Critical behavior for the model of random spatial permutations

by

John Kerl

A Dissertation Submitted to the Faculty of the

Department of Mathematics

In Partial Fulfillment of the Requirements
For the Degree of

Doctor of Philosophy

In the Graduate College

The University of Arizona

2 0 1 0

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

As members of the Dissertation Committee, we certify that we have read the dissertation prepared by John Kerl entitled
Critical behavior for the model of random spatial permutations
and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

______________________________ Date: March 22, 2010
Thomas Kennedy

______________________________ Date: March 22, 2010
Rabindra Bhattacharya

______________________________ Date: March 22, 2010
Kevin Lin

______________________________ Date: March 22, 2010
Daniel Ueltschi

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

______________________________ Date: March 22, 2010
Thomas Kennedy

Statement by Author

Dedication

For Steph, who insisted.

Acknowledgments

First and foremost, thanks go to co-advisors Daniel Ueltschi and Tom Kennedy. Much of what I have learned in this project has been learned through them; their patience has been endless and their knowledge has been irreplaceable.

My work in spring 2008 and spring 2009 was supported by research assistantships through National Science Foundation grant DMS-0601075. Summer 2009, fall 2009, and spring 2010 research assistantships were funded by the University of Arizona Department of Mathematics NSF VIGRE (Vertical InteGration in Research and Education) grant.

The July 2008 Summer School on Current Topics in Mathematical Physics at the Erwin Schrödinger Institute in Vienna, organized by Christian Hainzl and Robert Seiringer, was my first mathematical physics conference. There, I made several valuable contacts; I also benefited from extended collaboration time with Daniel Ueltschi. The March 2009 school on Entropy and the Quantum in Tucson, organized by Daniel Ueltschi and Bob Sims, allowed further networking opportunities. In particular, Volker Betz posed a question on upper and lower bounds for Brownian-bridge interactions, and gave unexpected advice on software architecture. Bob Sims' June 2009 workshop on Quantum Spin Systems and Applications in Quantum Computation in Tucson, at which I presented ongoing results of a side project in three-dimensional percolation, gave me valuable feedback on finite-size scaling which I was able to apply to my dissertation work. Daniel Gandolfo and Jean Ruiz, of the Université de Marseille and the Centre National de la Recherche Scientifique, respectively, were kind enough to host me for three days in July 2009 before a conference in Berlin. There, Gandolfo and Ueltschi gave me useful advice on my writing. Ueltschi's challenging questions on autocorrelation led me to write appendix B. Gandolfo and I discussed computational methods including lookup-table methods for the true Bose-gas interactions and the inapplicability of fourth-order-cumulant methods to our problem. The July 2009 conference on Stochastic Processes and Their Applications in Berlin allowed me to present my work to a probabilist audience. The February 2010 workshop of the Center for Simulational Physics at the University of Georgia provided invaluable contacts for me, including key ideas for presentation graphics due to Joan Adler of Technion, and a likely solution to the band-update problem due to Friederike Schmid of the University of Mainz.

Two reviewers for the Journal of Statistical Physics provided expert reassurance regarding the correctness of the methodology presented in an abbreviated version of this dissertation, while also helping to clarify the exposition. My committee members — Tom, Daniel, Kevin, and Rabi — likewise read this dissertation carefully and offered thoughtful suggestions for improvement.

Small-scale simulational processing for this project was done on the 8-processor chivo cluster at the University of Arizona Department of Mathematics. Large-scale processing was done on the massively parallel ICE cluster at the University of Arizona High Performance Computing center.

The University of Arizona Department of Mathematics has been a supportive, stimulating environment in which I have been fortunate to spend the last five years. After graduation I will miss this department; I will take its best practices with me.

Most importantly, I thank my wife, Sarah, for her patience and support. She and I have labored side by side as she progresses through a doctoral program in anthropology. Now, it's her turn to write a dissertation.

Table of Contents

Table of Contents—*Continued*

Table of Contents—*Continued*

Table of Contents—*Continued*

List of Figures

List of Figures—*Continued*

List of Tables

Abstract

We examine a phase transition in a model of random spatial permutations which originates in a study of the interacting Bose gas. Permutations are weighted according to point positions; the low-temperature onset of the appearance of arbitrarily long cycles is connected to the phase transition of Bose-Einstein condensates. In our simplified model, point positions are held fixed on the fully occupied cubic lattice and interactions are expressed as Ewens-type weights on cycle lengths of permutations. The critical temperature of the transition to long cycles depends on an interaction-strength parameter α. For weak interactions, the shift in critical temperature is expected to be linear in α with constant of linearity c. Using Markov chain Monte Carlo methods and finite-size scaling, we find $c = 0.618 \pm 0.086$. This finding matches a similar analytical result of Ueltschi and Betz. We also examine the mean longest cycle length as a fraction of the number of sites in long cycles, recovering an earlier result of Shepp and Lloyd for non-spatial permutations. The plan of this paper is as follows. We begin with a non-technical discussion of the historical context of the project, along with a mention of alternative approaches. Relevant previous works are cited, thus annotating the bibliography. The random-cycle approach to the BEC problem requires a model of spatial permutations. This model it is of its own probabilistic interest; it is developed mathematically, without reference to the Bose gas. Our Markov-chain Monte Carlo algorithms for sampling from the random-cycle distribution — the swap-only, swap-and-reverse, band-update, and worm algorithms — are presented, compared, and contrasted. Finite-size scaling techniques are used to obtain information about infinite-volume quantities from finite-volume computational data.

Chapter 1
Scientific context

1.1 Theory

In 1924, the physicist Satyendra Nath Bose examined the quantum statistics of photons. In 1925, collaborating with Bose, Albert Einstein realized that the same could be done with non-interacting massive particles. He also discovered the condensation phenomenon: a macroscopic occupation of the (single-particle) ground state of the external potential [LSSY]. Moreover, Einstein predicted a critical temperature for the phenomenon. This temperature was so low — at the nanokelvin scale — that Bose-Einstein condensation attracted little interest in the physics community.

Feynman in 1953, along with Penrose and Onsager in 1956 [Feynman, PO], developed the theoretical notion of long permutation cycles in the Feynman-Kac representation of the Bose gas. Feynman claimed that long cycles correspond to Bose-Einstein condensation.

András Sütő referred to the existence of long permutation cycles as cycle percolation. He proved in 1993 that BEC implies cycle percolation in the ideal (non-interacting) gas [Sütő1], and proved the converse in 2002. Sütő moreover proved in the 2002 paper that there are infinitely many macroscopic cycles in the condensation of the non-ideal Bose gas.

For the ideal Bose gas, BEC is defined as the macroscopic occupation of the single-particle ground state of the external potential. For an interacting Bose gas, Hamiltonian eigenfunctions do not factor and thus there are no single-particle ground states. BEC is carefully defined for interacting systems [LSSY] in terms of the largest eigenvalue of a density-matrix operator. The 1983 work of Buffet and Pulè [BP] examines the macroscopic occupation of the zero Fourier mode.

1.2 Experiments

Liquid helium was produced in the laboratory by Kammerlingh Onnes in 1908; Fritz London in 1938 [London] connected superfluidity of liquid helium with Bose-Einstein condensation. Atoms of liquid helium, however, are strongly interacting — they attract only weakly, due to helium being a noble gas, but there are strong repulsive effects due to the high density of the liquid. Thus, Einstein's non-interacting theory could not explain the phenomenon.

Several groups attempted during the 1990s to produce BECs in vapors of spin-polarized hydrogen, but were not able to achieve low enough temperatures. The group of Cornell and Wieman [AEMWC], using hybrid cooling methods, successfully brought rubidium atoms to well below the critical temperature and made numerous measurements on the resulting condensates. (Cornell, Wieman, and Ketterle received the 2001 Nobel prize in physics for this work.)

Interest in BECs was sparked by this experimental success: thousands of papers, both theoretical and experimental, have been published on BECs in the years since. The work of Cornell and Wieman was of interest for several reasons:

- Condensates were directly imaged. Measurements were taken of temperature, density, position, velocity, particle number, and the fraction of the condensate occupying the ground state of the 3D harmonic trapping potential.
- The method was able to vary temperature and density through wide ranges; the condensate fraction was varied from zero to 100 percent.
- The gaseous rubidium condensate was *weakly interacting* — permitting a perturbative analysis which liquid helium, with its strong interactions, did not allow. (Note in particular that recent mathematical studies are weak-interaction theories; they are valid only to first order in the scattering length.)

1.3 Critical temperature

Recall that Einstein predicted a critical temperature T_c for the ideal Bose gas. It is a long-standing question to discover the effects of scattering length a on the critical temperature. Moreover, one may fix the density $\rho_c(a)$ and obtain a critical temperature $T_c(a) = 1/\beta_c(a)$ or vice versa; both of these critical parameters depend on the scattering length a. One expects the critical combination of parameters to be a manifold in (ρ, β, a) space. (See figure 1.1.)

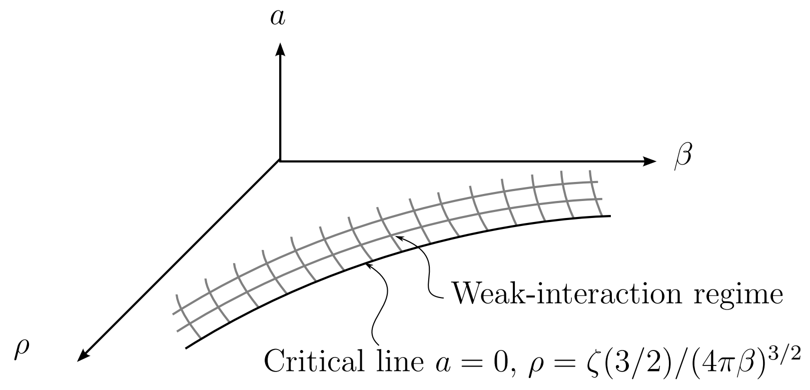

a
β
ρ
Weak-interaction regime
Critical line $a = 0$, $\rho = \zeta(3/2)/(4\pi\beta)^{3/2}$

FIGURE 1.1. Critical manifold in (ρ, β, a) for small a.

Much is known about the $a = 0$ line of this critical manifold; off $a = 0$, even the crude shape has been under debate. The following findings are described by [BBHLV]: The superfluid transition temperature of liquid helium is lower than that of an ideal gas of the same density. Thus, assuming that helium superfluidity is a strongly interacting BEC, one would expect interactions to decrease the critical temperature for the strongly interacting case. Various theoretical work (tabulated below) suggested either an increase or a decrease in critical temperature; path-integral simulations for low density (i.e. weak interactions) suggested that the critical temperature increases with scattering length, for small scattering length. The emerging consensus is that

$$\Delta T_c(a) = \frac{T_c(a) - T_c(0)}{T_c(0)}$$

is linear in a for small a, i.e.

$$\Delta T_c(a) = ca + O(a^2).$$

The following summary of the theoretical work on this question is found in [SU09]. (See also [AM, KPS, NL04] for a review on the widely varying analytical and simulational results on $T_c(a)$; see [BBHLV] for a thorough listing of the progress up to 2001 and [SU09] for an up-to-date survey.)

- 1964: *Huang*: $\Delta T_c(a) \sim (a\rho^{1/3})^{3/2}$, increases
- 1971: *Fetter & Walecka*: $\Delta T_c(a)$ decreases
- 1982: *Toyoda*: $\Delta T_c(a)$ decreases
- 1992: *Stoof*: $\Delta T_c(a) = c\, a\rho^{1/3} + o(a\rho^{1/3}), \quad c > 0$
- 1996: *Bijlsma & Stoof*: $c = 4.66$
- 1997: *Grüter, Ceperley, Laloë*: $c = 0.34$
- 1999: *Holzmann, Grüter, Laloë*: $c = 0.7$; *Holzmann, Krauth*: $c = 2.3$;
- 1999: *Baym et. al.*: $c = 2.9$
- 2000: *Reppy et. al.*: $c = 5.1$
- 2001: *Kashurnikov, Prokof'ev, Svistunov*: $c = 1.29$
- 2001: *Arnold, Moore*: $c = 1.32$
- 2004: *Kastening*: $c = 1.27$
- 2004: *Nho, Landau*: $c = 1.32$

1.4 Context of dissertation

The work of Ueltschi and Betz [BU07, U06, U07] extends the permutation point of view originated by Feynman, Penrose, and Onsager, drawing on the work of Sütő, Buffet, and Pulè [Feynman, PO, Sütő1, Sütő2, BP] to develop a model of random spatial permutations. (See chapter 2 for precise definition of the model.) In the permutation representation, this condensate transition manifests itself as the onset of long permutation cycles. The central point of this approach is that the system energy has been recast in terms of permutations, which are amenable to analysis and simulation. This permits a new perspective on an old question: the main goal of Ueltschi and Betz's long-term project is to quantify the shift in critical temperature, as a function of scattering length, for non-ideal Bose gases in the small-scattering-length regime.

The interaction terms for the permutation representation of the Bose gas are difficult to compute. Moreover, it is interesting to consider the model of random spatial permutations (which we sometimes refer to as the random-cycle model) for its own sake. Thus, Ueltschi, Betz, Gandolfo, Ruiz, and the author take various approaches with varying degrees of fidelity to the physical Bose-gas model. In section 2.1, we will see a random-cycle model for N particles which is parameterized by N *cycle weights* $\{\alpha_\ell\} = \alpha_1, \ldots, \alpha_N$ which encourage or discourage permutation cycles of lengths $\ell = 1, \ldots, N$. The remainder of this section involves results that may be obtained, analytically or simulationally, when various constraints are placed on the cycle weights.

In the papers [U07, BU07], Betz and Ueltschi examine the Bose-gas permutation weights with point positions allowed to vary in the continuum; an exact expression for the critical temperature is stated and proved for a simplified interaction model in which only two-cycles interact. The cycle-weight parameter α_2 is expressed in terms of the scattering length a; all other cycle weights are set to zero. In [BU08], this approach is extended to a model in which all the cycle weights α_ℓ may vary, but with the constraint that α_ℓ goes to zero faster than $1/\log(\ell)$. Here, an expression for the shift in critical temperature is found, as a function of all N cycle weights. It is key to note that these α_ℓ's are not computed directly from the physical scattering length a; rather, the result obtained is true for *any* cycle weights $\{\alpha_\ell\}$ satisfying the decaying-cycle-weight hypothesis. In [BU10], Ueltschi and Betz estimate, to first order, cycle weights $\{\alpha_\ell\}$ for the Bose gas.

Betz, Ueltschi, and Velenik [BUV09] examine cycle weights $\{\alpha_\ell\}$ with various hypotheses, including the Ewens [Ewens] case in which cycle weights are constant for all cycle lengths ℓ. These random permutations are non-spatial, i.e. $T = 0$ in the vocabulary of chapter 2. Their work is relevant to section 3.4 of this dissertation.

As is often the case in statistical mechanics, the study of this interacting system necessitates the use of computational methods — specifically, Markov-chain Monte Carlo. In [GRU], a simulational approach is taken for points held fixed on the cubic unit lattice in the non-interacting case ($\alpha_\ell \equiv 0$ in the language of chapter 2).

This dissertation, the only known simulation approach to the interacting model, applies MCMC methods to the case where $N = L^3$ points are held fixed on the fully occupied cubic unit lattice, with small additional probability weights depending on cycle lengths. This extends from[GRU] as well as [BUV09]. We find that even though lattice positions are used, and even though the decaying-cycle-weight hypothesis is invalidated, one nonetheless recovers the shift in critical temperature as predicted in the decaying-cycle-weight model of [BU08].

1.5 Literature review

In addition to the many references made in previous sections of this chapter, we point out the following.

The papers [GCL97], [KPS], [Ceperley], and [NL04] are among path-integral

Monte Carlo simulational approaches to Bose-Einstein condensation — perhaps the closest relatives to the numerical work done in [GRU] and in this dissertation.

