If we wish to attempt the replacement move only once, we observe that the Metropolis-Hastings acceptance probability will be more efficient than Eq. 4.59, as it is guaranteed to be 1 for one direction.

$$A(H_d^{\alpha_p} \rightarrow H_{d'}^{\alpha_p}) = \min \left(1, \frac{\langle \alpha_p | H_{d'} | \alpha_p \rangle}{\max_{\alpha'} \langle \alpha' | H_{d'} | \alpha' \rangle} \frac{\max_{\alpha} \langle \alpha | H_d | \alpha \rangle}{\langle \alpha_p | H_d | \alpha_p \rangle} \right) \quad (4.60)$$

The right-most fraction is guaranteed to be at least one, therefore this acceptance probability will be at least as large as Eq 4.59. We note that if the matrix element $\langle \alpha_p | H_{d'} | \alpha_p \rangle$ is a maximal matrix element of the proposed operator $H_{d'}$, then we can accept it right away without needing to compute the acceptance probability. When using the Metropolis-Hasting probability in Eq. 4.60 to accept the matrix element, we may no longer attempt operator replacements until acceptance, since in doing so we are no longer properly performing rejection sampling to obtain a sample from $q(d|\alpha_p)$, and we will therefore break detailed balance.

Note that, just like in the case presented in Sec. 4.1.4.1, if we elect to attempt the replacement move conditionally upon failure of the operator removal move, the extra factor $1 - A(C \odot_p H_d^{\alpha_p} \rightarrow C)$ that we must include in the selection probability will again cancel with the associated factor from the reverse process, so we may still directly use the expression given in Eq. 4.60.

* * *

We summarize the diagonal operator replacement move in Algorithm 6 and the full Three-Step diagonal update scheme in Algorithm 7. The constant N_r is used to set the number of times the diagonal operator replacement process is attempted and may be set to any non-negative integer (even zero). In the case where $N_r = 1$, we use the Metropolis-Hastings acceptance probability when performing the replacement move.

Algorithm 6 The Diagonal Operator Replacement Move

```

1: function DIAGONALREPLACEMENT( $\alpha \in \mathbf{A}, s \in \mathbf{S}, N_r \geq 0$ )
2:   if  $N_r = 1$  then
3:      $H_d \sim r(H_d)$ 
4:      $u_1 \sim U[0, 1]$ 
5:     if  $u_1 \leq \frac{\langle \alpha | H_d | \alpha \rangle}{\max_{\alpha'} \langle \alpha' | H_d | \alpha' \rangle} \frac{\max_{\alpha'} \langle \alpha' | H_s | \alpha' \rangle}{\langle \alpha | H_s | \alpha \rangle}$  then  $\triangleright$  Probability given by Eq. 4.60
6:       return  $d$ 
7:     end if
8:   else if  $N_r > 1$  then  $\triangleright$  Generate a diagonal operator using rejection sampling
9:     for  $i = 1, \dots, N_r$  do
10:       $H_d \sim r(H_d)$ 
11:       $u_2 \sim U[0, 1]$ 
12:      if  $u_2 \leq \frac{\langle \alpha | H_d | \alpha \rangle}{\max_{\alpha'} \langle \alpha' | H_d | \alpha' \rangle}$  then  $\triangleright$  Probability given by Eq. 4.59
13:        return  $d$ 
14:      end if
15:    end for
16:   end if
17:   return  $s$   $\triangleright$  If  $N_r$  was reached then leave the current operator untouched.
18: end function

```

Algorithm 7 The Three-Step Diagonal Update Algorithm

Require: $n \in \mathbb{Z}, \alpha_0 \in \mathbf{A}, \underline{s} \in \mathbf{S}^M, \beta \in \mathbb{R}, \mathcal{N} \in \mathbb{R}, N_r \in \mathbb{Z}$ and $n \geq 0, \beta > 0, \mathcal{N} > 0, N_r \geq 0$

```

1:  $|\alpha\rangle \leftarrow |\alpha_0\rangle$ 
2: for  $p = 1, \dots, M$  do
3:   if  $H_{s_p}$  is off-diagonal then
4:      $|\alpha\rangle \leftarrow H_{s_p} |\alpha\rangle$ 
5:      $\triangleright |\alpha\rangle$  remains normalized, ignore any coefficients applied by  $H_{s_p}$  ◁
6:   else if  $s_p = 0$  then  $\triangleright H_{s_p}$  is the identity operator
7:      $u_1 \sim U[0, 1]$ 
8:      $\triangleright$  Decide whether to insert a diagonal operator ◁
9:     if  $u_1 \leq \frac{\beta \mathcal{N}}{M-n}$  then  $\triangleright$  Probability given by Eq. 4.49
10:       $H_d \sim r(H_d)$   $\triangleright$  Sample  $d$  according to  $H_d$ 's largest matrix element
11:       $u_2 \sim U[0, 1]$ 
12:      if  $u_2 \leq \frac{\langle \alpha | H_d | \alpha \rangle}{\max_{\alpha'} \langle \alpha' | H_d | \alpha' \rangle}$  then
13:         $s_p \leftarrow d$ 
14:         $n \leftarrow n + 1$ 
15:      end if
16:    end if
17:   else  $\triangleright H_{s_p}$  is a diagonal operator
18:      $u_1 \sim U[0, 1]$ 
19:      $\triangleright$  Decide whether to remove the diagonal operator  $H_{s_p}$  ◁
20:     if  $u_1 \leq \frac{M-n+1}{\beta \mathcal{N}}$  then  $\triangleright$  Probability given by Eq. 4.55
21:        $s_p \leftarrow 0$ 
22:        $n \leftarrow n - 1$ 
23:     else
24:        $s_p \leftarrow \text{DIAGONALREPLACEMENT}(\alpha, s_p, N_r)$ 
25:     end if
26:   end if
27: end for

```

4.1.7 The Two-Step Scheme

The Three-Step scheme is quite effective, but the need for two separate acceptance steps is often unnecessary. By drawing from the proposal distribution first, we may combine the two acceptance steps together. As a result, our proposal distribution simplifies greatly: $g(C \rightarrow C \odot_p H_d^{\alpha_p}) = r(H_d)$. We may note that there is no longer any chance of proposing an identity operator. The proposal distribution now selects the diagonal operator H_d completely independent of the basis state α_p , from which the matrix element is chosen deterministically. The acceptance probability is then given by:

$$A(C \rightarrow C \odot_p H_d^{\alpha_p}) = \min \left(1, \frac{\Phi(C \odot_p H_d^{\alpha_p})g(C \odot_p H_d^{\alpha_p} \rightarrow C)}{\Phi(C)g(C \rightarrow C \odot_p H_d^{\alpha_p})} \right) \quad (4.61)$$

For the reverse process, the selection probability is of course unity.

$$A(C \rightarrow C \odot_p H_d^{\alpha_p}) = \min \left(1, \frac{\beta \langle \alpha_p | H_d | \alpha_p \rangle}{M - n} \frac{1}{r(H_d)} \right) \quad (4.62)$$

$$= \min \left(1, \frac{\beta \langle \alpha_p | H_d | \alpha_p \rangle}{M - n} \frac{\mathcal{N}}{\max_{\alpha} \langle \alpha | H_d | \alpha \rangle} \right) \quad (4.63)$$

$$= \min \left(1, \frac{\beta \mathcal{N}}{M - n} \cdot \frac{\langle \alpha_p | H_d | \alpha_p \rangle}{\max_{\alpha} \langle \alpha | H_d | \alpha \rangle} \right) \quad (4.64)$$

In the Three-Step Scheme we can, at our discretion, switch the order of sampling the diagonal operator and deciding whether to insert an operator via the operator insertion probability (Eq. 4.49)². The acceptance probabilities can then be combined by a simple multiplication, giving:

$$A(C \rightarrow C \odot_p H_d^{\alpha_p})r(H_d \rightarrow H_d^{\alpha_p}) = \min \left(1, \frac{\beta \mathcal{N}}{M - n} \right) \frac{\langle \alpha_p | H_d | \alpha_p \rangle}{\max_{\alpha} \langle \alpha | H_d | \alpha \rangle} \quad (4.65)$$

which is of course different from Eq. 4.64. We will now compare the two acceptance probabilities. While it is clear that $\langle \alpha_p | H_d | \alpha_p \rangle / (\max_{\alpha} \langle \alpha | H_d | \alpha \rangle) \leq 1$, the ratio $\beta \mathcal{N} / (M - n)$ has no such bound. By applying the super-multiplicativity property of the Metropolis function $m(a) = \min(1, a)$: $m(ab) \geq m(a)m(b)$ (see Appendix A for the general proof), we have

$$A(C \rightarrow C \odot_p H_d^{\alpha_p}) = \min \left(1, \frac{\beta \mathcal{N}}{M - n} \frac{\langle \alpha_p | H_d | \alpha_p \rangle}{\max_{\alpha} \langle \alpha | H_d | \alpha \rangle} \right) \quad (4.66)$$

$$\geq \min \left(1, \frac{\beta \mathcal{N}}{M - n} \right) \min \left(1, \frac{\langle \alpha_p | H_d | \alpha_p \rangle}{\max_{\alpha} \langle \alpha | H_d | \alpha \rangle} \right) \quad (4.67)$$

²We prefer not to do this as it would result in sampling from $r(H_d)$ more than necessary.

Put simply, the overall acceptance probability for a diagonal operator insertion in the Three-Step scheme can be lower than that of the Two-Step scheme. We may conclude that this new scheme is therefore better at inserting diagonal operators.

In order to satisfy detailed balance we must also modify the operator removal probabilities:

$$A(C \odot_p H_d^{\alpha_p} \rightarrow C) = \min \left(1, \frac{M - n + 1}{\beta \mathcal{N}} \cdot \frac{\max_{\alpha} \langle \alpha | H_d | \alpha \rangle}{\langle \alpha_p | H_d | \alpha_p \rangle} \right) \quad (4.68)$$

and we will perform a similar comparison for this probability as well. The quantity $(\max_{\alpha} \langle \alpha | H_d | \alpha \rangle) / \langle \alpha_p | H_d | \alpha_p \rangle$ is at least one, hence:

$$\frac{M - n + 1}{\beta \mathcal{N}} \cdot \frac{\max_{\alpha} \langle \alpha | H_d | \alpha \rangle}{\langle \alpha_p | H_d | \alpha_p \rangle} \geq \frac{M - n + 1}{\beta \mathcal{N}} \quad (4.69)$$

Thus, we have that the operator removal probability of the Three-Step scheme can be smaller than that of the Two-Step scheme, meaning that this scheme is also more efficient at removing diagonal operators.

The Two-Step scheme therefore has a few advantages. First, the two acceptance steps of the Three-Step scheme have been combined into one, simplifying the diagonal operator insertion procedure. Second, the insertion and removal acceptance probabilities for this scheme are at least as large as those of the previously discussed scheme.

On the other hand, in this scheme we draw a sample from the diagonal operator distribution $r(H_d)$ every time the diagonal update comes across an identity operator. This could be problematic in cases where drawing from $r(H_d)$ is expensive. Of course, if one uses the Alias Method to sample from $r(H_d)$ this is not an issue. However, if one needs to reconstruct or modify the diagonal operator distribution frequently, it may be necessary to instead use alternative sampling methods which allow for efficient modifications to the distribution at the cost of requiring more resources when drawing samples, in which case the previous scheme may be more practical.

Before concluding this discussion, we may ask why we still must define the distribution $r(H_d)$ as weighting the diagonal operators by their largest matrix element. Indeed, the assumption that the ratio $\langle \alpha_p | H_d | \alpha_p \rangle / w(H_d)$ be at most 1 originally came from the fact that $r(H_d \rightarrow H_d^{\alpha})$ is a probability. Writing the insertion and removal probabilities in terms of $w(H_d)$,

$$A(C \rightarrow C \odot_p H_d^{\alpha_p}) = \min \left(1, \frac{\beta \mathcal{N}}{M - n} \cdot \frac{\langle \alpha_p | H_d | \alpha_p \rangle}{w(H_d)} \right) \quad (4.70)$$

$$A(C \odot_p H_d^{\alpha_p} \rightarrow C) = \min \left(1, \frac{M - n + 1}{\beta \mathcal{N}} \cdot \frac{w(H_d)}{\langle \alpha_p | H_d | \alpha_p \rangle} \right) \quad (4.71)$$

We see that we may reduce $w(H_d)$ in order to increase the ratio $\langle \alpha_p | H_d | \alpha_p \rangle / w(H_d)$ and therefore the insertion probability. However, doing so would also reduce the removal probability through the reciprocal ratio $w(H_d) / \langle \alpha_p | H_d | \alpha_p \rangle$. The normalizing constant \mathcal{N} would decrease as well, reducing some of the gain that was achieved in the ratio $\langle \alpha_p | H_d | \alpha_p \rangle / w(H_d)$, and potentially even decreasing the overall insertion probability.

4.1.7.1 Diagonal Operator Replacement

We may still use the diagonal operator replacement move when operator removal fails in the Two-Step scheme.

If we generate the new diagonal operator by sampling repeatedly from $r(H_d)$ until an operator is accepted with probability $r(H_d \rightarrow H_d^{\alpha_p})$, then, once the operator has been selected, the acceptance probability will take the form:

$$A(H_d^{\alpha_p} \rightarrow H_d^{\alpha_p}) = \min \left(1, \frac{1 - A(C \odot_p H_d^{\alpha_p} \rightarrow C)}{1 - A(C \odot_p H_d^{\alpha_p} \rightarrow C)} \right) \quad (4.72)$$

This is a little bit troublesome, as the rejection sampling algorithm may have required several iterations to produce a sample, only to be discarded by the above condition.

For the Two-Step scheme it may be more beneficial to forego rejection sampling for the diagonal replacement move and only perform a single sampling attempt. The acceptance probability will be slightly more complicated and will take the form:

$$A(H_d^{\alpha_p} \rightarrow H_d^{\alpha_p}) = \min \left(1, \frac{\langle \alpha_p | H_d^{\alpha_p} | \alpha_p \rangle}{\max_{\alpha'} \langle \alpha' | H_d^{\alpha_p} | \alpha' \rangle} \frac{\max_{\alpha} \langle \alpha | H_d | \alpha \rangle}{\langle \alpha_p | H_d | \alpha_p \rangle} \frac{1 - A(C \odot_p H_d^{\alpha_p} \rightarrow C)}{1 - A(C \odot_p H_d^{\alpha_p} \rightarrow C)} \right) \quad (4.73)$$

where we of course compute the removal probabilities $A(C \odot_p H_d^{\alpha_p} \rightarrow C)$ using Eq. 4.68. Since the insertion and removal probabilities are now dependent on the specific operator matrix element that we are attempting to insert/remove, the extra factor of the form $1 - A(C \odot_p H_d^{\alpha_p} \rightarrow C)$ that we must include in the selection probability no longer cancels out with the associated factor from the reverse process. The Three-Step scheme does not have this issue, as the insertion and removal probabilities are independent of the particular matrix element. Additionally, due to the presence of these extra factors, we cannot short-circuit the acceptance probability computation in the case where $H_d^{\alpha_p}$ is a maximal matrix element. The exception is if $H_d^{\alpha_p}$ is *also* a maximal matrix element, resulting in both matrix element ratios evaluating to one and the removal probabilities becoming equal, and thereby giving an acceptance probability of unity.

Algorithm 8 The Diagonal Operator Replacement Move (Two-Step Scheme Variant)

```

1: function DIAGONALREPLACEMENT2STEP( $\alpha \in A, s \in S, N_r \geq 0$ )
2:   if  $N_r = 1$  then
3:      $H_d \sim r(H_d)$ 
4:      $u_1 \sim U[0, 1]$ 
5:      $\triangleright$  Probability given by Eq. 4.73  $\triangleleft$ 
6:     if  $u_1 \leq \frac{\langle \alpha | H_d | \alpha \rangle}{\max_{\alpha'} \langle \alpha' | H_d | \alpha' \rangle} \frac{\max_{\alpha'} \langle \alpha' | H_s | \alpha' \rangle}{\langle \alpha | H_s | \alpha \rangle} \frac{1 - A(C \odot_p H_d^{\alpha p} \rightarrow C)}{1 - A(C \odot_p H_s^{\alpha p} \rightarrow C)}$  then
7:       return  $d$ 
8:     end if
9:   else if  $N_r > 1$  then  $\triangleright$  Generate a diagonal operator using rejection sampling
10:    for  $i = 1, \dots, N_r$  do
11:       $H_d \sim r(H_d)$ 
12:       $u_2 \sim U[0, 1]$ 
13:      if  $u_2 \leq \frac{\langle \alpha | H_d | \alpha \rangle}{\max_{\alpha'} \langle \alpha' | H_d | \alpha' \rangle}$  then  $\triangleright$  Probability given by Eq. 4.59
14:         $\triangleright$  If a diagonal operator is successfully generated, then accept it with
15:          probability given by Eq. 4.72  $\triangleleft$ 
16:           $u_3 \sim U[0, 1]$ 
17:          if  $u_3 \leq \frac{1 - A(C \odot_p H_d^{\alpha p} \rightarrow C)}{1 - A(C \odot_p H_s^{\alpha p} \rightarrow C)}$  then
18:            return  $d$ 
19:          else
20:            return  $s$ 
21:          end if
22:        end if
23:      end for
24:    return  $s$   $\triangleright$  If  $N_r$  was reached then leave the current operator untouched
25: end function

```

* * *

We summarize the Two-Step diagonal update scheme in Algorithm 9 along with a compatible diagonal operator replacement move in Algorithm 8. As in the case of the Three-Step scheme, the constant N_r is used to set the number of times the diagonal operator replacement process is attempted and may be set to any non-negative integer (even zero).

Algorithm 9 The Two-Step Diagonal Update Algorithm

Require: $n \in \mathbb{Z}, \alpha_0 \in \mathbf{A}, \underline{s} \in \mathbf{S}^M, \beta \in \mathbb{R}, \mathcal{N} \in \mathbb{R}, N_r \in \mathbb{Z}$ and $n \geq 0, \beta > 0, \mathcal{N} > 0, N_r \geq 0$

```

1:  $|\alpha\rangle \leftarrow |\alpha_0\rangle$ 
2: for  $p = 1, \dots, M$  do
3:   if  $H_{s_p}$  is off-diagonal then
4:      $|\alpha\rangle \leftarrow H_{s_p} |\alpha\rangle$ 
5:      $\triangleright |\alpha\rangle$  remains normalized, ignore any coefficients applied by  $H_{s_p}$  ◁
6:   else if  $s_p = 0$  then  $\triangleright H_{s_p}$  is the identity operator
7:      $H_d \sim r(H_d)$   $\triangleright$  Sample  $d$  according to  $H_d$ 's largest matrix element
8:      $u \sim U[0, 1]$ 
9:      $\triangleright$  Decide whether to insert the diagonal operator  $H_d$  ◁
10:    if  $u \leq \frac{\beta \mathcal{N}}{M-n} \frac{\langle \alpha | H_d | \alpha \rangle}{\max_{\alpha'} \langle \alpha' | H_d | \alpha' \rangle}$  then  $\triangleright$  Probability given by Eq. 4.64
11:       $s_p \leftarrow d$ 
12:       $n \leftarrow n + 1$ 
13:    end if
14:  else  $\triangleright H_{s_p}$  is a diagonal operator
15:     $u_2 \sim U[0, 1]$ 
16:     $\triangleright$  Decide whether to remove the diagonal operator  $H_{s_p}$  ◁
17:    if  $u_2 \leq \frac{M-n+1}{\beta \mathcal{N}} \frac{\max_{\alpha'} \langle \alpha' | H_{s_p} | \alpha' \rangle}{\langle \alpha | H_{s_p} | \alpha \rangle}$  then  $\triangleright$  Probability given by Eq. 4.68
18:       $s_p \leftarrow 0$ 
19:       $n \leftarrow n - 1$ 
20:    else  $\triangleright$  Optional: replace the diagonal operator  $H_{s_p}$  with another
21:       $s_p \leftarrow \text{DIAGONALREPLACEMENT2STEP}(\alpha, s_p, N_r)$ 
22:    end if
23:  end if
24: end for

```

4.1.8 Full Diagonal Operator Replacement Sweep

We will now briefly discuss an update consisting entirely of diagonal operator replacements. That is, we perform a full sweep through the operator list without attempting any insertions or removals, only replacement moves. This procedure is summarized in Algorithm 10, and we note that the number of iterations N_r should be greater than 0, or else the update will do nothing. Although this style of update fulfills the goal of modifying the topology of the simulation cell, it will not change the expansion order, and should therefore not be used in place of the usual diagonal update schemes. In fact, even when used alongside the usual diagonal updates it is wasteful. If we perform a diagonal replacement sweep before the usual diagonal update sweep, there will be some computation wasted due to newly replaced diagonal operators being immediately removed. On the other hand, performing the replacement sweep afterwards will result in some of the computation in the standard diagonal update sweep being wasted due to newly inserted operators being immediately replaced.

Nevertheless, this update is still useful to us as it will be the only diagonal update that we will be able to use in the groundstate projector SSE simulations, as we will see in Sec. 4.3.

Algorithm 10 The Diagonal Operator Replacement Sweep

Require: $\alpha_0 \in \mathcal{A}$, $\underline{s} \in \mathcal{S}^M$, $N_r \in \mathbb{Z}$ and $N_r > 0$

```

1:  $|\alpha\rangle \leftarrow |\alpha_0\rangle$ 
2: for  $p = 1, \dots, M$  do
3:   if  $H_{s_p}$  is off-diagonal then
4:      $|\alpha\rangle \leftarrow H_{s_p} |\alpha\rangle$ 
5:   else if  $s_p \neq 0$  then  $\triangleright H_{s_p}$  is a diagonal (non-identity) operator
6:      $s_p \leftarrow \text{DIAGONALREPLACEMENT}(\alpha, s_p, N_r)$ 
7:   end if
8: end for

```

4.1.9 A Brief Note on the Truncation Order M

As mentioned in Sec. 3.1, the truncation order M is typically determined during the equilibration phase of the Monte Carlo simulation. We periodically check if the expansion order n is too close to M , in which case we remove all identity operators in the operator string

and set $M = \lceil cn_{\max} \rceil$ where the factor $c \in (1, 2)$ is a (heuristic) simulation constant³ and n_{\max} is the maximum expansion order seen so far. When this happens, we grow the operator string by adding identity operators to the end. At the end of the equilibration step, having kept track of the maximum expansion order thus far, we allow the configuration to grow one last time by setting $M = \lceil cn_{\max} \rceil$.

The initial value of M , which we will call M_{init} , is usually not discussed, though it obviously must be greater than zero. In the following, we hope to make some progress towards more well founded methods of setting each of these values.

Focusing on the Three-Step diagonal update scheme from Sec. 4.1.6, as it is slightly simpler to analyze for the sake of the current discussion, consider again the operator insertion and removal probabilities (Eqs. 4.49 and 4.55). As the number of operators n approaches M , the chance of inserting more diagonal operators increases, while the chance of removing them decreases. Conversely, as the number of identity operators, $M - n$ increases, the chance of inserting diagonal operators decreases, while the chance of removing them increases.

Therefore, we do not want either the number of Hamiltonian or identity operators to approach M . We may then consider it best to want both of these acceptance probabilities to be close to 1 on average, so we must tune M in pursuit of this goal. We would therefore like the number of identity operators, $M - n$, to be close to the quantity $\beta\mathcal{N}$. Recalling Eq. 3.56, the average number of operators is related to the expectation of the (shifted) Hamiltonian:

$$\langle n \rangle = -\beta \langle H \rangle \quad (4.74)$$

Hence, we would like M to be roughly

$$M_{\text{optimal}} = \beta\mathcal{N} + \langle n \rangle = \beta(\mathcal{N} - \langle H \rangle) \quad (4.75)$$

Due to the considerable efficiency of the diagonal update, the mean $\langle n \rangle$ tends to equilibrate quite fast. Hence, during the equilibration phase, we can compute the running average of the number of Hamiltonian operators, \bar{n} ⁴, and then set M to $\beta\mathcal{N} + \bar{n}$ (rounded to an integer⁵) at regular intervals. If one were worried about the incomplete equilibration of

³We do not want c to be too large as the time complexity of the Monte Carlo updates scale with M . Using a value of M which is too large will slow down the simulation with little benefit.

⁴We use the over-bar here in place of $\langle \cdot \rangle_{MC}$ in order to highlight that \bar{n} may not yet be an unbiased estimator for the true expectation value.

⁵Whether to round up or down here is a choice left to the programmer. Rounding up will result in a probabilities slightly favouring the operator removal move, while rounding down will slightly favour operator insertions. As β or the system size increase (thereby increasing the product $\beta\mathcal{N}$) the discrepancy decreases.

$\langle n \rangle$, one could instead compute \bar{n} as a windowed average over the most recent fraction (for example 10%) of configurations encountered. By the end of the equilibration phase, the bias in this estimator should be quite small.

It is straightforward to see that in the case of the Two-Step scheme, M_{optimal} can be estimated by $\beta\mathcal{N} \langle r(H_d \rightarrow H_d^{\alpha p}) \rangle + \langle n \rangle$. That is, we rescale the first term by the average diagonal matrix element ratio; again, this average can be easily estimated during the equilibration phase. We note that the Two-Step scheme is therefore capable of attaining optimal insertion and removal probabilities using a smaller simulation cell compared to the Three-Step scheme.

The issue of truncation errors may arise when using M_{optimal} if the off-diagonal part of the Hamiltonian is much stronger than the diagonal part. In this case $\beta\mathcal{N}$ may not be very large compared to the fluctuations in n , and we may find that M_{optimal} is not large enough to perform an effective simulation. As this thesis is predominantly focused on Hamiltonians which have very strong diagonal parts, we will not worry about this problem here.

Next, we move on to setting a good initial value of M , focusing on the Three-Step scheme. Splitting the Hamiltonian into diagonal and off-diagonal parts $H = -H_D - H_O$, where $H_D = \sum_d H_d$:

$$\frac{M}{\beta} = \mathcal{N} + \langle H_D \rangle + \langle H_O \rangle \leq \mathcal{N} + \max_{\alpha} \langle \alpha | H_D | \alpha \rangle + \langle H_O \rangle \quad (4.76)$$

and noting that, $\langle n \rangle \geq 0$, as well as:

$$\max_{\alpha} \langle \alpha | H_D | \alpha \rangle = \max_{\alpha} \sum_d \langle \alpha | H_d | \alpha \rangle \leq \sum_d \max_{\alpha} \langle \alpha | H_d | \alpha \rangle = \mathcal{N} \quad (4.77)$$

We have:

$$\beta\mathcal{N} \leq M < 2\beta\mathcal{N} + \beta \langle H_O \rangle \quad (4.78)$$

A priori, it is difficult to determine an upper bound on H_O , and indeed it is unnecessary as M_{init} does not need to be a very good estimate of the value in Eq. 4.75. By setting M_{init} to $2\beta\mathcal{N}$ at the start of the simulation, we may be overestimating the necessary size of the simulation cell. However, consider the first step of the diagonal update: as we iterate through time slices, we will encounter only identity operators, meaning the operator insertion move will be the only move that is attempted. The insertion probability will begin at $1/2$, and gradually decrease as the update progresses through the time-slices and once the diagonal update is complete, the operator string will contain a mixture of diagonal and identity operators. On the other hand, $M_{\text{init}} = \beta\mathcal{N}$ would be a much poorer choice, as the

first diagonal update would have unit insertion probability at every time-slice, resulting in an operator string that is almost entirely filled. The next few diagonal updates would then again proceed to perform even more insertion moves and very few removal moves, filling the cell until M is increased.

We will conclude by noting that this is one of the less important optimizations we can perform. Although using the usual $M_{\text{heuristic}} = \lceil cn_{\text{max}} \rceil$ may result in a simulation cell which is too small to give nearly optimal insertion and removal probabilities, the time complexity of SSE scales with M . Consequently, although the M_{optimal} may sometimes give better correlation times, using a value of M which is big enough to not risk encountering truncation errors, but smaller than M_{optimal} would reduce the runtime of each Monte Carlo step. On the other hand, there is also a chance that $M_{\text{heuristic}} > M_{\text{optimal}}$, for certain systems and simulation parameters. Hence, taking $M = \min(M_{\text{heuristic}}, M_{\text{optimal}})$ would be the best choice in the interest of reducing total runtime, so long as M_{optimal} is large enough to avoid truncation errors.

When using M_{optimal} it is no longer useful to include diagonal operator replacement moves in the simulation, as the chance of them ever being used is quite low due to the operator removal probabilities now being close to one. Conversely, if M_{optimal} is found to be too large (thereby introducing significant computational overhead) or too small (and hence introducing truncation errors), using $M_{\text{heuristic}}$ while employing replacement moves allows us to update the simulation cell topology much more frequently, even if the insertion or removal probabilities are less than unity.

4.1.10 Example: TFIM Diagonal Update

As a short example, we now return to the simple (ferromagnetic) TFIM and construct the matrix element distribution for the bond terms of that Hamiltonian.

$$H = -J \sum_{\langle i,j \rangle} \sigma_i^z \sigma_j^z - h \sum_i \sigma_i^x \quad (3.3)$$

Recall from Eq. 3.28, the ferromagnetic Ising bond operators have non-zero matrix elements $\langle 00|H_b|00 \rangle = \langle 11|H_b|11 \rangle = 2J$. Additionally, when performing an SSE simulation of the TFIM, we must add diagonal operators $h\mathbf{1}$ at each site, where h is the transverse field strength; the reason for this will be discussed further in Sec. 4.2. The maximal matrix elements of the diagonal sub-operators are $2J$ for the N_b bond operators, and h for the N site operators. The normalizing constant is therefore $\mathcal{N} = hN + 2JN_b$. A site operator is selected with probability proportional to h , while a bond operator is selected

with probability proportional to $2J$. We do not need to perform an acceptance step for the site operators as they are proportional to the identity and therefore all matrix element ratios will be one. When inserting bond operators we only need to check that the spins are aligned since $\langle 01|H_b|01\rangle = \langle 10|H_b|10\rangle = 0$, and for aligned spins the matrix elements are equal, $\langle 00|H_b|00\rangle = \langle 11|H_b|11\rangle = 2J$. Hence, the matrix element ratio will reduce to a Kronecker delta of the two sites that are connected by the bond operator.

The operator insertion and removal probabilities become:

$$A(n \rightarrow n + 1) = \min \left(1, \frac{\beta(hN + 2JN_b)}{M - n} \right) \quad (4.79)$$

$$A(n \rightarrow n - 1) = \min \left(1, \frac{M - n + 1}{\beta(hN + 2JN_b)} \right) \quad (4.80)$$

which match the existing formulas in the literature [136,173]. Since the Kronecker delta only takes the values zero and one, it can be included as a multiplicative factor inside of the Metropolis function of the above insertion probability or outside of it. This property means that the operator insertion and removal probabilities of both the Two- and Three-Step schemes are identical, and the only difference between them is the order in which one draws the operator and checks the alignment of spins and acceptance probability. In some sense the Kronecker delta allows the two steps to “commute” without requiring any adjustment of weight ratios.

4.1.11 Summary

In this section we developed several schemes to insert diagonal operators into the SSE simulation cell. As we iterated upon our update strategies we arrived at a pair of update schemes which each have their respective advantages and disadvantages. In cases where sampling from the distribution of diagonal operators may be costly, the Three-Step scheme should be preferred, as one only needs to sample the diagonal operator if the insertion move was accepted. On the other hand, the Two-Step scheme has operator insertion and removal probabilities that are at least as large as those of the Three-Step scheme, which could contribute to faster mixing of the Markov chain for certain systems. For both cases we may augment the schemes with diagonal operator replacement moves, ideally conditioned upon the failure of an operator removal, in order to increase the chances of an update to the simulation cell topology.