The doctoral dissertations of Peter Grüter and Markus Holzmann are paradigmatic examples of clear dissertation writing [Grüter, Holzmann]; the latter also provided insight into finite-size scaling.

Mean longest cycle for uniformly distributed (i.e. non-interacting, non-spatial) permutations ($T = 0$ and $\alpha = 0$ in the language of chapter 2) was discussed by [SL], following a question posed by Golomb on the basis of experimental data [Golomb]. See also sections 2.4 and 2.5. Non-uniformly distributed non-spatial permutations with constant cycle weights ($T = 0$ and $\alpha \neq 0$ in the language of chapter 2) arose in mathematical biology [Ewens].

Background in quantum mechanics and statistical mechanics may be found in [Griffiths], [Huang], and [Sakurai].

Worm algorithms for path-integral Monte Carlo, which inspired the random-spatial-permutation worm algorithm of chapter 7, are used throughout simulational physics. See in particular [BPS06] and [PST98].

Finite-size scaling techniques are employed for path-integral Monte Carlo simulations in [GCL97], [HK99], [KPS], [NL04], [PC87], [PGP08], and [PR92]. Citation trails in the above-cited works lead back to [Barber]. Some background information is found in [LB]. An excellent survey, encompassing and explicating all the above methods — truly a blessing for the aspiring learner — is [PV].

Markov chain Monte Carlo methods are discussed in [LB]; this dissertation has been influenced most heavily by [Berg]. Indeed, my appendix B is an elaboration on Berg's discussion of integrated autocorrelation time. The probability background necessary for either Landau and Binder or Berg may be found, with increasing levels of sophistication, in [Lawler], [GS], and [Øksendal]. The standard reference for statistical analysis, including confidence intervals, is [CB]. More practical aspects of the statistical reduction of experimental data are found in [Young].

The Mersenne Twister [MN] is the pseudo-random-number generator used in the this dissertation's computational work. Another good generator is pseudo-DES [NR]. Moreover, any numerical dissertation without a reference to *Numerical Recipes* is incomplete; its inclusion here is as good a point as any to end the literature review.

1.6 Originality of dissertation

Last, we delineate the originality of work presented in this dissertation. Chapters 1-3 are a rephrasing and an elaboration on [BU07, BU08, U07]. Chapter 4 is quite standard; the contribution made here is to present familiar general results in the specific context of random permutations. The essential SO algorithm of chapter 5, with a small modification, was presented in [GRU]; likewise for the SO ΔH computations in chapter 8. The treatment here is the first correctness proof of the SO algorithm. The SAR algorithm of chapter 5 was suggested by Daniel Ueltschi. The band-update algorithm (chapter 6) is due to the author. The worm algorithm and its correctness proof (chapter 7) are due to the author, along with the remaining ΔH computations of chapter 8. The remaining chapters, 9-12, are also original work. Appendix A briefly summarizes part of [BU07, U07]. Appendix B is a new take on an old question; see also [Berg]. The correlated-uniform Markov process is original, as is the explicit comparison of batched and non-batched means for exponentially correlated stationary Markov processes.

Chapter 2
The model of random spatial permutations

Here we review concepts from [BU07, BU08], fixing notation and intuition to be used in the rest of the paper.

2.1 The probability model

Our state space is

$$\Omega_{\Lambda,N} = \Lambda^N \times \mathcal{S}_N$$

where $\Lambda = [0, L]^3$ with periodic boundary conditions and $\mathcal{S}_N$ is the group of permutations of N points[1]. Point positions are $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ for $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \Lambda$. These are called *spatial permutations* in that they involve the permutation π as well as the N point positions $\mathbf{x}_1, \ldots, \mathbf{x}_N$. See figure 2.1.

The probability measure on this state space will be constructed via Gibbs measure, $P_{\text{Gibbs}} = e^{-H}/Z$, on a Hamiltonian H. The background probability measure is discrete (uniform) in π and continuous (Lebesgue) in $\mathbf{X}$. The Hamiltonian takes one of two forms. In the first, relevant to the Bose gas, we have

$$H(\mathbf{X}, \pi) = \frac{T}{4} \sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{x}_{\pi(i)}\|_\Lambda^2 + \sum_{1 \le i < j \le N} V(\mathbf{x}_i, \mathbf{x}_{\pi(i)}, \mathbf{x}_j, \mathbf{x}_{\pi(j)}) \tag{2.1.1}$$

where $T = 1/\beta$ and the V terms are interactions between permutation jumps. The notation $\| \cdot \|_\Lambda$ indicates the natural distance on the 3-torus:

$$\|\mathbf{x} - \mathbf{y}\|_\Lambda = \min_{\mathbf{n} \in \mathbb{Z}^3} \{ \|\mathbf{x} - \mathbf{y} + L\mathbf{n}\| \} \tag{2.1.2}$$

For the V terms in equation (2.1.1), the permutation jump $\mathbf{x}_i \mapsto \mathbf{x}_{\pi(i)}$ interacts with the permutation jump $\mathbf{x}_j \mapsto \mathbf{x}_{\pi(j)}$. The temperature scale factor $T/4$, not $\beta/4$, is

[1]One may of course consider $\Lambda = [0, L]^d$ for $d = 1, 2$, but for this paper, $d = 3$ only.

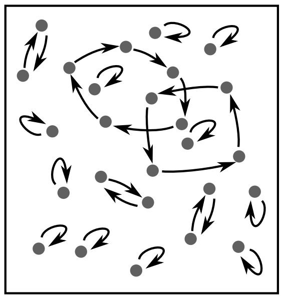


FIGURE 2.1. A spatial permutation on $N = 26$ points. There are 11 one-cycles, three two-cycles, one four-cycle, and one five-cycle. We say $r_1(\pi) = 11$, $r_2(\pi) = 3$, $r_4(\pi) = 1$, $r_5(\pi) = 1$, and $r_\ell(\pi) = 0$ for all other ℓ.

atypical in statistical mechanics. For purposes of the current work, this may be considered an ansatz; in [BU07], this choice of scale factor is shown to be appropriate for permutation representation of the Bose gas. In particular, as will be explained in more detail below, only the identity permutation appears at high T, and (with $V \equiv 0$) uniformly weighted permutations appear at zero T.

In the second form of the Hamiltonian, considered in this paper, we use interactions which are dependent solely on cycle lengths:

$$H(\mathbf{X}, \pi) = \frac{T}{4} \sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{x}_{\pi(i)}\|_\Lambda^2 + \sum_{\ell=1}^{N} \alpha_\ell r_\ell(\pi), \tag{2.1.3}$$

where $r_\ell(\pi)$ is the number of ℓ-cycles in π, for ℓ from 1 and N, and the α_ℓ's are free parameters, called *cycle weights*. One ultimately hopes to choose the α_ℓ's appropriately for the Bose gas; even if not, the model is well-defined and of its own mathematical interest.

The first contribution to the energy[2] is the sum of squares of permutation jump lengths. Since we will use a Gibbs distribution with $P_{\text{Gibbs}} = e^{-H}/Z$, the highest-probability permutations will be the ones with lowest energy. Thus, permutations

[2]The papers [BU07] and [U07] generalize from $\|\mathbf{x}_i - \mathbf{x}_{\pi(i)}\|_\Lambda$ to $\xi(\mathbf{x}_i, \mathbf{x}_{\pi(i)})$ where ξ is a spherically symmetric non-negative-valued function on $\mathbb{R}^d$ having integrable $e^{-\xi}$. This generalization is not of interest in this dissertation.

with long jumps will be disfavored; permutations with many short jumps will be less strongly disfavored. The second contribution to the energy involves cycle weights. We consider only small cycle weights, which perturb the critical temperature but which do not qualitatively modify the effects of the distance-related terms. More intuition for the model will be presented in section 2.3.

Choices of point positions $\mathbf{x}_1, \ldots, \mathbf{x}_N$ yield two cases: (1) In the *annealed model*, point positions are variable in the continuum and are averaged over. One has a particle density $\rho = N/L^3$. This model is examined analytically in [BU07, U07, BU08]. (2) In the *quenched model*, point positions are held fixed. Specifically, we consider $N = L^3$ points on the fully occupied integer-indexed sites of the $L \times L \times L$ cubic lattice. This model is examined simulationally in [GRU] and in this dissertation. We often write $H(\pi)$ in place of $H(\mathbf{X}, \pi)$ since we either work on a lattice where the $\mathbf{x}_i$'s are held fixed, or on the continuum where the $\mathbf{x}_i$'s are integrated out. Thus, the system energy H (as well as all other random variables we consider) is a function of the permutation π.

We consider two partition functions, for a fixed point configuration $\mathbf{X}$ and for an average over point configurations, respectively:

$$Y(\Lambda, \mathbf{X}) = \sum_{\sigma \in \mathcal{S}_N} e^{-H(\mathbf{X},\sigma)} \quad \text{and} \quad Z(\Lambda, N) = \frac{1}{N!} \int_{\Lambda^N} Y(\Lambda, \mathbf{X})\, d\mathbf{X}.$$

Fixing point positions $\mathbf{X}$, we have a discrete distribution on π:

$$Y(\Lambda, \mathbf{X}) = \sum_{\sigma \in \mathcal{S}_N} e^{-H(\mathbf{X},\sigma)}, \qquad P_{\text{Gibbs}}(\pi) = P_{\Lambda,\mathbf{X}}(\pi) = \frac{e^{-H(\mathbf{X},\pi)}}{Y(\Lambda, \mathbf{X})}. \tag{2.1.4}$$

For varying point positions (e.g. for considerations of the Bose gas), we have a joint distribution which is continuous in $\mathbf{X}$ and discrete in π:

$$P_{\Lambda,N}(\mathbf{X}, \pi)\, d\mathbf{X} = \frac{e^{-H(\mathbf{X},\pi)} d\mathbf{X}}{Z(\Lambda, N)}.$$

From this we obtain two marginal distributions. If we integrate over point configurations $\mathbf{X}$, then we obtain a discrete distribution on $\mathcal{S}_N$:

$$P_{\Lambda,N}(\pi) = \frac{1}{N!}\int_{\Lambda^N} d\mathbf{X}\, P_{\Lambda,N}(\mathbf{X},\pi) = \frac{\frac{1}{N!}\int_{\Lambda^N} d\mathbf{X}\, e^{-H(\mathbf{X},\pi)}}{\frac{1}{N!}\int_{\Lambda^N} d\mathbf{X}\, \sum_{\sigma\in\mathcal{S}_N} e^{-H(\mathbf{X},\sigma)}} = \frac{\int_{\Lambda^N} d\mathbf{X}\, e^{-H(\mathbf{X},\pi)}}{Z(\Lambda,N)\, N!}.$$

If, on the other hand, we sum over permutations, then we obtain a continuous distribution for point configurations:

$$P_{\Lambda,N}(\mathbf{X})\, d\mathbf{X} = \sum_{\pi\in\mathcal{S}_N} P_{\Lambda,N}(\mathbf{X},\pi)\, d\mathbf{X} = \frac{Y(\Lambda,\mathbf{X})\, d\mathbf{X}}{Z(\Lambda,N)}. \tag{2.1.5}$$

This continuous distribution is certainly of interest: it is the point distribution for the Bose gas, when the Hamiltonian is appropriately chosen. However, it is very difficult to compute: this is but one of several results in [LLS]. From here on, we consider the two discrete distributions on $\mathcal{S}_N$, namely, $P_{\Lambda,\mathbf{X}}(\pi)$ and $P_{\Lambda,N}(\pi)$.

For a random variable $X(\pi)$, we have

$$\mathbb{E}_{\Lambda,\mathbf{X}}[X] = \frac{\sum_{\pi\in\mathcal{S}_N} X(\pi) e^{-H(\mathbf{X},\pi)}}{Y(\Lambda,\mathbf{X})} \quad \text{and} \quad \mathbb{E}_{\Lambda,N}[X] = \frac{\int_{\Lambda^N} d\mathbf{X} \sum_{\pi\in\mathcal{S}_N} X(\pi) e^{-H(\mathbf{X},\pi)}}{Z(\Lambda,N) N!}.$$

In either case, we also write the probability as $P_{\text{Gibbs}}(\pi)$ and the expectation as

$$\mathbb{E}[X] = \sum_{\pi\in\mathcal{S}_N} P_{\text{Gibbs}}(\pi) X(\pi). \tag{2.1.6}$$

2.2 Model variants by choice of cycle weights

The model of random spatial permutations is in fact a family of models. As described in the previous section, point positions may be annealed or quenched, the latter being the case for this dissertation. Likewise, various constraints may be placed on the cycle weights. Recall from equation (2.1.3) that

$$H(\mathbf{X},\pi) = \frac{T}{4}\sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{x}_{\pi(i)}\|_\Lambda^2 + \sum_{\ell=1}^{N} \alpha_\ell r_\ell(\pi). \tag{2.2.1}$$

There are N free parameters α_ℓ, and thus many models of spatial permutations. (See also figure 2.2.)

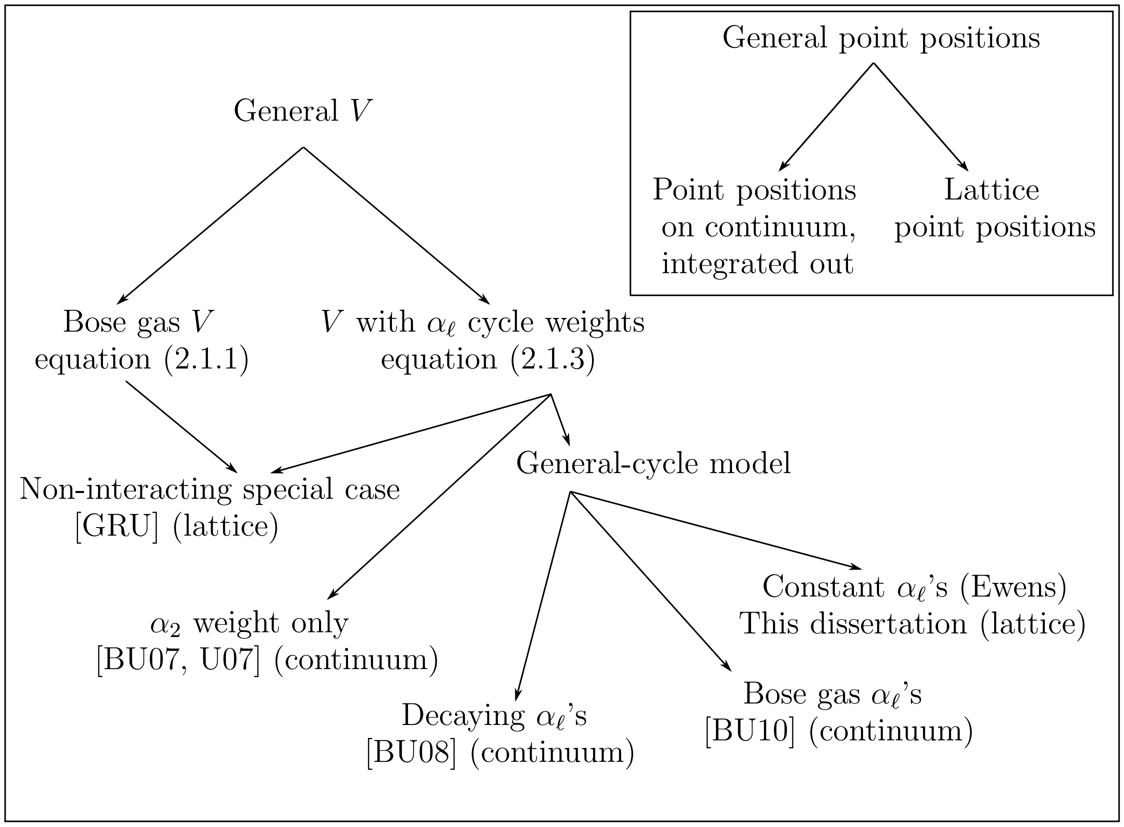

General point positions
Point positions on continuum, integrated out
Lattice point positions
General V
Bose gas V equation (2.1.1)
V with α_ℓ cycle weights equation (2.1.3)
General-cycle model
Non-interacting special case [GRU] (lattice)
α_2 weight only [BU07, U07] (continuum)
Constant α_ℓ's (Ewens) This dissertation (lattice)
Decaying α_ℓ's [BU08] (continuum)
Bose gas α_ℓ's [BU10] (continuum)

FIGURE 2.2. Model variants by choice of point positions and choice of cycle weights.

If $\alpha_\ell = 0$ for all ℓ, one obtains the *non-interacting case*. When $\alpha_2 = \alpha$ and $\alpha_\ell = 0$ for $\ell \neq 2$, we have the *two-cycle model* [BU07, U07] in which two-cycles are discouraged as α is increased. Otherwise, we have the *general-cycle model*. This splits into (at least) three submodels: Betz and Ueltschi, in [BU08], consider the case where α_ℓ tend toward zero faster than $1/\log(\ell)$. (Note that this includes the two-cycle model as a special case.) Ideally, one would have α_ℓ's which match the Brownian-bridge interactions for the Bose gas (appendix A). Work in this direction has recently been done by Betz and Ueltschi [BU10], but is beyond the scope of this dissertation.

For this dissertation, we consider the case where $\alpha_\ell = \alpha$ is constant in ℓ. We call this the *spatial Ewens distribution* [Ewens]. In summary, the model considered in this dissertation uses point positions held fixed on the fully occupied cubic unit lattice, with small non-negative Ewens cycle weights.

2.3 Qualitative characterization of long cycles

Having seen definitions for the probability model, one next asks what a typical random spatial permutation looks like. In this section we develop intuition; in section 3.7, we construct quantitative descriptions of the ideas presented here.

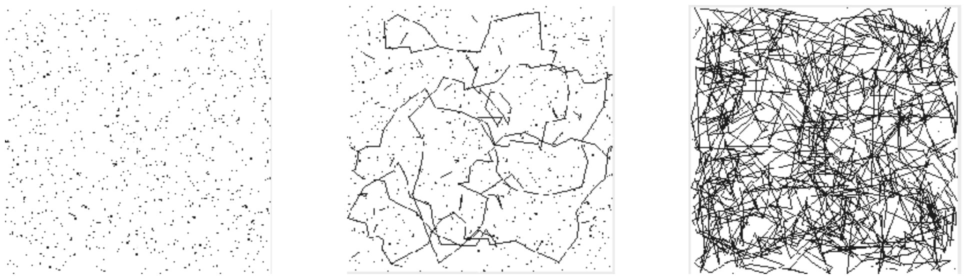


FIGURE 2.3. Points and permutation jumps, for a typical permutation at high T (there are only small cycles of short jumps), medium but subcritical T (all jump lengths are short, with occasional long cycles thereof), and low T (jump lengths are arbitrary).

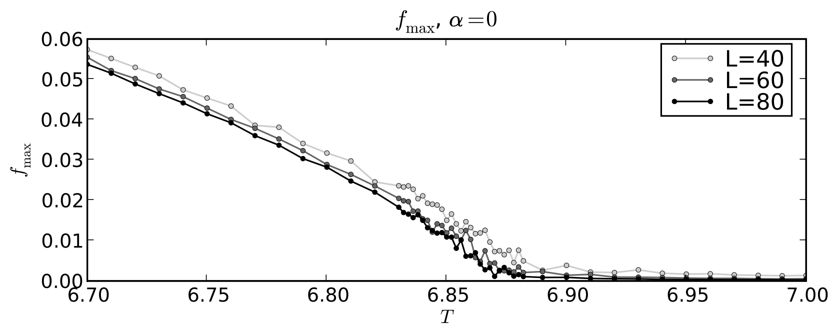
As $T \to \infty$, the probability measure becomes supported only on the identity permutation: the distance-dependent terms in equation (2.1.3) are large whenever any jump has non-zero length. For large but finite T, the length-dependent terms penalize permutation jumps from a site to any site other than itself. Thus, for large T we expect the identity permutation to be the most likely, with occasional 2-cycles, 3-cycles, etc. which involve nearby points. On the other extreme, as $T \to 0$, length-dependent terms go to zero and the probability measure approaches the uniform distribution on $\mathcal{S}_N$: the distance-dependent terms all go to zero. For intermediate T, one observes that the length $\|\pi(\mathbf{x}) - \mathbf{x}\|_\Lambda$ of each permutation jump remains small, increasing smoothly as T drops. See figure 9.8 on page 122 for more precise information.

The intermediate-temperature regime is the one of interest. As is found in theoretical and simulational work, as detailed through the rest of this dissertation, this regime has the following properties. There is a phase transition: for T below a critical temperature T_c, while individual jump lengths remain short (i.e. we work in the short-jump-length regime), *arbitrarily long cycles form.* See figure 2.3 for depictions of typical permutations at high T, subcritical T, and low T. In the non-interacting case, $T_c(0)$ is approximately 6.87; interactions — in the form of positive α terms — increase $T_c(\alpha)$. Quantifying that dependence, i.e.

$$\Delta T_c(\alpha) = \frac{T_c(\alpha) - T_c(0)}{T_c(0)}, \tag{2.3.1}$$

as a function of α for small positive α, is the central goal of this dissertation.

From figures such as 2.3, one can detect long cycles visually. How do we measure them numerically? Let $\ell_{\max}(\pi)$ be the length of the longest cycle in π, with $\mathbb{E}[\ell_{\max}]$ its mean over all permutations. We take $N = L^3$ points on the $L \times L \times L$ unit lattice with periodic boundary conditions. We observe that for T above T_c, $\mathbb{E}[\ell_{\max}]$ increases only perhaps as fast as $\log L$. That is to say, $\mathbb{E}[\ell_{\max}]/N$ goes to zero as $L \to \infty$. For T below T_c, the length of the longest cycle does increase as L increases — we find that $\mathbb{E}[\ell_{\max}]$ scales with N. (This is one of the results of [Sütő1]; it is perhaps surprising that the scaling is by $N = L^3$ rather than, say, L^2.) That is to say, $\mathbb{E}[\ell_{\max}]/N$ approaches a temperature-dependent constant as $L \to \infty$; there are arbitrarily long cycles, or infinite cycles, in the infinite-volume limit. See figure 2.4 for plots of $\mathbb{E}[\ell_{\max}]/N$ as a function of T for various system sizes with $N = L^3$. See also figure 9.10 on page 124. Precise information about $\mathbb{E}[\ell_{\max}]/N$ and other quantities is found in section 3.7.



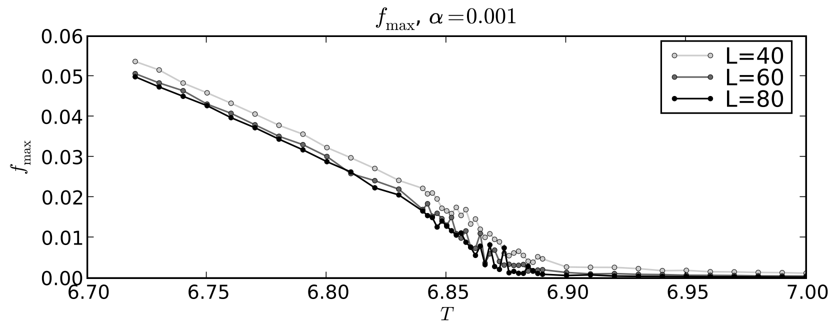


FIGURE 2.4. Order parameter $f_{\max} = \mathbb{E}[\ell_{\max}]/N$ for finite systems, with $\alpha = 0, 0.001$. Interactions increase the critical temperature.

2.4 Known results

In this dissertation we study chiefly the α-dependent shift in critical temperature for Ewens cycle weights and cubic-lattice point positions. We will also consider an α-dependent macroscopic-cycle quotient, to be defined below. Here we survey known results for related models — namely, other cycle weights as described in section 2.2, and point positions integrated over the continuum — before stating our conjectures for our model in section 2.5.

Known results for point locations averaged over the continuum (as discussed in section 2.1) are obtained largely using Fourier methods [BU08] which are unavailable for point positions held fixed on the lattice. Betz and Ueltschi have determined $\Delta T_c(\alpha)$, to first order in α, for two-cycle interactions [BU07] and decaying cycle weights [BU08]. The critical (ρ, T, α) manifold relates ρ_c to T_c. Specifically, they obtain the following, with the decaying-cycle-weight constraint on the cycle weights $\{\alpha_\ell\}$:

$$\rho_c(\alpha_1, \alpha_2, \ldots) = \frac{1}{(4\pi\beta)^{3/2}} \sum_{\ell \geq 1} e^{-\alpha_\ell} \ell^{-3/2}. \tag{2.4.1}$$

In the non-interacting special case, all cycle weights are zero, and we have

$$\rho_c(0) = \frac{1}{(4\pi\beta)^{3/2}} \sum_{\ell \geq 1} \ell^{-3/2} = \frac{\zeta(3/2)}{(4\pi\beta)^{3/2}} \tag{2.4.2}$$

where ζ is the Riemann zeta function. For the two-cycle special case [BU07], α_2 is non-zero, and the other cycle weights are zero. We have

$$\rho_c(\alpha) = \frac{1}{(4\pi\beta)^{3/2}} \left(e^{-\alpha} 2^{-3/2} + \sum_{\ell \neq 2} e^{-\alpha_\ell} \ell^{-3/2} \right) = \rho_c(0) + \frac{(e^{-\alpha} - 1)}{(8\pi\beta)^{3/2}}.$$

In addition to our main interest on the shift in critical temperature (here, phrased in terms of critical density), we also examine the fraction of sites in macroscopic cycles. As will be discussed in more detail in sections 3.4 and 3.5, $\ell_{\max}$ is the length of the longest cycle of a permutation; f_I will quantify the fraction of sites participating in long cycles. For $\alpha_\ell \equiv 0$ (the non-interacting model), one observes empirically that the *macroscopic-cycle quotient* $\mathbb{E}[\ell_{\max}]/N f_I$ is constant for T below but near T_c. (That is, the two order parameters f_I and $\mathbb{E}[\ell_{\max}]/N$ have the same critical exponent.) For uniform-random (non-spatial) permutations, Shepp and Lloyd 1966 [SL] solved Golomb's 1964 question [Golomb]: $\mathbb{E}[\ell_{\max}]/N \approx 0.6243$. Unpublished work of Betz and Ueltschi has found $\mathbb{E}[\ell_{\max}]/N f_I$ holds that same value for random spatial permutations in the non-interacting case $\alpha_\ell \equiv 0$. The intuition is that long cycles are uniformly distributed within the zero Fourier mode.

2.5 Conjectures

Equation (2.4.1) gives ρ_c as a function of $\beta = 1/T$. For the cubic unit lattice, we fix $\rho \equiv 1$ and thus obtain β_c, or equivalently T_c:

$$T_c(\alpha_1, \alpha_2, \ldots) = \frac{4\pi}{\left(\sum_{\ell \geq 1} e^{-\alpha_\ell} \ell^{-3/2}\right)^{2/3}}. \tag{2.5.1}$$

For the non-interacting special case, this is

$$T_c(0) = \frac{4\pi}{\zeta(3/2)^{2/3}} \approx 6.625. \tag{2.5.2}$$

The Ewens-cycle-weight case does not satisfy the decaying-cycle-weight constraint where the α_ℓ's must go to zero in ℓ faster than $1/\log(\ell)$; all the α_ℓ's are the same. Nonetheless, using equation (2.5.1), we obtain

$$T_c(\alpha) = \frac{4\pi}{\left[e^{-\alpha}\zeta(3/2)\right]^{2/3}} \approx 6.625 e^{2\alpha/3}.$$

Taylor-expanding in the small parameter α, the shift in critical temperature is then

$$\Delta T_c(\alpha) = \frac{T_c(\alpha) - T_c(0)}{T_c(0)} = e^{2\alpha/3} - 1 \approx \frac{2\alpha}{3} \quad \text{and} \quad c \approx 0.667. \tag{2.5.3}$$

(Note that this is not in conflict with the constant c in section 1.4, which through abuse of notation we also called c. There, one examines $\Delta T_c(a)$ where a is the scattering length of the interacting Bose gas; here, one has $\Delta T_c(\alpha)$ for free parameter α.)

As the primary goal of this dissertation, we inquire whether this result, obtained for decaying cycle weights with point positions varying on the continuum, holds for Ewens weights with point positions held fixed on the lattice. We suspect that the fine details of point positions are unimportant for the shift in critical temperature. For Ewens interactions, $\Delta T_c(\alpha)$ is theoretically unknown for points either on the continuum or on the lattice. The simulational treatment in this dissertation is the only known attack on this question.

Secondarily, we conjecture that $\mathbb{E}[\ell_{\max}]/Nf_I$, as discussed in section 2.4, is α-dependent but constant in T (for T below but near T_c) for our model of lattice point positions and Ewens cycle weights.

Chapter 3
Random variables

Having in hand the definition of the probability model from chapter 2, the next logical step is to define random variables. In particular, we seek *order parameters* — random variables which allow us to identify the phase transition to long cycles. Specific random variables used in this dissertation are as follows. For each, the relevant theory sections (in this chapter) and experiment sections (in chapter 9) are pointed to.

- System energy and energy density: these are as discussed in section 2.1; explicit computation is discussed in section 9.8.
- The number of ℓ-cycles in the permutation for $\ell = 1, 2, \ldots, N$: the definitions are familiar from elementary algebra; computation is discussed in section 9.9.
- Cycle length, spatial cycle length, and correlation length: sections 3.2 and 9.10.
- Mean jump length and maximum jump length: sections 3.3 and 9.11. These are used to confirm the hypothesis of short jump lengths as mentioned in section 3.6.
- Fraction of sites in cycles of specified lengths, and fraction of sites in long cycles: section 3.4 and (theory) and 9.12 (experiment).
- Longest cycle length and macroscopic-cycle quotient: sections 3.5 and 9.13.
- Winding numbers, scaled winding number, and fraction of sites in winding cycles: sections 3.6 and 9.14.

A word on notation: given a random variable X, let $Q = \mathbb{E}[X]$. For each quantity Q, one should distinguish between the finite-volume value $Q_L(T)$ and the infinite-volume limit $Q_\infty(T) = \lim_{L\to\infty} Q_L(T)$. For this chapter, omitted subscripts are disambiguated by context. The difference becomes significant in chapter 9; at that point, we will carefully distinguish between $Q_L(T)$ and $Q_\infty(T)$.