Finally, we should note that for practical implementations, it is recommended that the matrix element ratios $\langle \alpha|H_d|\alpha\rangle / (\max_{\alpha'} \langle \alpha'|H_d|\alpha'\rangle)$ be precomputed during the construc-

tion of the diagonal operator distribution $r(H_d)$ at the initialization phase of the simulation in order to avoid repeated computation during the diagonal update.

4.2 Off-Diagonal Updates

In order to introduce off-diagonal Hamiltonian operators into the simulation cell we must devise another update scheme. These updates will need to be non-local in the simulation cell, both for the sake of efficiency and in order to maintain the consistency of the resulting SSE configurations; they will therefore take the form of cluster updates. Many different off-diagonal update schemes have been proposed, each with its own realm of applicability in terms of the types of Hamiltonians it can efficiently simulate. Update schemes of note include the Loop algorithms [55], such as the Operator-Loop Update [172] along with its successor the Directed-Loop Update [194, 179], as well as the Multibranch Update [173] which we will focus on here.

Most common off-diagonal updates in SSE work by proposing moves which convert diagonal operators into off-diagonal operators and vice versa. As such, we often need to add another energy shift to the Hamiltonian in the form of diagonal operators proportional to the identity which are similar to the off-diagonal ones that are present in the Hamiltonian. In the case of the TFIM, we introduce single site operators $H_{e_i} = h_i \mathbf{1}_i$ where h_i is the strength of the transverse field on site i , and the identity operator only operates on that site. The off-diagonal update will then propose updates which switch between the diagonal operators H_{e_i} and off-diagonal operators $H_{f_i} = h_i \sigma_i^x$. Note that the operators H_{e_i} were given matrix elements equal to h_i so as to not produce a change in configuration weight during the updates. Recalling the discussion on operator tuples from Sec. 3.2.5, the TFIM simulation now has a total of four different types of operators: the trivial identity operators $(0, 0, 0)$, the bond operators (b, i, j) , the diagonal site operators $(e, i, 0)$, and finally the transverse field operators $(f, i, 0)$. Again, the integer constants b , e , and f can be set as the user pleases so long as none of them are equal.

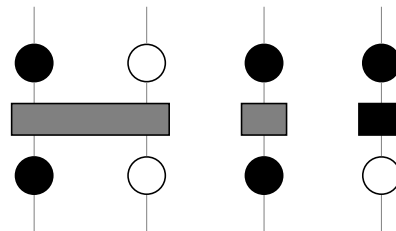


Figure 4.1: Examples of bond and site operators which may exist in an SSE simulation.

The update will proceed by forming a cluster within the simulation cell, connecting spins from different spatial locations and time-slices, and updating them all together. In doing this, the operators which are acting on those spins will be updated as well. We can

draw the bond and site operators of the simulation cell diagrammatically as in Fig. 4.1. In this section we will draw diagonal operators as gray rectangles while off-diagonal operators will be black; circles will denote spin states. It will be useful for the purposes of the cluster construction to view the operators as vertices which are connected by the spins' timelines, which we will often call "legs"⁶. Generally, when constructing a cluster, the cluster enters an operator vertex through one of its legs, and then may exit through one (or more) of the vertex's other legs, meaning those exit legs get added to the cluster. The specific rules the cluster must follow when selecting which legs to exit through are determined by the operator and the flavour of the cluster construction method. After exiting an operator vertex, the cluster then follows the edge down the time dimension of the simulation cell (possibly looping back around due to the periodic boundary conditions) until it encounters another operator vertex, at which point exit legs are added to the cluster following the predefined rules. Due to the way the cluster propagates through the configuration, we often use the term "worm-head" (or sometimes simply "worm") to refer to the end of the cluster which is moving through the simulation cell and adding vertex legs to the cluster. Cluster construction is often started by selecting a random vertex's⁷ leg in the simulation cell and placing the worm-head there. Given that that leg is set to be flipped, the configuration would then be inconsistent, meaning that this initial step has created a *defect* in the configuration. Hence, the worm-head can be seen as a defect which is spreading through the simulation cell. In a spin system, these defects would correspond to spin-flip operators⁸ which would propagate through the configuration, flipping spins along the way. Eventually the branches of a cluster will either terminate by closing on in themselves (worm-heads annihilating; defects "healing"), or will be blocked from proceeding further by a suitable operator vertex (worm-heads getting "stuck"). Once this occurs, the cluster can be flipped, meaning every vertex leg in the cluster is flipped and the operator matrix elements are updated accordingly. This of course must be done in such a way which respects detailed balance, either through the cluster construction rules (which may be probabilistic), through a Metropolis probability which decides whether the cluster flip can occur, or a combination of the two. Practitioners often endeavour to devise updates which do not require a Metropolis step that decides whether the cluster flip occurs, as the update would then require traversing the cluster twice: first to construct the cluster, then again to flip it if needed. If we only need to traverse a cluster once, we can actually flip the cluster as it is traversed, and it is therefore generally preferred to devise cluster construction

⁶The original loop algorithms were developed for classical statistical mechanical models known as "vertex models" [56, 57]. The operator and directed loop algorithms were heavily inspired by these classical update schemes, hence the common nomenclature.

⁷Which vertices may be chosen for this step is a detail of the specific cluster algorithm.

⁸For a spin-1/2 system expressed in the σ^z -basis, the σ^x operator would play this role.

rules which avoid the need for a Metropolis step. This can be done in one of two ways. One could construct probabilistic rules which introduce selection probabilities that cancel the weight change associated with a flip, as is done in the Operator-Loop [172] and Directed-Loop [194, 179] schemes. Alternatively, one can construct clusters deterministically which do not change the configuration weight upon flipping, as is done in the Multibranch scheme for the TFIM [173] which we will begin discussing now.

We will start simple and begin with the cluster construction rules when encountering site operators first and then we will move on to the rules for bond operators.

4.2.1 TFIM Site Operators

To perform updates of the form $H_{e_i} \leftrightarrow H_{f_i}$, we need to flip one leg of the site operator. Hence, we can impose the rule that a worm-head entering a leg of a site operator will terminate there. The worm is no longer allowed to move forward and it cannot turn and go backwards either—it is stuck to the site operator.

This construction rule is completely deterministic, and hence, this step of the cluster construction will have unit selection probability. Additionally, due to how we defined H_{e_i} , switching the operator type will also not change the configuration weight, since $\langle \circ | H_{e_i} | \circ \rangle = \langle \bullet | H_{e_i} | \bullet \rangle = \langle \circ | H_{f_i} | \bullet \rangle = \langle \bullet | H_{f_i} | \circ \rangle$. If our cluster consisted only of site operators we could therefore flip them with unit probability.

Viewing the worm-head as a σ_i^x operator, matrix element transitions such as $\langle \circ | H_{e_i} | \circ \rangle$ to $\langle \bullet | H_{f_i} | \circ \rangle$ simply correspond to the σ_i^x operator being “absorbed” into the operator H_{e_i} through multiplication, thereby turning it into the operator H_{f_i} :

$$\langle \bullet | \sigma_i^x | \circ \rangle \langle \circ | H_{e_i} | \circ \rangle \rightarrow \langle \bullet | \sigma_i^x H_{e_i} | \circ \rangle \rightarrow \langle \bullet | H_{f_i} | \circ \rangle$$

The possible site operator updates are laid out in Fig. 4.2. The arrow entering the site near the bottom of each sub-figure represents the worm entering the vertex leg. We draw

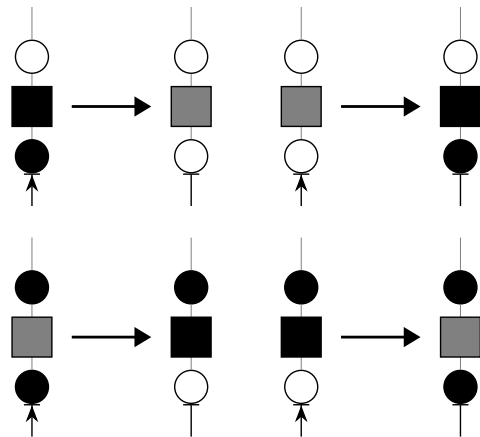


Figure 4.2: Cluster moves for the TFIM site operators. When a worm-head reaches a site operator, it is no longer allowed to proceed forward or backward.

a line perpendicular to the worm's path to show that the incoming arrow cannot proceed further. After the cluster update has been accepted, the spin state associated with the vertex leg has been flipped as well, and we see that the cluster has terminated at that point.

This update scheme therefore produces the following transitions between site operator matrix elements:

$$\begin{array}{ccc}
 \langle \circ | H_{e_i} | \circ \rangle & \longleftrightarrow & \langle \circ | H_{f_i} | \bullet \rangle \\
 \updownarrow & & \updownarrow \\
 \langle \bullet | H_{f_i} | \circ \rangle & \longleftrightarrow & \langle \bullet | H_{e_i} | \bullet \rangle
 \end{array}$$

In principle, we could also propose transitions along the diagonals of the above diagram by allowing the incoming worm to pass through the site operator vertex and exit along the opposite leg. Although the diagonal transitions can also result from a sequence of two moves along the perimeter of the transition diagram, we may hope that allowing these diagonal moves would result in lower autocorrelation times as a result of the Markov chain moving faster through the configuration space. However, doing so would result in the cluster construction process for the site operator to become non-deterministic as there are now two possible options for the incoming worm, which we would need to choose from uniformly at random in order to produce selection probabilities which cancel out. Additionally, consider that the site operator has two sides which, in the deterministic construction algorithm, may each be occupied by separate clusters. By allowing the worm to pass through the site operator we are now linking the two clusters on either side of the operator together, thereby preventing them from flipping independently of one another which could in fact hinder sampling efficiency, but this would depend on the exact system being simulated.

4.2.2 TFIM Bond Operators

Next we will consider how to perform updates to the TFIM bond operators. Recalling that, with the standard energy shift, the bond operator only has two non-zero matrix elements: $\langle \circ \circ | H_{b_{ij}} | \circ \circ \rangle = \langle \bullet \bullet | H_{b_{ij}} | \bullet \bullet \rangle = 2|J_{ij}|$ for a ferromagnetic bond, or $\langle \circ \bullet | H_{b_{ij}} | \circ \bullet \rangle = \langle \bullet \circ | H_{b_{ij}} | \bullet \circ \rangle = 2|J_{ij}|$ for an antiferromagnetic bond. As a result, in order to maintain a configuration with non-zero weight, we can only have matrix element transitions of the form $\langle \circ \circ | H_{b_{ij}} | \circ \circ \rangle \leftrightarrow \langle \bullet \bullet | H_{b_{ij}} | \bullet \bullet \rangle$ or $\langle \circ \bullet | H_{b_{ij}} | \circ \bullet \rangle \leftrightarrow \langle \bullet \circ | H_{b_{ij}} | \bullet \circ \rangle$. This means that a

worm entering a bond operator vertex must now *branch-out*⁹, and follow the three other vertex legs out of the bond.

In the picture where the worms are σ^x operators travelling through the simulation cell, a worm on site i entering the bond operator $H_{b_{ij}}$ corresponds to the multiplication $\sigma_i^x H_{b_{ij}}$. However, as we just mentioned, only certain spin configurations give non-zero matrix elements, and unlike in the site operator case from earlier, we cannot simply absorb the σ^x operator to turn the bond into one of a different type. We can proceed by instead making the worm infestation three times worse! By generating four new σ^x operators along each leg of the bond operator, one will annihilate with the incoming operator and the other three will be allowed to propagate further through the simulation cell. Here we illustrate an example of this process for an antiferromagnetic bond:

$$\begin{aligned}
& \langle \bullet_i | \sigma_i^x | \circ_i \rangle \langle \circ_i \bullet_j | H_{b_{ij}} | \circ_i \bullet_j \rangle \\
& \rightarrow \langle \bullet_i \bullet_j | \sigma_i^x H_{b_{ij}} | \circ_i \bullet_j \rangle \\
& \rightarrow \langle \bullet_i \bullet_j | \sigma_i^x \sigma_i^x \sigma_j^x H_{b_{ij}} \sigma_i^x \sigma_j^x | \circ_i \bullet_j \rangle \\
& \rightarrow \langle \bullet_i \bullet_j | \mathbf{1}_i \sigma_j^x H_{b_{ij}} | \bullet_i \circ_j \rangle \langle \bullet_i \circ_j | \sigma_i^x \sigma_j^x | \circ_i \bullet_j \rangle \\
& \rightarrow \langle \bullet_j | \sigma_j^x | \circ_j \rangle \langle \bullet_i \circ_j | H_{b_{ij}} | \bullet_i \circ_j \rangle \langle \bullet_i | \sigma_i^x | \circ_i \rangle \langle \circ_j | \sigma_j^x | \bullet_j \rangle
\end{aligned}$$

The multibranch cluster construction rule is illustrated in Fig. 4.3. We use the doubled arrowhead to denote the branching “worm”. Since all legs of the bond are deterministically added to the cluster, this step has unit selection probability. The non-zero matrix elements of the bond are also equal, due to the \mathbb{Z}_2 symmetry of the Ising interaction, hence flipping all legs of a bond vertex will not change the weight of the configuration.

4.2.3 The Multibranch Update for the TFIM

Due to the \mathbb{Z}_2 symmetry of the TFIM interaction, the full cluster update ends up being quite simple and elegant. It is completely deterministic and does not change the configuration weight, meaning no Metropolis acceptance step is required in order to determine whether a cluster flip should occur. As a result, the cluster can be flipped *while it is being constructed*.

The off-diagonal update for the TFIM proceeds as follows: the cluster construction begins by placing a worm at the leg of a site operator vertex, the worm follows the site’s

⁹Hence the name “multibranch” for this update algorithm. On another note, this is a case where the singular “worm” picture starts to break down, as the worm now needs to multiply whenever it encounters a bond operator.

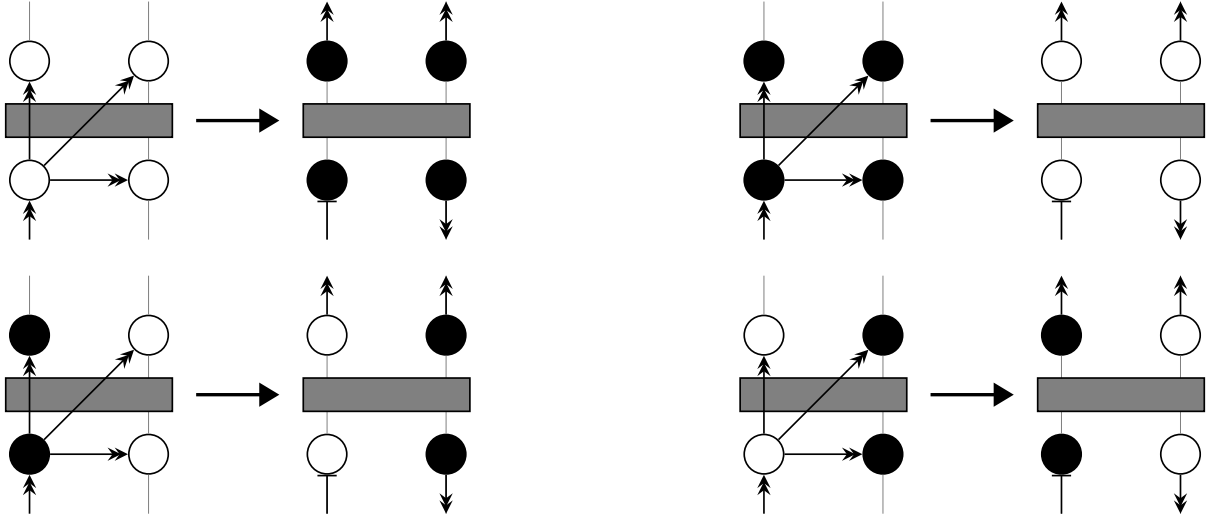


Figure 4.3: Cluster construction rules for diagonal bond operators that can be proposed by the Multibranch update scheme.

imaginary timeline until another operator is encountered. In a sense, the site operator switched type and in doing so has spawned a σ^x operator to travel along the site's timeline. If this operator is a bond operator, the worm splits and follows the three other legs of the bond. On the other hand, if a site operator is encountered the worm gets stuck at that vertex leg and that part of the cluster ceases to grow. This process continues until the entire simulation cell has been divided into distinct clusters. Since all clusters can be flipped with unit probability, we cannot actually flip all of them or else (due to \mathbb{Z}_2 symmetry) we will end up in an equivalent configuration to the current one. Instead we flip each cluster independently with probability $1/2$, meaning a random half of the clusters will be updated. As mentioned previously, the lack of a Metropolis acceptance move allows us to perform the cluster construction and flipping simultaneously. Hence, just before we begin constructing a cluster, we decide with probability $1/2$ whether the cluster will be flipped or not. If yes, the vertex legs are flipped as the cluster is being constructed, otherwise the cluster construction will still proceed, but no vertex legs are flipped. Some examples of multibranch clusters are illustrated in Fig. 4.4.

One may wonder why we bother constructing clusters that we know will not be flipped. We do this in order to label the cluster's vertex legs as already having been traversed and belonging to a cluster, otherwise we risk starting a new cluster construction process for this same cluster again, giving it another chance to be flipped with probability $1/2$. This

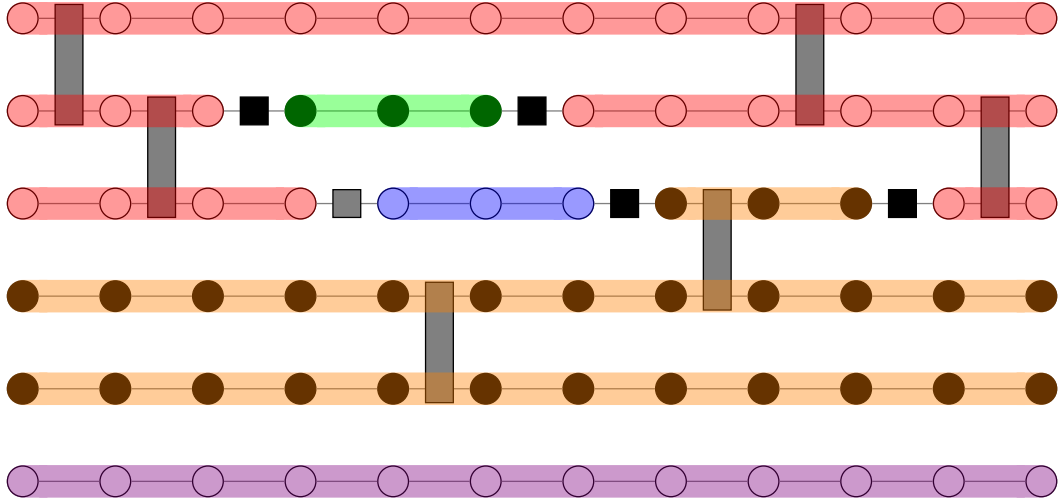


Figure 4.4: Example of an SSE simulation cell for a TFIM with 6 sites. The individual multibranch clusters have been constructed and are highlighted with different colours.

would then result in clusters effectively being flipped with probability greater than $1/2$, which would be less effective. On top of that, traversing the same cluster several times would be a waste of computational effort.

4.2.4 More General Transitions for Diagonal Bonds

If we consider instead models similar to the TFIM, but with bond operators having more non-zero diagonal matrix elements, we will need to consider different cluster construction rules for these bond operators. In general we may need to build update rules which can propose transitions several of the transitions depicted in the following diagram:

$$\begin{array}{ccc}
 \langle \circ \circ | H_b | \circ \circ \rangle & \longleftrightarrow & \langle \bullet \bullet | H_b | \bullet \bullet \rangle \\
 \updownarrow & \swarrow \nearrow & \updownarrow \\
 \langle \bullet \circ | H_b | \bullet \circ \rangle & \longleftrightarrow & \langle \circ \bullet | H_b | \circ \bullet \rangle
 \end{array}$$

The Multibranch cluster rule can only propose the transitions represented by the horizontal arrows. Next, we will introduce the *Line Update*, which will complete the above diagram by proposing transitions represented by the vertical and diagonal arrows.

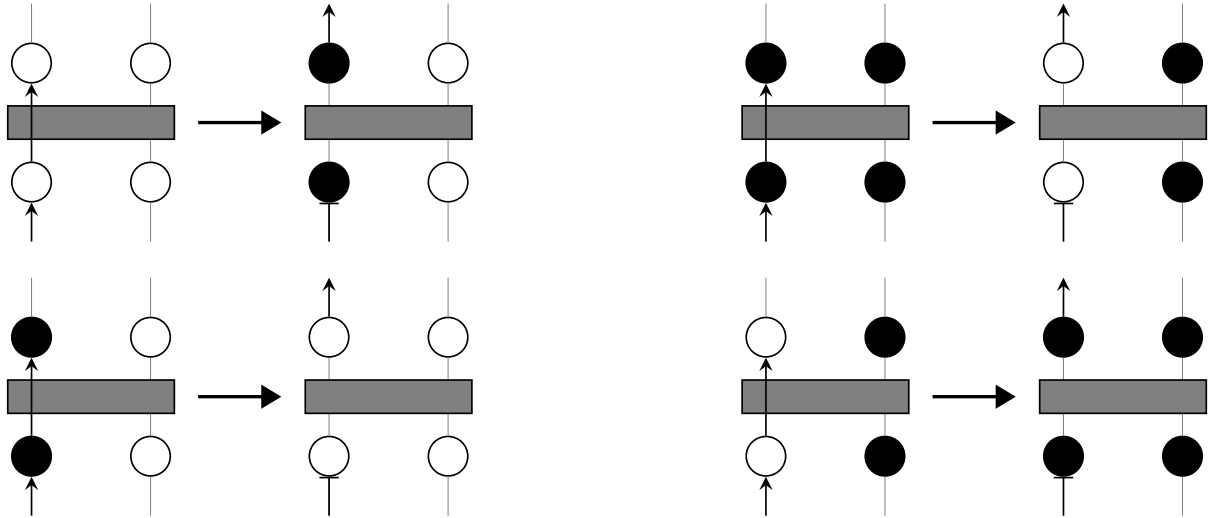


Figure 4.5: Cluster construction rules for diagonal bond operators that can be proposed by the Line update scheme.

4.2.5 The Line Update

The vertical and diagonal transitions illustrated in the above diagram require just flipping a single spin of a bond. The cluster rule for the line update is therefore quite simple: if a worm enters a bond operator we allow it to pass straight through, adding the exit vertex leg to the cluster. Examples of line update moves are illustrated in Fig. 4.5.

We can think of line clusters as multibranch clusters which have been split by site, allowing the sites to flip independently. This is depicted in Fig. 4.6, where the multibranch clusters are highlighted and the foreground dashed colours (if present) are used to distinguish the individual line clusters.

4.2.6 More General Diagonal Bonds

Of course, for a general diagonal bond operator, the various matrix elements may not have the same weight. Additionally, at a given bond operator we may be able to continue cluster construction by either a multibranch rule or a line rule. How can we decide? We could proceed in the simplest way by only using the line rule (after all, a multibranch update at a bond could potentially be accomplished by two line updates) and then using a Metropolis acceptance step to decide whether to flip the constructed cluster or not. Having to rely

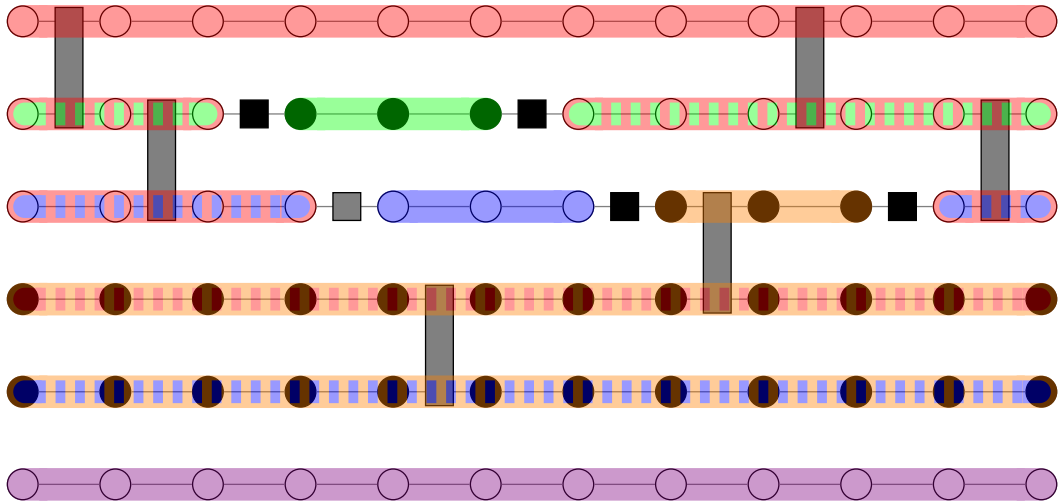


Figure 4.6: Example of an SSE simulation cell with 6 sites. The individual line clusters have been constructed and are highlighted with different colours and patterns.

on a Metropolis step is unwieldy, a more elegant cluster algorithm would probabilistically branch-out at a bond operator in such a way as to maintain a unit acceptance probability while potentially changing the configuration weight. We leave the development of such algorithms for future work.

4.2.7 High-Temperature Site Updates

Occasionally, we may find that there are no operators acting upon a site in the simulation cell. In these cases, we may randomly update the state of that site without changing the weight of the configuration. This scenario occurs more frequently in simulations performed at high temperature and in such cases these site updates are necessary to ensure efficient mixing of the Markov chain.

4.2.8 Summary

In this section we discussed cluster construction rules for performing off-diagonal updates in an SSE simulation. We focused on Hamiltonians similar to the TFIM which have a single-site σ_x field term and a completely diagonal 2-local interaction term. This led us

to the Line and Multibranch Updates, which flip one or both sites of a bond operator respectively.

4.3 Modifications for the Projector Method

As noted in Sec. 3.4.1, the configuration spaces of the thermal and projector SSE methods are extremely similar. Consequently, the Monte Carlo updates for the thermal case that we had discussed earlier in this chapter can still be used for the projector case with only minor modifications.

4.3.1 Diagonal Update Modifications

Since there are no identity operators present in the projector case, we cannot make use of the operator insertion and removal moves. Instead we must rely on the diagonal operator replacement moves in order to sample the diagonal operators in the simulation cell.

We summarize the diagonal update for the projector method in Algorithm 11. The constant N_r is again used to set the number of times the diagonal operator replacement process is attempted at each applicable time-slice. Unlike in the thermal case, N_r cannot be set to zero as this would result in no diagonal operators ever being replaced. Compare this algorithm against the full diagonal operator replacement sweep given in Algorithm 10. Here, we do not bother checking that H_{s_p} is not an identity operator as those should not be present in the projector simulation. If they are (which may be the case when the simulation has just begun), then we should attempt to replace them anyway. In the thermal case, replacing an identity operator with a diagonal one would change the expansion order, so we must specifically exclude that possibility from triggering the replacement move.

4.3.2 Off-Diagonal Update Modifications

In addition to allowing clusters to form starting from certain types of operator vertex legs, we also allow clusters to begin at each site of the trial states. For spin- $\frac{1}{2}$ systems, we essentially treat the site of the trial states as we would the H_{e_i} and H_{f_i} site operators from the TFIM, although we will need to account for the fact that the trial state may assign a different weight to the basis state upon flipping a site.

Simulating a groundstate typically requires a large simulation cell, hence the site update move of Sec. 4.2.7 is no longer necessary as it is very unlikely that a site will have no

Algorithm 11 The Diagonal Update Algorithm (Projector Variant)

Require: $\alpha_0 \in A, \underline{s} \in S^{2M}, N_r \in \mathbb{Z}$ and $N_r > 0$

```
1:  $|\alpha\rangle \leftarrow |\alpha_0\rangle$ 
2: for  $p = 1, \dots, 2M$  do
3:   if  $H_{s_p}$  is off-diagonal then
4:      $|\alpha\rangle \leftarrow H_{s_p} |\alpha\rangle$   $\triangleright |\alpha\rangle$  remains normalized, ignore any coefficients applied by  $H_{s_p}$ 
5:   else  $\triangleright H_{s_p}$  is a diagonal operator
6:      $s_p \leftarrow \text{DIAGONALREPLACEMENT}(\alpha, s_p, N_r)$ 
7:   end if
8: end for
```

operators acting on it. We may still include the move in our code just in case, but it may need to be modified in order to account for the trial state weight change.

4.4 Performing an SSE simulation

We conclude this section by summarizing how to perform an SSE simulation for both the thermal state and groundstate projector formalisms. In either case, we must begin by initializing the diagonal operator distribution $r(H_d)$, preferably using an Alias Table, as well as precomputing the matrix element ratios $\langle \alpha | H_d | \alpha \rangle / (\max_{\alpha'} \langle \alpha' | H_d | \alpha' \rangle)$.

4.4.1 Thermal SSE

When initializing the simulation, M should be set to some positive integer (not too large as we will be adjusting M to a suitable value anyway). The basis state α can be set to either a random basis state or, if we have some idea what the phase we are simulating looks like, we can set α to a configuration in that phase. Lastly, the operator list is initialized as an array of length M filled with indices referring to identity operators. A configuration consisting of a uniformly random basis state and an operator list filled only with identity operators is essentially an infinite-temperature configuration.

A single Monte Carlo update (or step) is defined as a full diagonal update sweep (and maybe also a diagonal operator replacement sweep) followed by an off-diagonal update. During the equilibration phase, we perform Monte Carlo updates until (we hope) the Markov chain has converged and is sampling from the target distribution of configurations. Since it is difficult to determine when exactly this convergence has occurred, we

typically allow the Markov chain to evolve for a pre-decided number of steps, usually a fixed fraction of the total computational budget. Simultaneously, we keep track of the maximum expansion order encountered during the equilibration n_{\max} , continually increasing M such that $M = \lceil 1.5n_{\max} \rceil$. Alternatively, for Hamiltonians whose diagonal part is much stronger than its off-diagonal part, we may set M as discussed in Sec. 4.1.9. Once the Markov chain has equilibrated, we can begin considering estimators computed using configurations generated from that point onward to be reliable.

4.4.2 Groundstate Projector SSE

Performing a projector simulation is quite similar to the thermal case, except we do not grow the expansion order dynamically. Instead we fix M to some positive integer, and initialize the operator string as an array of length $2M$ filled with identity operator indices. When performing our diagonal operator replacements we simply treat these identity operators as diagonal operators and replace them. For the identity operator, the matrix element ratios needed for the Metropolis-Hastings acceptance probability, $\langle \alpha_p | \mathbf{1} | \alpha_p \rangle / (\max_{\alpha} \langle \alpha | \mathbf{1} | \alpha \rangle)$, are simply one. As there is no way to re-introduce identity operators into the simulation cell, by the time our simulation is equilibrated there will be no identity operators remaining.

The basis states on the edges of the simulation cell, α_{ℓ} and α_r , can be set to either a random basis state or a configuration in the phase being investigated. It is important to initialize α_{ℓ} and α_r to the same basis state, as the initial operator string is an identity operator and using two distinct basis states will produce an initial configuration of zero weight, which will break the Monte Carlo simulation.

Just like in the thermal case, a single Monte Carlo update is defined as a diagonal update sweep¹⁰ followed by an off-diagonal update. The equilibration phase consists of repeated Monte Carlo updates which converge the Markov chain to the target distribution, after which point we can begin computing the estimators we desire.

¹⁰In this case the diagonal update sweep *is* the diagonal operator replacement sweep.

Chapter 5

Simulating Rydberg Atom Arrays

This chapter contains results and material from Ref. [139], along with additional material not published elsewhere.

5.1 Abstract

Arrays of Rydberg atoms are a powerful platform to realize strongly-interacting quantum many-body systems. A common Rydberg Hamiltonian is free of the sign problem, meaning that its equilibrium properties are amenable to efficient simulation by quantum Monte Carlo (QMC). In this chapter, we develop a Stochastic Series Expansion QMC algorithm for Rydberg atoms interacting on arbitrary lattices. We describe a cluster update that allows for the efficient sampling and calculation of physical observables for typical experimental parameters, and show that the algorithm can reproduce experimental results on large Rydberg arrays in one and two dimensions.

5.2 Introduction

Arrays of neutral atoms provide one of the most coherent and well-controlled experimental quantum many-body platforms available today [84, 17]. In a typical experiment, individual atoms, such as rubidium, can be trapped by laser light and driven to transition between

their groundstate and a *Rydberg* state: an atomic state with a large principal quantum number. With the use of optical tweezers, multiple such atoms, called Rydberg atoms, can be manipulated into arrays or lattices. Within an array, Rydberg atoms separated by a distance R_{ij} (typically a few micrometers or less) experience a dipole-dipole interaction. The power-law decay of this interaction depends on how pairs of Rydberg atoms are experimentally prepared [17]; it is common to prepare pairs such that a $1/R_{ij}^6$ van der Waals (VDW) interaction is the leading-order behaviour. The resulting VDW interactions penalize the simultaneous excitation of two atoms in close proximity to each other. This effect, called the Rydberg blockade [98, 129, 58], results in a strongly-interacting Hamiltonian that can be tuned with a high degree of control to realize a variety of lattices of interest to condensed matter and quantum information physicists[54, 9].

Experimental studies are proceeding rapidly, demonstrating the creation of novel phases and phase transitions in lattice Hamiltonians in one [12] and two dimensions [50]. Theoretical studies have shown that Rydberg arrays are capable of realizing extremely rich groundstate phase diagrams [125, 227, 7, 152]. Numerical techniques have played a critical role in this theoretical exploration, providing evidence of the existence of a number of compelling phenomena, including novel quantum critical points [166, 167], floating phases [161, 31], and topologically ordered spin liquid phases [168, 187]. For these reasons, we are interested in developing a quantum Monte Carlo (QMC) algorithm for the most common Rydberg Hamiltonian. Based on the Stochastic Series Expansion (SSE) framework pioneered by Sandvik [178, 170], our algorithm provides a starting point for the exploration of a wide variety of equilibrium statistical phenomena in Rydberg arrays using this powerful and efficient QMC method.

The Hamiltonian that we consider acts on the two electronic levels of each atom $i \in \{1, 2, \dots, N\}$: the groundstate $|g\rangle \equiv |0\rangle$ and a Rydberg state $|r\rangle \equiv |1\rangle$. The Hamiltonian can be written as

$$H = \frac{\Omega}{2} \sum_{i=1}^N \sigma_i^x - \delta \sum_{i=1}^N n_i + \sum_{i < j} V_{ij} n_i n_j \quad (5.1)$$

where N is the total number of Rydberg atoms. Here, the natural computational basis is the Rydberg state occupation basis, which is defined by the eigenstates of the occupation operator $n_i = |1\rangle\langle 1|_i$. The eigenequations are $n_i |0\rangle_j = 0$ for all i, j , and $n_i |1\rangle_j = \delta_{i,j} |1\rangle_j$. We define $\sigma_i^x = |0\rangle\langle 1|_i + |1\rangle\langle 0|_i$ which is an off-diagonal operator in this basis. Physically, the parameter Ω that couples to σ_i^x is the Rabi frequency which quantifies the atomic groundstate and Rydberg state energy difference, and δ is the laser detuning which acts as a longitudinal field. As mentioned previously, a pair of atoms which are both excited into

Rydberg states will experience a VDW interaction decaying as

$$V_{ij} = \Omega \left(\frac{R_b}{r_{ij}} \right)^6 \quad (5.2)$$

Here $r_{ij} = |\mathbf{x}_i - \mathbf{x}_j|/a$ is the distance between the atoms, which is controlled in the experiment by tuning the lattice spacing a . R_b is called the blockade radius, and we treat R_b/a as a free parameter in the simulations below with $a = 1$. The blockade mechanism, which penalizes simultaneous excitation of atoms within the blockade radius, results in a strongly-interacting quantum Hamiltonian that produces a plethora of rich phenomena on a wide variety of lattices accessible to current and near-term experiments.

In this chapter, we develop an algorithm for simulating Rydberg Hamiltonians based on the SSE method [178, 170, 175, 194, 176, 136]. Our algorithm follows Sandvik’s development of the spin-1/2 transverse-field Ising model [173], generalized to the Rydberg Hamiltonian Eq. 5.1 The remaining sections of this chapter are organized as follows. In Sec. 5.3, our SSE framework as it applies to the Hamiltonian in Eq. 5.1 is outlined for finite-temperature and groundstate simulations. We then show results for simulations in one and two dimensions in Sec. 5.4, and give concluding remarks in Sec. 5.6.

5.3 SSE Implementation for Rydberg atoms

The previous chapters presented some generalities of the SSE framework in both the finite-temperature and groundstate projector formalisms. To translate these formalisms into simulating the Rydberg Hamiltonian Eq. 5.1, we must define the basis states $\{|\alpha\rangle\}$, Hamiltonian operator breakup, and the update strategy. Naturally, the choice of computational basis is that of the Rydberg occupation basis: $\{|\alpha\rangle\} = \{\otimes_{i=1}^N |n_i\rangle, n_i = 0, 1\}$.

To make progress on defining the Hamiltonian sub-operators, as well as the update strategy, the specific form of the Hamiltonian must be considered. The Rydberg Hamiltonian Eq. 5.1 takes the form of a quantum Ising model with transverse and longitudinal fields. Since the transverse-field term is positive in the Rydberg occupation basis, we must devise a sign cure. As shown in Sec. 3.2.3, the unitary transformation $U = \otimes_{i=1}^N \sigma_i^z = \otimes_{i=1}^N [1 - 2n_i]$ provides the sign cure we need. The Hamiltonian Eq. 5.1 is transformed to

$$U^\dagger H U = -\frac{\Omega}{2} \sum_{i=1}^N \sigma_i^x - \delta \sum_{i=1}^N n_i + \sum_{i<j} V_{ij} n_i n_j \quad (5.3)$$

which is now free of a sign problem in the Rydberg occupation basis.

Now that the sign problem has been alleviated, we can proceed with breaking up the Hamiltonian Eq. 5.3 into a sum of local operators. Motivated by Refs. [173, 194], we define the sub-operators as

$$H_0 = \mathbf{1} \quad (5.4a)$$

$$H_{f_i} = \frac{\Omega}{2} \sigma_i^x \quad (5.4b)$$

$$H_{e_i} = \frac{\Omega}{2} \mathbf{1} \quad (5.4c)$$

$$H_{b_{ij}} = -V_{ij} n_i n_j + \delta_b (n_i + n_j) + C_{ij} \quad (5.4d)$$

Here $\delta_b = \delta / (N - 1)$ is the reduced detuning parameter since the sum $\delta \sum_i n_i$ has been moved into the sum over pairs $\sum_{i < j}$, and $C_{ij} = |\min(0, \delta_b, 2\delta_b - V_{ij})| + \varepsilon |\min(\delta_b, 2\delta_b - V_{ij})|$ is added to $H_{b_{ij}}$ so that all of its matrix elements remain non-negative, where $\varepsilon \geq 0$. Note that Eq. 5.4a is only used for finite temperature simulations. The additional ε term in the definition of C_{ij} is typically employed to aid numerics [194]. In contrast to Ref. [194], we define ε as a multiplicative constant as opposed to an additive one, since the different C_{ij} s may vary greatly in magnitude.

It is helpful to show the matrix elements of each of these local operators since these values are the foundation of importance sampling for each of the local operators. The matrix elements in the Rydberg occupation basis are

$$\langle 1 | H_{f_i} | 0 \rangle = \langle 0 | H_{f_i} | 1 \rangle = \frac{\Omega}{2} \quad (5.5a)$$

$$\langle 1 | H_{e_i} | 1 \rangle = \langle 0 | H_{e_i} | 0 \rangle = \frac{\Omega}{2} \quad (5.5b)$$

$$H_{ij}^{(1)} \equiv \langle 00 | H_{b_{ij}} | 00 \rangle = C_{ij} \quad (5.5c)$$

$$H_{ij}^{(2)} \equiv \langle 01 | H_{b_{ij}} | 01 \rangle = \delta_b + C_{ij} \quad (5.5d)$$

$$H_{ij}^{(3)} \equiv \langle 10 | H_{b_{ij}} | 10 \rangle = \delta_b + C_{ij} \quad (5.5e)$$

$$H_{ij}^{(4)} \equiv \langle 11 | H_{b_{ij}} | 11 \rangle = -V_{ij} + 2\delta_b + C_{ij} \quad (5.5f)$$

where we defined $H_{ij}^{(1,2,3,4)}$ as a shorthand for the diagonal bond operator matrix elements. Fig. 5.1 shows an example of a zero-temperature SSE simulation cell of such an operator breakup. A finite temperature simulation cell would look very similar, except translational invariance in imaginary time forces the Rydberg occupation configurations on the left and right edges to be the same.

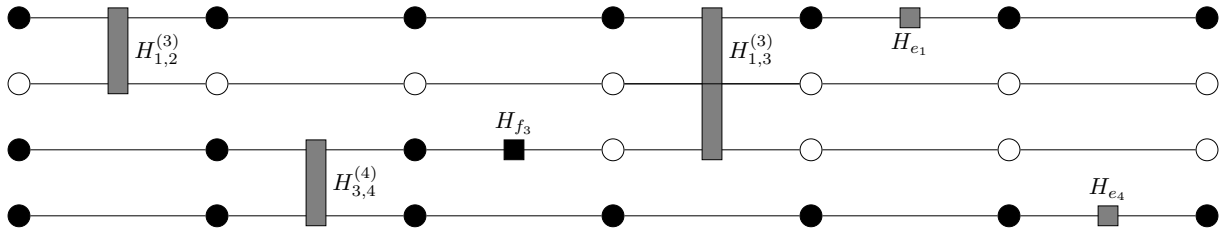


Figure 5.1: A groundstate projector SSE simulation cell example of the Rydberg Hamiltonian operator breakup in Eq. 5.4 with matrix elements given in Eq. 5.5 for $2M = 6$. Rydberg occupations labelled with a filled (unfilled) circle denote $n_i = 1(0)$. The line passing over the bond operator matrix element $H_{1,3}^{(3)}$ in the fourth time-slice indicates that the bond operator does not act on site 2. The occupation configuration on the left is $\langle \alpha_\ell |$, and on the right is $|\alpha_r \rangle$.

5.3.1 Diagonal Update

As discussed in the Chapter 4, updates to the $(d + 1)$ -dimensional configurations in the SSE QMC framework typically occur via a number of separate steps – most importantly a *diagonal update* followed by a non-local *cluster update* (often called an *off-diagonal update*).

For finite temperature simulations, we make use of the Three-Step diagonal update scheme of Sec. 4.1.6 to simulate the Rydberg Hamiltonian¹. We do not augment the update with diagonal replacement moves.

In the projector simulations, we only use diagonal replacement moves (as explained in Sec. 4.3), without imposing a maximum number of iterations (therefore taking $N_r \rightarrow \infty$). When performing the diagonal replacement, we compute the acceptance probabilities as the matrix element ratios $\langle \alpha_p | H_d | \alpha_p \rangle / (\max_\alpha \langle \alpha | H_d | \alpha \rangle)$ instead of the Metropolis-Hastings formula of Eq. 4.60².

¹We developed the Three-Step scheme because initially we were not using alias tables to sample the diagonal operator distributions. Although we did end up switching to the Alias method, we hadn't derived the Two-Step scheme until much later, long after the paper on which this chapter is based was already on the arXiv.

²There is no practical justification for this besides being able to reuse the function which computes the matrix element acceptance probability from the Three-Step scheme. We simply hadn't yet thought of using the Metropolis probability when we wrote the work that this chapter is based on.

5.3.2 Cluster Updates

The diagonal update procedures in Sec. 5.3.1 allow for new diagonal operators to replace current ones. However these updates alone are not ergodic, as they clearly do not sample operators H_{f_i} , i.e. they do not alter the world line configurations. Thus, each diagonal update in the SSE is followed by a non-local cluster update. To devise an ergodic algorithm for the Rydberg Hamiltonian, we use the cluster update devised by Sandvik called the *multibranch* cluster update, which is described in Refs. [136, 173, 93] and that we had discussed in Sec. 4.2.3. This is a highly non-local update originally designed for the SSE implementation of the transverse-field Ising model.

Recall that the operator vertices from the elementary bond operator $H_{b_{ij}}$ comprise of four *legs* (two Rydberg occupation states from the ket and bra), while vertices from site operators H_{e_i} and H_{f_i} have two legs (one Rydberg occupation state from the ket and bra). The operators H_0 in the finite-temperature case are ignored. Multibranch clusters are formed by beginning at one of the legs of a random site operator vertex and traversing away from this operator in the imaginary time direction. If a bond vertex is encountered, all four vertex legs are added to the cluster and the cluster continues to grow by branching out of all three remaining exit legs. If a site vertex is encountered, the cluster terminates at that newly encountered leg. For finite-temperature simulations, if the edge of the simulation is reached by the cluster, it must loop around to the opposite edge in order to respect periodic boundary conditions in imaginary time. If the edge of the simulation is reached in a groundstate projector simulation, the cluster terminates at the boundary edge.

Fig. 5.2 shows an example of a groundstate projector SSE simulation cell wherein a multibranch cluster is pictured by the green region. Updating clusters consists of flipping all legs (Rydberg occupations) and operator vertex types that are within the cluster in a corresponding fashion. Since cluster weights may change when flipping, detailed balance must be satisfied by flipping clusters with the Metropolis probability

$$P_{\text{flip}} = \min \left(1, \frac{\mathcal{W}'}{\mathcal{W}} \right) \quad (5.6)$$

where \mathcal{W} is weight of the cluster defined as the product of operator vertices s_p (matrix elements with values $W(s_p) = \langle \alpha_{p-1} | H_{s_p} | \alpha_p \rangle$) found in Eq. 5.5) belonging to the cluster \mathcal{C} :

$$\mathcal{W} = \prod_{s_p \in \mathcal{C}} W(s_p) \quad (5.7)$$

\mathcal{W}' denotes the weight of cluster \mathcal{C} from flipping it, therefore changing the vertex types $s_p \in \mathcal{C}$. For instance, the upper pane of Fig. 5.2 shows a multibranch cluster (green) that

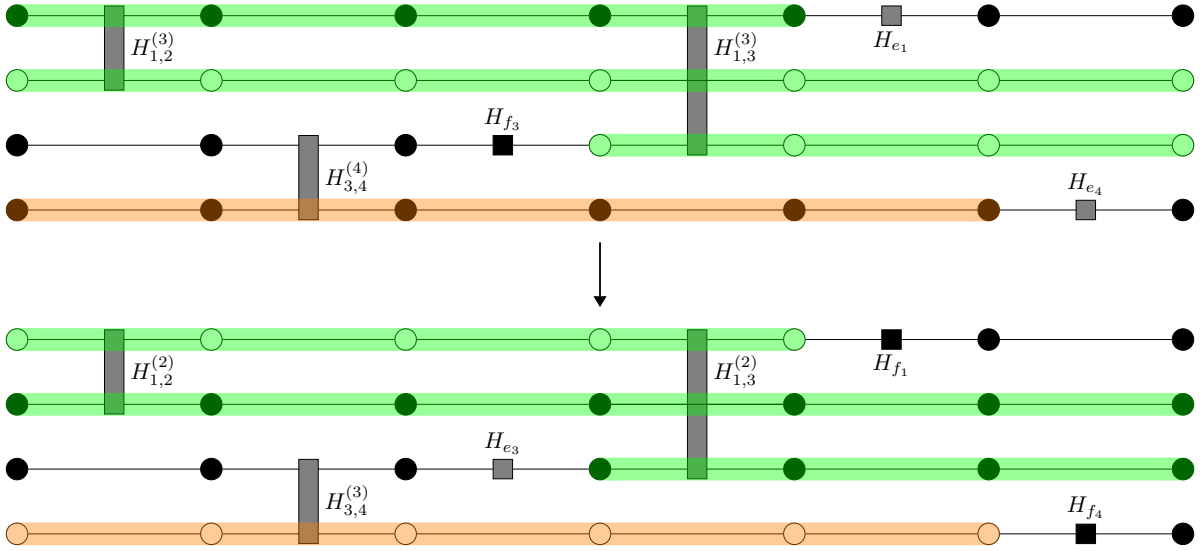


Figure 5.2: A groundstate projector SSE simulation cell example of the Rydberg Hamiltonian operator breakup in Eq. 5.4 with matrix elements given in Eq. 5.5 for $2M = 6$. Rydberg occupations labelled with a filled (unfilled) circle denote $n_i = 1(0)$, with $\langle\alpha_\ell|$ on the left edge and $|\alpha_r\rangle$ on the right edge. In the upper simulation cell, we show some examples of the multibranch (green) and line (orange) clusters (note: these are not all of the possible clusters that may be formed in the depicted simulation cell). These clusters are probabilistically flipped according Eq. 5.6. If each cluster highlighted here is flipped, the lower simulation cell is what results.

has a weight $\mathcal{W} \propto H_{1,2}^{(3)} \times H_{1,3}^{(3)} \times \langle 1|H_{e_1}|1\rangle \times \langle 1|H_{f_3}|0\rangle$. When flipped (lower pane), it has a weight $\mathcal{W}' \propto H_{1,2}^{(2)} \times H_{1,3}^{(2)} \times \langle 0|H_{f_1}|1\rangle \times \langle 1|H_{e_3}|1\rangle$. Note that if the simulation cell's outer edge states are initialized to $\bigotimes_{i=1}^N \frac{1}{\sqrt{2}}(|0\rangle_i + |1\rangle_i)$ (i.e. simulation cell edge states are randomly initialized), weight changes do not manifest from flipping Rydberg occupations at the simulation cell edges. As we are now taking the weight change into account, we may need to visit every leg in a cluster twice: once to accumulate the weights and then again to flip each of these legs if the update was accepted.

The multibranch cluster works exceptionally well for the transverse-field Ising model partially owing to the fact that this update results in efficient, highly non-local configuration changes. In particular, multibranch clusters for the TFIM are formed deterministically and do not accrue a weight change upon flipping, allowing the update to be accepted with

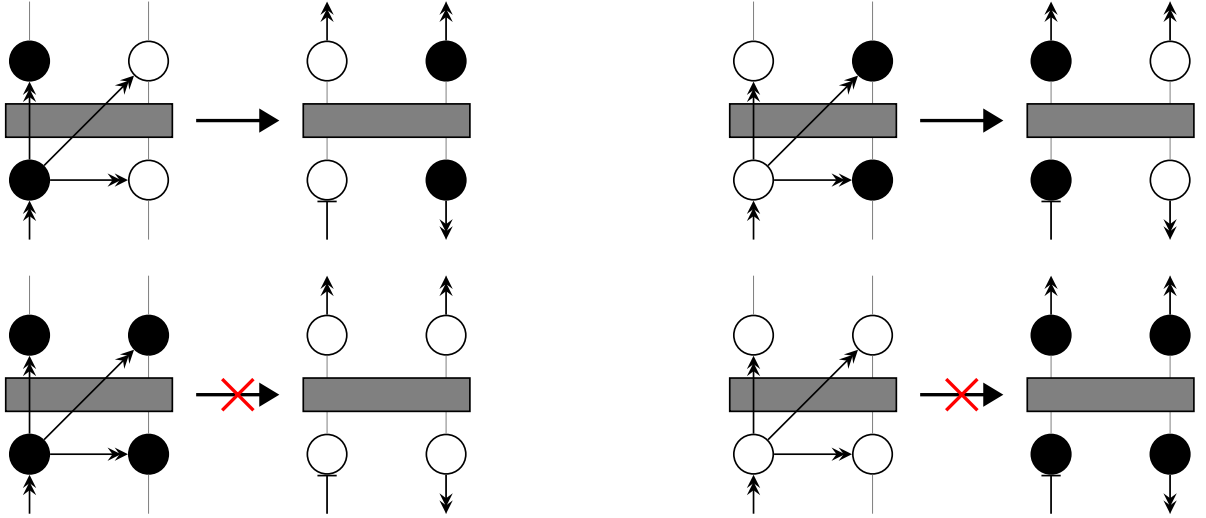


Figure 5.3: Cluster construction rules for the diagonal Rydberg bond operator that can be proposed by the Multibranch update scheme. Moves which will result in a matrix element with zero weight (or weight close to zero) are forbidden (or very unlikely) for the Rydberg Hamiltonian and are therefore marked with a red \times .

probability $1/2$ [173]. In the case of the Rydberg Hamiltonian Eq. 5.1, the presence of the laser detuning δ and the nature of the interactions $n_i n_j$ require that the ratio of weights in Eq. 5.6 must be considered for every update, though the clusters are still constructed deterministically.

Intuitively, we expect the multibranch update to be inefficient for many R_b and δ combinations, as any cluster containing either the matrix element $\langle 00|H_{b_{ij}}|00\rangle$ or $\langle 11|H_{b_{ij}}|11\rangle$ will be frozen since the flipped counterpart has weight zero (or, in the case of a small non-zero ε , a weight close to zero). Additionally, we expect the long-range interactions to increase the number of frozen clusters as each cluster will have a higher likelihood of containing a $\langle 00|H_{b_{ij}}|00\rangle$ or $\langle 11|H_{b_{ij}}|11\rangle$ matrix element. Thus we need an update which instead of proposing moves $\langle 00|H_{b_{ij}}|00\rangle \leftrightarrow \langle 11|H_{b_{ij}}|11\rangle$, proposes moves such as $\langle 00|H_{b_{ij}}|00\rangle \leftrightarrow \langle 01|H_{b_{ij}}|01\rangle$, $\langle 00|H_{b_{ij}}|00\rangle \leftrightarrow \langle 10|H_{b_{ij}}|10\rangle$ and so on, flipping only a single spin of a bond. As discussed in Sec. 4.2.5, the Line update does exactly what we need.

From an alternative combinatorial perspective, a spatially non-local cluster like that in Fig. 5.2 touches K physical sites and thus has (in general) 2^K states. Due to the $\sigma_z \rightarrow -\sigma_z$ symmetry of the transverse-field Ising model, the bond operator (after adding the constant energy shift) has only two non-zero matrix elements and the cluster therefore

only has two possible configurations with non-zero weights which the multibranch update alternates between. The multibranch update is thus optimal for this case. However, in the Rydberg case most bonds have more than two non-zero weights. The multibranch update is therefore no longer sufficient to explore all $2^{O(K)}$ configurations of each cluster.

The line cluster update is a local-in-space and non-local-in-imaginary-time cluster inspired by Ref. [148] (similar updates have also been proposed in Refs. [13, 48]). Like the multibranch clusters, the line clusters also terminate on site operators and are thus deterministically constructed. However, if a bond operator is encountered, only the adjacent leg in imaginary time is added to the cluster and it continues to propagate in the imaginary time direction until reaching a site operator. This cluster is then flipped with the Metropolis probability as in Eq. 5.6. For instance, the orange line cluster in Fig. 5.2 has a weight $\mathcal{W} \propto H_{3,4}^{(4)} \times \langle 1|H_{e_4}|1\rangle$. When flipped, it has a weight $\mathcal{W}' \propto H_{3,4}^{(3)} \times \langle 0|H_{e_4}|1\rangle$.

In our simulations, we define a Monte Carlo step as a diagonal update followed by an off-diagonal update in which all possible clusters are constructed and flipped independently according to the Metropolis condition. The specific type of off-diagonal update we use (line or multibranch) is selected before the cluster update begins. We generate pseudo-random numbers using the Xoroshiro128+ algorithm [14].

5.3.3 Groundstate Energy Estimator

Given the normalization in Eq. 3.75, we require a compact expression for the groundstate energy,

$$E_0 = \langle H \rangle = \frac{1}{Z} \langle \alpha_\ell | (-H)^M H (-H)^M | \alpha_r \rangle \quad (5.8)$$

in terms of properties which can be computed from the projector SSE simulation cell. It can be shown [139, 41] that

$$\frac{E_0}{N} = \frac{\langle H \rangle}{N} = -\frac{\Omega}{2} \frac{2M + 1}{\langle n_e + 1 \rangle} \quad (5.9)$$

where n_e is the number of H_e operators present in the simulation cell.

5.4 Results

Numerous recent experimental works have showcased the future potential of Rydberg atoms as a platform for quantum computation and for realizing a host of quantum many-body

phenomena. Motivated in particular by the experiments of Bernien *et al.* [12] and Ebadi *et al.* [50], we present results that showcase the line update algorithm for a 51 atom one-dimensional (1D) chain and a 16×16 square array of Rydberg atoms, both with open boundary conditions. All results reported in this section take $\Omega = 1$ and $R_b = 1.2$.

5.4.1 51 atom 1D chain

At finite temperature, an SSE simulation is allowed to grow in imaginary time during the equilibration phase. Therefore, a suitably-converged simulation cell size is automatically calculated during equilibration (see Sec. 4.4). Fig. 5.4 shows the estimated energy density, calculated using Eq. 3.56, and the corresponding simulation cell size M for various δ/Ω values. The line update was chosen as the cluster update for each simulation. As expected, for higher (lower) temperatures we observe that the automatically-calculated simulation cell size is smaller (larger).

Sec. 5.3.2 outlined the two cluster updates we have implemented for our SSE QMC algorithm. The question of which cluster update is best to employ will undoubtedly depend on R_b , δ/Ω , and system size. However, Monte Carlo observables like the finite- (Eq. 3.56) or zero-temperature (Eq. 5.9) energies that strictly depend on SSE simulation-cell parameters and not the basis states $\{|\alpha\rangle\}$ are extremely robust to the choice of cluster update; the mechanics of the diagonal update are far more important since the diagonal updates do not modify $\{|\alpha\rangle\}$.

At zero temperature we do not automatically grow the simulation cell size / projector length $2M$ — typically, it is manually converged. For our example value of the blockade radius, $R_b = 1.2$, we consider a value of $\delta/\Omega = 1.1$ which is near a quantum phase transition (QPT) in 1D [12]. Fig. 5.5 shows the estimated groundstate energy, calculated using Eq. 5.9, versus projector lengths $2M$. The line update was chosen as the cluster update for each simulation. From this, a suitably-converged projector length $2M$ can be interpolated. We observe that $2M = 2.4 \times 10^4$ gives energies converged to well within error bars of those with larger projector lengths. We use this projector length henceforth for the 51 Rydberg atom results.

Fig. 5.6 shows the estimated absolute value of the staggered magnetization,

$$|M_s| = \left| \sum_{j=1}^N (-1)^j \left(n_j - \frac{1}{2} \right) \right| \quad (5.10)$$

where $n_j = 0, 1$ is the Rydberg state occupation at site j , which clearly resolves the QPT. The domain wall density (DWD) is another indicator of the onset of the QPT [12].

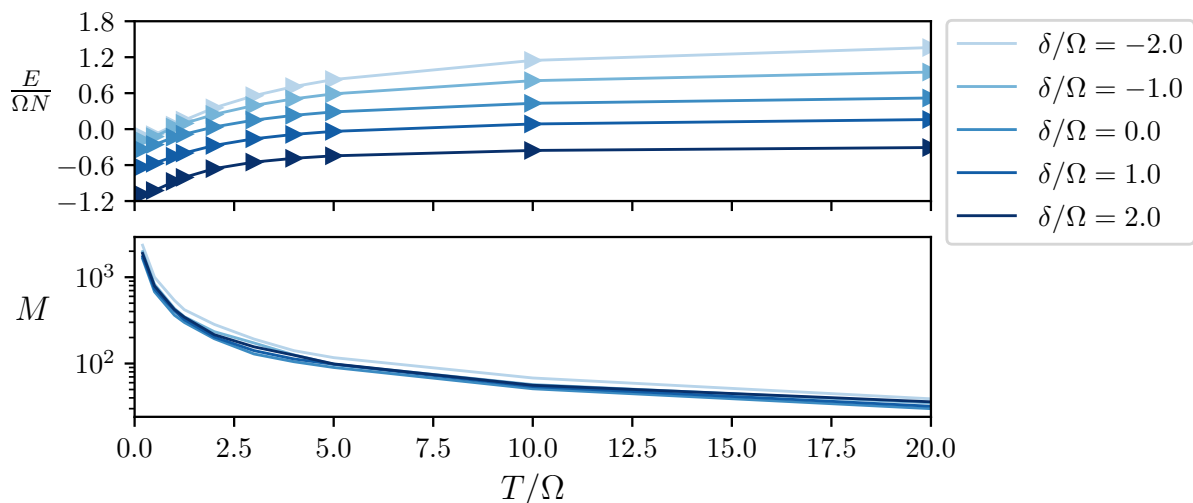


Figure 5.4: The estimated energy density $E/\Omega N$ and the equilibrated simulation cell size M for an $N = 51$ 1D chain of Rydberg atoms with $R_b = 1.2$ as a function of temperature T/Ω and δ/Ω . Error bars in the energy density are smaller than the markers. Each data point represents an independent simulation (line cluster updates only—see Sec. 5.3.2) wherein 10^7 successive measurements were taken and placed into 500 bins. These 500 binned measurements were then used to calculate statistics.

Domain walls are defined as neighbouring Rydberg atoms in the same state or a Rydberg atom not in a Rydberg state on the open boundaries. The bottom pane of Fig. 5.6 shows the simulated DWD versus δ/Ω . The behaviour of $|M_s|$ and the DWD across the range of δ/Ω values matches that from the experimental results in Figure 5 from Bernien *et al.* [12] extremely well.