3.1 Differences and distances on the torus

We first define the natural difference-vectors and distances on the 3-torus. Namely, for $\mathbf{z} \in \Lambda$, we define a zero-centered modulus vector $\mathbf{m}_L(\mathbf{z})$. For $\mathbf{x}, \mathbf{y} \in \Lambda$, this gives rise to a difference vector $\mathbf{d}_\Lambda(\mathbf{x}, \mathbf{y})$ and a distance $\|\mathbf{x} - \mathbf{y}\|_\Lambda$. The former are needed for winding numbers (section 3.6); the latter are needed for the Hamiltonian (section 2.1), spatial cycle length and correlation length (section 3.2), and jump length (section 3.3). Specifically, we have the following:

$$\mathbf{m}_L(\mathbf{z}) = \begin{pmatrix} m_L(z_1) \\ m_L(z_2) \\ m_L(z_3) \end{pmatrix} \tag{3.1.1}$$

$$n_L(z) = n \in \mathbb{Z} \text{ which minimizes } |z + nL| \tag{3.1.2}$$

$$m_L(z) = z + n_L(z)L \tag{3.1.3}$$

$$\mathbf{d}_\Lambda(\mathbf{x}, \mathbf{y}) = \mathbf{m}_\Lambda(\mathbf{x} - \mathbf{y}) \tag{3.1.4}$$

$$\|\mathbf{z}\|_\Lambda = \|\mathbf{m}_\Lambda(\mathbf{z})\|. \tag{3.1.5}$$

For example, suppose $L = 20$, $\mathbf{x} = (0, 0, 18)$, and $\mathbf{y} = (0, 0, 1)$. Then $\mathbf{d}_\Lambda(\mathbf{x}, \mathbf{y}) = (0, 0, -3)$ and $\|\mathbf{x}-\mathbf{y}\|_\Lambda = 3$. This is called a zero-centered modulus since $m_L(z)$ takes values from $-L/2$ to $L/2$. There is an antipodal problem when L is even: the distance is well-defined on the torus, but differences are ambiguous at $L/2$ in any of the three slots. For example, if $L = 20$, $\mathbf{x} = (0, 0, 18)$, and $\mathbf{y} = (0, 0, 8)$, then $\|\mathbf{x} - \mathbf{y}\|_\Lambda = 10$ but $\mathbf{d}_\Lambda(\mathbf{x}, \mathbf{y}) = (0, 0, 10)$ or $(0, 0, -10)$. However, as mentioned in sections 2.3, 3.6, 6.2, and 9.11, we work in the short-jump-length regime. Specifically, in section 9.11

we find that for T near T_c, jump length remains below 5, with probability very near 1, regardless of how big L is.

We now show that equation (3.1.5) is compatible with the definition of $\|\cdot\|_\Lambda$ from equation (2.1.2) on page 23. namely,

$$\|\mathbf{x}-\mathbf{y}\|_\Lambda = \min_{\mathbf{n}\in\mathbb{Z}^3}\{\|\mathbf{x}-\mathbf{y}+L\mathbf{n}\|\}.$$

Proposition 3.1.6. *We have*

$$\|\mathbf{x}-\mathbf{y}\|_\Lambda = \|\mathbf{d}_\Lambda(\mathbf{x},\mathbf{y})\|. \tag{3.1.7}$$

Proof. For brevity, let $\mathbf{z}=\mathbf{x}-\mathbf{y}$. Since the square-root function is one-to-one and increasing on non-negative reals, it suffices to show

$$\|\mathbf{m}_\Lambda(\mathbf{z})\|^2 = \min_{\mathbf{n}\in\mathbb{Z}^3}\{\|\mathbf{z}+\mathbf{n}L\|^2\}.$$

Starting with the right-hand side, we have

$$\begin{aligned}\min_{\mathbf{n}\in\mathbb{Z}^3}\{\|\mathbf{z}+\mathbf{n}L\|^2\} &= \min_{n_1\in\mathbb{Z}}\min_{n_2\in\mathbb{Z}}\min_{n_3\in\mathbb{Z}}\left\{(z_1+n_1L)^2+(z_2+n_2L)^2+(z_3+n_3L)^2\right\}\\ &= \min_{n_1\in\mathbb{Z}}\left\{(z_1+n_1L)^2\right\}+\min_{n_2\in\mathbb{Z}}\left\{(z_2+n_2L)^2\right\}+\min_{n_3\in\mathbb{Z}}\left\{(z_3+n_3L)^2\right\}\\ &= m_L(z_1)^2+m_L(z_2)^2+m_L(z_3)^2\\ &= \|\mathbf{m}_\Lambda(\mathbf{z})\|^2.\end{aligned}$$

□

3.2 Cycle lengths and correlation length ξ

Definition 3.2.1. Fixing $\pi\in\mathcal{S}_N$ and $\mathbf{x}\in\Lambda$, the cycle length $\ell_\mathbf{x}(\pi)$ is the smallest positive integer a such that $\pi^a(\mathbf{x})=\mathbf{x}$. This is nothing more than the length of the cycle containing $\mathbf{x}$: if $\mathbf{x}$ is in a 3-cycle, then $\ell_\mathbf{x}(\pi)=3$. We define the spatial cycle length

$$s_\mathbf{x}(\pi) = \sum_{j=1}^{\ell_\mathbf{x}(\pi)} \|\pi^j(\mathbf{x})-\pi^{j-1}(\mathbf{x})\|_\Lambda.$$

This is simply the sum of Euclidean jump distances for all permutation jumps in the cycle containing $\mathbf{x}$. We may at times instead write, respectively,

$$\ell_i(\pi) = \ell_{\mathbf{x}_i}(\pi) \qquad \text{or} \qquad s_i(\pi) = s_{\mathbf{x}_i}(\pi).$$

For example, for the point configuration $\mathbf{X}$ and the permutation π in figure 3.1, we have

$$\ell_1(\pi) = \ell_2(\pi) = \ell_3(\pi) = \ell_4(\pi) = 4, \quad \ell_5(\pi) = \ell_6(\pi) = \ell_7(\pi) = 3, \quad \text{and} \quad \ell_8(\pi) = 1.$$

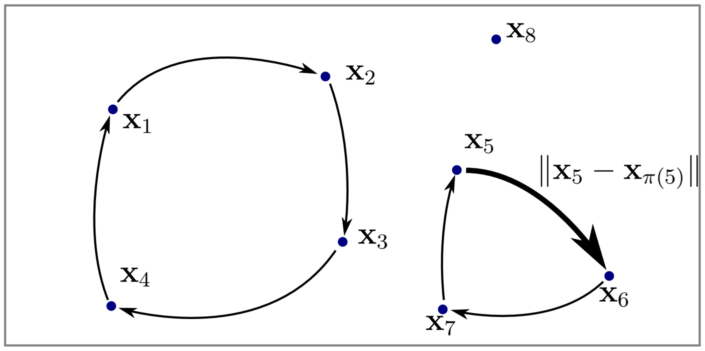


FIGURE 3.1. A configuration of $\mathbf{X}$ and π with $N = 8$.

Definition 3.2.2. If $\mathbf{x}$ and $\mathbf{y}$ are in a common cycle of π, we say that $\mathbf{x}$ is connected to $\mathbf{y}$; otherwise we say that $\mathbf{x}$ is not connected to $\mathbf{y}$. These are written

$$\mathbf{x} \circ\!\!-\!\!\circ \mathbf{y} \qquad \text{and} \qquad \mathbf{x} \circ\!\!\not-\!\!\circ \mathbf{y}.$$

In the former case, we write

$$\ell_{\mathbf{x},\mathbf{y}}(\pi)$$

for the smallest positive integer a such that $\pi^a(\mathbf{x}) = \mathbf{y}$. This is the number of permutation jumps from $\mathbf{x}$ to $\mathbf{y}$.

Definition 3.2.3. We may average cycle lengths and spatial cycle lengths (definition 3.2.1) over all points $\mathbf{x}$:

$$\overline{\ell}(\pi) = \frac{1}{N} \sum_{\mathbf{x} \in \Lambda} \ell_{\mathbf{x}}(\pi) \qquad\qquad \overline{s}(\pi) = \frac{1}{N} \sum_{\mathbf{x} \in \Lambda} s_{\mathbf{x}}(\pi).$$

Now, $\mathbb{E}[\ell_{\mathbf{x}}] = \mathbb{E}[\ell_{\mathbf{y}}]$ for all $\mathbf{x}, \mathbf{y} \in \Lambda$, and both are the same as $\mathbb{E}[\overline{\ell}]$ as shown in the lemma below. Experimentally, however, as discussed at the beginning of chapter 4, we cannot compute expectations over all $\pi \in \mathcal{S}_N$; we must content ourselves with sample means over some sequence of M permutations $\pi_1, \ldots, \pi_M$, obtained in an MCMC simulation. We may increase the sample size by a factor N (thereby decreasing the sample variance by a factor of N, by the central limit theorem) if we average over all points $\mathbf{x}$. For each permutation, we may compute $\overline{\ell}(\pi)$ by averaging $\ell_{\mathbf{x}}(\pi)$ over all N point positions, and likewise for $s_{\mathbf{x}}(\pi)$ and $\overline{s}$.

Lemma 3.2.4. *For all* $\mathbf{x} \in \Lambda$, $\mathbb{E}[\overline{\ell}] = \mathbb{E}[\ell_{\mathbf{x}}]$.

Proof. The left-hand side is

$$\mathbb{E}[\overline{\ell}] = \sum_{\pi \in \mathcal{S}_N} P_{\text{Gibbs}}(\pi)\overline{\ell}(\pi).$$

Since both sums are finite, we have

$$\begin{aligned}\mathbb{E}[\overline{\ell}] &= \sum_{\pi \in \mathcal{S}_N} P_{\text{Gibbs}}(\pi)\overline{\ell}(\pi) = \sum_{\pi \in \mathcal{S}_N} P_{\text{Gibbs}}(\pi)\frac{1}{N}\sum_{\mathbf{x}\in\Lambda} \ell_{\mathbf{x}}(\pi) \\ &= \frac{1}{N}\sum_{\mathbf{x}\in\Lambda}\sum_{\pi \in \mathcal{S}_N} P_{\text{Gibbs}}(\pi)\ell_{\mathbf{x}}(\pi) = \overline{\mathbb{E}[\ell_{\mathbf{x}}]}.\end{aligned} \tag{3.2.5}$$

The equality $\overline{\mathbb{E}[\ell_{\mathbf{x}}]} = \mathbb{E}[\ell_{\mathbf{x}}]$ follows from translation invariance on the 3-torus. □

In summary, we have

$$\begin{aligned} \ell_{\mathbf{x}}(\pi) &= \min\{a > 0 : \pi^a(\mathbf{x}) = \mathbf{x}\} & s_{\mathbf{x}}(\pi) &= \sum_{j=1}^{\ell_{\mathbf{x}}(\pi)} \|\pi^j(\mathbf{x}) - \pi^{j-1}(\mathbf{x})\|_\Lambda \\ \mathbb{E}[\ell_{\mathbf{x}}] &= \sum_{\pi \in \mathcal{S}_N} P_{\text{Gibbs}}(\pi)\ell_{\mathbf{x}}(\pi) & \mathbb{E}[s_{\mathbf{x}}] &= \sum_{\pi \in \mathcal{S}_N} P_{\text{Gibbs}}(\pi)s_{\mathbf{x}}(\pi) \\ \overline{\ell}(\pi) &= \frac{1}{N}\sum_{\mathbf{x}\in\Lambda} \ell_{\mathbf{x}}(\pi) & \overline{s}(\pi) &= \frac{1}{N}\sum_{\mathbf{x}\in\Lambda} s_{\mathbf{x}}(\pi) \\ \mathbb{E}[\overline{\ell}] &= \sum_{\pi \in \mathcal{S}_N} P_{\text{Gibbs}}(\pi)\overline{\ell}(\pi) & \mathbb{E}[\overline{s}] &= \sum_{\pi \in \mathcal{S}_N} P_{\text{Gibbs}}(\pi)\overline{s}(\pi). \end{aligned}$$

The quantities on the second line are convenient for theoretical use; the quantities on the fourth line (due to the larger sample size) are preferable for experimental use.

We define a correlation length ξ to be

$$\xi = \mathbb{E}[\overline{s}]. \tag{3.2.6}$$

Computational details are in section 9.10.

3.3 Mean and maximum jump length

Definition 3.3.1. Let

$$j_{\mathbf{x}}(\pi) = \|\pi(\mathbf{x}) - \mathbf{x}\|_\Lambda$$

be the length of the permutation jump starting at site $\mathbf{x}$; let

$$j(\pi) = \frac{1}{N}\sum_{i=1}^{N} j_{\mathbf{x}_i}(\pi).$$

The mean jump length at site $\mathbf{x}$ is simply

$$\mathbb{E}[j_{\mathbf{x}}] = \sum_{\pi \in \mathcal{S}_N} P_{\text{Gibbs}}(\pi)\|\pi(\mathbf{x}) - \mathbf{x}\|_\Lambda.$$

By linearity of expectation, this is the same as the average over all sites:

$$\mathbb{E}[j(\pi)] = \mathbb{E}[j_{\mathbf{x}_1}(\pi)].$$

As was the case for $\overline{\ell}$ and $\overline{s}$ in section 3.2, we approximate the uncomputably large sum over all $N!$ permutations by a random sequence of M permutations, and the sample mean is random. By the central-limit theorem argument in section 3.2, the variance of the sample mean of j is a factor of N smaller than the variance of the sample mean of $j_{\mathbf{x}}$, since the sample size is MN instead of M. Additional computational details are discussed in section 9.11.

3.4 Fraction of sites in infinite cycles f_I

If one wants to quantify the temperature-dependent onset of long cycles (section 2.3), then one can define a random variable which counts the fraction of sites in long cycles. In the infinite limit, one looks for infinite cycles. One could define

$$f_I(\infty) = 1 - \sum_{k \geq 1} P_{\text{Gibbs}}(\ell_0 = k) \tag{3.4.1}$$

where $\ell_0(\pi)$ is as defined in section 3.2. For finite volume, where simulations are done, the right-hand side of (3.4.1) is always 0: every site is in a cycle of some finite length. One might then define

$$f_I(N) = 1 - \sum_{k < \varepsilon(N)} P_{\text{Gibbs}}(\ell_0 = k) \tag{3.4.2}$$

where $\varepsilon(N)$ is such that $\varepsilon(N) \to \infty$ as $N \to \infty$ but $\varepsilon(N)/N \to 0$. For example, one may take $\varepsilon(N) = \sqrt{N}$. This is chosen so that as $N \to \infty$, one obtains $f_I(\infty)$. In [BU07] it is found, among other results, that mesoscopic cycles are unimportant: for $T > T_c$, there are only microscopic cycles; for $T < T_c$, there are only microscopic and macroscopic cycles. The $\varepsilon(N)$ cutoff is designed to separate the former from the latter.

For practical computation, [GRU] begin by defining

$$\rho = \frac{N}{V},$$

where $V = L^3$ is the volume, i.e. ρ is the particle density. For $1 \leq m \leq n \leq N$, define

$$\varrho_{m,n}(\pi) = \frac{1}{V} \, \# \{ i = 1, \ldots, N : m \leq \ell_i(\pi) \leq n \}$$

This random variable, taking values between 0 and ρ, is the density of sites in cycles of specified length. One may also consider the related random variable

$$f_{m,n}(\pi) = \frac{1}{N} \, \# \{ i = 1, \ldots, N : m \leq \ell_i(\pi) \leq n \}$$

which is $\varrho_{m,n}/\rho$. (Of course, on the unit lattice where $N = L^3$ and $\rho = 1$, the two random variables $f_{m,n}(\pi)$ and $\varrho_{m,n}(\pi)$ are identical.) This runs from 0 to 1 and is the fraction of sites in cycles of specified length. For figure 3.1 on page 37, we have $f_{2,3}(\pi) = 3/8$. Then $\mathbb{E}[f_{\varepsilon(N),N}]$ matches equation (3.4.2) as follows:

$$\begin{aligned}\mathbb{E}[f_{\varepsilon(N),N}(\pi)] &= \frac{1}{N}\mathbb{E}\left[\#\{i = 1, \ldots, N : \ell_i(\pi) \geq \varepsilon(N)\}\right] \\ &= \frac{1}{N}\mathbb{E}\left[\sum_{i=1}^{N} 1_{\ell_i(\pi)\geq\varepsilon(N)}(\pi)\}\right] = \frac{1}{N}P_{\text{Gibbs}}(\ell_i(\pi) \geq \varepsilon(N)).\end{aligned}$$

This is the same as $P_{\text{Gibbs}}(\ell_0(\pi) \geq \varepsilon(N))$ by translation invariance. Then

$$P_{\text{Gibbs}}(\ell_0(\pi) \geq \varepsilon(N)) = \sum_{k\geq\varepsilon(N)} P_{\text{Gibbs}}(\ell_0 = k) = 1 - \sum_{k<\varepsilon(N)} P_{\text{Gibbs}}(\ell_0 = k) = f_I(N).$$

In practice, a single cutoff of the form $\varepsilon(N)$ is not used; one estimates the infinite-limit behavior in a different way. To see how to do this, we next invoke results of [Sütő2] and [BUV09] regarding the behavior of $\mathbb{E}[f_{1,k}]$ as a function of k/N in the infinite limit. The former, [Sütő2], applies in the non-interacting case; the latter, [BUV09], applies in the Ewens case but with non-spatial permutations, which are equivalent to random spatial permutations with $T = 0$. For the non-interacting case, $\mathbb{E}[f_{1,k}]$ is a straight line of slope 1 as shown in the upper left of figure 3.2. At $T = 0$, it fills the full diagonal; cycle lengths have uniform distribution. At $T > T_c$, the diagonal vanishes into the upper-left corner; there are no long cycles. The transition to criticality occurs at T such that the diagonal becomes visible. See also [Lugo].

In finite volume, $\mathbb{E}[f_{1,k}]$ is rounded as shown in the bottom left of figure 3.2. Thus, one wishes to draw a tangent-line approximation for the infinite-volume behavior, and take f_I to be one minus the vertical intercept. This avoids use of a specific, arbitrary cutoff $\varepsilon(N)$, replacing it instead with a graphical estimator which makes use of all available data. See also section 9.12 for computational details, including the handling of sampling variability.

For Ewens interactions, the diagonals are curved as shown in the upper right of

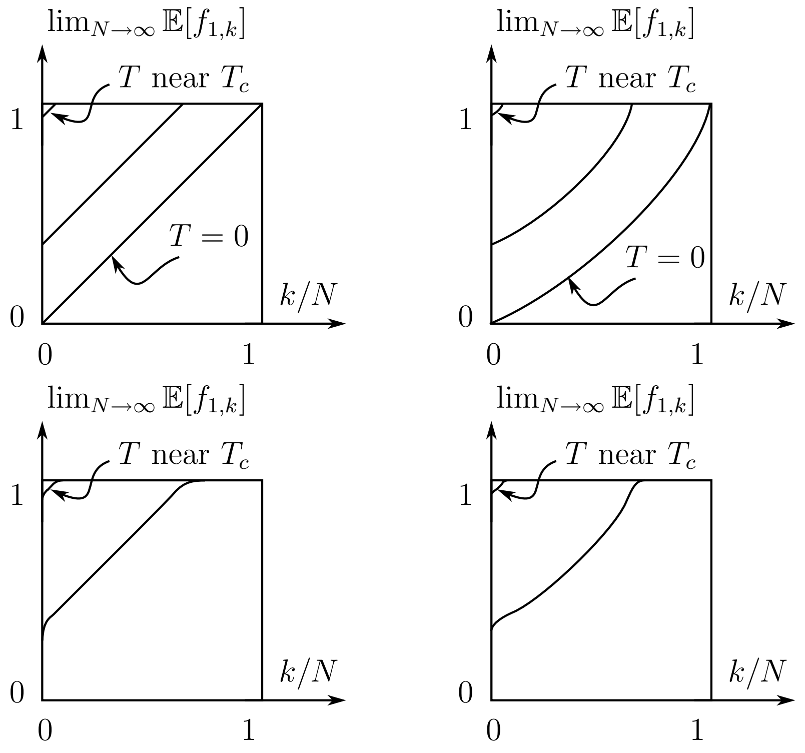


FIGURE 3.2. Qualitative behavior of $\lim_{N\to\infty}\mathbb{E}[f_{1,k}]$ as a function of k/N in the non-interacting (upper left) and non-spatial large-α Ewens-interacting (upper right) cases. Lower left and lower right show the behavior for finite N.

figure 3.2. For the small α values considered in this dissertation, however, this Betz-Ueltschi-Velenik sag in the curve is less than the sampling variability in the data itself (see the plots in section 9.12). Thus, the sag is not relevant to our discussion. It should also be remarked that the author has performed larger-α simulations for which the sag is indeed observed simulationally.

3.5 Macroscopic-cycle quotient $f_{\max}/f_I$

Definition 3.5.1. For a permutation π in $\mathcal{S}_N$, define

$$\ell_{\max}(\pi) = \max_{1\le i\le N} \ell_i(\pi). \tag{3.5.2}$$

For a spatial permutation, this is precisely the same as

$$\ell_{\max}(\pi) = \max_{\mathbf{x}\in\Lambda} \ell_{\mathbf{x}}(\pi). \tag{3.5.3}$$

We write

$$f_{\max} = \mathbb{E}[\ell_{\max}]/N. \tag{3.5.4}$$

Definition 3.5.5. The *macroscopic cycle quotient*, written $f_{\max}/f_I$ for brevity, is given by

$$\text{macroscopic cycle quotient} = \begin{cases} \frac{\mathbb{E}[\ell_{\max}]}{N f_I}, & f_I \neq 0 \\ 0, & f_I = 0. \end{cases} \tag{3.5.6}$$

Intuition was discussed in sections 2.4 and 2.5; computational details are discussed in section 9.13.

3.6 Winding numbers, f_S, and f_W

The box $\Lambda = [0, L]^3$ with periodic boundary conditions is topologically equivalent to the 3-torus. Permutation cycles wind around the 3-torus some number of times in the x, y, and/or z directions. The sub-T_c onset of long cycles corresponds to the appearance of cycles which wrap around the torus in one or more of the three axes. If a cycle goes around once in the clockwise direction, we want to say it has sign $+1$; likewise, we want sign -1 for the counterclockwise direction. The following definition formalizes this intuition.

Definition 3.6.1. The *winding number* (really a 3-tuple of numbers) of a permutation π is

$$\mathbf{W}(\pi) = (W_x(\pi), W_y(\pi), W_z(\pi)) = \frac{1}{L}\sum_{i=1}^{N} \mathbf{d}_\Lambda(\pi(\mathbf{x}_i), \mathbf{x}_i) \tag{3.6.2}$$

where $\mathbf{d}_\Lambda$ is the difference vector defined in equation (3.1.4). This simply counts the integer number of wraps of π's cycles around the 3-torus in each of the three directions. We also write

$$\mathbf{W}^2(\pi) = \mathbf{W}(\pi) \cdot \mathbf{W}(\pi) = W_x(\pi)^2 + W_y(\pi)^2 + W_z(\pi)^2. \tag{3.6.3}$$

Definition 3.6.4. The *scaled winding number* is

$$f_S = \frac{\mathbb{E}[\mathbf{W}^2]L^2}{3\beta N} = \frac{\mathbb{E}[\mathbf{W}^2]T}{3L}.$$

See [PC87] for the physical derivation. For us, it simply needs to be scaled by $L^2/N = 1/L$ in order to be an intensive parameter.

Definition 3.6.5. Let $c_{\mathbf{x}}(\pi)$ consist of all sites in the same cycle as $\mathbf{x}$:

$$c_{\mathbf{x}}(\pi) = \{\mathbf{y} \in \Lambda : \mathbf{y} = \pi^a(\mathbf{x}), a = 0, 1, 2, \ldots, N-1\}.$$

Let $\mathbf{w}_{\mathbf{x}}(\pi)$ be the *winding vector* for $\mathbf{x}$ (which is clearly the same for all sites in the same cycle as $\mathbf{x}$):

$$\mathbf{w}_{\mathbf{x}}(\pi) = \frac{1}{L} \sum_{\mathbf{y} \in c_{\mathbf{x}}(\pi)} \mathbf{d}_\Lambda(\pi(\mathbf{y}), \mathbf{y}),$$

where the difference vectors are again interpreted as in section 3.1. Note that all three slots of $\mathbf{w}_{\mathbf{x}}(\pi)$ are necessarily integer-valued. We say that π *winds through* $\mathbf{x}$ if $\mathbf{x}$ has a non-zero winding vector:

$$t_{\mathbf{x}}(\pi) = \begin{cases} 1, & \mathbf{w}_{\mathbf{x}}(\pi) \neq (0,0,0); \\ 0, & \mathbf{w}_{\mathbf{x}}(\pi) = (0,0,0). \end{cases}$$

We use these to define the *fraction of sites in winding cycles*:

$$f_W(\pi) = \mathbb{E}\left[\frac{1}{N} \sum_{\mathbf{x} \in \Lambda} t_{\mathbf{x}}(\pi)\right].$$

Computational details are discussed in section 9.14.

3.7 Order parameters: quantifying long cycles

Of the random variables presented in this chapter, the following, referred to as *order parameters*, may be used to locate the critical temperature $T_c(\alpha)$ below which long cycles begin to appear. The first four are non-zero for $T < T_c$ and zero for $T > T_c$. Since ξ blows up below T_c, $1/\xi$ goes to zero below T_c. See also figure 3.3.

- $f_{\max} := \mathbb{E}[\ell_{\max}]/N$.
- Fraction of sites in long ("infinite") cycles f_I.
- Scaled winding number f_S.
- Fraction f_W of sites in cycles which wind.
- Reciprocal correlation length $1/\xi$.

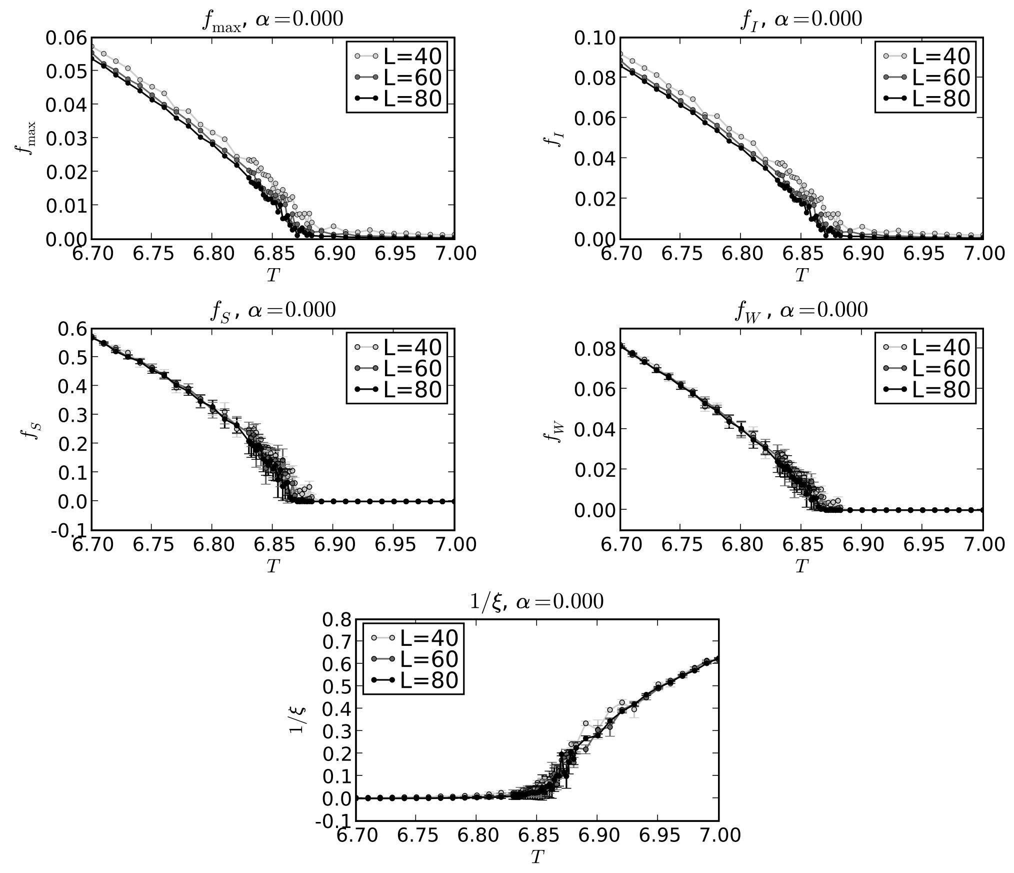


FIGURE 3.3. Behavior of order parameters as functions of L and T, for the non-interacting model. Each of the following occurs at a critical temperature T_c, in the limit $L \to \infty$: onset of N-scaling of the length of the longest cycle ($f_{\max}$), onset of long cycles (f_I), onset of winding cycles (f_S and f_W), and blow-up of correlation length (vanishing of $1/\xi$). For finite L, the transitions are smooth; they sharpen toward non-analyticity as $L \to \infty$. Interactions increase the critical temperature, shifting these graphs to the right.

Chapter 4
Markov chain Monte Carlo methods

In this chapter we discuss the need for random-sampling methods, and justify their use rigorously. Given a random variable $X(\pi)$, such as any of those presented in chapter 3, the expectation of X is (equation (2.1.6))

$$\mathbb{E}[X] = \sum_{\pi \in \mathcal{S}_N} P_{\text{Gibbs}}(\pi) X(\pi).$$

This is a real number, with no uncertainty. The problem is that the number of permutations, $N!$, grows intractably in N: even for $L = 10$ (and we consider L up to 80), $N = 1000$ and $N!$ is a number with over 5,000 digits. The true expectation is effectively incomputable. Expectations are instead *estimated* by summing over some number M (in the current work, 10^5 or 10^6) of typical permutations. The sample mean

$$\langle X \rangle_M = \frac{1}{M} \sum_{k=1}^{M} X(\pi_k) \tag{4.0.1}$$

depends on the random sequence $\pi_1, \ldots, \pi_M$. It is now a random variable with its own variance. The two main sources of error in MCMC simulations are *initialization bias* and *sampling variability.* The former involves *thermalization* (section 9.6) and *multimodality of distributions* (sections 5.4 and 7.8); the latter involves *autocorrelation* (section 9.15 and appendix B).

To create such a sequence of system states (for us, permutations), the method used throughout the computational physics community [Berg, LB] is Markov chain Monte Carlo. Namely, given a permutation π_k, one selects a successor permutation π_{k+1} in some random way. This is the Monte Carlo part. Moreover, the transition probabilities from π_k to each candidate π_{k+1} depend only on π_k, and not on any previous choices. This is the Markov chain part.

In the next sections we show (1) we can construct Markov chains which sample from the Gibbs distribution $P_{\text{Gibbs}}(\pi)$ (equation (2.1.4)); (2) other distributions (induced by the selection of initial state) converge to the Gibbs distribution; (3) the sample mean $\langle X \rangle_M$ converges almost surely to the true expectation $\mathbb{E}[X]$; and (4) the variance of $\langle X \rangle_M$ can be estimated, allowing us to place error bars on our estimates of $\mathbb{E}[X]$. Most of the material in this chapter is familiar: see [Berg, LB, CB, FG, GS] to name only a few. Results are restated here for self-containment of presentation.

4.1 Markov chains and Markov matrices

Before continuing to discuss random permutations and the Gibbs measure, we spend some time discussing more general random sequences, including Markov chains as a special case. This will turn out to be worthwhile: one of the strengths of this dissertation, in the author's estimation, is the careful disambiguation of some misleading notation and terminology (principally, overuse of the single letter P) which are encountered from time to time in the literature.

Let Ω be a finite set, and put $\#\Omega = K$. (For example, $\Omega = \mathcal{S}_N$ with $K = N!$.) The set of all sequences of elements of Ω, indexed by the non-negative integers, is $\Omega^{\mathbb{N}}$. Just as we can have an arbitrary probability measure P on Ω, we can have an arbitrary probability measure $\mathbf{P}$ on $\Omega^{\mathbb{N}}$. Marginal distributions on the kth slot are written P_k. If S_k is an Ω-valued random variable, e.g. a random selection from Ω at the kth slot, then $S_0, S_1, S_2, \ldots$ is a *random sequence*, or *discrete-time random process*. Since $\mathbf{P}$ is arbitrary, the P_i are not necessarily the same distributions, and samples S_i and S_j at the ith and jth slots are not necessarily independent.