Interestingly, depending on the cluster update type that is employed throughout these simulations, we observe drastically different autocorrelation times [5, 217] for $|M_s|$. The right-hand pane of Fig. 5.6 shows the autocorrelation times for three different update procedures: performing line updates exclusively, performing a line update or a multibranch update with equal probabilities at every Monte Carlo step, or performing multibranch updates exclusively. Each autocorrelation time curve shows a peak near the QPT, but the line update offers orders-of-magnitude better autocorrelation times compared to multibranch updates. Whether this critical slowing can be ameliorated further is a problem we leave for future work. Additionally, we see that introducing a non-zero ε as mentioned in

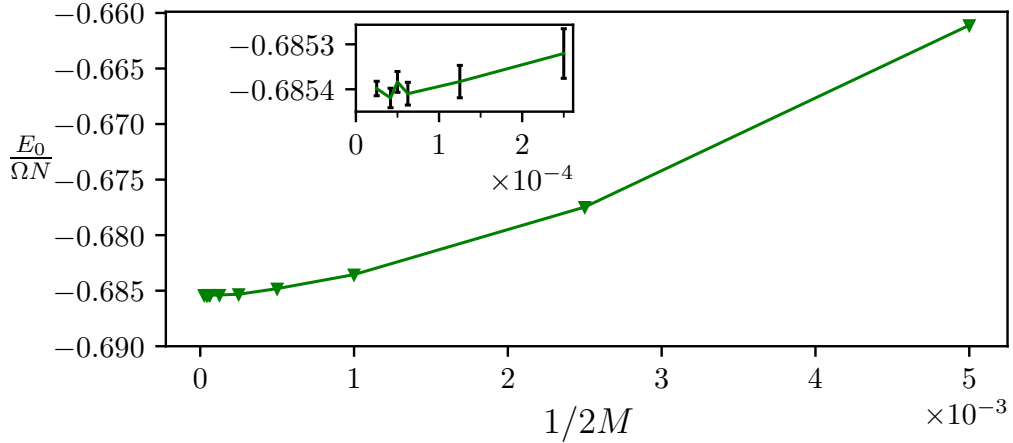


Figure 5.5: The estimated energy density $E_0/\Omega N$ (Eq. 5.9) vs the simulation cell size $2M$ for an $N = 51$ 1D chain of Rydberg atoms with $R_b = 1.2$ and $\delta/\Omega = 1.1$ as a function of the inverse projector length $1/2M$. Each data point represents an independent simulation (line cluster updates only—see Sec. 5.3.2) wherein 10^7 successive measurements were taken and placed into 500 bins. These 500 binned measurements were then used to calculate statistics via a standard jackknife routine. In the main plot, error bars are smaller than the plot markers.

Sec. 5.3 has little effect on the actual performance of the algorithm. Although this may differ depending on R_b , δ/Ω , and system size, these results illustrate how choice of update (or combination of the updates) is crucial to simulation efficiency.

5.4.2 256 atom 2D array

Next, we performed groundstate simulations of a 16×16 square lattice Rydberg array with open boundary conditions. We set $M = 10^5$, which we found gave sufficient energy convergence during preliminary runs. Independent simulations were performed over the range $\delta/\Omega \in [0, 1.75]$ in increments of 0.05, each performing 10^5 equilibration steps followed by 10^6 measurements.

For the value of $R_b = 1.2$, Samajdar *et al.* reported the existence of a QPT from a disordered to checkerboard phase in two spatial dimensions on a square lattice [167]. The top left pane of Fig. 5.7 shows the absolute value of the staggered magnetization

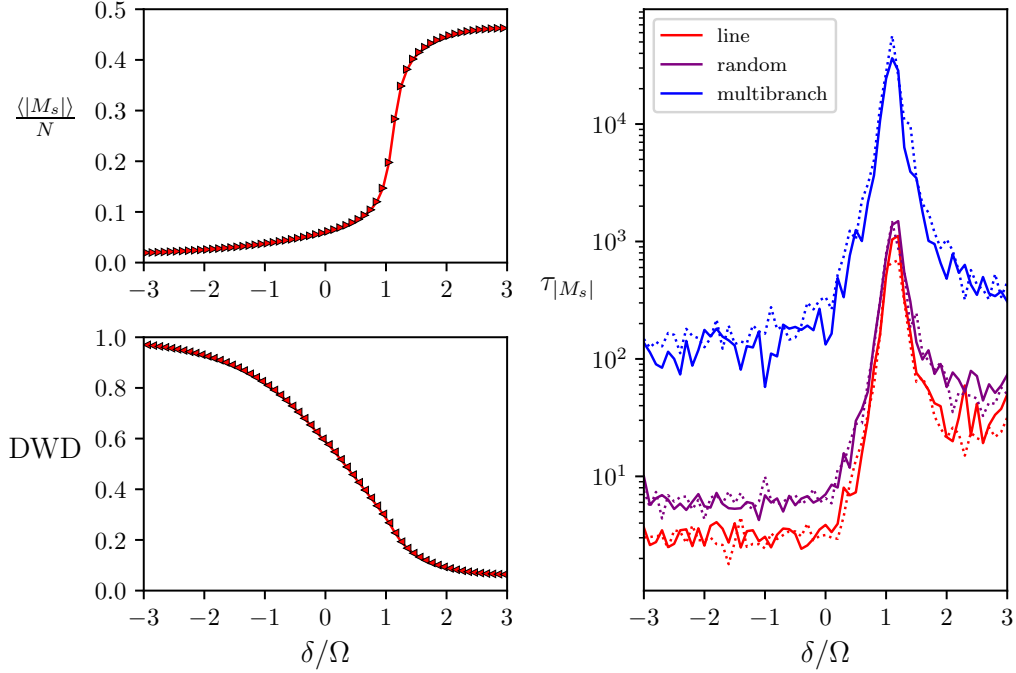


Figure 5.6: Absolute value of the staggered magnetization density $\langle |M_s| \rangle / N$ (top left), and the corresponding staggered magnetization autocorrelation times $\tau_{|M_s|}$ (right pane) for three different update procedures—line updates exclusively (red), randomly choosing line or multibranch updates at every MC step (purple), or multibranch updates exclusively (blue)—and different ϵ values: $\epsilon = 0$ (solid lines), or $\epsilon = 0.1$ (dotted lines). The estimated DWD for an $N = 51$ 1D chain of Rydberg atoms with $R_b = 1.2$ as a function of δ / Ω (bottom left). Each data point represents an independent SSE QMC simulation wherein 10^7 successive measurements were taken and placed into 500 bins. These 500 binned measurements were then used to calculate statistics. Error bars for the plots on the left are smaller than the markers. A logarithmic binning analysis was performed on the full dataset to estimate the autocorrelation times.

density where we observe this transition, and the top right pane shows the corresponding autocorrelation times [5, 217] for exclusive multibranch updates, exclusive line updates, and randomly choosing between line and multibranch updates at every MC step. The orders-of-magnitude improvement in autocorrelation time when using line updates exclusively is apparent again for this system. Not only this, but the autocorrelation time for the

multibranch curve does not show a peak near the transition into the checkerboard phase. This is most likely attributed to the fact that the staggered magnetization error bar sizes and non-monotonicity of the multibranch (blue) curve indicate non-ergodic behaviour.

Motivated by reported experimental results, Fig. 5.7 also shows the Rydberg excitation $\langle \hat{n} \rangle$, which shows good agreement in qualitative behaviour with the experimental results in Extended Data Figure 7 from Ebadi *et al.* [50], though this experimental data was extracted at a different value of R_b . Lastly, the autocorrelation time of the Rydberg excitation density is shown in the bottom right of Fig. 5.7, which again demonstrates that the line update's performance drastically exceeds that of the multibranch update in this parameter range.

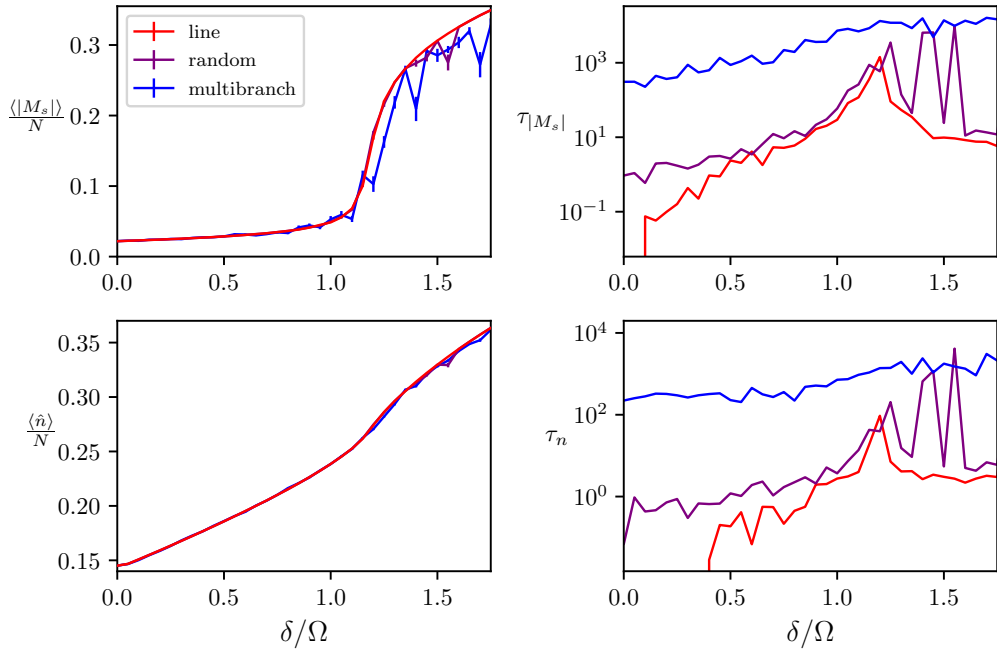


Figure 5.7: Absolute value of the staggered magnetization density $\langle |M_s| \rangle / N$ (top left), and the corresponding autocorrelation times $\tau_{|M_s|}$ (top right) for three different update procedures. The Rydberg excitation density and its autocorrelation time are plotted in the bottom row. Each data point represents an independent SSE QMC simulation of a 16×16 Rydberg array with $R_b = 1.2$, wherein 10^6 successive measurements were taken and a logarithmic binning analysis was performed to estimate the autocorrelation times.

In order to further pin down exactly *why* the line update is so much more efficient than

the multibranch update, we construct frequency histograms of both the counts and sizes of accepted and rejected clusters near the disordered-to-checkerboard phase transition. The cluster count histograms are constructed by counting the number of clusters in the simulation cell during each Monte Carlo step. Cluster size histograms are constructed similarly. In Fig. 5.8 we plot the relative frequencies of clusters against their sizes. First we must note that only certain cluster sizes are valid for each update; cluster sizes with frequency zero were not plotted. We see that the line update constructs clusters of more diverse sizes, with a gradual decay in frequency of larger clusters. On the other hand, the distribution of clusters constructed by the multibranch scheme is bimodal, with the dominant mode showing a very rapid decay with cluster size, followed by a smaller mode of very large rejected clusters. This indicates that the multibranch update tends to create a few very large clusters which will then rarely be flipped. Noting the distribution of rejected line clusters is much wider than that of the accepted clusters, it is clear that while the line update does build clusters of a greater variety of sizes, the larger clusters will not be flipped. We leave the possibility of a scheme which can flip larger clusters for future work.

In Fig. 5.9 we plot histograms of the number of clusters constructed in a single Monte Carlo step, as well as the mean cluster size, both as functions of the interaction truncation. Truncation was performed by eliminating interactions beyond the k^{th} nearest-neighbour, where $k = \infty$ corresponds to no truncation. The cluster count histograms at each truncation approximately follow a Gaussian distribution, except for the rejected multibranch clusters which show a slight skew. We see that the multibranch update has a tendency to accept relatively few clusters in each Monte Carlo step while only rejecting a handful of clusters. Additionally, the mean cluster size shows that the accepted clusters constructed by the multibranch update are on average quite small (predominantly consisting of trivial clusters containing only two site operators) while the rejected clusters tend to grow quickly with interaction distance. In the case of the line update, increasing the truncation distance results in growth of both the accepted and rejected clusters, though the rejected clusters grow faster³. Combined with the data from Fig. 5.8, we can conclude that the multibranch update constructs a small number of very large clusters which will almost always be rejected. By breaking the clusters into smaller spatially-local slices, the line update is able to propose many more successful updates to the simulation cell.

³One may ask why the line update is sensitive to the interaction truncation in the first place as it is a spatially local update. While this is true, we must also keep in mind that more bond operators means there will on average be more bonds between two site operators in the SSE simulation cell, causing the temporal extent of the line clusters to grow.

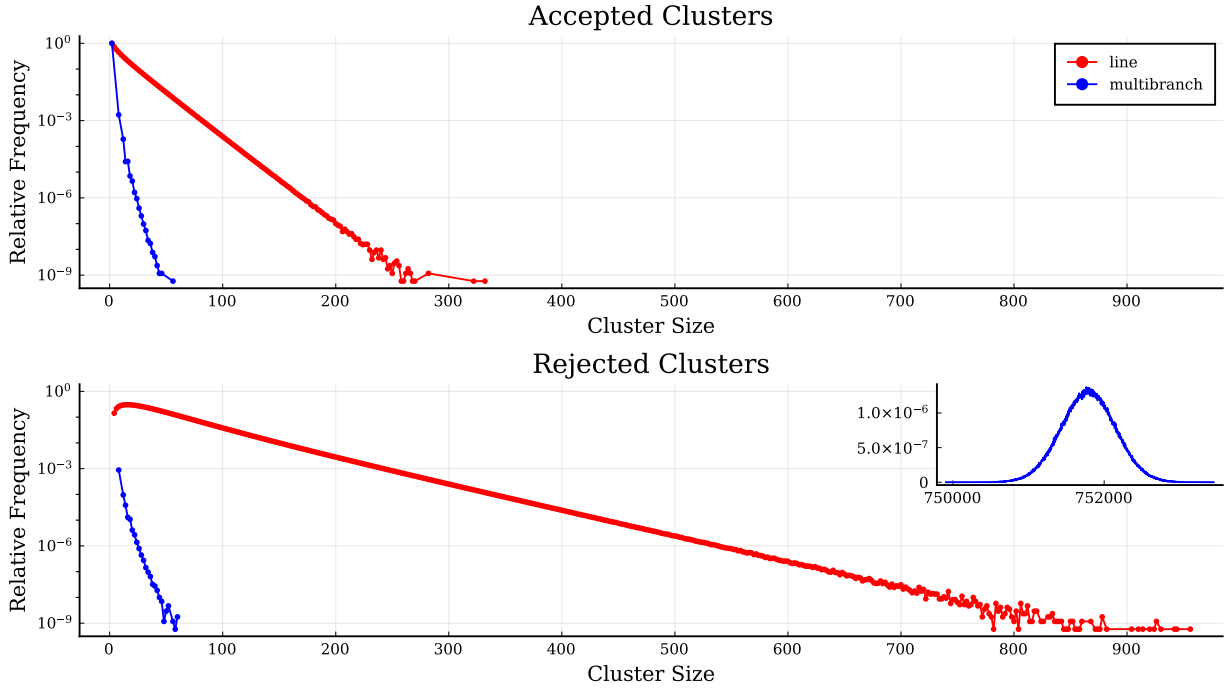


Figure 5.8: Cluster size histograms for the 2D Rydberg array at $R_b = 1.2$, $\delta/\Omega = 1.1$ for the two update types on a semi-log plot. Note that only certain cluster sizes are valid for each update; cluster sizes with frequency zero are not shown. Inset shows the second mode of the rejected cluster size histogram of the multibranch update on a linear plot.

5.5 Future Directions

The cluster updates presented in this chapter, though effective, can clearly be improved upon. In the following we will propose a new cluster construction procedure which we believe will be more efficient, but whose practical implementation we were unable to complete.

As noted in Sec. 5.4.2, the Line update produces clusters with a wider variety of sizes than the Multibranch update. Consequently, the Line update is able to more effectively modify the simulation cell. However, the accepted clusters are still smaller than the rejected clusters on average. We therefore need an update scheme that can propose clusters that are larger than the Line update is capable of, but that can still be flipped easily.

Consider the case where the detuning is zero, the diagonal bond operator now has all

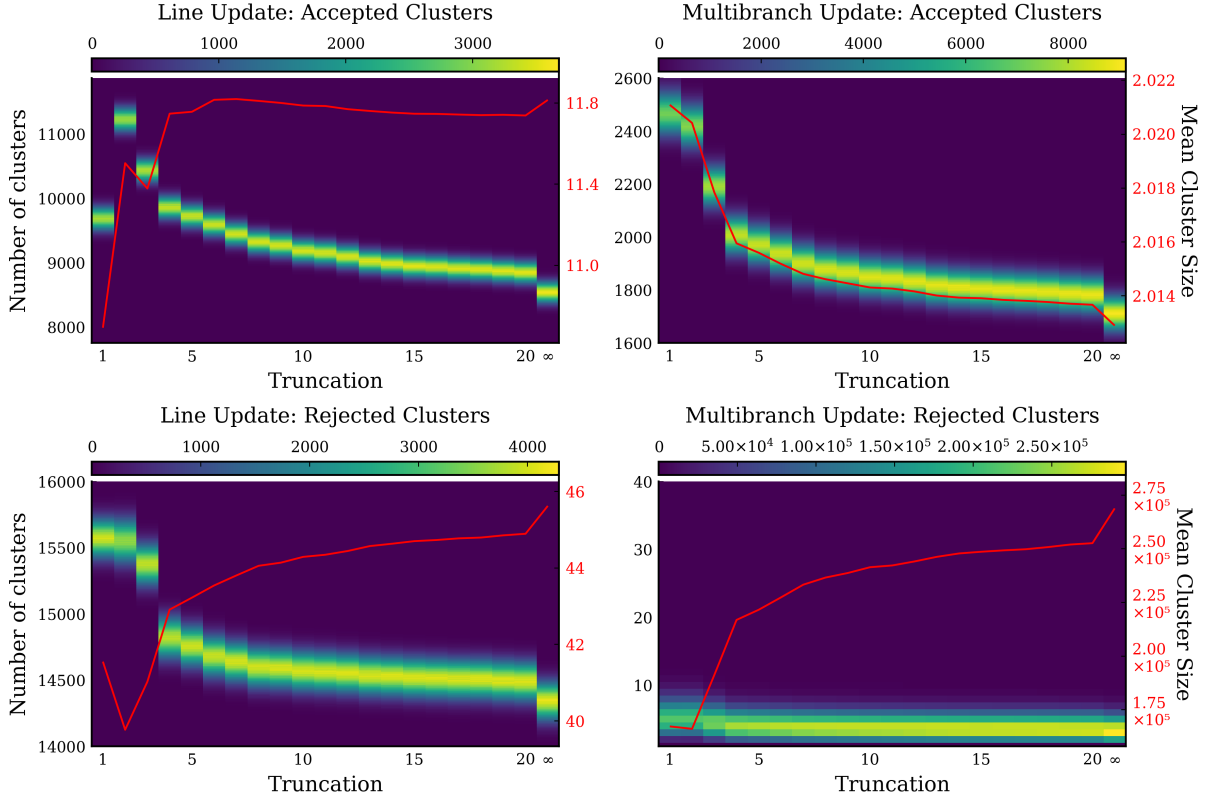


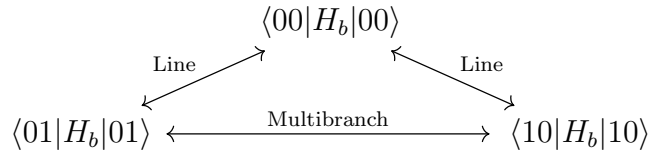
Figure 5.9: Frequency heatmap of cluster counts vs interaction truncation for the 2D Rydberg array at $R_b = 1.2$, $\delta/\Omega = 1.1$. The red line tracks the mean cluster size vs interaction truncation.

equal non-zero matrix elements:

$$V_{ij}\mathbf{1} - V_{ij}n_i n_j = \begin{bmatrix} V_{ij} & & & \\ & V_{ij} & & \\ & & V_{ij} & \\ & & & 0 \end{bmatrix}$$

which means the transitions between all three diagonal matrix elements will not change the weight of the configuration. Additionally, the diagonal update will only reject this operator if placed on two sites containing Rydberg states, otherwise the matrix element ratio is unity. An off-diagonal update which takes advantage of this should be explored and the detuning should therefore not be included in the bond term.

If the updating worm enters the bond operator matrix elements $\langle 01|H_b|01\rangle$ or $\langle 10|H_b|10\rangle$ on the site in the state $|0\rangle$, the cluster construction rule applied should be the multibranch rule, thereby giving the transitions $\langle 01|H_b|01\rangle \leftrightarrow \langle 10|H_b|10\rangle$. If, on the other hand the worm enters a 1 site on either of those bond operator matrix elements, or a 0 site of the matrix element $\langle 00|H_b|00\rangle$, then the line construction rule should be used. We obviously cannot use the multibranch construction rule on the matrix element $\langle 00|H_b|00\rangle$ as this would produce the matrix element $\langle 11|H_b|11\rangle$ which is zero. Hence, overall we will get the transition diagram:



The problem which remains is to properly handle the diagonal detuning operators δn_i . The transverse field already requires the presence of a diagonal single site operator $(\Omega/2)\mathbf{1}$, and adding the two together gives a matrix similar to

$$\begin{bmatrix} \frac{\Omega}{2} & 0 \\ 0 & \delta + \frac{\Omega}{2} \end{bmatrix}$$

Inspecting this matrix, we observe that we will have a chance of rejection when the diagonal update tries to insert this operator, which we did not have before for site operators. When building clusters, we will now need to devise new rules for this site operator. If a worm encounters an off-diagonal site operator $\langle 0|H_f|1\rangle$ from the 0 side, it can flip the 0 to a 1, converting the matrix element to $\langle 1|H_e|1\rangle$ *increasing* the weight from $\Omega/2$ to $\delta + \Omega/2$. Coming in from the 1 side, letting the worm get stuck would convert the operator matrix element to $\langle 0|H_1|0\rangle$, resulting in no weight change.

If instead we begin with the matrix element $\langle 0|H_e|0\rangle$, we can convert this to either matrix element of the off-diagonal operator H_f since there will be no weight change. The issue arises if we encounter $\langle 1|H_e|1\rangle$, in this case, trying to convert the matrix element, by letting the worm “stick” to one side, would decrease the weight from $\delta + \Omega/2$ to $\Omega/2$. Hence, in order to maintain detailed balance we must allow some chance of rejection. A rejection would mean the worm must turn around and return along the path it came from; we say that the worm “bounced”. Setting the probabilities for each site operator transition process will likely require solving the directed loop equations [194, 179]. It should be noted that the directed loop approach was analyzed in Ref. [41] for the case of the Rydberg bond operators with detunings absorbed into them as in Eq. 5.4d, but that approach was found to be unfruitful.

Alternatively, we could propose to convert the matrix element regardless (i.e. deterministically) and then consult a Metropolis condition to flip the entire cluster once construction has been completed.

We should mention that δ can be negative, in which case we simply add the shift $|\delta|\mathbf{1}$ to the diagonal site operator, and switch the roles of the 0 and 1 states in the above discussion.

Although the update described above could potentially improve the efficiency of SSE simulations of Rydberg Hamiltonians for R_b values near one, it will likely be necessary to depart from the SSE formalism when performing simulations at larger R_b values. Recall that the time and space complexity of SSE in the thermal case scales as $O(\beta \langle H \rangle)$, where $\langle H \rangle$ is the energy given by the *shifted* Hamiltonian. The matrix elements of the Rydberg interactions scale as $(R_b/r_{ij})^6$. In cases where the interactions are dominant, this means the time and space complexity of SSE actually scales as $O(\beta R_b^6)$! It may therefore be necessary to switch to a perturbative scheme such as the Continuous Imaginary Time flavour of QMC algorithms [73, 148, 100, 65] when we wish to probe the behaviour of Rydberg systems with larger blockade radii. When constructing such a simulation, we would take the diagonal part of the Rydberg Hamiltonian (that is, the interactions and detunings) as the “base” Hamiltonian, H_0 , and treat the off-diagonal⁴ field term as a perturbation, V . Writing the perturbation operators in the imaginary-time interaction representation, $V(\tau) = e^{\tau H_0} V e^{-\tau H_0}$, the partition function is then written as [176]:

$$Z = \sum_{n=0}^{\infty} (-1)^n \int_0^{\beta} d\tau_1 \int_0^{\tau_1} d\tau_2 \int_0^{\tau_2} d\tau_3 \cdots \int_0^{\tau_{n-1}} d\tau_n \text{Tr} \{ e^{-\beta H_0} V(\tau_n) V(\tau_{n-1}) \cdots V(\tau_1) \} \quad (5.11)$$

and we expand the trace in the basis in which H_0 is diagonal, allowing us to easily manipulate the weight contributions from the exponentials. We then expand out the sum over individual operators within the perturbing Hamiltonian V similarly to how we constructed the operator list in SSE. The simulation would then proceed by sampling the basis state, the expansion order, the strings of perturbing operators, as well as the different imaginary-times τ_i that each perturbing operator $V(\tau_i)$ has evolved for. This flavour of QMC has the advantage that the temporal extent of the simulation cell scales as $O(\beta \langle V \rangle)$, that is, only the energy contribution from the perturbation affects the size of the simulation cell. Although the space complexity of the operator list only scales with $O(\beta \langle V \rangle)$, the cluster update will still need to account for the interaction term, resulting in a time and space scaling of $O(\beta R_b^6)$ once again. However, these cluster updates are much more similar to cluster

⁴We may include some additional diagonal terms in V if required by the update scheme.

updates in classical Monte Carlo, which means we could potentially make use of much of the work done in the development of cluster updates for frustrated classical systems.

5.6 Conclusions

We have introduced a QMC algorithm within the SSE formalism that can efficiently simulate finite-temperature and groundstate properties of Rydberg atom arrays in arbitrary dimensions. We have outlined the algorithm in both the finite-temperature and groundstate projector formalism, emphasizing the theoretical frameworks as well as details required for practical implementation. In particular, we provide details of the Hamiltonian breakup into local operators, and introduce a modification of Sandvik’s multibranch cluster update [173], suitable for Rydberg Hamiltonians with strong detuning. We also present an efficient estimator for the groundstate energy, which is valid for any SSE algorithm containing an elementary operator that is a scalar multiple of the identity (including that for the transverse-field Ising model [173]).

In order to characterize the behaviour of the SSE algorithm, we study its efficiency in simulating recent results from experimental Rydberg arrays in one and two dimensions. In addition to convergence properties, we focus on Monte Carlo autocorrelation times for estimators of physical observables in the vicinity of quantum phase transitions which occur as a function of the detuning parameter. We compare in particular the original multibranch cluster update to a modified *line* update which is local in space but non-local in imaginary time. For some detunings near criticality, this new line update shows improvements of at least an order of magnitude in the autocorrelation time for some observables.

Our results show that this simple SSE QMC algorithm is very capable of simulating typical groundstate observables measured in current state-of-the-art Rydberg array experiments. Considerable refinements of our algorithm are possible with straightforward modifications, including larger (plaquette) Hamiltonian breakups, and multicanonical sampling methods like parallel tempering. These simulations will be able to offer more numerical insights into exotic physics contained in Rydberg atom arrays through detailed finite size scaling analyses, and will make available the wide array of well-developed SSE techniques, such as replica measurements of the Rényi entanglement entropies [115, 80, 94, 93].

Our SSE algorithm will be useful in directly characterizing equilibrium groundstate properties on Rydberg arrays of the exact size and lattice geometry of current experiments [12, 17, 39, 50]. In addition, QMC simulations such as this will be crucial for providing data for pre-training generative machine learning models, which are poised to become important tools in state reconstruction and tomography [23, 201, 186, 40]. To this point, it

is foreseeable that our SSE algorithm will be required to access system sizes beyond current experiments to facilitate the aforementioned numerical studies. We expect our SSE algorithm to set the standard for the performance of numerical simulation methods going forward. Finally, although the Rydberg Hamiltonian is fundamentally free of the sign problem – and hence lies in a complexity class where its groundstate properties are theoretically known to be amenable to efficient simulation – we have illustrated that devising an efficient algorithm is nontrivial in practice. The question we leave open is whether an efficient global SSE cluster update is available for all Rydberg interaction geometries which can be engineered in current and future experiments. Without algorithmic studies like the present to advance QMC and other simulation technologies forward [166, 167, 168, 208, 22], even sign-problem free Hamiltonians like those found in Rydberg arrays may stake a claim to experimental quantum advantage in the surprisingly near future.

Chapter 6

Simulating Rydberg Atom Arrays on a Kagome Lattice

This chapter contains results and material from Ref. [90], along with additional commentary not published elsewhere.

6.1 Abstract

Rydberg atom array experiments have demonstrated the ability to act as powerful quantum simulators, preparing strongly-correlated phases of matter which are challenging to study for conventional computer simulations. A key direction has been the implementation of interactions on frustrated geometries, in an effort to prepare exotic many-body states such as spin liquids and glasses. In this paper, we apply two-dimensional recurrent neural network (RNN) wave functions to study the ground states of Rydberg atom arrays on the kagome lattice. We implement an annealing scheme to find the RNN variational parameters in regions of the phase diagram where exotic phases may occur, corresponding to rough optimization landscapes. For Rydberg atom array Hamiltonians studied previously on the kagome lattice, our RNN ground states show no evidence of exotic spin liquid or emergent glassy behavior. In the latter case, we argue that the presence of a non-zero Edwards-Anderson order parameter is an artifact of the long autocorrelations times experienced with quantum Monte Carlo simulations. This result emphasizes the utility of autoregressive

models, such as RNNs, to explore Rydberg atom array physics on frustrated lattices and beyond.

6.2 Introduction

Rydberg atom arrays have emerged as a rich playground for quantum simulation of many-body problems [17]. A key property of these arrays is their high degree of programmability, which enables the realization of multiple Hamiltonians on different lattice geometries and parameter ranges. This programmability facilitates the simulation of a wide array of phases of matter [50, 233] and enables the solution to challenging combinatorial optimization problems [233, 49, 151]. Remarkably, the preparation of spin liquid phases—disordered phases of matter characterized by the presence of anyonic excitations, topological invariants, and long-range entanglement—has been demonstrated in programmable Rydberg arrays, potentially serving as building blocks of future generation of fault-tolerant qubits [43, 109, 110].

Recent numerical studies have investigated the physics of the ground state of Rydberg atom arrays in different lattice geometries, in particular in one [166] and two spatial dimensions in various geometries [167, 100, 123, 168, 208, 237, 112]. In lattices such as ruby and honeycomb lattices, strong numerical evidence favours the existence of a spin liquid phase in agreement with experiments [208, 112]. Another recent example is the kagome lattice, where Density Matrix Renormalization Group (DMRG) [224, 182] studies provided evidence that Rydberg atom arrays host a liquid-like regime [168], while Quantum Monte Carlo (QMC) simulations predicted the existence of a spin glass phase [236]. These systems display frustration arising from lattice geometry and Hamiltonian interactions, leading to the existence of a large number of quantum states with nearly degenerate energies but markedly different properties. This makes it computationally difficult to accurately approximate the ground state of these systems.

Here we focus on applying recurrent neural network (RNNs) wave functions [86, 163] to a Rydberg array of atoms on the kagome lattice. The effectiveness of RNNs and Transformer language models has already been demonstrated in Rydberg atom arrays on the square lattice [146, 190, 37]. RNNs possess two key properties that make them particularly well-suited for studying frustrated systems. Firstly, their ability to perform exact sampling helps mitigate frustration-induced ergodicity issues in quantum Monte Carlo. Secondly, the ability to define them in any spatial dimension without incurring additional computational intractability helps address challenges faced by techniques like DMRG, such as

the increased computational cost stemming from increased entanglement in higher dimensions [86, 89].

Our findings reveal that in the highly frustrated and highly entangled regimes of the system, the RNN predicts a paramagnetic phase without topological order, consistent with earlier QMC simulations [236]. However, in contrast to the QMC results in Ref. [236], the RNN suggests the absence of a spin-glass phase. Nevertheless, in agreement with QMC, our numerical simulations indicate the emergence of a rugged optimization landscape, necessitating more optimization steps and thermal-like fluctuations to mitigate local minima in the RNN’s parameter landscape.

Overall, our results showcase the remarkable applicability and advantages of machine learning-based wave functions, particularly RNNs, in tackling challenging problems at the forefront of Rydberg atom array physics. These findings pave the way for further exploration of exotic phases and phenomena in highly frustrated quantum systems, harnessing the power of modern machine learning techniques to advance our understanding in this field.