Repeatedly using the conditional-probability formula $P(E \mid F) = P(E, F)/P(F)$ for events E and F, we can always split up the probability of a finite sequence of samples into a sequencing of initial and conditional probabilities:

$$\begin{aligned}\mathbf{P}(S_1 = \omega_1, S_2 = \omega_2, \ldots, S_n = \omega_n) =& \mathbf{P}(S_1 = \omega_1)\\ &\cdot \mathbf{P}(S_2 = \omega_2 \mid S_1 = \omega_1)\\ &\cdot \mathbf{P}(S_3 = \omega_3 \mid S_1 = \omega_1, S_2 = \omega_2)\\ &\cdot \mathbf{P}(S_n = \omega_n \mid S_1 = \omega_1, \cdots, S_{n-1} = \omega_{n-1}).\end{aligned} \tag{4.1.1}$$

A *Markov process* (or *Markov chain* if the state space Ω is finite) is a discrete-time random process such that for all $k > 0$,

$$\mathbf{P}(S_k = \omega_k \mid S_1 = \omega_1, S_2 = \omega_2, \ldots, S_{k-1} = \omega_{k-1}) = \mathbf{P}(S_k = \omega_k \mid S_{k-1} = \omega_{k-1}).$$

That is, if the probability of moving from one state to another depends only on the previous sample, and on nothing farther into the past, then the process is Markov. Now we have

$$\begin{aligned}\mathbf{P}(S_1 = \omega_1, \ldots, S_n = \omega_n) &= \mathbf{P}(S_1 = \omega_1)\\ &\cdot \mathbf{P}(S_2 = \omega_2 \mid S_1 = \omega_1) \cdots \mathbf{P}(S_n = \omega_n \mid S_{n-1} = \omega_{n-1}).\end{aligned} \tag{4.1.2}$$

We have the initial distribution for the first state, then transition probabilities for subsequent states. Precisely, one says a Markov chain is a discrete-time random process with this Markov property. With slight abuse of notation, though, we also refer to the probability distribution $\mathbf{P}$ as a Markov chain if it has this property, since given $\mathbf{P}$ we can always construct a discrete-time random process $S_0, S_1, S_2, \ldots$. Additionally, if for all $\omega, \omega' \in \Omega$ the conditional probabilities $\mathbf{P}(S_{k+1} = \omega' \mid S_k = \omega)$ are the same for all k, then we say the Markov chain is *homogeneous*.

Observe that $\mathbf{P}(S_{k+1} = \omega_j \mid S_k = \omega_i)$ is a $K \times K$ matrix of numbers between zero and one, with the property that rows sum to one (since each ω_i must transition to *some* ω_j). Such a matrix is called a *stochastic matrix* or *Markov matrix*. We might as well name that matrix A_k, with the entry in the ith row and jth column given by

$$(A_k)_{ij} = \mathbf{P}(S_{k+1} = \omega_j \mid S_k = \omega_i).$$

If the chain is homogeneous, we omit the subscript and write A. The key to making linear algebra out of this setup is the following *law of total probability*:

$$\begin{aligned}\mathbf{P}(S_{k+1} = \omega_j) &= \sum_{\omega_i} \mathbf{P}(S_k = \omega_i, S_{k+1} = \omega_j) \\ &= \sum_{\omega_i} \mathbf{P}(S_k = \omega_i)\mathbf{P}(S_{k+1} = \omega_j \mid S_k = \omega_i) \\ &= \sum_{\omega_i} \mathbf{P}(S_k = \omega_i)(A_k)_{ij}.\end{aligned} \tag{4.1.3}$$

The probability mass functions P_k are row vectors. The PMF P_{k+1} of S_{k+1} is the PMF P_k of S_k times the Markov matrix A_k. In vector/matrix notation,

$$P_{k+1} = P_k A_k.$$

Throughout this section, we supposed we had been given a probability distribution $\mathbf{P}$ on $\Omega^{\mathbb{N}}$ which satisfied the Markov property; we obtained Markov transition matrices. If, on the other hand, we start with an initial distribution P_0 and stochastic matrices $A_0, A_1, \ldots$, then we can re-use equation (4.1.3) to obtain $P_1 = P_0 A_0$ and, inductively, $P_{k+1} = P_k A_k$. We can then use equation (4.1.2) in reverse to obtain a probability distribution $\mathbf{P}$ on $\Omega^{\mathbb{N}}$. Note that now the process is Markov by construction.

In summary, a Markov chain is specified by a sequence-space distribution $\mathbf{P}$ with the Markov property, or an initial distribution P_0 and a sequence of transition matrices. Furthermore, we can go back and forth between these two points of view:

$$\mathbf{P} \longleftrightarrow (P_0, A_0, A_1, A_2, \ldots).$$

If the chain is homogeneous, we write

$$\mathbf{P} \longleftrightarrow (P_0, A).$$

4.2 Invariant and limiting distributions

The only probability distribution for random permutations considered thus far is the Gibbs distribution P_{Gibbs} of equation (2.1.4) on page 25. However, others exist. If one is asked for a permutation π and one always replies with the identity, then the distribution of that answer is the singleton measure supported on the identity. This is not the Gibbs measure (unless $\alpha = 0$ and $\beta = 0$). A third distribution is the uniform distribution, where each permutation has probability $1/N!$. (This is the same as the Gibbs measure only for $\alpha = 0$ and $T = 0$.)

Following the construction of the final paragraph of section 4.1, suppose we select an initial permutation π_0 from some probability distribution: for computational work done in this dissertation, this will be the singleton measure supported at the identity permutation, although a uniform-random π_0 is another possibility. Given Markov transition matrices A_k to be constructed in sections 5.2 and 7.6, we obtain a random sequence of permutations π_0, π_1, π_2, ... We write $\mathbf{P}^{(\pi_0)}$ for the probability distribution on this sequence space, with $P_k^{(\pi_0)}$ being the marginal at the kth slot.

Below, we will construct Markov transition matrices A_k which preserve the Gibbs measure P_{Gibbs}, i.e. $P_{\text{Gibbs}} = P_{\text{Gibbs}} A_k$. The following terminology applies.

Definition 4.2.1. A probability distribution P is *invariant* for a Markov transition matrix A if $P = PA$, that is, if for all $j = 1, \ldots, K$,

$$P(S_2 = \omega_j) = \sum_{i=1}^{K} A_{ij} P(S_1 = \omega_i).$$

In vector/matrix notation, this means

$$P = PA.$$

In other words, P is invariant for A if A-transitions preserve the distribution P. If S_1 is distributed according to P, then S_2 will also be distributed according to P, and so on. Such a sequence of states $S_1, S_2, \ldots$, while not in general independent, is identically distributed.

Remark 4.2.2. A homogeneous chain is not the same as a stationary sequence. In the former case, the transition matrix is the same at each time step; in the latter case, the probability distributions are the same at each time step.

Example 4.2.3. $\triangleright$ An illustrative example uses die-tipping. Recall that an ordinary six-sided die has pips on opposite faces summing to seven. There are six states, which we assume to be uniformly distributed if the die is rolled. Given that the die has n pips facing up, we may *tip* the die by picking one of the four sides at uniform random and putting that side up. E.g. if 1 is up, then after the tip, 2, 3, 4, or 5 will appear each with probability 1/4; 1 or 6 will appear with probability 0.

The transition matrix, with rows indexing the current state and columns indicating the successor state, is

$$A = \begin{pmatrix} 0 & 1/4 & 1/4 & 1/4 & 1/4 & 0 \\ 1/4 & 0 & 1/4 & 1/4 & 0 & 1/4 \\ 1/4 & 1/4 & 0 & 0 & 1/4 & 1/4 \\ 1/4 & 1/4 & 0 & 0 & 1/4 & 1/4 \\ 1/4 & 0 & 1/4 & 1/4 & 0 & 1/4 \\ 0 & 1/4 & 1/4 & 1/4 & 1/4 & 0 \end{pmatrix}.$$

Suppose the die is initially set on table with 1 up, i.e.

$$P_0 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Then

$$P_1 = P_0 A = \begin{pmatrix} 0 & 1/4 & 1/4 & 1/4 & 1/4 & 0 \end{pmatrix}$$

If, instead, the die is initially rolled, then P_0 is uniform:

$$P_0 = \begin{pmatrix} 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \end{pmatrix}.$$

One computes P_1 to be uniform as well:

$$P_1 = P_0 A = \begin{pmatrix} 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \end{pmatrix}.$$

The 1-up distribution is not invariant for the die-tipping transition rule, but the die-roll distribution is. $\triangleleft$

Given a Markov matrix A, one may wish to find an invariant distribution P. Going the other way, given a distribution P, one may wish to construct a Markov matrix A such that P is invariant with respect to A. The latter is our main goal here: the distribution of interest is the Gibbs distribution P_{Gibbs}. The following theorem is key. First, we define the terminology necessary to state it.

Definition 4.2.4. A Markov chain $\mathbf{P}$ on a state space Ω is *irreducible* if for all $\omega, \omega' \in \Omega$ there exists an $n > 0$ such that $\mathbf{P}(S_n = \omega' \mid S_0 = \omega) > 0$.

Definition 4.2.5. The *period* of $\omega \in \Omega$ is

$$p(\omega) = \gcd\{n : \mathbf{P}(S_n = \omega \mid S_0 = \omega) > 0\}$$

We say that ω has period p if it reappears with probability 1 after every p steps. A state ω is *aperiodic* if $p(\omega) = 1$. A chain is *aperiodic* if $p(\omega) = 1$ for every ω.

Remark. An irreducible, aperiodic chain on a finite state space is sometimes called *ergodic.*

Definition 4.2.6. A Markov matrix A on a state space Ω (with $\#\Omega = K$) and a distribution P on Ω are *reversible*, or satisfy *detailed balance*, if for all $1 \le i, j \le K$,

$$A_{ij}\, P(\omega_i) = A_{ji}\, P(\omega_j).$$

Theorem 4.2.7 (Invariant-distribution theorem)**.** *Fix a finite set Ω. Let A be a Markov transition matrix on Ω, as above, and let P be a probability distribution on Ω. If the homogeneous Markov chain (P, A) is irreducible, aperiodic, and satisfies detailed balance, then P is invariant for A. That is, the Markov chain (P, A) is stationary. Furthermore, if another homogeneous Markov chain (P_0, A) is irreducible and aperiodic, then for all $\omega \in \Omega$, $P_n(\omega) \to P(\omega)$ as $n \to \infty$.*

Remark. Some authors call this the *ergodic theorem*; yet, others call our theorem 4.2.9 the ergodic theorem. It also may be thought of as a special case of the Perron-Frobenius theorem, applied to the Markov transition matrix A.

Proof. See [Lawler], sections 1.2 and 1.3. $\square$

We say P (P_{Gibbs} in the context of random permutations) is the *stationary distribution* or *invariant distribution* for A. We say that it is also a *limiting distribution* for any initial distribution P_0 satisfying the above hypotheses.

Example 4.2.8. $\triangleright$ Continuing example 4.2.3: If we initially roll the die, we start with the uniform distribution which is stationary for the die-tipping rule:

$$\begin{aligned} P_0 &= \begin{pmatrix} 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 \end{pmatrix} \\ P_1 = P_0\, A &= \begin{pmatrix} 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 \end{pmatrix} \\ &\vdots \end{aligned}$$

If we start with the one-face up, we begin with the singleton initial distribution which is not stationary for the die-tipping rule. Yet, subsequent tips have a distribution which tends toward the limiting, uniform distribution:

$$\begin{aligned} P_0 &= \begin{pmatrix} 1.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \end{pmatrix} \\ P_1 = P_0\, A &= \begin{pmatrix} 0.0000 & 0.2500 & 0.2500 & 0.2500 & 0.2500 & 0.0000 \end{pmatrix} \\ P_2 = P_1\, A &= \begin{pmatrix} 0.2500 & 0.1250 & 0.1250 & 0.1250 & 0.1250 & 0.2500 \end{pmatrix} \\ P_3 = P_2\, A &= \begin{pmatrix} 0.1250 & 0.1875 & 0.1875 & 0.1875 & 0.1875 & 0.1250 \end{pmatrix} \\ P_4 = P_3\, A &= \begin{pmatrix} 0.1875 & 0.1562 & 0.1562 & 0.1562 & 0.1562 & 0.1875 \end{pmatrix} \\ P_5 = P_4\, A &= \begin{pmatrix} 0.1562 & 0.1719 & 0.1719 & 0.1719 & 0.1719 & 0.1562 \end{pmatrix} \\ &\vdots \\ P_{14} = P_{13}\, A &= \begin{pmatrix} 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 \end{pmatrix} \\ P_{15} = P_{14}\, A &= \begin{pmatrix} 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 \end{pmatrix} \\ &\vdots \end{aligned}$$

$\triangleleft$

Importantly, for simulations using the model of random spatial permutations, we need not know *a priori* what a typical permutation looks like. We may start always with the identity permutation, i.e. P_0 is the singleton distribution supported on the identity permutation. We may then run the Markov chain, producing a sequence of permutations. As the number of iterations goes to infinity, the distribution of permutations approaches P_{Gibbs}: for all $\pi \in \mathcal{S}_N$, $P_k^{(\pi_0)}(\pi) \to P_{\text{Gibbs}}(\pi)$ as $k \to \infty$.

The specific number k of iterations needed for convergence of $P_k^{(\pi_0)}$ to P_{Gibbs} is another matter entirely. The theory exposited by [Lawler], as noted above, guarantees that a Markov matrix A with P_{Gibbs} as its invariant distribution has no other invariant distribution: A has a single eigenvalue 1 with eigenvector P_{Gibbs}. The rate of convergence of $P_k^{(\pi_0)}$ to P_{Gibbs} depends on the second-largest eigenvalue for A, which one in general does not know how to compute. In practice, this *mixing time* (or *burn-in time* or *thermalization time*) is estimated using techniques such as those in section 9.6.

The theory above also does not tell us how to construct a Markov matrix A having a desired distribution P_{Gibbs} as its invariant distribution — it simply tells us what we can do once we have constructed such a matrix. A specific construction is due to Nicholas Metropolis [Berg, LB, CB]. The essence is that if the invariant probability distribution P_{Gibbs} is defined as a Gibbs measure via an energy function H on Ω, then proposed successor states π_{k+1} of π_k are accepted with probability $\min\{1, e^{-\Delta H}\}$ where ΔH is the energy difference for the state change. In this dissertation, I directly prove that such methods result in detailed balance. Thus, the reader is referred to propositions 5.3.6 and 7.7.5 for details.

We next show that the sample mean $\langle X \rangle_M$ (equation (4.0.1) on page 47) converges to the true mean $\mathbb{E}[X]$ (equation (2.1.6) on page 26).

Theorem 4.2.9 (Ergodic sampling theorem)**.** *Let X be a random variable on the finite probability space $(\Omega, 2^\Omega, P)$ where 2^Ω is the power set of Ω. If the stationary,* *homogeneous Markov chain (P, A) satisfies the hypotheses of theorem 4.2.7, then*

$$\frac{1}{M}\sum_{k=1}^{M} X(S_k) \to \mathbb{E}[X] \quad \textit{as} \quad M \to \infty.$$

Remark. Some authors call this this *ergodic theorem*; yet, others call our theorem 4.2.7 the ergodic theorem.

Proof. This follows from theorem 4.2.7 using the central-limit theorem for identically distributed but non-independent S_i's. □

4.3 Sample means and their variances

There is one caveat to replacing the true mean $\mathbb{E}[X]$ with the MCMC sample mean $\langle X \rangle_M$: the naive computation of the standard deviation of the sample mean, which is correct for independent identically distributed states, is a significantly incorrect underestimate for the standard deviation of the sample mean in the case of identically distributed but correlated states. This issue is so important that appendix B is devoted to it. The key result of that appendix is that the standard deviation (error bar) of the sample mean $\langle X \rangle_M$ is off by a factor of the square root of the *integrated autocorrelation time*, $\sqrt{\hat{\tau}_{\text{int}}(X)}$, which is computed as described in section B.10.