6.3 Methods

We focus our attention on an array Rydberg atoms on the kagome lattice with periodic boundary conditions (PBC). The Hamiltonian of this system is given in Eq. 5.3, which we restate here in full form for convenience:

$$H = \Omega \sum_{i < j} \left(\frac{R_b}{r_{ij}} \right)^6 n_i n_j - \delta \sum_{i=1}^N n_i - \frac{\Omega}{2} \sum_{i=1}^N \sigma_i^x \quad (6.1)$$

where Ω is the Rabi frequency, δ is the laser detuning, and R_b is the blockade radius. Finally, we note that for this study, the sum over all possible pairs is truncated to a sum over neighbours separated by a distance cutoff $R_c = 2$ or $R_c = 4$. The choice $R_c = 2$ is taken to compare with the DMRG results reported in Ref. [168] as well as with the QMC findings in Ref. [236].

6.3.1 Two dimensional RNNs

The Rydberg Hamiltonian is stoquastic in nature [15], which implies that the groundstate wavefunction contains only positive amplitudes. This offers the opportunity to model the

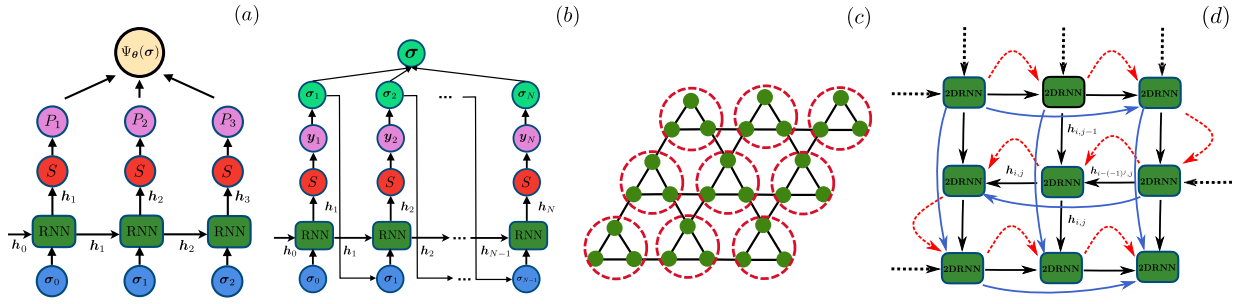


Figure 6.1: (a) An illustration of a positive RNN wavefunction. Each RNN cell receives an input σ_{n-1} and a hidden state \mathbf{h}_{n-1} and outputs a new hidden state \mathbf{h}_n . This vector is taken as an input to the Softmax layer (denoted S) that computes the conditional probability P_i . (b) RNN autoregressive sampling scheme: after obtaining the probability vector \mathbf{y}_i from the Softmax layer (S) in step i , we sample it to produce σ_i . The latter is taken again as an input to the RNN along with the hidden state \mathbf{h}_i to sample the following degree of freedom σ_{i+1} . (c) Mapping of a Kagome lattice to a square lattice by embedding three atoms in a larger local Hilbert space. (d) A two-dimensional (2D) RNN with periodic boundary conditions for a 3×3 lattice for illustration purposes. A bulk RNN cell receives two hidden states $\mathbf{h}_{i,j-1}$ and $\mathbf{h}_{i-(-1)^j,j}$, as well as two input vectors $\sigma_{i,j-1}$ and $\sigma_{i-(-1)^j,j}$ (not shown) as illustrated by the black solid arrows. RNN cells at the boundary receive additional hidden states $\mathbf{h}_{i,j+1}$ and $\mathbf{h}_{i+(-1)^j,j}$, as well as two input vectors $\sigma_{i,j+1}$ and $\sigma_{i+(-1)^j,j}$ (not shown), as demonstrated by the blue curved and solid arrows. The sampling path is taken as a zigzag path, as demonstrated by the dashed red arrows. The initial memory states of the 2D RNN and the initial inputs are null vectors, as indicated by the dashed black arrows.

groundstate with an RNN wavefunction with only positive amplitudes [86] which we adopt below. Complex extensions of RNN wavefunctions for non-stoquastic Hamiltonians have been explored in Refs. [86, 163, 89]. To model a positive RNN wavefunction, we can express our ansatz in the computational basis as:

$$\Psi_{\boldsymbol{\theta}}(\boldsymbol{\sigma}) = \sqrt{p_{\boldsymbol{\theta}}(\boldsymbol{\sigma})},$$

such that $\boldsymbol{\theta}$ corresponds to the variational parameters of the ansatz $|\Psi_{\boldsymbol{\theta}}\rangle$, and $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \dots, \sigma_N)$ is a configuration of the Rydberg atoms. The main advantage of using RNN wavefunctions is the possibility of estimating observables through autoregressive sampling, which allows obtaining uncorrelated samples by construction [86]. To do so, we model the joint probability $p_{\boldsymbol{\theta}}(\boldsymbol{\sigma})$ by constructing the conditionals $p_{\boldsymbol{\theta}}(\sigma_i|\sigma_{<i})$ by taking advantage of the probability chain rule

$$p_{\boldsymbol{\theta}}(\boldsymbol{\sigma}) = p_{\boldsymbol{\theta}}(\sigma_1)p_{\boldsymbol{\theta}}(\sigma_2|\sigma_1) \cdots p_{\boldsymbol{\theta}}(\sigma_N|\sigma_{N-1}, \dots, \sigma_2, \sigma_1).$$

These conditional probabilities are obtained through a Softmax layer as follows:

$$p_{\boldsymbol{\theta}}(\sigma_i|\sigma_{<i}) = \mathbf{y}_i \cdot \boldsymbol{\sigma}_i.$$

Here $\mathbf{y}_i = \text{Softmax}(U\mathbf{h}_i + \mathbf{c})$ where U and \mathbf{c} are, respectively, trainable weights and biases, and ‘Softmax’ corresponds to the normalizing Softmax activation function. Additionally, the memory (hidden) state \mathbf{h}_i is obtained recursively as [124]:

$$\mathbf{h}_i = f(W[\mathbf{h}_{i-1}; \boldsymbol{\sigma}_{i-1}] + \mathbf{b}), \quad (6.2)$$

such that $[\cdot; \cdot]$ is a concatenation of two vectors, while $\boldsymbol{\sigma}_{i-1}$ is a one-hot encoding of σ_{i-1} . These computations are illustrated in Fig. 6.1(a). W and \mathbf{b} are also trainable weights and biases, and f is a user-defined activation function.

By virtue of the ‘Softmax’ activation function, the conditionals $p_{\boldsymbol{\theta}}(\sigma_i|\sigma_{<i})$ are normalized to one. This property implies that the RNN joint probability $p_{\boldsymbol{\theta}}$ is also normalized [86]. Furthermore, by sampling the conditionals $p_{\boldsymbol{\theta}}(\sigma_i|\sigma_{<i})$ sequentially, as illustrated in Fig. 6.1(b), we can extract exact samples from the joint RNN probability $p_{\boldsymbol{\theta}}$. An attractive property of this scheme is the possibility to efficiently generate uncorrelated samples from different modes present in $p_{\boldsymbol{\theta}}$, whereas traditional MCMC sampling scheme may get stuck in only one mode.

The atom configurations of a Rydberg atom array on a kagome lattice can be seen as an $L \times L \times 3$ array of binary degrees of freedom where L is the size of each side of the lattice. As illustrated in Fig. 6.1(c), we can map our kagome lattice with a local Hilbert

space of 2 to a square lattice with an enlarged Hilbert space of size $2^3 = 8$ which we can study using our two-dimensional (2D) RNN wavefunction [88, 190].

To construct a 2D RNN ansatz that can handle PBC, we modify our RNN recursion in Eq. (6.2) to a two-dimensional recursion relation as:

$$\mathbf{h}_{i,j} = f\left(W[\text{Neighbours}(\mathbf{h}_{i,j}); \text{Neighbours}(\boldsymbol{\sigma}_{i,j})] + \mathbf{b}\right). \quad (6.3)$$

$\mathbf{h}_{i,j}$ is a memory state with two indices for each atom in the two-dimensional lattice. Here ‘Neighbours($\boldsymbol{\sigma}_{i,j}$)’ returns a concatenation of the neighbours of $\boldsymbol{\sigma}_{i,j}$. The same observation goes for ‘Neighbours($\mathbf{h}_{i,j}$)’. These neighbours correspond to incoming vectors indicated by the black and blue arrows as illustrated in Fig. 6.1(d). More specifically, we define

$$\text{Neighbours}(\mathbf{h}_{i,j}) \equiv [\mathbf{h}_{i-(-1)^j,j}; \mathbf{h}_{i,j-1}; \mathbf{0}; \mathbf{0}]$$

on the bulk. On the boundaries, we take

$$\begin{aligned} \text{Neighbours}(\mathbf{h}_{i,j}) \equiv & [\mathbf{h}_{i-(-1)^j,j}; \mathbf{h}_{i,j-1}; \\ & \mathbf{h}_{i+(-1)^j,j}; \mathbf{h}_{i,j+1}]. \end{aligned}$$

Note that PBC on the indices is assumed. The additional inputs $\boldsymbol{\sigma}_{i+(-1)^j,j}$, $\boldsymbol{\sigma}_{i,j+1}$ and hidden states $\mathbf{h}_{i+(-1)^j,j}$, $\mathbf{h}_{i,j+1}$ allow to take PBC into account and to introduce correlations between degrees of freedom at the boundaries. During the autoregressive sampling procedure, the input and hidden vectors are initialized to a null vector if not previously defined to preserve the autoregressive nature of our scheme, as illustrated in Fig. 6.1(b). Also, note that the particular choice of the indices is motivated by the zigzag sampling path. In this study, we use an advanced version of 2D RNNs incorporating the gating mechanism as previously done in Refs. [89, 27, 130]. More details can be found in Ref. [90]. Finally, since $\mathbf{h}_{i,j}$ is a summary of the history of the generated $\sigma_{<i,j}$, it is used to compute the conditional probabilities as follows:

$$p_{\theta}(\sigma_{i,j} | \sigma_{<i,j}) = \text{Softmax}(U\mathbf{h}_{i,j} + \mathbf{c}) \cdot \boldsymbol{\sigma}_{i,j}. \quad (6.4)$$

6.3.2 Variational monte carlo (VMC)

To optimize the energy expectation value of our RNN wavefunction Ψ_{θ} , we use the Variational Monte Carlo (VMC) scheme [10], which consists of generating samples from a

parametrized wavefunction *ansatz* to estimate the energy expectation value $E_\theta = \langle \Psi_\theta | H | \Psi_\theta \rangle$ as follows [10, 86]:

$$E_\theta \approx \frac{1}{M} \sum_{i=1}^M E_{\text{loc}}(\boldsymbol{\sigma}^{(i)}), \quad (6.5)$$

where the local energies E_{loc} are defined as

$$E_{\text{loc}}(\boldsymbol{\sigma}) = \sum_{\boldsymbol{\sigma}'} H_{\boldsymbol{\sigma}\boldsymbol{\sigma}'} \frac{\Psi_\theta(\boldsymbol{\sigma}')}{\Psi_\theta(\boldsymbol{\sigma})}. \quad (6.6)$$

Here the configurations $\{\boldsymbol{\sigma}^{(i)}\}_{i=1}^M$ are sampled from our RNN *ansatz* using autoregressive sampling. The choice of M is a hyperparameter that can be tuned. Similarly, the gradients can be estimated as

$$\partial_\theta E_\theta \approx \frac{1}{M} \sum_{i=1}^M \partial_\theta \log(\Psi_\theta^*(\boldsymbol{\sigma}^{(i)})) (E_{\text{loc}}(\boldsymbol{\sigma}^{(i)}) - E_\theta). \quad (6.7)$$

Subtracting the mean energy E_θ is helpful to achieve convergence as it reduces the variance of the gradients without biasing its expectation value [86, 88]. The gradient descent steps are performed using the Adam optimizer [107]. Similarly to the stochastic energy estimation, we can implement a similar procedure for the estimation of the variational pseudo-free energy F_θ in Eq. (6.8). Ref. [87] provides more details in the supplementary information.

6.3.3 Supplementing RNN optimization with annealing

To reach the groundstate of the Rydberg atoms array Hamiltonian on the kagome lattice, we minimize the energy expectation value $E_\theta = \langle \Psi_\theta | H | \Psi_\theta \rangle$ using the Variational Monte Carlo scheme. Due to the frustrated nature of the kagome lattice which can induce local minima in the VMC scheme, we leverage annealing with thermal-like fluctuations to mitigate local minima. This technique has been suggested and implemented in Refs. [163, 87, 89, 164, 105, 88]. In this case, we obtain a free-energy like cost function, defined as

$$F_\theta(n) = E_\theta - T(n) S_{\text{classical}}(p_\theta), \quad (6.8)$$

where F_θ is a variational pseudo Free energy and $S_{\text{classical}}$ is the classical Shannon entropy:

$$S_{\text{classical}}(p_\theta) = - \sum_{\boldsymbol{\sigma}} p_\theta(\boldsymbol{\sigma}) \log(p_\theta(\boldsymbol{\sigma})). \quad (6.9)$$

The previous sum goes over all classical Rydberg configurations $\{\sigma\}$ in the computational z -basis. Note that $S_{\text{classical}}$ is a pseudo-entropy that can be efficiently estimated using our RNN wavefunction as opposed to the quantum von Neumann entropy. Additionally, $T(n)$ is a pseudo-temperature that is annealed from some initial value T_0 to zero as follows: $T(n) = T_0(1 - n/N_a)$ where $n \in [0, N_a]$ and N_a is the total number of annealing steps. For more details about the hyperparameters of the training scheme refer to the Appendix of the full paper [90].

Although we can perform annealing in the SSE simulations, preliminary runs for $L = 6$ and $R_b = 1.95, \delta = 3.3$ found that it does not seem to affect the simulation results much, with almost all chains giving observable estimates consistent with each other, indicating that the transition kernel is able to successfully converge to the correct distribution (or at least one of its modes) without much difficulty.

6.4 Results

According to the RNN numerics, our results show that the ground state at $R_b = 1.95$ and $\delta = 3.3$, which is suggested to be in the spin-liquid phase according to Ref. [168], is rather a disordered state with no topological order. We first plot the correlations $\langle n_{\mathbf{0}} n_{\mathbf{r}} \rangle$ in Fig. 6.2(a). The results indicate that the extracted state has short-range correlations.

To investigate the existence of a spin liquid in this regime, we calculate the TEE γ using the Kitaev-Preskill construction [111] for a system size $L = 8$, and for different values of $\delta \in [2.0, 3.7]$ and $R_c = 2, 4$ at $R_b = 1.95$. We also do the same using the Levin-Wen construction [96]. Our results, illustrated in Fig. 6.2(b) suggest that the TEE extracted by the RNN is consistent with zero and different from $\ln(2)$ within error bars. These results suggest the non-existence of a spin liquid within our settings and also suggest that the state we find in this regime is a disordered state. Our findings are further corroborated by a recent QMC study [236] and also by previous results in the literature suggesting that the paramagnetic ‘liquid’ phase in Ising systems on the kagome lattice is not exotic [152, 142, 143].

In the aforementioned QMC study [236], it was suggested that the region, around $R_b = 1.95$ and the values of δ used in our study, contains an emergent spin-glass phase instead of a paramagnetic state. To verify this claim, we compute the Edwards-Anderson (EA) order parameters [51, 162], defined as:

$$q_{\text{EA}} = \frac{\sum_{i=1}^N \langle n_i - \rho \rangle^2}{N\rho(1 - \rho)}, \quad (6.10)$$

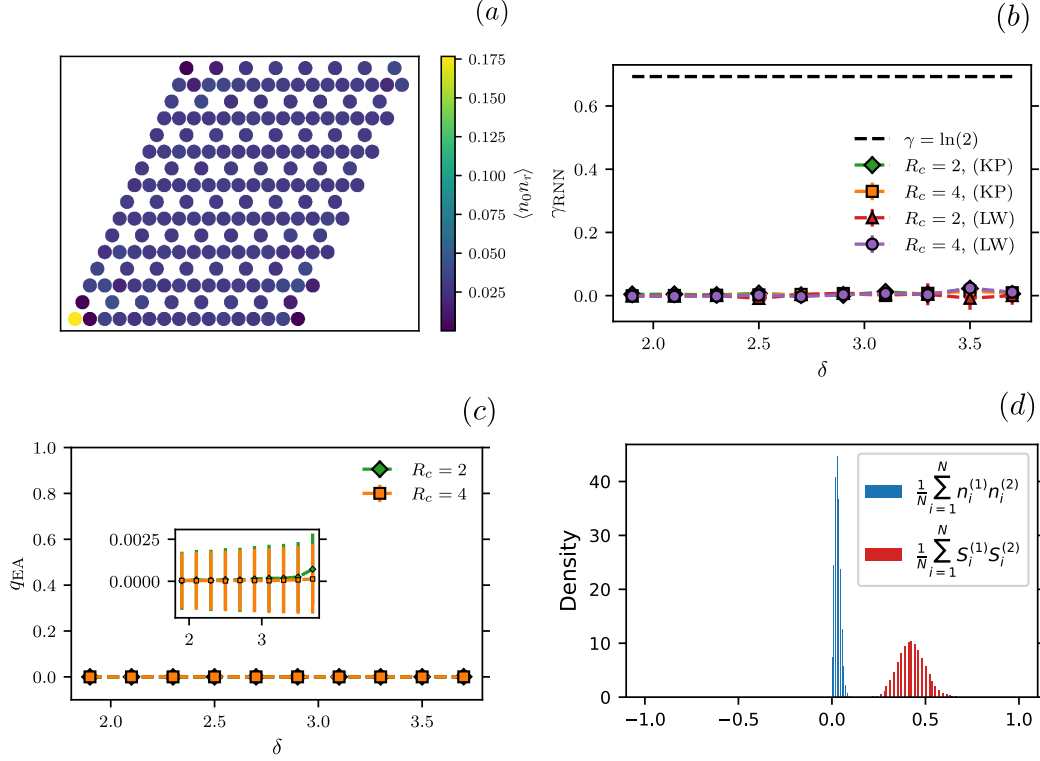


Figure 6.2: In these panels, we focus on the Blockade radius $R_b = 1.95$. (a) Plot of two point correlations $\langle n_{\mathbf{0}} n_{\mathbf{r}} \rangle$ with $\delta = 3.3$ and for a system size $N = 8 \times 8 \times 3$ and $R_c = 2$. (b) Plots of the topological entanglement entropy versus δ for two different values of the cutoff radius R_c , using the Levin-Wen (LW) construction and the Kitaev-Preskill (KP) construction, for $N = 8 \times 8 \times 3$. (c) A plot of the Edwards-Anderson order parameter q_{EA} defined in Eq. (6.10) as a function of δ for $N = 8 \times 8 \times 3$. (d) A plot of the density overlaps $\frac{1}{N} \sum_{i=1}^N n_i^{(1)} n_i^{(2)}$ and the spin overlaps $\frac{1}{N} \sum_{i=1}^N S_i^{(1)} S_i^{(2)}$ at $\delta = 3.3$. Here $S_i = 2n_i - 1$, and (1) and (2) are labels for two sets of samples obtained from our optimized RNN, that are aggregated from 10 different training seeds, for $N = 6 \times 6 \times 3$ and $R_c = 2$. For each seed, we generate 2×10^5 independent samples and divide them into two sets.

System Parameters	QMC	2DRNN
$R_b = 1.7, \delta = 3.3 (R_c = 2)$	-0.790975(14)	-0.790964(5)
$R_b = 1.7, \delta = 3.3 (R_c = 4)$	-0.77546(1)	-0.775412(4)
$R_b = 1.95, \delta = 3.3 (R_c = 2)$	-0.59785(1)	-0.59657(1)
$R_b = 1.95, \delta = 3.3 (R_c = 4)$	-0.56445(1)	-0.56401(3)

Table 6.1: A table of the energies per site obtained by QMC and 2DRNN for a system size $8 \times 8 \times 3$ with fully periodic boundary conditions and for different values of the cutoff radius R_c . The error bars in parentheses correspond to the standard error of the mean.

where N is the system size, n_i is the occupation number of site i and $\rho = (\sum_{i=1}^N n_i)/N$. Clearly, for a system with translational invariance, q_{EA} should be zero. Deviations of this order parameter from zero values are signals of the existence of a spin-glass phase. In Fig. 6.2(c), we plot this order parameter as a function of δ with $R_c = 2, 4$ and $R_b = 1.95$. We find that the values of the order parameter are consistent with zero, as opposed to the results of QMC in Ref. [236]. Furthermore, we report in Fig. 6.2(d) the density-density overlap $\frac{1}{N} \sum_{i=1}^N n_i^{(1)} n_i^{(2)}$ and the spin-spin overlap $\frac{1}{N} \sum_{i=1}^N S_i^{(1)} S_i^{(2)}$ between different RNN samples at $R_b = 1.95, \delta = 3.3$, and $R_c = 2$. Here labels (1) and (2) correspond to two independent sets of samples, which are obtained from optimized RNNs with 10 different training seeds. The Gaussian nature of the overlap distribution in both representations is another indicator that there is no static signature of a spin-glass order [28].

The discrepancy in our results and previous QMC findings [236] could be related to emergent glassy dynamics in the QMC simulations, which results in very long auto-correlations times and thus in a non-ergodic behavior. To corroborate our findings, we run QMC simulations [139] for larger inverse temperatures compared to Ref. [236], namely for $\beta \geq 200$ and using 2.2×10^6 Monte Carlo samples. We find that the QMC prediction for the EA order parameter is given as $q_{\text{EA}}^{\text{QMC}} = 0.000018(5)$ for $R_b = 1.95, \delta = 3.3$, a system size $8 \times 8 \times 3$, and for a radius cutoff $R_c = 2$. The previous result agrees very well with our RNN findings in Fig. 6.2(c). This result is also confirmed by the good agreement between the RNN energies and the QMC energies as shown in Table 6.1. Our findings are further supported by the results of Ref. [235], which suggests the possibility of transition in a quantum dimer model between nematic to paramagnetic to staggered states. In conclusion, our numerical investigation suggests that the long auto-correlation time could be a limiting factor in the QMC results reported in Ref. [236].

6.4.1 Numerical comparisons

In Table 6.1, we show a comparison between QMC’s and RNN’s energies per site for a system size $8 \times 8 \times 3$ and for a detuning $\delta = 3.3$ and at the blockade radii $R_b = 1.7, 1.95$. These points correspond to the nematic and disordered phases, respectively. We note that our RNN-based ansatz provides energies with a relative error of less than 0.2% compared to the QMC energies. The QMC simulations we run for Rydberg atom arrays are introduced in Ref. [139]. We use a finite-temperature QMC scheme run at several different values of β until we observe convergence to the ground state. For each β , five independent simulations are taken and the convergence of observables is observed at $\beta = 200$. Thus, to compute observables, we treat simulations with $\beta \geq 200$ as additional independent chains, giving us a total of 25 independent Markov chains at each parameter point. Each chain is allowed to warm-up for 10^4 steps, after which 10^6 sequential measurements were taken. With respect to the computation of the Edwards-Anderson order parameter, q_{EA} , we note that the analysis given in Ref. [236] can give different results in the case of imperfect sampling. Ref. [236] computes the order parameter independently for each Markov chain and then averages the results. This procedure can produce different results as each chain will only explore a subset of the QMC configuration space due to the presence of frustrated interactions. As a result, each chain’s estimate of the one-point function can be biased. Since q_{EA} is a non-linear function of the one-point function, we must first aggregate the one-point functions generated by each Markov chain, and then compute q_{EA} . Lastly, to compute an estimate of the error in q_{EA} , we must account for auto-correlations and non-linearity simultaneously. This step is done by combining jackknife resampling with a binning procedure. To deal with auto-correlations, we first compute the one-point function on sequential “bins” of data; we found a bin size of 10^4 to be sufficient, giving 100 bins for each chain. Thus, we can consider each bin’s one-point function to be nearly uncorrelated, allowing us to directly apply the jackknife resampling procedure to these approximately independent bins.

By computing q_{EA} separately for each chain before averaging, what Ref. [236] is doing is essentially performing a disorder average of the Edwards-Anderson order parameter, but treating each individual chain as a disorder instance. Assuming that this is indeed a valid method of detecting glassy dynamics, we can only conclude that the dynamics of the *MCMC simulation* is glassy, not the actual physical system, as MCMC dynamics do not correspond to physical dynamics.

6.5 Conclusions and Outlooks

In this chapter, we demonstrate a successful application of recurrent neural network (RNN) wavefunctions to the task of investigating topological order on Rydberg atom arrays on kagome lattice. We use these architectures to estimate the second Renyi entropies using the swap trick [86]. The latter allows us to compute the TEEs using the Kitaev-Preskill [111] and the Levin-Wen [121] constructions. Furthermore, with the possibility of handling periodic boundary conditions in RNNs, the boundary effects on the TEE are reduced compared to DMRG, which has challenges with boundary effects on cylinders [70].

Our main finding, suggested by the two-dimensional RNN wavefunctions results, points out that Rydberg atom arrays on the kagome lattice do not establish a Z_2 spin liquid in the highly entangled regime. This observation is also consistent with previous QMC studies [236]. Our RNN numerics also suggest that the highly entangled region corresponds to a trivial paramagnetic state and that there is no signature for spin glass order as opposed to the observations outlined in Ref. [236]. We believe that the ability of RNNs to generate uncorrelated samples from a multimodal distribution is a crucial factor in ruling out the spin-glass phase. Furthermore, supplementing RNNs with annealing turns out to be a valuable tool for mitigating local minima induced by the frustrated nature of the kagome lattice in the highly entangled regime. Additionally, we conclude that autocorrelation could be the main factor behind the observed spin glass phase observed in previous QMC simulations [236].

Finally, we note that our method can be generalized to study other systems with potential topological order, such as the Rydberg atom arrays on the Ruby lattice [208, 187, 68]. One could also use quantum state tomography with RNNs [23] in a wide variety of quantum simulators and also combine data with VMC to improve the variational results [11, 37, 146]. We also believe in the potential of RNN wavefunctions ansätze in the discovery of new phases of matter with topological order. Overall, these results highlight the promising future of RNN wavefunctions [86, 163], language-model based wavefunctions, and neural quantum states [21] for investigating open questions and discovering new physics within the condensed matter community and beyond.

Chapter 7

Machine Learning Applications of Rydberg Simulation Data

This chapter contains results and material from Refs. [37, 62].

In this chapter we will briefly discuss some Machine Learning applications for the data generated using the Rydberg SSE simulation developed in Chapter 5. For convenience, we restate the Rydberg Hamiltonian here:

$$H = \Omega \sum_{i < j} \left(\frac{R_b}{r_{ij}} \right)^6 n_i n_j - \delta \sum_i n_i - \frac{\Omega}{2} \sum_i \sigma_i^x \quad (7.1)$$

We will first discuss the idea of *data enhanced* VMC, in which a generative model (in this case an RNN) used as a wavefunction ansatz is first trained on a small amount of Rydberg occupation basis data from the true groundstate system (which we generate using QMC), before switching to a standard VMC training procedure. Since the groundstate is positive and real valued, a single basis is sufficient to perform quantum state reconstruction. This data-driven pretraining step is found to greatly accelerate the overall convergence of the model, in some cases allowing the ansatz to converge to the groundstate in less than a third of the number of training iterations required otherwise.

We then move on to the RydbergGPT model, a generative pre-trained transformer model which is trained on Rydberg occupation basis measurements taken at various differ-

ent Hamiltonian parameters and system sizes. We find that the model is able to interpolate quite well over the laser detuning parameter δ , giving proper predictions for various observables, both diagonal and off-diagonal.

As we've already discussed VMC training in Chapter 6, we will now briefly review data-driven training of generative models.

7.1 Data-Driven Training

Given a generative model with parameters θ , such as an RNN or Transformer, we have available to us some method to compute the probability of a sample (which we denote by σ) as $p_\theta(\sigma)$.

Now, let's say we have a dataset of \mathcal{D} samples $\{\sigma_1, \sigma_2, \dots, \sigma_{\mathcal{D}}\}$ and we want to train the model to be able to generate samples which are similar to those found in the dataset. Assuming that the dataset was drawn from some underlying probability distribution $q(\sigma)$, we would therefore like to train the model such that p_θ approximates q . To do this we must perform an optimization procedure by tuning the parameters θ in order to minimize some objective function which measures the deviation of p_θ from q . For probability distributions, the KL-divergence is the standard choice.

$$D(q||p_\theta) = \sum_{\{\sigma\}} q(\sigma) \log \frac{q(\sigma)}{p_\theta(\sigma)} \quad (7.2)$$

$$= \sum_{\{\sigma\}} q(\sigma) (\log q(\sigma) - \log p_\theta(\sigma)) \quad (7.3)$$

where the summation is over all configurations σ in the support of q . The first term is simply the entropy of the data distribution, which is a constant with respect to the parameters θ and therefore irrelevant to our optimization. The second term is known as the *negative log-likelihood*, and is the quantity we will seek to minimize.

$$\text{NLL}(q||p_\theta) = - \sum_{\{\sigma\}} q(\sigma) \log p_\theta(\sigma) \quad (7.4)$$

$$\approx - \frac{1}{\mathcal{D}} \sum_{i=1}^{\mathcal{D}} \log p_\theta(\sigma_i) \quad (7.5)$$

where in the last line we have approximated the negative log-likelihood using the samples from q . As we mentioned above, we now update the parameters θ in order to minimize

the negative log-likelihood. This is typically done using a gradient-descent procedure with the gradients being computed using automatic differentiation software [1, 156].

7.2 Data-Enhanced Variational Monte Carlo Simulations for Rydberg Atom Arrays

Rydberg atom arrays are programmable quantum simulators capable of preparing interacting qubit systems in a variety of quantum states. Due to long experimental preparation times, obtaining projective measurement data can be relatively slow for large arrays, which poses a challenge for state reconstruction methods such as tomography. Today, novel groundstate wavefunction ansätze like recurrent neural networks can be efficiently trained not only from projective measurement data, but also through Hamiltonian-guided variational Monte Carlo. In this section, we demonstrate how pretraining modern RNNs on even small amounts of data significantly reduces the convergence time for a subsequent variational optimization of the wavefunction. This suggests that essentially any amount of measurements obtained from a state prepared in an experimental quantum simulator could provide significant value for neural-network-based VMC strategies.

The quantum state of an array is probed with fluorescent imaging techniques, which provide projective measurements in the Rydberg occupation basis [53, 50, 234]. Since each measurement is destructive, the repetition rate at which they can be performed is limited by a number of factors, in particular the preparation time of the target quantum state. The probabilistic loading of the array requires a non-trivial rearrangement of atoms, resulting in repetition rates on the order of a few measurements per second [53, 50], with exact time scales depending on the specific experimental setup. Thus, data acquisition is limited, especially when compared to competing quantum simulation platforms, such as ion traps which allow for hundreds of measurements per second [144], or superconducting circuits where orders of magnitude more measurements per second can be achieved [218].

Data acquisition rates have obvious consequences for state reconstruction and characterization, as well as the direct estimation of operator expectation values, which suffer variances that scale inversely proportional to the number of independent measurements. Recently, neural network wavefunctions have been explored as tools for leveraging limited measurement data, as they provide a powerful ansatz for representing quantum states with systematically tunable expressivity [196, 21, 200, 197, 23, 86, 132]. For example, standard generative models adopted from the field of machine learning have been used to tomographically reconstruct quantum states [198, 201, 195, 149, 213, 35, 119] and have demonstrated

the ability to significantly reduce the amount of measurements required for the accurate reconstruction of operator expectation values [195]. In addition to their designed ability for *data-driven* learning, these ansätze have the ability to find groundstate wavefunctions of a given Hamiltonian by variational energy minimization, via the same *Hamiltonian-driven* training methods common in variational Monte Carlo (VMC) [137, 22, 10]. Modern neural network strategies provide VMC ansätze that can systematically be made powerful enough that their expressiveness is no longer the limiting bottleneck. Instead, the optimization process often requires long convergence times [19, 223, 205, 87, 145] and physics-inspired modifications of the network structure are sometimes needed to reach accuracies comparable to traditional VMC approaches [59, 210]. In addition, the power of wisely chosen initializations to improve convergence has already been demonstrated in traditional VMC methods [158, 204].

In this section, we leverage these unique features of neural network wavefunctions to explore the effect of combined data- and Hamiltonian-driven learning [11]. Beginning with a randomly initialized recurrent neural network (RNN) [86], we first optimize network parameters using a limited amount of simulated [173, 139, 100] Rydberg occupation data drawn from a two-dimensional array in the vicinity of a quantum phase transition. Then, we continue optimizing the network variationally, in the spirit of the recent work by Carrasquilla and Torlai [22]. We find a significant enhancement in variationally obtaining the groundstate wavefunction by pretraining the RNN on a limited amount of quantum data.

7.2.1 Background

We consider a system of $N = L \times L$ atoms arranged on a square lattice with spacing $a = 1$ and open boundary conditions. Each atom can be found in a groundstate $|g\rangle$ or in an excited (Rydberg) state $|r\rangle$. The system is driven by the Rydberg Hamiltonian. We fix $\delta = \Omega = 1$ and $R_b = 7^{1/6}$, so that $V_{ij} = 7/|r_{ij}|^6$, in the following. This brings the system in the vicinity of the transition between the disordered and the striated phase [50].

Given sufficient projective measurements, a quantum state can be tomographically reconstructed, e.g. via a neural network wavefunction ansatz [195, 201, 200, 198, 22, 197, 23, 3, 29, 149]. Here, we focus on RNNs to represent quantum states, and choose the gated recurrent unit (GRU) [32] as network cell, inspired by Refs. [86, 22]. Unlike the study in the previous chapter, we use a 1-dimensional RNN here, using the “snake” path to define a sampling sequence through the 2-dimensional lattice. The RNN generates an output based on a sequence of inputs σ and the state of N_h hidden neurons per network cell. During the training process, the network parameters θ are tuned to generate a target output. The

amount of tunable parameters is defined by N_h , which can be increased to improve the network expressivity. More details on RNNs are given in [86, 22, 32].

RNNs are naturally designed to encode probability distributions [86]. The network output at iteration i can be interpreted as the conditional probability distribution $p_\theta(\sigma_i|\sigma_{<i})$, providing the joint distribution $p_\theta(\boldsymbol{\sigma}) = \prod_i p_\theta(\sigma_i|\sigma_{<i})$. To represent a wavefunction, single qubit states are chosen as network input, which is iterated over the entire qubit system. The RNN is then trained to encode the probability distribution underlying projective measurements in the computational basis. In the case of positive real-valued wavefunctions, such as the groundstate of Eq. 7.1, the RNN represents the full quantum state $\Psi_\theta(\boldsymbol{\sigma}) = \langle \boldsymbol{\sigma} | \Psi_\theta \rangle = \sqrt{p_\theta(\boldsymbol{\sigma})}$. Further modifications can be used to reconstruct arbitrary complex wavefunctions or density matrices [86, 22].

Finally, samples from the generative step of the RNN-encoded distribution emulate projective measurement outcomes. The quantum state of the full system can be sampled by iteratively drawing single qubit states. We use these samples to evaluate the energy expectation value $H_{\text{RNN}} = \langle \Psi_\theta | H | \Psi_\theta \rangle$ via Eq. 6.5, which is calculated and averaged over N_s samples $\boldsymbol{\sigma}$ drawn from $p_\theta(\boldsymbol{\sigma})$ and can be evaluated efficiently for local off-diagonal operators [21, 22, 200, 137, 86].

7.2.2 RNN training procedures

We first explore the reconstruction of the groundstate of a Rydberg atom array based on a data set $\{\boldsymbol{\sigma}_i\}_{i=1}^D$ consisting solely of projective measurements taken in the Rydberg occupation basis.

In this data-driven setting, we optimize an RNN to approximate the probability distribution $p_\theta(\boldsymbol{\sigma}) \approx q(\boldsymbol{\sigma})$ underlying the data points, by minimizing the negative log-likelihood given in Eq. 7.4. We use the Adam optimizer [107] to train a Glorot uniform initialized [69] RNN by determining parameters $\boldsymbol{\theta}$ that minimize the loss function.

In order to produce a dataset, we use the quantum Monte Carlo (QMC) algorithm introduced in Chapter 5, which is able to accurately emulate projective measurements of Rydberg atoms in the groundstate of the Hamiltonian in Eq. 7.1. We consider systems with up to $N = 16 \times 16$ atoms, relevant for state-of-the-art experimental realizations [50], for which an unbiased estimate of the energy H_{QMC} is easily obtained. We generate 10^5 QMC samples for all considered system sizes, which gives estimates of the groundstate energy density H_{QMC}/N with errors on the order of $\sim 10^{-4}$ (caption of Fig. 7.1).

We implement our RNN based on the code provided in [22] and choose similar network hyperparameters, fixing the learning rate to $\eta = 0.001$ unless otherwise stated. We first

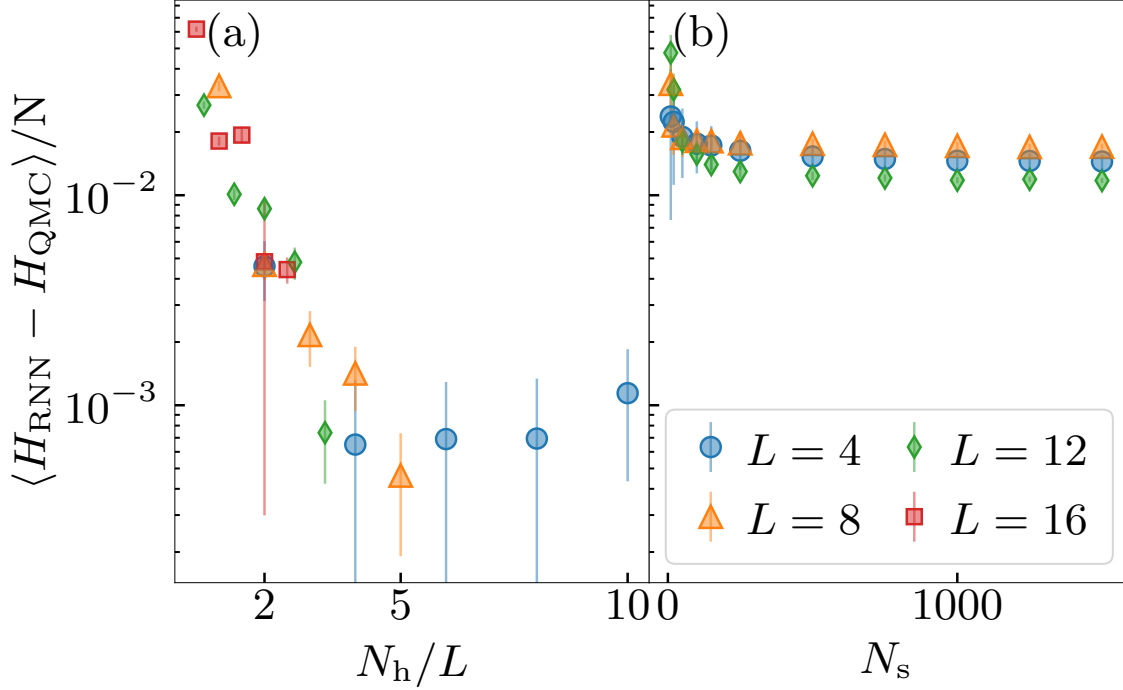


Figure 7.1: (a) Energy density difference of data-driven trained RNN states for different system sizes $N = L \times L$ as a function of the number of hidden neurons N_h per network cell. Results are averaged over iterations 1350 to 1400 (950 to 1000 for $N = 16 \times 16$) and error bars denote standard deviations. The RNN is trained on 10^5 data points with learning rate $\eta = 10^{-4}$. (b) Energy density difference of an RNN trained with the Hamiltonian-driven procedure using $\eta = 10^{-3}$. Results are plotted versus the number of RNN samples N_s for different system sizes, with $N_h = 2L$. Data is averaged over iterations 9500 to 10000 with standard deviations as error bars. The QMC estimates for the energy densities are given in Table 7.1

System Size	$L = 4$	$L = 8$	$L = 12$	$L = 16$
Energy Density H_{RNN}/N	-0.4534(1)	-0.4052(2)	-0.3885(2)	-0.3805(2)

Table 7.1: Table of QMC energy densities.

analyze the expressivity of the RNN in the data-driven approach to training. In Fig. 7.1(a) we train the network on the entire set of 10^5 QMC data points for different system sizes of $N = L \times L$ atoms and plot the energy density difference $\langle H_{\text{RNN}} - H_{\text{QMC}} \rangle / N$ as a function of the number of hidden neurons N_{h} per network cell. The energy density differences are averaged over optimization iterations 1350 to 1400, where the training is approximately converged. Each iteration corresponds to one training epoch where the full input dataset is given to the RNN in batches of 100 randomly chosen samples and the network parameters are updated. The energy expectation value H_{RNN} is calculated on 1000 samples drawn from $p_{\theta}(\sigma)$. For $N = 16 \times 16$ we average the energy densities over iterations 950 to 1000 due to long computational runtimes, leading to larger variances as convergence is not yet reached. For all system sizes the energy density error shows a clear decreasing trend with increasing N_{h} , which corresponds to higher network expressivity. We find that the observed differences reach values below 10^{-2} for $N_{\text{h}}/L \geq 2$ in all cases. While higher reconstruction accuracies can be reached, increasing N_{h} comes at the price of higher computational costs. We thus focus on $N_{\text{h}} = 2L$ as a practical compromise in the following.

Next, we train the RNN to represent the groundstate of the same $N = L \times L$ system using the Hamiltonian-driven approach. In this procedure, the RNN parameters are optimized such that the energy expectation value H_{RNN} is minimized, corresponding to VMC. This becomes our new loss function for the VMC phase, and we again use the Adam optimizer [107] to find optimal network parameters θ . Here we evaluate the VMC loss, given by Eq. 6.5, on N_{s} samples drawn from $p_{\theta}(\sigma)$.

In Fig. 7.1(b) we consider the energy density difference as a function of N_{s} for different system sizes. The largest system size, $N = 16 \times 16$, is not shown due to exceeding computational runtimes. We choose $N_{\text{h}} = 2L$ and average the measured energy densities over optimization iterations 9500 to 10000. The measured differences decrease with increasing N_{s} , before they saturate at $\sim 10^{-2}$ for $N_{\text{s}} \geq 500$. In accordance with [22] we thus fix $N_{\text{s}} = 1000$ in the following. All system sizes show larger energy density differences than in Fig. 7.1(a). This demonstrates the limitation of the Hamiltonian-driven training procedure even after a large number of iterations, which require prohibitively long computation times. This large amount of required optimization steps is commonly caused by local optima in the parameter landscape and has led to various model-inspired modifications of VMC [204, 158, 145, 19, 205, 87]. Below, we show that variational optimization can incur significant performance improvements without system-specific modifications, by finding suitable initializations resulting from data-driven pretraining.

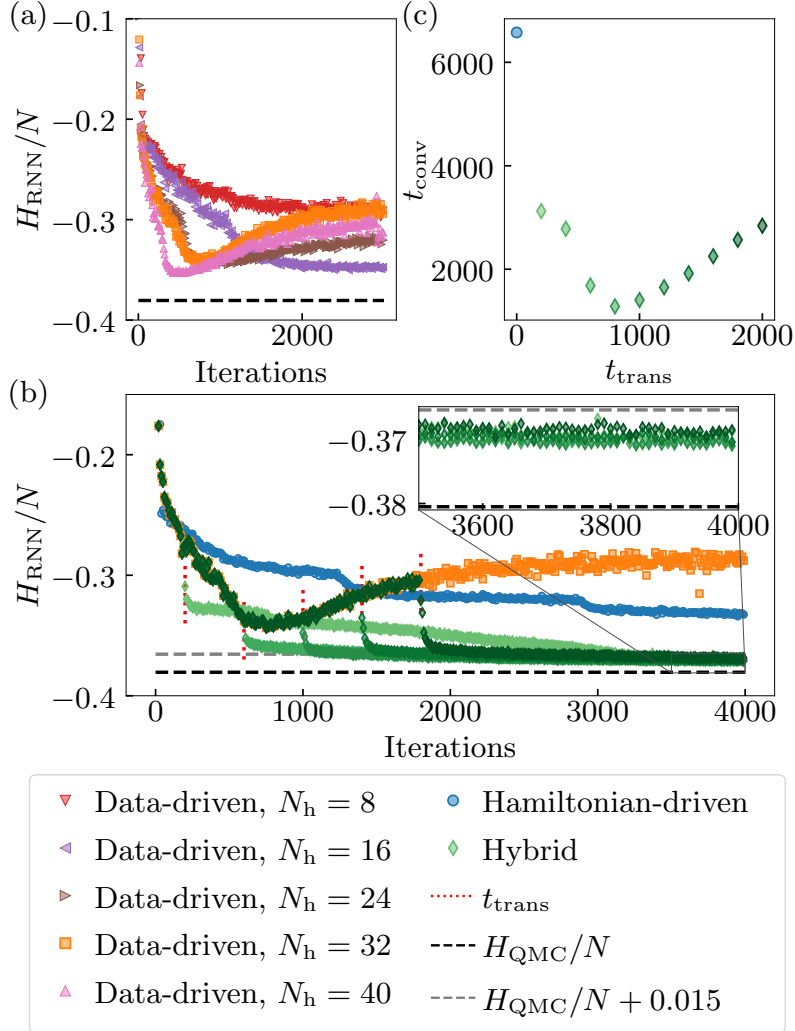


Figure 7.2: Quantum state reconstruction of an $N = 16 \times 16$ atom array. (a) Energy density H_{RNN}/N as a function of optimization iterations using the data-driven training on 1000 data points. Different shapes and colors show different N_h compared to the target energy density H_{QMC}/N . (b) Energy density evolution during data-driven, Hamiltonian-driven, and hybrid training with different t_{trans} (darker green shows larger t_{trans}) and $N_h = 32$. Inset emphasizes the convergence to the target solution after ≥ 3500 iterations within a margin of 0.015 for all $t_{\text{trans}} > 0$. (c) Convergence time t_{conv} (see main text for definition) as a function of transition point t_{trans} in the hybrid training of an RNN with $N_h = 32$.

7.2.3 Data-enhanced VMC

In the above, we have shown that RNN wavefunctions can be made sufficiently expressive to represent the groundstate of large Rydberg arrays, however naive variational optimization of the neural network parameters does not lead to accurate energies in a reasonable amount of simulation time. Further, due to the long state-preparation times in modern Rydberg experiments [53, 50], it is reasonable to hypothesize that accurate data-driven reconstruction will be challenging on large arrays due to limited data. To test this, we generate a randomly chosen subset of data containing 1000 projective Rydberg occupation measurements, representing a typical amount of experimental data. In Fig. 7.2(a) we show the energy density expectation value when training networks with different numbers of hidden neurons N_h on the Rydberg groundstate of $N = 16 \times 16$ atoms. Instead of converging towards the estimated value, the energies for $N_h \geq 24$ reach a minimum at ~ 700 iterations. This phenomenon is easily recognized as overfitting.

Clearly, such limited datasets are insufficient for accurate state reconstruction. However, we now demonstrate the ability of small datasets to enhance the performance of VMC in a *hybrid* training procedure, defined with a simple change of loss function. Namely, we begin training the RNN with the data-driven loss function given in Eq. 7.5, before switching to Hamiltonian-driven training via the loss given in Eq. 6.5 after t_{trans} iterations. Fig. 7.2(b) illustrates the effectiveness of this simple hybrid procedure. The green diamonds show the evolution of the energy using 1000 data points in data-driven training, before switching to Hamiltonian-driven training after t_{trans} iterations. As a comparison, we also plot purely Hamiltonian-driven training results (i.e. $t_{\text{trans}} = 0$). We explore a number of different choices of t_{trans} , which show significantly better convergence than the pure Hamiltonian-driven variational method out to 4000 optimization steps. All hybrid-trained simulations reach similar energy densities at $\gtrsim 3500$ iterations, which approximate the estimated value within a margin of 0.015 (see Fig. 7.2(b) inset).

In order to quantify the performance improvement of our algorithm, we define a convergence time t_{conv} as the iteration after which the energy density difference reaches $(H_{\text{RNN}} - H_{\text{QMC}})/N \leq 0.015$ for the first time. Results are illustrated in Fig. 7.2(c), where for better accuracy we consider the running average over 50 iterations, $\frac{1}{50} \sum_{i=-24}^{25} H_{\text{RNN}}(t+i)$, with t denoting the iterations. The convergence time is significantly reduced for all $t_{\text{trans}} > 0$, while Hamiltonian-driven training converges after $t_{\text{conv}} \sim 6600$ iterations. The shortest convergence time for our hybrid algorithm is observed for $t_{\text{trans}} = 800$, which is around the minimum of the data-driven training curve. Numerical studies on smaller system sizes have shown similar improvements in the hybrid training approach. While overfitting in the data-driven training starts at different points for different system sizes and

amounts of hidden neurons, Fig. 7.2(c) proposes that t_{trans} does not need to be optimized to ensure convergence within reasonable computational runtimes.

7.3 RydbergGPT

In this section we describe a generative pretrained transformer (GPT) designed to learn the measurement outcomes of an array of Rydberg atoms. Based on a vanilla transformer, our encoder-decoder architecture takes as input the Hamiltonian and outputs an autoregressive sequence of Rydberg occupation probabilities. Its performance is studied in the vicinity of a disordered-to-checkerboard quantum phase transition on a square lattice array. We explore the ability of the architecture to generalize, by producing groundstate measurements for Hamiltonian parameters not seen in the training set.

7.3.1 Introduction

Generative models have emerged as a key technology for predicting the probabilistic behaviour of a quantum system. Their most basic task is to produce a target sequence representing qubit measurement outcomes; for example, projective measurements distributed according to the Born rule of quantum mechanics. Thus far, many proposed models have been trained to output computational basis states from a single parameterized distribution.

In this section, we train an attention-based transformer [206] to learn the distribution of projective measurement outcomes corresponding to a Hamiltonian describing an array of interacting Rydberg atoms. We construct RydbergGPT to take the system parameters of a Rydberg atom array as input, and then produce computational basis states which approximately follow the Born probability distribution of the quantum state of the array.

We use SSE simulated Rydberg occupation data from a two-dimensional square-lattice array to train three different models. Once trained, each model can produce new data via autoregressive sampling. By varying the Hamiltonian that is input to the model, we will find that the model is able to accurately predict measurement outcomes in regions outside of the training regime.

7.3.2 Rydberg atom array physics

We consider a system of $N = L \times L$ Rydberg atoms arranged on a square lattice. For the purposes of our study, the array is considered to be in a thermal state at temperature

$T = 1/\beta$. The parameters defining the thermal state are therefore given by the vector $\mathbf{x} = (\Omega, \delta/\Omega, R_b, \mathbf{V}, \beta\Omega)$, where \mathbf{V} is the matrix containing the interactions $V_{ij} = \Omega(R_b/r_{ij})^6$. As mentioned in the previous section, the groundstate of the Rydberg Hamiltonian is positive and real-valued, meaning a single basis is sufficient for performing quantum state reconstruction. We will therefore use measurements taken in the Rydberg occupation basis as training data for our generative models, which we will now introduce.

7.3.3 Transformer architecture

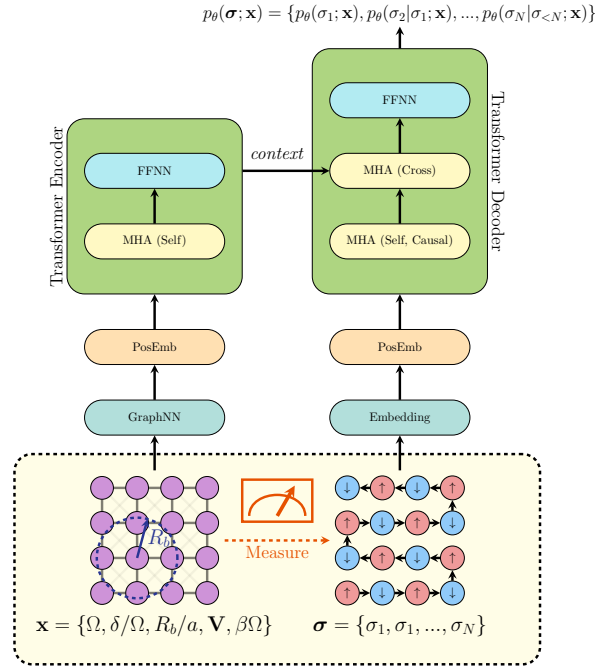


Figure 7.3: Overview of RydbergGPT. The bottom section represents the data from the physical system used in the architecture. On the left is a diagram representing the experimental settings \mathbf{x} of the system and on the right is a corresponding projective measurement σ . The experimental settings are inputs to the encoder portion of the model, composed of a graph neural network and the transformer encoder, and is processed to form a context vector. This context vector is combined with the sample to obtain a chain of conditional probabilities from the transformer decoder, the product of which is the probability of obtaining the sample from the system. The sequence of conditional probabilities is mapped onto the 2D lattice with a “snake” path as illustrated.

In order to learn the qubit measurement distribution of the Rydberg atom array, we employ the transformer encoder-decoder architecture [206]. The encoder maps the experimental Hamiltonian settings to a context vector can be used to condition this decoder. The encoder-decoder architecture that we use is illustrated in Fig. 7.3 and follows the overall design introduced by Vaswani et al. [206]. Transformers belong to the class of generative models known as autoregressive models, which, as we saw in Chapter 6, learn conditional probability distributions over the N individual variables, $p_{\theta}(\sigma_i|\sigma_{<i})$. Given a sample, the decoder outputs a probability, and just like an RNN wavefunction, the decoder can generate samples one site at a time. In contrast to the RNN wavefunction, the decoder conditions its outputs on the context vectors it receives from the encoder, as well as the previously generated site configurations. For more details on the model architecture, see Ref. [62]. We assume that the data distribution can be accurately described by a positive real-valued wavefunction, hence the transformer quantum state takes the form $\Psi_{\theta}(\sigma) = \sqrt{p_{\theta}(\sigma)}$ [146, 232], where θ denotes the model parameters.

7.3.4 Training dataset generation

To train the models, we will follow a data-driven training procedure, where we train transformers on a dataset composed of pairs of source and target sequences. The former corresponds to the system settings \mathbf{x} , input to the encoder, and the latter corresponds to binary Rydberg occupation measurement data, input to the decoder. Our target sequences are simulated Rydberg occupation measurements, produced via the Stochastic Series Expansion Quantum Monte Carlo method described in Chapter 5.

We employ our QMC to produce correlated samples, which we collect across a wide range of settings. Working in units of $\Omega = 1$ —which we will continue to do for the remainder of this section—we choose values of the inverse temperature β within the set $\{0.5, 1, 2, 4, 8, 16\}$, the detuning parameter δ over the range $[-0.364, 3.173]$, the Rydberg blockade R_b within $\{1.05, 1.15, 1.3\}$, and the linear system size L for the values $\{5, 6, 11, 12, 15, 16\}$. The total number of parameter configurations was 1080. For each configuration of Hamiltonian parameters, we generate a dataset by running an SSE simulation for those parameters. We collect 10^6 sequential samples at each parameter configuration. In order to ensure that the models are exposed to a sufficiently diverse subset of the configuration space, we would like the training data to be uncorrelated, thereby giving an effective sample size which is closer to the actual number of samples given to the model for training. We therefore partially de-correlate the data by keeping only every 10 samples, resulting in each Hamiltonian parameter configuration’s dataset containing 10^5 samples of Rydberg occupations. Although this does definitely not guarantee that the samples are

uncorrelated, it was the best we were able to do given the available computational budget. The datasets are selected to explore δ values near a quantum phase transition in the square-lattice Rydberg atom arrays, which occurs at approximately $\delta = 1.1$ [139].

7.3.5 Training transformer models

As described in Sec. 7.1, the transformer architecture can be trained in an unsupervised manner on data by minimizing the negative log-likelihood of a dataset $\{\sigma_1, \sigma_2, \dots, \sigma_{\mathcal{D}}\}$.

We trained three models with different subsets of the simulated data. Specifically, the models are all trained on data at $R_b = 1.15$ and across the values for δ and β . However, each model is trained with different sets of linear size L : model M_1 is trained with data for systems of size $L = 5, 6$, model M_2 with $L = 5, 6, 11, 12$ and model M_3 with $L = 5, 6, 11, 12, 15, 16$. Since we generate 10^5 samples for each lattice size, the larger lattice sizes have more tokens compared to the smaller ones. In addition to the models M_2 and M_3 needing to train on larger datasets, the samples within the datasets they do train on are larger. As a result, models M_2 and M_3 have relatively less time to learn from each sample, and overall perform fewer training iterations compared to M_1 given the same computational budget. Indeed, when trained on an Nvidia A100 for 85 hours, the models M_1 , M_2 , and M_3 were able to complete 116, 40, and 17 full training iterations respectively over their training datasets.

7.3.6 Results

We estimate certain physical observables of interest using the trained models and study the disordered-to-checkerboard phase transition of the Rydberg atom array, governed by the detuning δ . The model’s predictions are compared against estimates obtained from QMC simulations [139]. For Rydberg atom arrays, QMC is able to obtain an estimate of the observables that can be considered exact, upto statistical errors. In showing that our model’s predictions of these observables are in good agreement with the QMC values, we illustrate the efficacy of data-driven training as a technique for predicting Rydberg atom array measurement outcomes for groundstates.

It is important to emphasize that any estimates of off-diagonal observables are appropriate only when the output of the decoder can be interpreted as the normalized wavefunction amplitude, $\Psi_{\theta}(\sigma) = \sqrt{p_{\theta}(\sigma)}$. In particular, this correspondence is valid when the system is a pure state and the wavefunction can be assumed to be real and positive in the occupation basis, as is the case for the Rydberg Hamiltonian in its groundstate at $T = 0$. At

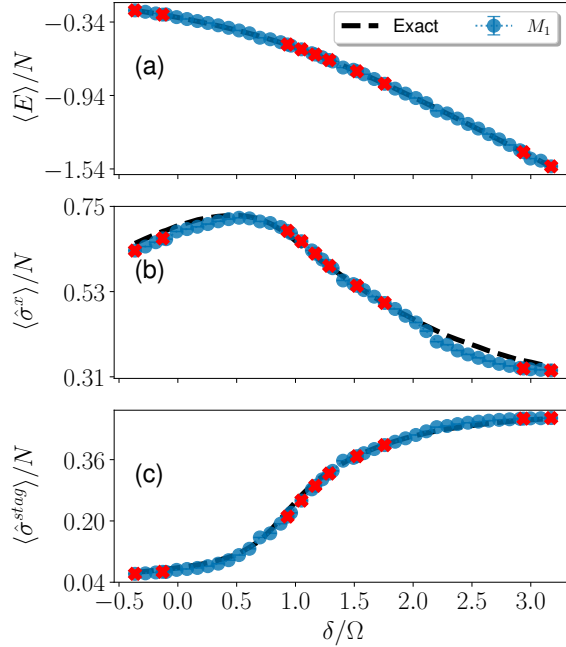


Figure 7.4: Model predictions of multiple observables across the disordered-to-checkerboard phase transition, which is governed by the detuning, δ . The observables include, (a) energy, (b) site-averaged x -magnetization and (c) the staggered magnetization. Here, blue points correspond to observables estimated from projective measurements sampled from the trained model (i.e., out-of-distribution), red points indicate parameters contained within the training dataset (i.e., in-distribution), and dotted black lines correspond to the exact observables.

$\beta = 16$, we find that the temperature is sufficiently low for the Rydberg atom array to be in its groundstate to a good approximation. Importantly, the use of the decoder output as a wavefunction amplitude is no longer valid for finite temperature states. While the above estimates can still be computed, $\Psi_{\theta}(\boldsymbol{\sigma})$ is not able to capture the physics of the thermal state, as such, we will not discuss the performance of the models at temperatures for which the state of the Rydberg array deviates significantly from the groundstate and instead refer the reader to the full paper [62].

In Fig. 7.4 we illustrate the performance of model M_1 on the three observables: the staggered magnetization, the x -magnetization, and the energy, all as functions of δ , for $R_b = 1.15, \beta = 16.0$. The estimators are calculated over $N_s = 10^5$ samples drawn from the trained model. The predictions are compared against the QMC observables calculated

from 10^5 partially de-correlated samples, which we take as the ground truth. We avoided using the full dataset of 10^6 correlated samples for the QMC estimates in order to offer a fair comparison for the models. The model shows good agreement with the QMC estimates for the energy and staggered magnetization, with slight deviations from the SSE estimates in the x -magnetization, near the tails. It's possible that this is due to the training set containing data from thermal states at various temperatures, thereby contaminating the learning of off-diagonal observables for the groundstate. Meanwhile the energy is dominated by its diagonal part, so deviations in the Rabi oscillation term of H are hidden. We conclude that the trained model is able to generalize well to detunings that lie outside of the training data set when the system is restricted to the groundstate and therefore well approximated by a classical probability distribution.

7.4 Conclusions

In this chapter, we have explored the use of recurrent neural networks (RNNs) and transformer models for studying groundstate wavefunctions of interacting Rydberg atom arrays of sizes currently accessible to experiments. We consider RNNs both trained from projective measurement data that could be produced by a typical experiment, as well as trained variationally with knowledge of the target Hamiltonian. We found that naive variational training of RNNs becomes challenging for arrays approaching current experimental sizes, meanwhile RNNs which undergo a preliminary data-driven training phase with a small amount of measurement data converge to accurate groundstate energies with significantly fewer optimization steps than their purely variationally trained counterparts. This result indicates the value of projective measurement data, obtained from Rydberg arrays and other quantum simulators, as a mechanism to significantly improve the convergence times of variational Monte Carlo simulations. This strategy should be immediately accessible to RNNs and other neural network wavefunction ansätze, which are amenable to both data-driven and Hamiltonian-driven training.

Going further down the route of data-driven training, we explored the training of transformer models trained solely on Hamiltonian parameters and computational basis measurement data. In the case where a quantum state can be assumed to be pure, real and positive, the probability distribution defined by the transformer decoder has a one-to-one mapping to the wavefunction, giving the output of the decoder a powerful interpretation. In the groundstate of Rydberg atom arrays governed by Hamiltonian given in Eq. 7.1, this assumption is valid, and we use it to demonstrate that a trained transformer model can accurately reproduce physical estimators in this case.