4.4 Simple example: 1D Ising

For an example which is more complex than die-tipping but less complex than random spatial permutations, consider the 1D N-point Ising model. Namely, the configuration space is $\Omega = \{\pm 1\}^N$, i.e. N particles which may be in either an up (filled) or a down (hollow) state:

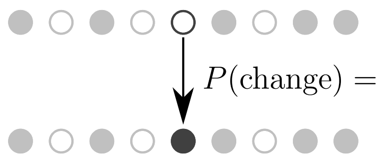
● ○ ● ○ ○ ● ○ ● ●

A state is described by $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_n)$. The configuration space Ω has 2^N possible configurations. The system is endowed with an energy function. For the 1D Ising model, one has

$$H(\boldsymbol{\omega}) = \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij}\omega_i\omega_j + \sum_{i=1}^{n} h_i\omega_i.$$

where the C_{ij}'s are interaction terms (non-interacting, nearest neighbor, mean-field, etc.) and the h_i's are magnetization terms. Given a temperature-related parameter β, one sets the (Gibbs) probability of each configuration to be proportional to $e^{-\beta H}$.

One picks an initial configuration. There are three obvious choices: (1) Start with all spins down, i.e. $\boldsymbol{\omega} = (-1, \ldots, -1)$. (2) Start with all spins up, i.e. $\boldsymbol{\omega} = (+1, \ldots, +1)$. (3) Start with $\boldsymbol{\omega}$ selected from a uniform probability distribution on Ω. Then, one selects a site i and decides whether to flip ω_i to $-\omega_i$.



This decision is made using the Metropolis prescription, namely:

- One computes the change in energy $\Delta H = H(\boldsymbol{\omega}') - H(\boldsymbol{\omega})$ which would be obtained if $\boldsymbol{\omega}$ were sent to $\boldsymbol{\omega}'$ by flipping ω_i.
- One may compute ΔH by separately computing $H(\boldsymbol{\omega}')$ and $H(\boldsymbol{\omega})$ and subtracting the two. However, since the only change is at the site i, one may do some ad-hoc algebra to derive an expression for ΔH which is less computationally expensive.
- One accepts the change with probability $\min\{1, e^{-\Delta H}\}$.

This is called a *Metropolis step*.

Looping through all n sites from $i = 1$ to $i = n$, performing a Metropolis step at each site i, is called a *Metropolis sweep*. If one realizes a random variable $X(\boldsymbol{\omega})$ at each of M sweeps, averaging X over the M sweeps, one obtains an approximation $\langle X \rangle_M$ for the expectation $\mathbb{E}[X]$.

As discussed at the end of section 4.2, one should first run some number B of Metropolis sweeps of the system until it is thermalized, i.e. until the Gibbs distribution has been approached. One should discard the B realizations of the random variable X obtained during thermalization, before running the M sweeps in which data are accumulated. The B sweeps are called the thermalization phase; the M sweeps are called the accumulation phase.

4.5 Recipe for MCMC algorithms

The naive outline of an MCMC run is simple:

- Use a Markov chain, discussed at the end of this section, to generate a sequence $\pi_1, \ldots, \pi_M$ of permutations.
- For each permutation π_k, for each random variable X of interest, remember the value $X_k = X(\pi_k)$.
- Compute the sample mean $\overline{X} = \frac{1}{M} \sum_{k=1}^{M} X_k$. Also compute the sample standard deviation, and any other desired statistics.
- Display the statistics.

Since the initial permutation is the identity, the initial distribution is the singleton supported at the identity, which is not the Gibbs distribution P_{Gibbs}. Furthermore, as a very low-energy state, the identity is highly non-typical with respect to the Gibbs distribution (equation (2.1.4)) for the model of spatial permutations. As discussed at the end of section 4.2, one runs the chain until it is thermalized, i.e. until the Gibbs distribution has been approached. (The number of steps τ required for this is random, but it turns out to fall within a narrow range.) Renumbering π_τ to π_0, one then accumulates statistics over the M permutations $\pi_1, \ldots, \pi_M$. See section 9.6 for the thermalization-detection algorithm used in this dissertation.

Thus the computational recipe is as follows:

- Start with the initial permutation being the identity permutation.
- Run the Markov chain, generating a sequence of permutations until thermalization has been detected. At that point, rename the current permutation π_0.
- Continue generating a sequence $\pi_1, \ldots, \pi_M$ of permutations. The mechanics of transitioning from π_k to π_{k+1} comprises a series of *steps*, which collectively form the kth *sweep*. Various types of sweep — swap-only, swap-and-reverse, swaps with band updates, and worm — are presented in chapters 5, 6, and 7.
- For each permutation π_k, for each random variable X of interest, remember the value $X_k = X(\pi_k)$.
- After all M permutations have been generated, compute the sample mean $\overline{X} = \frac{1}{M}\sum_{k=1}^{M} X_k$. Also compute the sample standard deviation and its error bar (using estimated integrated autocorrelation time), and any other desired statistics, such as histograms.
- Display the statistics.

Given the framework established by previous sections of this chapter, the recipe to prove correctness of this algorithm reduces to the following: define a Markov chain, then prove irreducibility, aperiodicity, and detailed balance. We devote the next chapters to present three Markov chains: the swap-only algorithm, the swap-and-reverse algorithms, and the worm algorithm.

Chapter 5
The swap-only and swap-and-reverse algorithms

This chapter (along with chapter 7) presents Markov chains for MCMC sampling of the model of random spatial permutations (chapter 2) within the MCMC-recipe framework of chapter 4. The algorithms are proved correct, then compared and contrasted. Computational results using the swap-and-reverse algorithm are given in chapter 11.

5.1 The swap-only algorithm

The swap-only algorithm for transitioning from π to π', within the context of the recipe in section 4.5, is as follows. One sweeps through sites $\mathbf{x}$ of the lattice in either sequential or uniform-random order. In either case[1], one obtains a lattice site $\mathbf{x}$. One then does a *Metropolis step* at site $\mathbf{x}$:

- Choose a site $\pi(\mathbf{y})$ from among the six nearest neighbors of $\pi(\mathbf{x})$.
- Propose to change π to the permutation π' which has $\pi'(\mathbf{z}) = \pi(\mathbf{z})$ for all $\mathbf{z} \neq \mathbf{x}, \mathbf{y}$ but $\pi'(\mathbf{x}) = \pi(\mathbf{y})$ and $\pi'(\mathbf{y}) = \pi(\mathbf{x})$. (See figure 5.1.)
- With probability $\min\{1, e^{-\Delta H}\}$ where $\Delta H = H(\pi') - H(\pi)$, accept the change. (If the change is rejected, set $\pi' = \pi$.)

Definition 5.1.1. A swap is *trivial* if $\mathbf{x} = \mathbf{y}$.

[1]For computational results presented in chapter 11, site selection was sequential. Experiments show that sequential site selection and random site selection produce indistinguishable results, for our model and for the near-critical temperature range we consider.

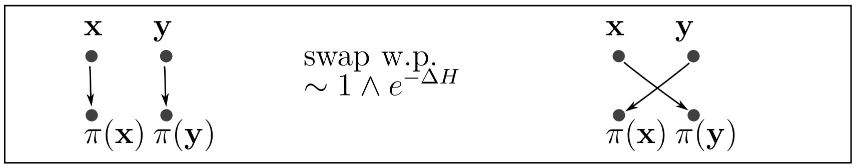


FIGURE 5.1. Metropolis moves for the swap-only algorithm.

5.2 Explicit construction of the Markov matrices

For section 5.3 we will need an explicit construction of the Markov matrices corresponding to the swap-only algorithm as described in section 5.1.

The Markov perspective on the algorithm is that, given a probability distribution $P_k^{(\pi_0)}(\pi)$, the distribution for the subsequent permutation is

$$P_{k+1}^{(\pi_0)}(\pi') = \sum_{\pi \in \mathcal{S}_N} P_k^{(\pi_0)}(\pi) A_k(\pi, \pi')$$

or, in matrix/vector notation, $P_{k+1}^{(\pi_0)} = P_k^{(\pi_0)} A_k$. In this section we precisely describe the matrices A_k; in section 5.3 we show that $P_k^{(\pi_0)}$ approaches the Gibbs distribution P_{Gibbs} (equation (2.1.4)) as $k \to \infty$.

The matrices A_k are $N! \times N!$, with rows indexed by $\pi_1, \ldots, \pi_{N!}$ and columns indexed by $\pi'_1, \ldots, \pi'_{N!}$. Most of the entries of A_k are zero: Metropolis steps change only two permutation sites whereas most π, π' differ at more than two sites.

Definition 5.2.1. For $\pi, \pi' \in \mathcal{S}_N$, define

$$d(\pi, \pi') = \#\{i = 1, 2, \ldots, N : \pi(i) \neq \pi'(i)\}.$$

Remark. Note that $d(\pi, \pi') \neq 1$ since if two permutations agree on $N-1$ sites, they must agree on the remaining site.

Lemma 5.2.2. *The function $d(\pi, \pi')$ is a metric on $\mathcal{S}_N$.*

Proof. Symmetry is obvious, as is non-negativity. For positive definiteness, note that $d(\pi, \pi') = 0$ iff $\pi = \pi'$. For the triangle inequality, let $\pi, \pi', \pi'' \in \mathcal{S}_N$. Partition the set $\{1, 2, \ldots, N\}$ into the four disjoint sets

$$\begin{aligned}
A &= \{i = 1, 2, \ldots, N : \pi(i) = \pi'(i), \pi'(i) = \pi''(i)\}, \\
B &= \{i = 1, 2, \ldots, N : \pi(i) = \pi'(i), \pi'(i) \neq \pi''(i)\}, \\
C &= \{i = 1, 2, \ldots, N : \pi(i) \neq \pi'(i), \pi'(i) = \pi''(i)\}, \\
D &= \{i = 1, 2, \ldots, N : \pi(i) \neq \pi'(i), \pi'(i) \neq \pi''(i)\}.
\end{aligned}$$

Then $\pi = \pi''$ on all of A; $\pi \neq \pi''$ on all of B and C; and π, π'' may or may not agree on various elements of D:

A	B	C	D
$\pi = \pi'$	$\pi = \pi'$	$\pi \neq \pi'$	$\pi \neq \pi'$
$\pi' = \pi''$	$\pi' \neq \pi''$	$\pi' = \pi''$	$\pi' \neq \pi''$
$\pi = \pi''$	$\pi \neq \pi''$	$\pi \neq \pi''$	Varies

That is,

$$\begin{aligned}
d(\pi, \pi') &= \#C + \#D, \\
d(\pi', \pi'') &= \#B + \#D, \\
\#B + \#C \leq d(\pi, \pi'') &\leq \#B + \#C + \#D.
\end{aligned}$$

Then

$$d(\pi, \pi'') \leq \#B + \#C + \#D \leq \#B + \#C + 2\#D = d(\pi, \pi') + d(\pi', \pi'').$$

$\square$

Definition 5.2.3. Lattice sites $\mathbf{x}$ and $\mathbf{y}$ are *nearest-neighbor* if $\|\mathbf{x} - \mathbf{y}\|_\Lambda = 1$.

Definition 5.2.4. For $\pi \in \mathcal{S}_N$ and $\mathbf{x} \in \Lambda$, define

$$R_{\mathbf{x}}(\pi) = \{\pi' \in \mathcal{S}_N : d(\pi, \pi') = 2 \text{ and } \|\pi(\mathbf{x}) - \pi(\mathbf{y})\|_\Lambda = 1\}$$

where the $\mathbf{x}$ and $\mathbf{y}$ are taken to be the two points at which π and π' differ. Then $R_{\mathbf{x}}(\pi)$ is the set of permutations π' reachable from π on a swap involving site $\mathbf{x}$. This is used for sequential site selection. Likewise, for use with random site selection, define

$$R(\pi) = \{\pi' \in \mathcal{S}_N : d(\pi, \pi') = 2 \text{ and } \|\pi(\mathbf{x}) - \pi(\mathbf{y})\|_\Lambda = 1\}$$

where the $\mathbf{x}$ and $\mathbf{y}$ are taken to be the two points at which π, π' differ. Then $R(\pi)$ is the set of permutations π' reachable from π on a swap. We also write

$$\pi' \multimap\!\!\circ\, \pi$$

if $\pi' \in R_{\mathbf{x}}(\pi)$ or $\pi' = \pi$ (for sequential site selection), or $\pi' \in R(\pi)$ or $\pi' = \pi$ (for random site selection).

The Metropolis steps are then described as follows. First consider sequential site selection. For each $\pi \in \mathcal{S}_N$,

$$A_{\mathbf{x}}(\pi, \pi') = \begin{cases} \frac{1}{6}\left(1 \wedge e^{-H(\pi')+H(\pi)}\right), & \pi' \in R_{\mathbf{x}}(\pi), \\ 1 - \displaystyle\sum_{\pi'' \in R(\pi)} \frac{1}{6}\left(1 \wedge e^{-H(\pi'')+H(\pi)}\right), & \pi = \pi'; \\ 0, & \text{otherwise.} \end{cases} \tag{5.2.5}$$

To justify the choice of prefactor 1/6, note that we choose one of six $\pi(\mathbf{y})$ at uniform random from the six nearest-neighbor lattice sites of $\pi(\mathbf{x})$. If the change is accepted, then we obtain π by swapping at $\mathbf{x}$ and $\mathbf{y}$; otherwise, we take $\pi' = \pi$.

Next we construct the Markov matrix for random site selection. For each $\pi \in \mathcal{S}_N$,

$$A(\pi, \pi') = \begin{cases} \frac{1}{3N}\left(1 \wedge e^{-H(\pi')+H(\pi)}\right), & \pi' \in R(\pi), \\ 1 - \displaystyle\sum_{\pi'' \in R(\pi)} \frac{1}{3N}\left(1 \wedge e^{-H(\pi'')+H(\pi)}\right), & \pi = \pi'; \\ 0, & \text{otherwise.} \end{cases} \tag{5.2.6}$$

To justify the choice of prefactor $1/3N$, note that there are N choices of lattice points $\mathbf{x}$. For each $\mathbf{x}$, there are 6 choices of $\pi(\mathbf{y})$ which are nearest neighbors to $\pi(\mathbf{x})$. This double-counts the $3N$ distinct choices of π' reachable from π in a single Metropolis step, since choosing $\mathbf{x}$ and then $\mathbf{y}$ results in the same Metropolis step as choosing $\mathbf{y}$ and then $\mathbf{x}$.

The use of the Markov matrices in practice is as follows. Number the lattice sites $\mathbf{x}_1, \ldots, \mathbf{x}_N$. When using sequential site selection, the kth Metropolis sweep (as described in section 4.5) begins with a permutation π distributed according to $P_k^{(\pi_0)}$. A Metropolis step is done at site $\mathbf{x}_1$, using transition matrix $A_{\mathbf{x}_1}$, followed by a Metropolis step at site $\mathbf{x}_2$, using transition matrix $A_{\mathbf{x}_2}$, and so on up to site $\mathbf{x}_N$. The sweep is then complete, and the distribution of π' is

$$P_{k+1}^{(\pi_0)}(\pi') = P_k^{(\pi_0)}(\pi)\, A_k, \qquad\qquad A_k = A_{\mathbf{x}_N} \cdots A_{\mathbf{x}_1}.$$

The chain is non-homogeneous, if we consider all the intermediate permutations after each Metropolis step. Yet, at the level of Metropolis sweeps, the chain is homogeneous since at each sweep we apply the composite transition matrix $A_{\mathbf{x}_N} \cdots A_{\mathbf{x}_1}$ to obtain π' from π.