Chapter 8

Conclusions

8.1 Summary

In Chapters 2 and 3 of this thesis, we provided detailed derivations and explanations of Monte Carlo methods and of the Stochastic Series Expansion formalism for simulations of quantum many-body systems. We discussed how to estimate various different types of observables as well as their error bars, while also providing derivations for estimators that are difficult to find in the existing literature. We also provided advice on how to effectively compute certain estimators which suffer from poor numerical stability. Chapter 4 then focused on developing various update algorithms in the SSE formalism from first principles. Several different diagonal update schemes were developed and their various advantages and disadvantages were discussed thoroughly, leading to some advice on which scheme to use in a given scenario. Following this, we discussed off-diagonal updates specific to Hamiltonians similar to the TFIM. We detailed two cluster construction rules that could be employed in order to develop an efficient off-diagonal update scheme: the multibranch and line updates.

These results were then used in the development of an SSE algorithm for the Rydberg Hamiltonian in Chapter 5. We discussed why the multibranch algorithm fails to provide an efficient update, first through rough intuition-based arguments, and then later through concrete data on the distributions of cluster sizes for both off-diagonal update schemes.

We then move on to applications of the Rydberg SSE simulations. Investigations of a possible glassy phase were detailed in Chapter 6, followed by discussions on proper analysis of Markov Chain Monte Carlo data, particularly in respect to conclusions regarding physical dynamics based on simulations that do not actually follow physical dynamics.

Some relevant machine learning applications of measurement data generated by the Rydberg SSE simulations were then detailed in Chapter 7. These included the pre-training of variational models, showing that although VMC is very useful due to its lighter computational requirements and ability to more easily compute general off-diagonal observables, exact methods like SSE QMC are able to complement these methods by providing data which can help accelerate the convergence of these variational models. Additionally, we discussed a generative pretrained transformer model which was able to effectively train on multiple parts of the phase diagram and correctly predict the behaviour of groundstates it was not trained on.

8.2 Discussion and Future Directions

Although the Rydberg Hamiltonian is free of the sign-problem and therefore amenable to exact simulation with QMC, the construction of efficient update schemes is nevertheless non-trivial and the presence of strong autocorrelation can hinder scaling of simulations drastically. While the SSE simulation schemes detailed in this thesis for the Rydberg Hamiltonian are effective, there is still much that can be done to improve. New off-diagonal update schemes are absolutely necessary for performing larger scale simulations, however, probing larger R_b values may well require a departure from the SSE formalism due to the unfavourable scaling dependence of the simulation cell size on R_b .

In contrast, VMC methods typically have lighter computational requirements and a lower barrier to entry, allowing new users to more quickly obtain useful results pertaining to groundstate properties of quantum many-body systems. However, VMC ansatzes often suffer from some bias due to finite training time and optimization difficulties. By pre-training a VMC ansatz on a small amount of exact groundstate data generated via QMC, we can potentially ameliorate the bias somewhat. Nevertheless, simulations of finite-temperature systems remain difficult to solve variationally, and must therefore be performed using QMC. In this sense, QMC and VMC are complementary techniques.

Letters of Copyright Permission



01-Apr-2025

This license agreement between the American Physical Society ("APS") and Ejaaz Merali ("You") consists of your license details and the terms and conditions provided by the American Physical Society and SciPris.

Licensed Content Information

License Number: RNP/25/APR/089668
License date: 01-Apr-2025
DOI: 10.1103/PhysRevB.105.205108
Title: Data-enhanced variational Monte Carlo simulations for Rydberg atom arrays
Author: Stefanie Czischek et al.
Publication: Physical Review B
Publisher: American Physical Society
Cost: USD \$ 0.00

Request Details

Does your reuse require significant modifications: Yes

Modification Details:

I will be cutting out a lot, and mostly writing about the parts of the study that I worked on. I do have permission.

Specify intended distribution locations: Canada
Reuse Category: Reuse in a thesis/dissertation
Requestor Type: Author of requested content
Items for Reuse: Whole Article
Format for Reuse: Electronic

Information about New Publication:

University/Publisher: University of Waterloo
Title of dissertation/thesis: Quantum Monte Carlo Simulations of Rydberg Atom Arrays
Author(s): Ejaaz Merali
Expected completion date: Apr. 2025

License Requestor Information

Name: Ejaaz Merali
Affiliation: Individual
Email Id: emerali@uwaterloo.ca
Country: Canada



TERMS AND CONDITIONS

The American Physical Society (APS) is pleased to grant the Requestor of this license a non-exclusive, non-transferable permission, limited to Electronic format, provided all criteria outlined below are followed.

1. You must also obtain permission from at least one of the lead authors for each separate work, if you haven't done so already. The author's name and affiliation can be found on the first page of the published Article.
2. For electronic format permissions, Requestor agrees to provide a hyperlink from the reprinted APS material using the source material's DOI on the web page where the work appears. The hyperlink should use the standard DOI resolution URL, [http://dx.doi.org/\[DOI\]](http://dx.doi.org/[DOI]). The hyperlink may be embedded in the copyright credit line.
3. For print format permissions, Requestor agrees to print the required copyright credit line on the first page where the material appears: "Reprinted (abstract/excerpt/figure) with permission from [(FULL REFERENCE CITATION) as follows: Author's Names, APS Journal Title, Volume Number, Page Number and Year of Publication.] Copyright (YEAR) by the American Physical Society."
4. Permission granted in this license is for a one-time use and does not include permission for any future editions, updates, databases, formats or other matters. Permission must be sought for any additional use.
5. Use of the material does not and must not imply any endorsement by APS.
6. APS does not imply, purport or intend to grant permission to reuse materials to which it does not hold copyright. It is the requestor's sole responsibility to ensure the licensed material is original to APS and does not contain the copyright of another entity, and that the copyright notice of the figure, photograph, cover or table does not indicate it was reprinted by APS with permission from another source.
7. The permission granted herein is personal to the Requestor for the use specified and is not transferable or assignable without express written permission of APS. This license may not be amended except in writing by APS.
8. You may not alter, edit or modify the material in any manner.
9. You may translate the materials only when translation rights have been granted.
10. APS is not responsible for any errors or omissions due to translation.
11. You may not use the material for promotional, sales, advertising or marketing purposes.
12. The foregoing license shall not take effect unless and until APS or its agent, Aptara, receives payment in full in accordance with Aptara Billing and Payment Terms and Conditions, which are incorporated herein by reference.
13. Should the terms of this license be violated at any time, APS or Aptara may revoke the license with no refund to you and seek relief to the fullest extent of the laws of the USA. Official written notice will be made using the contact information provided with the permission request. Failure to receive such notice will not nullify revocation of the permission.
14. APS reserves all rights not specifically granted herein.
15. This document, including the Aptara Billing and Payment Terms and Conditions, shall be the entire agreement between the parties relating to the subject matter hereof.

References

- [1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- [2] Rajeev Acharya, Igor Aleiner, Richard Allen, Trond I. Andersen, Markus Ansmann, Frank Arute, Kunal Arya, Abraham Asfaw, Juan Atalaya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Joao Basso, Andreas Bengtsson, Sergio Boixo, Gina Bortoli, Alexandre Bourassa, Jenna Bovaird, Leon Brill, Michael Broughton, Bob B. Buckley, David A. Buell, Tim Burger, Brian Burkett, Nicholas Bushnell, Yu Chen, Zijun Chen, Ben Chiaro, Josh Cogan, Roberto Collins, Paul Conner, William Courtney, Alexander L. Crook, Ben Curtin, Dripto M. Debroy, Alexander Del Toro Barba, Sean Demura, Andrew Dunsworth, Daniel Eppens, Catherine Erickson, Lara Faoro, Edward Farhi, Reza Fatemi, Leslie Flores Burgos, Ebrahim Forati, Austin G. Fowler, Brooks Foxen, William Giang, Craig Gidney, Dar Gilboa, Marissa Giustina, Alejandro Grajales Dau, Jonathan A. Gross, Steve Habegger, Michael C. Hamilton, Matthew P. Harrigan, Sean D. Harrington, Oscar Higgott, Jeremy Hilton, Markus Hoffmann, Sabrina Hong, Trent Huang, Ashley Huff, William J. Huggins, Lev B. Ioffe, Sergei V. Isakov, Justin Iveland, Evan Jeffrey, Zhang Jiang, Cody Jones, Pavol Juhas, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Tanuj Khattar, Mostafa Khezri, Mária Kieferová, Seon Kim, Alexei Kitaev, Paul V. Klimov, Andrey R. Klots, Alexander N. Korotkov, Fedor Kostritsa, John Mark Kreikebaum, David Landhuis, Pavel

Laptev, Kim-Ming Lau, Lily Laws, Joonho Lee, Kenny Lee, Brian J. Lester, Alexander Lill, Wayne Liu, Aditya Locharla, Erik Lucero, Fionn D. Malone, Jeffrey Marshall, Orion Martin, Jarrod R. McClean, Trevor McCourt, Matt McEwen, Anthony Megrant, Bernardo Meurer Costa, Xiao Mi, Kevin C. Miao, Masoud Mohseni, Shirin Montazeri, Alexis Morvan, Emily Mount, Wojciech Mruczkiewicz, Ofer Naaman, Matthew Neeley, Charles Neill, Ani Nersisyan, Hartmut Neven, Michael Newman, Jiun How Ng, Anthony Nguyen, Murray Nguyen, Murphy Yuezhen Niu, Thomas E. O'Brien, Alex Opremcak, John Platt, Andre Petukhov, Rebecca Potter, Leonid P. Pryadko, Chris Quintana, Pedram Roushan, Nicholas C. Rubin, Negar Saei, Daniel Sank, Kannan Sankaragomathi, Kevin J. Satzinger, Henry F. Schurkus, Christopher Schuster, Michael J. Shearn, Aaron Shorter, Vladimir Shvarts, Jindra Skruzny, Vadim Smelyanskiy, W. Clarke Smith, George Sterling, Doug Strain, Marco Szalay, Alfredo Torres, Guifre Vidal, Benjamin Villalonga, Catherine Vollgraff Heidweiller, Theodore White, Cheng Xing, Z. Jamie Yao, Ping Yeh, Juhwan Yoo, Grayson Young, Adam Zalcman, Yaxing Zhang, Ningfeng Zhu, and Google Quantum AI. Suppressing quantum errors by scaling a surface code logical qubit. *Nature*, 614(7949):676–681, February 2023.

- [3] S. Ahmed, C. Sánchez Muñoz, F. Nori, and A. F. Kockum. Quantum state tomography with conditional generative adversarial networks. *Phys. Rev. Lett.*, 127:140502, 2021.
- [4] A. Fabricio Albuquerque, Fabien Alet, Clément Sire, and Sylvain Capponi. Quantum critical scaling of fidelity susceptibility. *Phys. Rev. B*, 81:064418, February 2010.
- [5] Vinay Ambegaokar and Matthias Troyer. Estimating errors reliably in monte carlo simulations of the ehrenfest model. *American Journal of Physics*, 78(2):150–157, 02 2010.
- [6] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.
- [7] Mari Carmen Bañuls, Rainer Blatt, Jacopo Catani, Alessio Celi, Juan Ignacio Cirac, Marcello Dalmonte, Leonardo Fallani, Karl Jansen, Maciej Lewenstein, Simone Montangero, Christine A. Muschik, Benni Reznik, Enrique Rico, Luca Tagliacozzo, Karel Van Acoleyen, Frank Verstraete, Uwe-Jens Wiese, Matthew Wingate, Jakub Zakrzewski, and Peter Zoller. Simulating lattice gauge theories within quantum technologies. *Eur. Phys. J. D*, 74(8):165, August 2020.

- [8] Leon Balents. Spin liquids in frustrated magnets. *Nature*, 464(7286):199–208, March 2010.
- [9] Daniel Barredo, Vincent Lienhard, Sylvain de Léséleuc, Thierry Lahaye, and Antoine Browaeys. Synthetic three-dimensional atomic structures assembled atom by atom. *Nature*, 561(7721):79–82, September 2018.
- [10] F. Becca and S. Sorella. *Quantum Monte Carlo Approaches for Correlated Systems*. Cambridge University Press, 2017.
- [11] E. R. Bennewitz, F. Hopfmueller, B. Kulchytsky, J. Carrasquilla, and P. Ronagh. Neural error mitigation of near-term quantum simulations. [quant-ph], 2021.
- [12] H. Bernien, S. Schwartz, A. Keesling, H. Levine, A. Omran, Hannes Pichler, Soonwon Choi, Alexander S. Zibrov, Manuel Endres, Markus Greiner, Vladan Vuletić, and Mikhail D. Lukin. Probing many-body dynamics on a 51-atom quantum simulator. *Nature*, 551(7682):579–584, November 2017.
- [13] Sounak Biswas and Kedar Damle. Efficient quantum cluster algorithms for frustrated transverse field ising antiferromagnets and ising gauge theories. *arXiv: 1812.05326*, 2018.
- [14] David Blackman and Sebastiano Vigna. Scrambled linear pseudorandom number generators, 2022.
- [15] Sergey Bravyi. Monte carlo simulation of stoquastic hamiltonians, 2015.
- [16] Sergey Bravyi, David P. DiVincenzo, Roberto I. Oliveira, and Barbara M. Terhal. The complexity of stoquastic local hamiltonian problems, 2007.
- [17] Antoine Browaeys and Thierry Lahaye. Many-body physics with individually controlled rydberg atoms. *Nature Physics*, 16(2):132–142, February 2020.
- [18] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

- [19] Marin Bukov, Markus Schmitt, and Maxime Dupont. Learning the ground state of a non-stoquastic quantum hamiltonian in a rugged neural network landscape. *SciPost Physics*, 10(6), June 2021.
- [20] Zi Cai and Jinguo Liu. Approximating quantum many-body wave functions using artificial neural networks. *Phys. Rev. B*, 97:035116, January 2018.
- [21] Giuseppe Carleo and Matthias Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, February 2017.
- [22] J. Carrasquilla and G. Torlai. How to use neural networks to investigate quantum many-body physics. *PRX Quantum*, 2:040201, November 2021.
- [23] J. Carrasquilla, G. Torlai, R. G. Melko, and L. Aolita. Reconstructing quantum states with generative models. *Nat. Mach. Intell.*, 1(3):155–161, March 2019.
- [24] Juan Carrasquilla. Machine learning for quantum matter. *Advances in Physics: X*, 5(1):1797528, January 2020.
- [25] Juan Carrasquilla, Di Luo, Felipe Pérez, Ashley Milsted, Bryan K. Clark, Maksims Volkovs, and Leandro Aolita. Probabilistic simulation of quantum circuits using a deep-learning architecture. *Physical Review A*, 104(3):032610, September 2021.
- [26] Juan Carrasquilla and Roger G. Melko. Machine learning phases of matter. *Nature Physics*, 13(5):431–434, May 2017.
- [27] Corneel Casert, Tom Vieijra, Stephen Whitelam, and Isaac Tamblyn. Dynamical large deviations of two-dimensional kinetically constrained models using a neural-network state ansatz. *Phys. Rev. Lett.*, 127:120602, September 2021.
- [28] Tommaso Castellani and Andrea Cavagna. Spin-glass theory for pedestrians. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(05):P05012, May 2005.
- [29] P. Cha, P. Ginsparg, F. Wu, J. Carrasquilla, P. L. McMahon, and E.-A. Kim. Attention-based quantum tomography, machine learning: Science and technology 3. *01LT01*, 2021.
- [30] Peter Cha, Paul Ginsparg, Felix Wu, Juan Carrasquilla, Peter L. McMahon, and Eun-Ah Kim. Attention-based quantum tomography. *Machine Learning: Science and Technology*, 3(1):01LT01, March 2022.

- [31] Natalia Chepiga and Frédéric Mila. Lifshitz point at commensurate melting of chains of rydberg atoms. *Phys. Rev. Research*, 3:023049, April 2021.
- [32] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (Association for Computational Linguistics, Doha, Qatar)*, pages 1724–1734, 2014.
- [33] Kenny Choo, Antonio Mezzacapo, and Giuseppe Carleo. Fermionic neural-network states for ab-initio electronic structure. *Nature Communications*, 11:2368, 2020.
- [34] I. Cong, S.-T. Wang, H. Levine, A. Keesling, and M. D. Lukin. Hardware-efficient, fault-tolerant quantum computation with rydberg atoms. [quant-ph], 2021.
- [35] M. Cramer, M. B. Plenio, S. T. Flammia, R. Somma, D. Gross, S. D. Bartlett, O. Landon-Cardinal, D. Poulin, and Y.-K. Liu. Efficient quantum state tomography. *Nat. Commun.*, 1:149, 2010.
- [36] Stefanie Czischek, Martin Gärttner, and Thomas Gasenzer. Quenches near ising quantum criticality as a challenge for artificial neural networks. *Phys. Rev. B*, 98:024311, July 2018.
- [37] Stefanie Czischek, M. Schuyler Moss, Matthew Radzihovsky, Ejaaz Merali, and Roger G. Melko. Data-enhanced variational monte carlo simulations for rydberg atom arrays. *Physical Review B*, 105(20):205108, May 2022.
- [38] S. de Léséleuc, V. Lienhard, P. Scholl, D. Barredo, S. Weber, N. Lang, H. P. Büchler, T. Lahaye, and A. Browaeys. Observation of a symmetry-protected topological phase of interacting bosons with rydberg atoms. *Science*, 365(6455):775–780, 2019.
- [39] Daniel Ohl de Mello, Dominik Schäffner, Jan Werkmann, Tilman Preuschoff, Lars Kohfahl, Malte Schlosser, and Gerhard Birkel. Defect-free assembly of 2d clusters of more than 100 single-atom quantum systems. *Phys. Rev. Lett.*, 122(20):203601, May 2019.
- [40] I. J. S. De Vlucht, D. Iouchtchenko, E. Merali, P.-N. Roy, and R. G. Melko. Reconstructing quantum molecular rotor ground states. *Phys. Rev. B*, 102(3):035108, 2020.

- [41] Isaac De Vlugt. Data-driven and hamiltonian-driven quantum simulations with generative models. Master’s thesis, University of Waterloo, January 2022. Publisher: University of Waterloo.
- [42] Dong-Ling Deng, Xiaopeng Li, and S. Das Sarma. Machine learning topological states. *Physical Review B*, 96(19), November 2017.
- [43] Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. Topological quantum memory. *Journal of Mathematical Physics*, 43(9):4452–4505, September 2002.
- [44] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.
- [45] Yi-Ming Ding, Yin Tang, Zhe Wang, Zhiyan Wang, Bin-Bin Mao, and Zheng Yan. Tracking the variation of entanglement rényi negativity: an efficient quantum monte carlo method, 2024.
- [46] Ansgar Dorneich and Matthias Troyer. Accessing the dynamics of large many-particle systems using the stochastic series expansion. *Phys. Rev. E*, 64:066701, November 2001.
- [47] Feyza Duman Keles, Pruthuvi Mahesakya Wijewardena, and Chinmay Hegde. On the computational complexity of self-attention. In Shipra Agrawal and Francesco Orabona, editors, *Proceedings of The 34th International Conference on Algorithmic Learning Theory*, volume 201 of *Proceedings of Machine Learning Research*, pages 597–619. PMLR, 20 Feb–23 Feb 2023.
- [48] Maxime Dupont, Snir Gazit, and Thomas Scaffidi. From trivial to topological paramagnets: The case of z_2 and z_{23} symmetries in two dimensions. *Physical Review B*, 103(14), April 2021.
- [49] S. Ebadi, A. Keesling, M. Cain, T. T. Wang, H. Levine, D. Bluvstein, G. Semeghini, A. Omran, J.-G. Liu, R. Samajdar, X.-Z. Luo, B. Nash, X. Gao, B. Barak, E. Farhi, S. Sachdev, N. Gemelke, L. Zhou, S. Choi, H. Pichler, S.-T. Wang, M. Greiner, V. Vuletić, and M. D. Lukin. Quantum optimization of maximum independent set using rydberg atom arrays. *Science*, 376(6598):1209–1215, 2022.
- [50] Sepehr Ebadi, Tout T. Wang, Harry Levine, Alexander Keesling, Giulia Semeghini, Ahmed Omran, Dolev Bluvstein, Rhine Samajdar, Hannes Pichler, Wen Wei Ho, Soonwon Choi, Subir Sachdev, Markus Greiner, Vladan Vuletić, and Mikhail D.

- Lukin. Quantum phases of matter on a 256-atom programmable quantum simulator. *Nature*, 595(7866):227–232, July 2021.
- [51] S F Edwards and P W Anderson. Theory of spin glasses. *Journal of Physics F: Metal Physics*, 5(5):965, May 1975.
- [52] Salaheddine El Adlouni, Anne-Catherine Favre, and Bernard Bobée. Comparison of methodologies to assess the convergence of markov chain monte carlo methods. *Computational Statistics & Data Analysis*, 50(10):2685–2701, 2006.
- [53] M. Endres, H. Bernien, A. Keesling, H. Levine, E. R. Anschuetz, A. Krajenbrink, C. Senko, V. Vuletic, M. Greiner, and M. D. Lukin. Atom-by-atom assembly of defect-free one-dimensional cold atom arrays. *Science*, 354:1024, 2016.
- [54] Manuel Endres, Hannes Bernien, Alexander Keesling, Harry Levine, Eric R. Anschuetz, Alexandre Krajenbrink, Crystal Senko, Vladan Vuletic, Markus Greiner, and Mikhail D. Lukin. Atom-by-atom assembly of defect-free one-dimensional cold atom arrays. *Science*, 354(6315):1024–1027, November 2016.
- [55] H. G. Evertz. The loop algorithm. *Advances in Physics*, 52(1):1–66, January 2003.
- [56] Hans Gerd Evertz, Gideon Lana, and Mihai Marcu. Cluster algorithm for vertex models. *Phys. Rev. Lett.*, 70:875–879, February 1993.
- [57] Hans Gerd Evertz and Mihai Marcu. The loop-cluster algorithm for the case of the 6 vertex model. *Nuclear Physics B - Proceedings Supplements*, 30:277–280, 1993. Proceedings of the International Symposium on.
- [58] Paul Fendley, K. Sengupta, and Subir Sachdev. Competing density-wave orders in a one-dimensional hard-boson model. *Phys. Rev. B*, 69(7):075106, February 2004.
- [59] F. Ferrari, F. Becca, and J. Carrasquilla. Neural gutzwiller-projected variational wave functions. *Phys. Rev. B*, 100:125131, 2019.
- [60] Matthew Fishman, Steven White, and Edwin Stoudenmire. The itensor software library for tensor network calculations. *SciPost Physics Codebases*, August 2022.
- [61] David Fitzek and Yi Hong Teoh. RydbergGPT, 2024. Code repository available at: <https://github.com/PIQuIL/RydbergGPT/tree/main>.
- [62] David Fitzek, Yi Hong Teoh, Hin Pok Fung, Gebremedhin A. Dagnew, Ejaaz Merali, M. Schuyler Moss, Benjamin MacLellan, and Roger G. Melko. RydbergGPT, 2024.

- [63] Steven T. Flammia, Alioscia Hamma, Taylor L. Hughes, and Xiao-Gang Wen. Topological entanglement rényi entropy and reduced density matrix structure. *Physical Review Letters*, 103(26), December 2009.
- [64] Emilio Flores-Sola, Martin Weigel, Ralph Kenna, and Bertrand Berche. Cluster monte carlo and dynamical scaling for long-range interactions. *The European Physical Journal Special Topics*, 226(4):581–594, April 2017.
- [65] Kouki Fukui and Synge Todo. Order-n cluster monte carlo method for spin systems with long-range interactions. *Journal of Computational Physics*, 228(7):2629–2642, April 2009.
- [66] Shunsuke Furukawa and Grégoire Misguich. Topological entanglement entropy in the quantum dimer model on the triangular lattice. *Physical Review B*, 75(21), June 2007.
- [67] Xun Gao and Lu-Ming Duan. Efficient representation of quantum many-body states with deep neural networks. *Nature Communications*, 8(1):662, September 2017.
- [68] Giuliano Giudici, Mikhail D Lukin, and Hannes Pichler. Dynamical preparation of quantum spin liquids in rydberg atom arrays, 2022.
- [69] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In Y. W. Teh and M. Titterington (pmlr, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research, Vol. 9*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Sardinia, Italy, 2010, 13–15 May 2010. Chia Laguna Resort.
- [70] Shou-Shu Gong, Wei Zhu, D. N. Sheng, Olexei I. Motrunich, and Matthew P. A. Fisher. Plaquette ordered phase and quantum phase diagram in the spin- $\frac{1}{2}$ j_1 - J_2 square heisenberg model. *Phys. Rev. Lett.*, 113:027201, July 2014.
- [71] Roberto Gozalo-Brizuela and Eduardo C. Garrido-Merchan. Chatgpt is not all you need. a state of the art review of large generative ai models, 2023.
- [72] Alex Graves. Supervised sequence labelling. In *Supervised sequence labelling with recurrent neural networks*, pages 5–13. Springer, 2012.
- [73] James Gubernatis, Naoki Kawashima, and Philipp Werner. *Quantum Monte Carlo Methods: Algorithms for Lattice Models*. Cambridge University Press, 2016.

- [74] Lalit Gupta and Itay Hen. Elucidating the interplay between non-stoquasticity and the sign problem. *Advanced Quantum Technologies*, 3(1):1900108, 2020.
- [75] Alioscia Hamma, Radu Ionicioiu, and Paolo Zanardi. Bipartite entanglement and entropic boundary law in lattice spin systems. *Physical Review A*, 71(2), February 2005.
- [76] Alioscia Hamma, Radu Ionicioiu, and Paolo Zanardi. Ground state entanglement and geometric entropy in the kitaev model. *Physics Letters A*, 337(1-2):22–28, March 2005.
- [77] D. C. Handscomb. The monte carlo method in quantum statistical mechanics. *Mathematical Proceedings of the Cambridge Philosophical Society*, 58(4):594–598, 1962.
- [78] D. C. Handscomb. A monte carlo method applied to the heisenberg ferromagnet. *Mathematical Proceedings of the Cambridge Philosophical Society*, 60(1):115–122, January 1964.
- [79] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with numpy. *Nature*, 585(7825):357–362, September 2020.
- [80] Matthew B. Hastings, Iván González, Ann B. Kallin, and Roger G. Melko. Measuring renyi entanglement entropy in quantum monte carlo simulations. *Physical Review Letters*, 104(15):157201, April 2010.
- [81] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.
- [82] Douglas Hendry, Hongwei Chen, and Adrian Feiguin. Neural network representation for minimally entangled typical thermal states. *Phys. Rev. B*, 106:165111, October 2022.
- [83] Patrik Henelius and Anders W. Sandvik. Sign problem in monte carlo simulations of frustrated quantum spin systems. *Phys. Rev. B*, 62(2):1102–1113, July 2000.

- [84] Loïc Henriët, Lucas Beguin, Adrien Signoles, Thierry Lahaye, Antoine Browaeys, Georges-Olivier Reymond, and Christophe Jurczak. Quantum computing with neutral atoms. *Quantum*, 4:327, September 2020. Publisher: Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften.
- [85] Jan Hermann, Zeno Schätzle, and Frank Noé. Deep-neural-network solution of the electronic schrödinger equation. *Nature Chemistry*, 12(10):891–897, September 2020.
- [86] Mohamed Hibat-Allah, Martin Ganahl, Lauren E. Hayward, Roger G. Melko, and Juan Carrasquilla. Recurrent neural network wave functions. *Physical Review Research*, 2(2):023358, June 2020.
- [87] Mohamed Hibat-Allah, Estelle M. Inack, Roeland Wiersema, Roger G. Melko, and Juan Carrasquilla. Variational neural annealing. *Nature Machine Intelligence*, 3(11):952–961, October 2021.
- [88] Mohamed Hibat-Allah, Roger G. Melko, and Juan Carrasquilla. Investigating topological order using recurrent neural networks. *Phys. Rev. B*, 108:075152, August 2023.
- [89] Mohamed Hibat-Allah, Roger G. Melko, and Juan Carrasquilla. Supplementing recurrent neural network wave functions with symmetry and annealing to improve accuracy, 2024.
- [90] Mohamed Hibat-Allah, Ejaaz Merali, Giacomo Torlai, Roger G Melko, and Juan Carrasquilla. Recurrent neural network wave functions for rydberg atom arrays on kagome lattice, 2024.
- [91] J. E. Hirsch, R. L. Sugar, D. J. Scalapino, and R. Blankenbecler. Monte carlo simulations of one-dimensional fermion systems. *Phys. Rev. B*, 26:5033–5055, November 1982.
- [92] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.
- [93] Stephen Inglis. *Implementations and applications of Renyi entanglement in Monte Carlo simulations of spin models*. PhD thesis, University of Waterloo, September 2013. Publisher: University of Waterloo.
- [94] Stephen Inglis and Roger G. Melko. Wang-landau method for calculating renyi entropies in finite-temperature quantum monte carlo simulations. *Phys. Rev. E*, 87(1):013306, January 2013.

- [95] S. V. Isakov, Yong Baek Kim, and A. Paramekanti. Spin-liquid phase in a spin-1/2 quantum magnet on the kagome lattice. *Phys. Rev. Lett.*, 97:207204, November 2006.
- [96] Sergei V. Isakov, Matthew B. Hastings, and Roger G. Melko. Topological entanglement entropy of a bose–hubbard spin liquid. *Nature Physics*, 7(10):772–775, July 2011.
- [97] Pierre E. Jacob, John O’Leary, and Yves F. Atchadé. Unbiased markov chain monte carlo methods with couplings. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 82(3):543–600, 05 2020.
- [98] D. Jaksch, J. I. Cirac, P. Zoller, S. L. Rolston, R. Côté, and M. D. Lukin. Fast quantum gates for neutral atoms. *Phys. Rev. Lett.*, 85(10):2208–2211, September 2000.
- [99] Hong-Chen Jiang, Zhenghan Wang, and Leon Balents. Identifying topological order by entanglement entropy. *Nature Physics*, 8(12):902–905, November 2012.
- [100] M. Kalinowski, R. Samajdar, R. G. Melko, M. D. Lukin, S. Sachdev, and S. Choi. Bulk and boundary quantum phase transitions in a square rydberg atom array. *Phys. Rev. B*, 105:174417, May 2021.
- [101] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.
- [102] Ribhu K. Kaul, Roger G. Melko, and Anders W. Sandvik. Bridging lattice-scale physics and continuum field theory with quantum monte carlo simulations. *Annual Review of Condensed Matter Physics*, 4(1):179–215, 2013.
- [103] Naoki Kawashima and Kenji Harada. Recent developments of world-line monte carlo methods. *J. Phys. Soc. Jpn.*, 73(6):1379–1414, June 2004. [_eprint: https://doi.org/10.1143/JPSJ.73.1379](https://doi.org/10.1143/JPSJ.73.1379).
- [104] Alexander Keesling, Ahmed Omran, Harry Levine, Hannes Bernien, Hannes Pichler, Soonwon Choi, Rhine Samajdar, Sylvain Schwartz, Pietro Silvi, Subir Sachdev, Peter Zoller, Manuel Endres, Markus Greiner, Vladan Vuletić, and Mikhail D. Lukin. Quantum kibble–zurek mechanism and critical dynamics on a programmable rydberg simulator. *Nature*, 568(7751):207–211, April 2019.

- [105] Shoummo Ahsan Khandoker, Jawaril Munshad Abedin, and Mohamed Hibat-Allah. Supplementing recurrent neural networks with annealing to solve combinatorial optimization problems. *Machine Learning: Science and Technology*, 4(1):015026, February 2023.
- [106] Mária Kieferová and Nathan Wiebe. Tomography and generative training with quantum boltzmann machines. *Phys. Rev. A*, 96(6):062327, December 2017. tex.numpages: 13.
- [107] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.
- [108] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017.
- [109] Alexei Kitaev. Anyons in an exactly solved model and beyond. *Annals of Physics*, 321(1):2–111, January 2006.
- [110] Alexei Kitaev and Chris Laumann. Topological phases and quantum computation, 2009.
- [111] Alexei Kitaev and John Preskill. Topological entanglement entropy. *Phys. Rev. Lett.*, 96:110404, March 2006.
- [112] Milan Kornjača, Rhine Samajdar, Tommaso Macrì, Nathan Gemelke, Sheng-Tao Wang, and Fangli Liu. Trimer quantum spin liquid in a honeycomb array of rydberg atoms. *Communications Physics*, 6(1):358, December 2023.
- [113] Simon Kothe and Peter Kirton. Liouville space neural network representation of density matrices, 2023.
- [114] Sebastian Krinner, Nathan Lacroix, Ants Remm, Agustin Di Paolo, Elie Genois, Catherine Leroux, Christoph Hellings, Stefania Lazar, Francois Swiadek, Johannes Herrmann, Graham J. Norris, Christian Kraglund Andersen, Markus Müller, Alexandre Blais, Christopher Eichler, and Andreas Wallraff. Realizing repeated quantum error correction in a distance-three surface code. *Nature*, 605(7911):669–674, May 2022.

- [115] Bohdan Kulchytskyy. *Probing universality with entanglement entropy via quantum Monte Carlo*. PhD thesis, University of Waterloo, August 2019. Publisher: University of Waterloo.
- [116] Bohdan Kulchytskyy, C. M. Herdman, Stephen Inglis, and Roger G. Melko. Detecting goldstone modes with entanglement entropy. *Physical Review B*, 92(11):115146, September 2015.
- [117] Hannah Lange, Guillaume Bornet, Gabriel Emperauger, Cheng Chen, Thierry Lahaye, Stefan Kienle, Antoine Browaeys, and Annabelle Bohrdt. Transformer neural networks and quantum simulators: a hybrid approach for simulating strongly correlated systems. *Quantum*, 9:1675, March 2025.
- [118] Hannah Lange, Fabian Döschl, Juan Carrasquilla, and Annabelle Bohrdt. Neural network approach to quasiparticle dispersions in doped antiferromagnets, 2023.
- [119] B. P. Lanyon, C. Maier, M. Holzäpfel, T. Baumgratz, C. Hempel, P. Jurcevic, I. Dhand, A. S. Buyskikh, A. J. Daley, M. Cramer, M. B. Plenio, R. Blatt, and C. F. Roos. Efficient tomography of a quantum many-body system. *Nature Phys*, 13:1158, 2017.
- [120] Peter D. Lax. *Functional Analysis*. Wiley-Interscience, 2002.
- [121] Michael Levin and Xiao-Gang Wen. Detecting topological order in a ground state wave function. *Physical Review Letters*, 96(11), March 2006.
- [122] H. Levine, A. Keesling, G. Semeghini, A. Omran, T. T. Wang, S. Ebadi, H. Bernien, M. Greiner, V. Vuletić, H. Pichler, and M. D. Lukin. Parallel implementation of high-fidelity multiqubit gates with neutral atoms. *Phys. Rev. Lett.*, 123:170503, 2019.
- [123] Chang-Xiao Li, Sheng Yang, and Jing-Bo Xu. Quantum phases of rydberg atoms on a frustrated triangular-lattice array. *Opt. Lett.*, 47(5):1093–1096, March 2022.
- [124] Zachary C. Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning, 2015.
- [125] Fangli Liu, Seth Whitsitt, Przemyslaw Bienias, Rex Lundgren, and Alexey V. Gorshkov. Realizing and probing baryonic excitations in rydberg atom arrays. *arXiv:2007.07258*, July 2020.
- [126] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

- [127] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.
- [128] Erik Luijten and Henk W. J. Blöte. Classical critical behavior of spin models with long-range interactions. *Physical Review B*, 56(14):8945–8958, October 1997.
- [129] M. D. Lukin, M. Fleischhauer, R. Cote, L. M. Duan, D. Jaksch, J. I. Cirac, and P. Zoller. Dipole blockade and quantum information processing in mesoscopic atomic ensembles. *Phys. Rev. Lett.*, 87(3):037901, June 2001.
- [130] Di Luo, Zhuo Chen, Kaiwen Hu, Zhizhen Zhao, Vera Mikyoung Hur, and Bryan K. Clark. Gauge-invariant and anyonic-symmetric autoregressive neural network for quantum lattice models. *Phys. Rev. Res.*, 5:013216, March 2023.
- [131] Di Luo and Bryan K. Clark. Backflow transformations via neural networks for quantum many-body wave functions. *Phys. Rev. Lett.*, 122:226401, June 2019.
- [132] H. Ma, D. Dong, I. R. Petersen, C.-J. Huang, and G.-Y. Xiang. A comparative study on how neural networks enhance quantum state tomography. [quant-ph], 2021.
- [133] Vasilios I. Manousiouthakis and Michael W. Deem. Strict detailed balance is unnecessary in monte carlo simulation. *The Journal of Chemical Physics*, 110(6):2753–2756, 02 1999.
- [134] George Marsaglia, Wai Wan Tsang, and Jingbo Wang. Fast generation of discrete random variables. *Journal of Statistical Software*, 11(i03), 2004.
- [135] Roger G. Melko. Simulations of quantum xxz models on two-dimensional frustrated lattices. *J. Phys.: Condens. Matter*, 19(14):145203, April 2007.
- [136] Roger G. Melko. Stochastic series expansion quantum monte carlo. In Adolfo Avella and Ferdinando Mancini, editors, *Strongly Correlated Systems: Numerical Methods*, volume 176, pages 185–206. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
- [137] Roger G. Melko, Giuseppe Carleo, Juan Carrasquilla, and J. Ignacio Cirac. Restricted boltzmann machines in quantum physics. *Nature Physics*, 15(9):887–892, September 2019.
- [138] Roger G. Melko, Ann B. Kallin, and Matthew B. Hastings. Finite-size scaling of mutual information in monte carlo simulations: Application to the spin- $\frac{1}{2}$ xxz model. *Phys. Rev. B*, 82:100409, September 2010.

- [139] Ejaaz Merali, Isaac J. S. De Vlucht, and Roger G. Melko. Stochastic series expansion quantum monte carlo for rydberg arrays. *SciPost Phys. Core*, 7:016, 2024.
- [140] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra, Second Edition*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2023.
- [141] C. Miles, R. Samajdar, S. Ebadi, T. T. Wang, H. Pichler, S. Sachdev, M. D. Lukin, M. Greiner, K. Q. Weinberger, and E.-A. Kim. Machine learning discovery of new phases in programmable quantum simulator snapshots. *Phys. Rev. Res.*, 5:013026, January 2021.
- [142] R. Moessner and S. L. Sondhi. Ising models of quantum frustration. *Phys. Rev. B*, 63:224401, May 2001.
- [143] R. Moessner, S. L. Sondhi, and P. Chandra. Two-dimensional periodic frustrated ising models in a transverse field. *Physical Review Letters*, 84(19):4457–4460, May 2000.
- [144] C. Monroe, W. C. Campbell, L.-M. Duan, Z.-X. Gong, A. V. Gorshkov, P. W. Hess, R. Islam, K. Kim, N. M. Linke, G. Pagano, P. Richerme, C. Senko, and N. Y. Yao. Programmable quantum simulations of spin systems with trapped ions. *Rev. Mod. Phys.*, 93:025001, 2021.
- [145] Stewart Morawetz, Isaac J. S. De Vlucht, Juan Carrasquilla, and Roger G. Melko. U(1)-symmetric recurrent neural networks for quantum state reconstruction. *Physical Review A*, 104(1):012401, July 2021.
- [146] M. Schuyler Moss, Sepehr Ebadi, Tout T. Wang, Giulia Semeghini, Annabelle Bohrdt, Mikhail D. Lukin, and Roger G. Melko. Enhancing variational monte carlo simulations using a programmable quantum simulator. *Phys. Rev. A*, 109:032410, March 2024.
- [147] Kouhei Nakaji, Lasse Bjørn Kristensen, Jorge A. Campos-Gonzalez-Angulo, Mohammad Ghazi Vakili, Haozhe Huang, Mohsen Bagherimehrab, Christoph Gorgulla, FuTe Wong, Alex McCaskey, Jin-Sung Kim, Thien Nguyen, Pooja Rao, and Alan Aspuru-Guzik. The generative quantum eigensolver (gqe) and its application for ground state search, 2024.
- [148] Tota Nakamura. Efficient monte carlo algorithm in quasi-one-dimensional ising spin systems. *Phys. Rev. Lett.*, 101(21):210602, November 2008.

- [149] M. Neugebauer, L. Fischer, A. Jäger, S. Czischek, S. Jochim, M. Weidemüller, and M. Gärttner. Neural-network quantum state tomography in a two-qubit experiment. *Phys. Rev. A*, 102:042604, 2020.
- [150] M. E. J. Newman and G. T. Barkema. *Monte Carlo Methods in Statistical Physics*. Oxford University Press, 02 1999.
- [151] Minh-Thi Nguyen, Jin-Guo Liu, Jonathan Wurtz, Mikhail D. Lukin, Sheng-Tao Wang, and Hannes Pichler. Quantum optimization with arbitrary connectivity using rydberg atom arrays. *PRX Quantum*, 4(1), February 2023.
- [152] P. Nikolić and T. Senthil. Theory of the kagome lattice ising antiferromagnet in weak transverse fields. *Phys. Rev. B*, 71(2):024401, January 2005.
- [153] Yusuke Nomura. Helping restricted boltzmann machines with quantum-state representation by restoring symmetry. *Journal of Physics: Condensed Matter*, 33(17):174003, April 2021.
- [154] Yusuke Nomura, Andrew S. Darmawan, Youhei Yamaji, and Masatoshi Imada. Restricted boltzmann machine learning for solving strongly correlated quantum systems. *Phys. Rev. B*, 96:205152, November 2017.
- [155] Gaopei Pan and Zi Yang Meng. The sign problem in quantum monte carlo simulations. In Tapash Chakraborty, editor, *Encyclopedia of Condensed Matter Physics (Second Edition)*, pages 879–893. Academic Press, Oxford, second edition edition, 2024.
- [156] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.
- [157] David Pfau, James S. Spencer, Alexander G. D. G. Matthews, and W. M. C. Foulkes. Ab initio solution of the many-electron schrödinger equation with deep neural networks. *Physical Review Research*, 2(3), September 2020.
- [158] S. Pilati, E. M. Inack, and P. Pieri. Self-learning projective quantum monte carlo simulations guided by restricted boltzmann machines. *Phys. Rev. E*, 100:043301, 2019.

- [159] Christopher C.J. Potter and Robert H. Swendsen. Guaranteeing total balance in metropolis algorithm monte carlo simulations. *Physica A: Statistical Mechanics and its Applications*, 392(24):6288–6299, 2013.
- [160] Nikolay Prokof'ev and Boris Svistunov. Worm algorithms for classical statistical models. *Phys. Rev. Lett.*, 87(16):160601, September 2001.
- [161] Michael Rader and Andreas M. Läuchli. Floating phases in one-dimensional rydberg ising chains. *arXiv: 1908.02068*, 2019.
- [162] Peter M. Richards. Spin-glass order parameter of the random-field ising model. *Phys. Rev. B*, 30:2955–2957, September 1984.
- [163] Christopher Roth. Iterative retraining of quantum spin models using recurrent neural networks, 2020.
- [164] Christopher Roth, Attila Szabó, and Allan MacDonald. High-accuracy variational monte carlo for frustrated magnets with deep neural networks, 2022.
- [165] Subir Sachdev. *Quantum Phase Transitions*. Cambridge University Press, 2 edition, 2011.
- [166] Rhine Samajdar, Soonwon Choi, Hannes Pichler, Mikhail D. Lukin, and Subir Sachdev. Numerical study of the chiral Z_3 quantum phase transition in one spatial dimension. *Phys. Rev. A*, 98(2):023614, August 2018.
- [167] Rhine Samajdar, Wen Wei Ho, Hannes Pichler, Mikhail D. Lukin, and Subir Sachdev. Complex density wave orders and quantum phase transitions in a model of square-lattice rydberg atom arrays. *Physical Review Letters*, 124(10):103601, March 2020.
- [168] Rhine Samajdar, Wen Wei Ho, Hannes Pichler, Mikhail D. Lukin, and Subir Sachdev. Quantum phases of rydberg atoms on a kagome lattice. *Proceedings of the National Academy of Sciences*, 118(4):e2015785118, January 2021.
- [169] A. W. Sandvik and K. S. D. Beach. Monte carlo simulations of quantum spin systems in the valence bond basis. *arXiv:0704.1469*, April 2007.
- [170] Anders W. Sandvik. A generalization of handscomb's quantum monte carlo scheme-application to the 1d hubbard model. *Journal of Physics A: Mathematical and General*, 25(13):3667, July 1992.

- [171] Anders W. Sandvik. Finite-size scaling of the ground-state parameters of the two-dimensional heisenberg model. *Phys. Rev. B*, 56(18):11678–11690, November 1997.
- [172] Anders W. Sandvik. Stochastic series expansion method with operator-loop update. *Phys. Rev. B*, 59:R14157–R14160, June 1999.
- [173] Anders W. Sandvik. Stochastic series expansion method for quantum ising models with arbitrary interactions. *Phys. Rev. E*, 68(5):056701, November 2003.
- [174] Anders W. Sandvik. Ground state projection of quantum spin systems in the valence-bond basis. *Phys. Rev. Lett.*, 95(20):207203, November 2005.
- [175] Anders W. Sandvik. Computational studies of quantum spin systems. *arXiv:1101.3281*, pages 135–338, 2010.
- [176] Anders W. Sandvik. Stochastic series expansion methods. In Eva Pavarini, Erik Koch, and Schiwei Zhang, editors, *Many-Body Methods for Real Materials*, volume 9 of *Schriften des Forschungszentrums Jülich. Modeling and Simulation*, page getr. Zählung, Jülich, September 2019. Autumn School on Correlated Electrons, Jülich (Germany), 16 Sep 2019 - 20 Sep 2019, Forschungszentrum Jülich GmbH Zentralbibliothek, Verlag.
- [177] Anders W. Sandvik and Hans Gerd Evertz. Loop updates for variational and projector quantum monte carlo simulations in the valence-bond basis. *Phys. Rev. B*, 82(2):024407, July 2010.
- [178] Anders W. Sandvik and Juhani Kurkijärvi. Quantum monte carlo simulation method for spin systems. *Phys. Rev. B*, 43(7):5950–5961, March 1991.
- [179] Anders W. Sandvik and Olav F. Syljuåsen. The directed-loop algorithm. *AIP Conference Proceedings*, 690(1):299–308, November 2003.
- [180] K. J. Satzinger, Y.-J Liu, A. Smith, C. Knapp, M. Newman, C. Jones, Z. Chen, C. Quintana, X. Mi, A. Dunsworth, C. Gidney, I. Aleiner, F. Arute, K. Arya, J. Atalaya, R. Babbush, J. C. Bardin, R. Barends, J. Basso, A. Bengtsson, A. Bिल्mes, M. Broughton, B. B. Buckley, D. A. Buell, B. Burkett, N. Bushnell, B. Chiaro, R. Collins, W. Courtney, S. Demura, A. R. Derk, D. Eppens, C. Erickson, L. Faoro, E. Farhi, A. G. Fowler, B. Foxen, M. Giustina, A. Greene, J. A. Gross, M. P. Harrigan, S. D. Harrington, J. Hilton, S. Hong, T. Huang, W. J. Huggins, L. B. Ioffe, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, T. Khattar, S. Kim, P. V. Klimov, A. N. Korotkov, F. Kostritsa, D. Landhuis, P. Laptev, A. Locharla,

- E. Lucero, O. Martin, J. R. McClean, M. McEwen, K. C. Miao, M. Mohseni, S. Montazeri, W. Mruczkiewicz, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, T. E. O'Brien, A. Opremcak, B. Pató, A. Petukhov, N. C. Rubin, D. Sank, V. Shvarts, D. Strain, M. Szalay, B. Villalonga, T. C. White, Z. Yao, P. Yeh, J. Yoo, A. Zalcman, H. Neven, S. Boixo, A. Megrant, Y. Chen, J. Kelly, V. Smelyanskiy, A. Kitaev, M. Knap, F. Pollmann, and P. Roushan. Realizing topologically ordered states on a quantum processor. *Science*, 374(6572):1237–1241, December 2021.
- [181] P. Scholl, M. Schuler, H. Williams, A. Eberharter, D. Barredo, K.-N. Schymik, V. Lienhard, L.-P. Henry, T. Lang, T. Lahaye, A. Läuchli, and A. Browaeys. Quantum simulation of 2d antiferromagnets with hundreds of rydberg atoms. *Nature*, 595:233, 2021.
- [182] Ulrich Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, 326(1):96–192, January 2011.
- [183] Raoul D. Schram and Gerard T. Barkema. Monte carlo methods beyond detailed balance. *Physica A: Statistical Mechanics and its Applications*, 418:88–93, 2015. Proceedings of the 13th International Summer School on Fundamental Problems in Statistical Physics.
- [184] David Schwandt, Fabien Alet, and Sylvain Capponi. Quantum monte carlo simulations of fidelity at magnetic quantum phase transitions. *Phys. Rev. Lett.*, 103:170501, October 2009.
- [185] Keith Schwarz. Darts, dice, and coins: Sampling from a discrete distribution, December 2011.
- [186] Dan Sehayek, Anna Golubeva, Michael S. Albergro, Bohdan Kulchytskyy, Giacomo Torlai, and Roger G. Melko. Learnability scaling of quantum states: Restricted boltzmann machines. *Phys. Rev. B*, 100(19):195125, November 2019.
- [187] G. Semeghini, H. Levine, A. Keesling, S. Ebadi, T. T. Wang, D. Bluvstein, R. Verresen, H. Pichler, M. Kalinowski, R. Samajdar, A. Omran, S. Sachdev, A. Vishwanath, M. Greiner, V. Vuletić, and M. D. Lukin. Probing topological spin liquids on a programmable quantum simulator. *Science*, 374(6572):1242–1247, December 2021.
- [188] Huitao Shen. Mutual information scaling and expressive power of sequence models, 2019.

- [189] V. V. Sivak, A. Eickbusch, B. Royer, S. Singh, I. Tsioutsios, S. Ganjam, A. Miano, B. L. Brock, A. Z. Ding, L. Frunzio, S. M. Girvin, R. J. Schoelkopf, and M. H. Devoret. Real-time quantum error correction beyond break-even. *Nature*, 616(7955):50–55, April 2023.
- [190] Kyle Sprague and Stefanie Czischek. Variational monte carlo with large patched transformers. *Communications Physics*, 7(1):90, March 2024.
- [191] E.M. Stoudenmire and Steven R. White. Studying two-dimensional systems with the density matrix renormalization group. *Annual Review of Condensed Matter Physics*, 3(1):111–128, March 2012.
- [192] Hidemaro Suwa. *Geometrically Constructed Markov Chain Monte Carlo Study of Quantum Spin-phonon Complex Systems*. Springer Japan, Tokyo, 2014.
- [193] Masuo Suzuki, Seiji Miyashita, and Akira Kuroda. Monte carlo simulation of quantum spin systems. i. *Progress of Theoretical Physics*, 58(5):1377–1387, November 1977.
- [194] Olav F. Syljuåsen and Anders W. Sandvik. Quantum monte carlo with directed loops. *Phys. Rev. E*, 66(4):046701, October 2002.
- [195] G. Torlai, G. Mazzola, G. Carleo, and A. Mezzacapo. Precise measurement of quantum observables with neural-network estimators. *Phys. Rev. Research*, 2:022060, 2020. R.
- [196] G. Torlai and R. G. Melko. Learning thermodynamics with boltzmann machines. *Phys. Rev. B*, 94:165134, 2016.
- [197] G. Torlai and R. G. Melko. Latent space purification via neural density operators. *Phys. Rev. Lett.*, 120:240503, 2018.
- [198] G. Torlai and R. G. Melko. Machine-learning quantum states in the nisq era. *Annual Review of Condensed Matter Physics*, 11:325, 2020.
- [199] Giacomo Torlai and Matthew Fishman. PastaQ: A package for simulation, tomography and analysis of quantum computers, 2020.
- [200] Giacomo Torlai, Guglielmo Mazzola, Juan Carrasquilla, Matthias Troyer, Roger Melko, and Giuseppe Carleo. Neural-network quantum state tomography. *Nature Physics*, 14(5):447–450, May 2018.

- [201] Giacomo Torlai, Brian Timar, Evert P. L. van Nieuwenburg, Harry Levine, Ahmed Omran, Alexander Keesling, Hannes Bernien, Markus Greiner, Vladan Vuletić, Mikhail D. Lukin, Roger G. Melko, and Manuel Endres. Integrating neural networks with a quantum simulator for state reconstruction. *Phys. Rev. Lett.*, 123:230504, Dec 2019.
- [202] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, February 2023.
- [203] Matthias Troyer and Uwe-Jens Wiese. Computational complexity and fundamental limitations to fermionic quantum monte carlo simulations. *Phys. Rev. Lett.*, 94:170201, May 2005.
- [204] C. J. Umrigar, J. Toulouse, C. Filippi, S. Sorella, and R. G. Hennig. Alleviation of the fermion-sign problem by optimization of many-body wave functions. *Phys. Rev. Lett.*, 99:179902, 2007. E.
- [205] A. Valenti, E. Greplova, N. H. Lindner, and S. D. Huber. Correlation-enhanced neural networks as interpretable variational quantum states. *Phys. Rev. Research*, 4, 2022. Article L012010.
- [206] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30:5999–6009, 2017.
- [207] R. Verresen, M. D. Lukin, and A. Vishwanath. Prediction of toric code topological order from rydberg blockade. *Phys. Rev. X*, 11:031005, 2021.
- [208] Ruben Verresen, Mikhail D. Lukin, and Ashvin Vishwanath. Prediction of toric code topological order from rydberg blockade. *Physical Review X*, 11(3), July 2021.
- [209] F. Verstraete, M. M. Wolf, D. Perez-Garcia, and J. I. Cirac. Criticality, the area law, and the computational power of projected entangled pair states. *Phys. Rev. Lett.*, 96:220601, June 2006.
- [210] Luciano Loris Viteritti, Francesco Ferrari, and Federico Becca. Accuracy of restricted boltzmann machines for the one-dimensional $j_1 - j_2$ heisenberg model. *SciPost Phys.*, 12:166, 2022.

- [211] Luciano Loris Viteritti, Riccardo Rende, and Federico Becca. Transformer variational wave functions for frustrated quantum spin systems. *Phys. Rev. Lett.*, 130:236401, June 2023.
- [212] Luciano Loris Viteritti, Riccardo Rende, Alberto Parola, Sebastian Goldt, and Federico Becca. Transformer wave function for the shastry-sutherland model: emergence of a spin-liquid phase, 2024.
- [213] K. Vogel and H. Risken. Determination of quasiprobability distributions in terms of probability distributions for the rotated quadrature phase. *Phys. Rev. A*, 40:2847, 1989.
- [214] Michael D. Vose. A linear algorithm for generating random numbers with a given distribution. *IEEE Trans. Softw. Eng.*, 17(9):972–975, September 1991.
- [215] Alastair J. Walker. New fast method for generating discrete random numbers with arbitrary frequency distributions. *Electronics Letters*, 10:127–128, May 1974.
- [216] Alastair J. Walker. An efficient method for generating discrete random variables with general distributions. *ACM Trans. Math. Softw.*, 3(3):253–256, September 1977.
- [217] Markus Wallerberger. Efficient estimation of autocorrelation spectra. *arXiv preprint arXiv:1810.05079*, 2018.
- [218] T. Walter, P. Kurpiers, S. Gasparinetti, P. Magnard, A. Potočnik, Y. Salathé, M. Pechal, M. Mondal, M. Oppliger, C. Eichler, and A. Wallraff. Rapid high-fidelity single-shot dispersive readout of superconducting qubits. *Phys. Rev. Appl.*, 7:054020, May 2017.
- [219] Haoxiang Wang, Maurice Weber, Josh Izaac, and Cedric Yen-Yu Lin. Predicting properties of quantum systems with conditional generative models, November 2022.
- [220] Lei Wang, Ye-Hua Liu, Jakub Imriška, Ping Nang Ma, and Matthias Troyer. Fidelity susceptibility made simple: A unified quantum monte carlo approach. *Phys. Rev. X*, 5:031007, July 2015.
- [221] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Trans. Graph.*, 38(5), October 2019.
- [222] Zhaoyou Wang and Emily J. Davis. Calculating rényi entropies with neural autoregressive quantum states. *Physical Review A*, 102(6), December 2020.

- [223] T. Westerhout, N. Astrakhantsev, K. S. Tikhonov, M. I. Katsnelson, and A. A. Bagrov. Generalization properties of neural network approximations to frustrated magnet ground states. *Nat. Commun.*, 11:1593, 2020.
- [224] Steven R. White. Density matrix formulation for quantum renormalization groups. *Phys. Rev. Lett.*, 69:2863–2866, November 1992.
- [225] Steven R. White. Density-matrix algorithms for quantum renormalization groups. *Phys. Rev. B*, 48(14):10345–10356, October 1993.
- [226] Steven R. White. Minimally entangled typical quantum states at finite temperature. *Phys. Rev. Lett.*, 102:190601, May 2009.
- [227] Seth Whitsitt, Rhine Samajdar, and Subir Sachdev. Quantum field theory for the chiral clock transition in one spatial dimension. *Phys. Rev. B*, 98(20):205118, November 2018.
- [228] Julia Wildeboer, Alexander Seidel, and Roger G. Melko. Entanglement entropy and topological order in resonating valence-bond quantum spin liquids. *Physical Review B*, 95(10), March 2017.
- [229] Dian Wu, Riccardo Rossi, Filippo Vicentini, and Giuseppe Carleo. From tensor network quantum states to tensorial recurrent neural networks, 2022.
- [230] Kai-Hsin Wu, Tsung-Cheng Lu, Chia-Min Chung, Ying-Jer Kao, and Tarun Grover. Entanglement renyi negativity across a finite temperature transition: A monte carlo study. *Phys. Rev. Lett.*, 125:140603, September 2020.
- [231] Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. Bloomberggpt: A large language model for finance, March 2023.
- [232] Xiaoling Wu, Xinhui Liang, Yaoqi Tian, Fan Yang, Cheng Chen, Yong-Chun Liu, Meng Khoon Tey, and Li You. A concise review of rydberg atom based quantum computation and quantum simulation. *Chinese Physics B*, 30(2):020305, February 2021.
- [233] Jonathan Wurtz, Alexei Bylinskii, Boris Braverman, Jesse Amato-Grill, Sergio H. Cantu, Florian Huber, Alexander Lukin, Fangli Liu, Phillip Weinberg, John Long, Sheng-Tao Wang, Nathan Gemelke, and Alexander Keesling. Aquila: Quera’s 256-qubit neutral-atom quantum computer, 2023.

- [234] Wenchao Xu, Aditya V. Venkatramani, Sergio H. Cantú, Tamara Šumarac, Valentin Klüsener, Mikhail D. Lukin, and Vladan Vuletić. Fast preparation and detection of a rydberg qubit using atomic ensembles. *Phys. Rev. Lett.*, 127(5):050501, July 2021.
- [235] Zheng Yan, Rhine Samajdar, Yan-Cheng Wang, Subir Sachdev, and Zi Yang Meng. Triangular lattice quantum dimer model with variable dimer density. *Nature Communications*, 13(1):5799, October 2022.
- [236] Zheng Yan, Yan-Cheng Wang, Rhine Samajdar, Subir Sachdev, and Zi Yang Meng. Emergent glassy behavior in a kagome rydberg atom array. *Phys. Rev. Lett.*, 130:206501, May 2023.
- [237] Sheng Yang and Jing-Bo Xu. Density-wave-ordered phases of rydberg atoms on a honeycomb lattice. *Phys. Rev. E*, 106:034121, September 2022.
- [238] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.
- [239] Peter Young. Jackknife and bootstrap resampling methods in statistical analysis to correct for bias.
- [240] Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J. Reddi, and Sanjiv Kumar. Are transformers universal approximators of sequence-to-sequence functions?, February 2020.
- [241] Chaoning Zhang, Chenshuang Zhang, Sheng Zheng, Yu Qiao, Chenghao Li, Mengchun Zhang, Sumit Kumar Dam, Chu Myaet Thwal, Ye Lin Tun, Le Luang Huy, Donguk kim, Sung-Ho Bae, Lik-Hang Lee, Yang Yang, Heng Tao Shen, In So Kweon, and Choong Seon Hong. A complete survey on generative ai (aigc): Is chatgpt from gpt-4 to gpt-5 all you need?, 2023.
- [242] Yi Zhang, Tarun Grover, Ari Turner, Masaki Oshikawa, and Ashvin Vishwanath. Quasiparticle statistics and braiding from ground-state entanglement. *Physical Review B*, 85(23), June 2012.
- [243] Yuan-Hang Zhang and Massimiliano Di Ventra. Transformer quantum state: A multipurpose model for quantum many-body problems. *Physical Review B*, 107(7), February 2023.
- [244] Guo-Bing Zhou, Jianxin Wu, Chen-Lin Zhang, and Zhi-Hua Zhou. Minimal gated unit for recurrent neural networks, 2016.

APPENDICES

Appendix A

Super-Multiplicativity of the Metropolis Function

In this Appendix we will prove the Super-Multiplicativity property of the Metropolis Function $m(a) = \min(1, a)$ used in the main text.

Theorem A.1 (Super-Multiplicativity). *Let $a, b \in \mathbb{R}_{\geq 0}$, the Metropolis function of the product ab is always greater than or equal to the product $m(a)m(b)$.*

$$m(ab) \geq m(a)m(b) \tag{A.1}$$

with equality when either a and b are both less than one, both greater than one, or at least one of them is equal to zero or one.

Proof. There are four cases to handle for this proof: (I) $a, b \leq 1$, (II) $a \leq 1, b > 1$, (III) $a > 1, b \leq 1$, and (IV) $a, b > 1$.

For case (I), we have $a, b \leq 1$ hence $ab \leq 1$, and the Metropolis function becomes multiplicative:

$$m(ab) = \min(1, ab) = ab = \min(1, a) \min(1, b) = m(a)m(b) \tag{A.2}$$

Cases (II) and (III) are equivalent, so we will only handle the former. Since $b > 1$, $ab \geq a$ (with equality when $a = 0$) and $m(b) = 1$, so we have:

$$ab \geq a \tag{A.3}$$

$$\min(1, ab) \geq \min(1, a) \tag{A.4}$$

$$m(ab) \geq m(a) \tag{A.5}$$

$$m(ab) \geq m(a)m(b) \tag{A.6}$$

with equality when either $a = 0$ or $a = 1$ (or $b = 0$ or $b = 1$ in case (III)).

Lastly, for case (IV) we have $a, b > 1$, which implies $ab > 1$ and $m(a) = m(b) = 1$, giving:

$$m(ab) = 1 = m(a)m(b) \tag{A.7}$$

which completes the proof. □