When we use random site selection, the kth Metropolis sweep begins with a permutation π distributed according to $P_k^{(\pi_0)}$. We do N Metropolis steps, each using the transition matrix A, each selecting a site $\mathbf{x}$ at uniform random from among $\mathbf{x}_1, \ldots, \mathbf{x}_N$. The sweep is then complete, and the distribution of π' is

$$P_{k+1}^{(\pi_0)}(\pi') = P_k^{(\pi_0)}(\pi)\, A_k, \qquad\qquad A_k = A^N.$$

The chain is homogeneous, whether viewed at the level of Metropolis steps or Metropolis sweeps.

5.3 Correctness of the swap-only algorithm

It is clear that the swap-only algorithm produces a sequence of permutations, but with what distribution? From the Markov-chain theory in section 4.2, we know that if the chain is irreducible, aperiodic, and satisfies detailed balance, then the chain has the Gibbs distribution (equation (2.1.4)) as its unique invariant distribution. All the results in this section will apply for sequential or random site selection; in this section, we write A to refer to either A or $A_{\mathbf{x}}$.

Proposition 5.3.1 (Irreducibility). *For all π, π', there is an n such that $A^n(\pi, \pi') > 0$. That is, any permutation is reachable from any other.*

Proof. Transpositions generate $\mathcal{S}_N$ [DF]. Thus, for all $\pi \in \mathcal{S}_N$, there exist transpositions $\sigma_1, \ldots, \sigma_m$ such that $\pi = \prod_{j=1}^m \sigma_j$. Thus, it suffices to show that given any permutation π and any two points $\mathbf{x}$ and $\mathbf{z}$, so $\pi : \mathbf{x} \mapsto \pi(\mathbf{x})$ and $\pi : \mathbf{z} \mapsto \pi(\mathbf{z})$, we can construct a sequence of swaps sending π to π' so that $\pi' : \mathbf{x} \mapsto \pi(\mathbf{z})$, $\pi' : \mathbf{z} \mapsto \pi(\mathbf{x})$, and $\pi'(\mathbf{y}) = \pi(\mathbf{y})$ for all $\mathbf{y} \neq \mathbf{x}, \mathbf{z}$. (If $\pi(\mathbf{x})$ and $\pi(\mathbf{z})$ are nearest-neighbor lattice sites, of course, then a single swap does the job.)

Define $G_{\mathbf{a},\mathbf{b}} : \mathcal{S}_N \to \mathcal{S}_N$ to be the swap operator for nearest-neighbor lattice sites $\pi(\mathbf{a})$ and $\pi(\mathbf{b})$, i.e. $\pi' = G_{\mathbf{a},\mathbf{b}}\pi$. Given $\mathbf{x}$ and $\mathbf{z}$, there is a (non-unique) sequence of lattice sites $\mathbf{y}_0, \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n$ such that $\mathbf{y}_0 = \mathbf{x}$, $\mathbf{y}_n = \mathbf{z}$, and $\|\pi(\mathbf{y}_{i+1}) - \pi(\mathbf{y}_i)\|_\Lambda = 1$ for $i = 0, 1, \ldots, n-1$. (See figure 5.2.) We will construct a sequence of swaps along this nearest-neighbor path whose end result is to swap the permutation arrows starting at $\mathbf{x}$ and $\mathbf{z}$, leaving all other arrows unchanged. We first need a lemma about compositions of swaps.

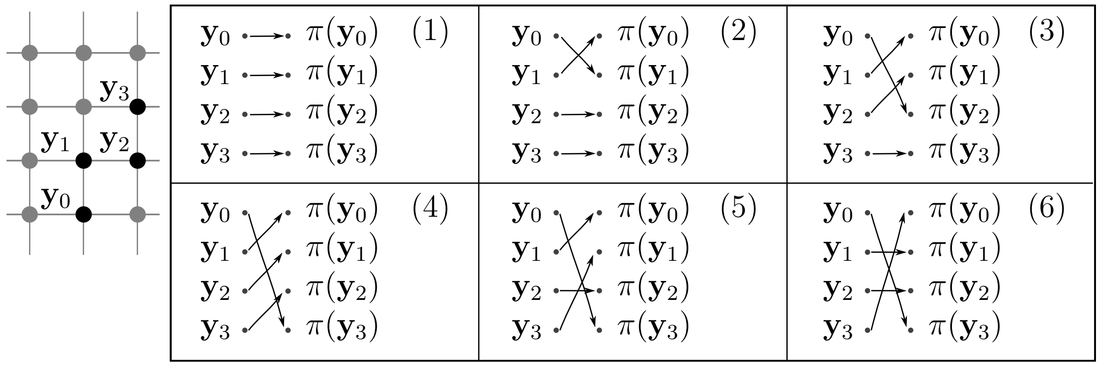


FIGURE 5.2. A sequence of (nearest-neighbor) swaps which results in a non-nearest-neighbor swap.

Notation 5.3.2. Given $\mathbf{x}_1, \ldots, \mathbf{x}_N$ and a permutation π, we may write π as an *image* *map* with the $\mathbf{x}_i$'s along the top row and their images along the bottom row:

$$\begin{pmatrix} \mathbf{x}_1 & \dots & \mathbf{x}_N \\ \pi(\mathbf{x}_1) & \dots & \pi(\mathbf{x}_N) \end{pmatrix}$$

Lemma 5.3.3. *The composition $G_{\mathbf{a},\mathbf{c}} \circ G_{\mathbf{a},\mathbf{b}}$ behaves as follows:*

$$\pi = \begin{pmatrix} \dots & \mathbf{a} & \mathbf{b} & \mathbf{c} & \dots \\ \dots & \pi(\mathbf{a}) & \pi(\mathbf{b}) & \pi(\mathbf{c}) & \dots \end{pmatrix} \mapsto \pi' = \begin{pmatrix} \dots & \mathbf{a} & \mathbf{b} & \mathbf{c} & \dots \\ \dots & \pi(\mathbf{c}) & \pi(\mathbf{a}) & \pi(\mathbf{b}) & \dots \end{pmatrix}$$

Proof. The first map, $G_{\mathbf{a},\mathbf{b}}$, does

$$\begin{pmatrix} \dots & \mathbf{a} & \mathbf{b} & \mathbf{c} & \dots \\ \dots & \pi(\mathbf{a}) & \pi(\mathbf{b}) & \pi(\mathbf{c}) & \dots \end{pmatrix} \mapsto \begin{pmatrix} \dots & \mathbf{a} & \mathbf{b} & \mathbf{c} & \dots \\ \dots & \pi(\mathbf{b}) & \pi(\mathbf{a}) & \pi(\mathbf{c}) & \dots \end{pmatrix};$$

$G_{\mathbf{a},\mathbf{c}}$ sends this to

$$\begin{pmatrix} \dots & \mathbf{a} & \mathbf{b} & \mathbf{c} & \dots \\ \dots & \pi(\mathbf{c}) & \pi(\mathbf{a}) & \pi(\mathbf{b}) & \dots \end{pmatrix}.$$

□

Corollary 5.3.4. *The composition $G_{\mathbf{y}_0,\mathbf{y}_n} \circ G_{\mathbf{y}_0,\mathbf{y}_{n-1}} \circ \dots \circ G_{\mathbf{y}_0,\mathbf{y}_2} \circ G_{\mathbf{y}_0,\mathbf{y}_1}$, sending $\pi \mapsto \pi'$, performs the right cyclic shift on images of $\mathbf{y}_0, \dots, \mathbf{y}_n$ given by*

$$\begin{pmatrix} \mathbf{y}_0 & \mathbf{y}_1 & \dots & \mathbf{y}_{n-1} & \mathbf{y}_n \\ \pi(\mathbf{y}_0) & \pi(\mathbf{y}_1) & \dots & \pi(\mathbf{y}_{n-1}) & \pi(\mathbf{y}_n) \end{pmatrix} \mapsto \begin{pmatrix} \mathbf{y}_0 & \mathbf{y}_1 & \dots & \mathbf{y}_{n-1} & \mathbf{y}_n \\ \pi(\mathbf{y}_n) & \pi(\mathbf{y}_0) & \dots & \pi(\mathbf{y}_{n-2}) & \pi(\mathbf{y}_{n-1}) \end{pmatrix}.$$

Likewise, $G_{\mathbf{y}_n,\mathbf{y}_1} \circ G_{\mathbf{y}_n,\mathbf{y}_2} \circ \dots \circ G_{\mathbf{y}_n,\mathbf{y}_{n-2}} \circ G_{\mathbf{y}_n,\mathbf{y}_{n-1}}$ leaves the image of $\mathbf{y}_0$ unchanged and performs the left cyclic shift on images of $\mathbf{y}_1, \dots, \mathbf{y}_n$ given by

$$\begin{pmatrix} \mathbf{y}_0 & \mathbf{y}_1 & \dots & \mathbf{y}_{n-1} & \mathbf{y}_n \\ \pi(\mathbf{y}_0) & \pi(\mathbf{y}_1) & \dots & \pi(\mathbf{y}_{n-1}) & \pi(\mathbf{y}_n) \end{pmatrix} \mapsto \begin{pmatrix} \mathbf{y}_0 & \mathbf{y}_1 & \dots & \mathbf{y}_{n-1} & \mathbf{y}_n \\ \pi(\mathbf{y}_0) & \pi(\mathbf{y}_2) & \dots & \pi(\mathbf{y}_n) & \pi(\mathbf{y}_1) \end{pmatrix}.$$

Proof. These follow from the lemma by induction on n. □

Composing these two maps, we find that

$$(G_{\mathbf{y}_n,\mathbf{y}_1} \circ G_{\mathbf{y}_n,\mathbf{y}_2} \circ \dots \circ G_{\mathbf{y}_n,\mathbf{y}_{n-2}} \circ G_{\mathbf{y}_n,\mathbf{y}_{n-1}}) \circ (G_{\mathbf{y}_0,\mathbf{y}_n} \circ G_{\mathbf{y}_0,\mathbf{y}_{n-1}} \circ \dots \circ G_{\mathbf{y}_0,\mathbf{y}_2} \circ G_{\mathbf{y}_0,\mathbf{y}_1})$$

swaps the images of $\mathbf{x} = \mathbf{y}_0$ and $\mathbf{z} = \mathbf{y}_n$ while leaving all other images unchanged, that is,

$$\begin{pmatrix} \mathbf{y}_0 & \mathbf{y}_1 & \dots & \mathbf{y}_{n-1} & \mathbf{y}_n \\ \pi(\mathbf{y}_0) & \pi(\mathbf{y}_1) & \dots & \pi(\mathbf{y}_{n-1}) & \pi(\mathbf{y}_n) \end{pmatrix} \mapsto \begin{pmatrix} \mathbf{y}_0 & \mathbf{y}_1 & \dots & \mathbf{y}_{n-1} & \mathbf{y}_n \\ \pi(\mathbf{y}_n) & \pi(\mathbf{y}_1) & \dots & \pi(\mathbf{y}_{n-1}) & \pi(\mathbf{y}_0) \end{pmatrix}.$$

This ends the proof of propostion 5.3.1. □

Remark. Below we will discuss winding cycles, and the empirical fact that the swap-only algorithm reaches them only rarely. The chain is irreducible but the non-zero transition probabilities can still be very small.

Proposition 5.3.5 (Aperiodicity)**.** *The swap-only algorithm's Markov chain is aperiodic.*

Proof. This follows from irreducibility, which says in particular that for every π, there is an integer m such that $A^m(\pi,\pi) > 0$. Then $A^n(\pi,\pi) > 0$ for all $n > m$, implying $p(\pi) = 1$. □

Proposition 5.3.6 (Detailed balance)**.** *For all $\pi, \pi' \in \mathcal{S}_N$,*

$$P_{\text{Gibbs}}(\pi)A(\pi,\pi') = P_{\text{Gibbs}}(\pi')A(\pi',\pi). \tag{5.3.7}$$

Remark. For the swap-only algorithm, this is a trivial result. We work through the details in order to foreshadow the non-trivial construction of proposition 7.7.5 for the worm algorithm.

Proof. The detailed-balance statement in terms of the Gibbs distribution (equation (2.1.4)) and the swap-only Metropolis transition matrices (equations (5.2.5) and (5.2.6)) is

$$\frac{e^{-H(\pi)}}{Z}\left(1 \wedge e^{-H(\pi')}e^{H(\pi)}\right) \overset{?}{=} \frac{e^{-H(\pi')}}{Z}\left(1 \wedge e^{-H(\pi)}e^{H(\pi')}\right).$$

The Z's cancel. The lemma below shows that $A(\pi,\pi') \neq 0$ iff $A(\pi',\pi) \neq 0$. If $A(\pi,\pi') = 0$, then detailed balance holds. If $A(\pi,\pi') \neq 0$, then there are two cases. If $H(\pi') \leq H(\pi)$, then

$$e^{-H(\pi)}\left(1\right) = e^{-H(\pi')}\left(e^{-H(\pi)}e^{H(\pi')}\right).$$

If $H(\pi') > H(\pi)$,

$$e^{-H(\pi)}\left(e^{-H(\pi')}e^{H(\pi)}\right) = e^{-H(\pi')}\left(1\right).$$

In all cases, detailed balance holds. □

Lemma 5.3.8. *For all $\pi, \pi' \in \mathcal{S}_N$,*

$$A(\pi, \pi') \neq 0 \iff A(\pi', \pi) \neq 0.$$

Proof. As a direct consequence of definition 5.2.4 of $R(\pi)$, $\pi' \in R(\pi)$ if and only if $\pi \in R(\pi')$. The same holds for $A_{\mathbf{x}}$ and $R_{\mathbf{x}}(\pi)$. $\square$

This lemma completes the proof that the swap-only algorithm satisfies detailed balance and thus has the Gibbs distribution as its invariant distribution.

The following proposition is not a correctness result, but rather a sanity check. It shows that cycles may grow or shrink upon swaps.

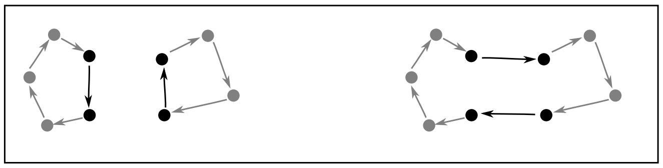


FIGURE 5.3. Swaps merge disjoint cycles and split single cycles. The left-hand permutation can be reached from the right-hand permutation via a swap, and vice versa.

Proposition 5.3.9. *If $\mathbf{x}$ and $\mathbf{y}$ are in disjoint cycles before a non-trivial swap at $\mathbf{x}$ and $\mathbf{y}$, then they are in the same cycle afterward and vice versa (see figure 5.3).*

Proof. First suppose that $\mathbf{x}$ and $\mathbf{y}$ are in disjoint cycles. Let the respective cycle lengths be $\ell(\mathbf{x}) = a$ and $\ell(\mathbf{y}) = b$. Recall that $1 \leq a, b \leq N$. Those cycles are

$$\mathbf{x} \mapsto \pi(\mathbf{x}) \mapsto \pi^2(\mathbf{x}) \mapsto \ldots \mapsto \pi^{a-1}(\mathbf{x}) \mapsto \mathbf{x}$$

and

$$\mathbf{y} \mapsto \pi(\mathbf{y}) \mapsto \pi^2(\mathbf{y}) \mapsto \ldots \mapsto \pi^{b-1}(\mathbf{y}) \mapsto \mathbf{y}.$$

Since these are disjoint cycles, all elements listed are distinct lattice sites. After the swap, we have

$$\mathbf{y} \mapsto \pi(\mathbf{x}) \mapsto \pi^2(\mathbf{x}) \mapsto \ldots \mapsto \pi^{a-1}(\mathbf{x}) \mapsto \mathbf{x}$$

and

$$\mathbf{x} \mapsto \pi(\mathbf{y}) \mapsto \pi^2(\mathbf{y}) \mapsto \ldots \mapsto \pi^{b-1}(\mathbf{y}) \mapsto \mathbf{y}.$$

This is a single cycle of length $a+b$, starting with $\mathbf{y}$, including $\mathbf{x}$, and returning to $\mathbf{y}$.

Second, suppose that $\mathbf{x}$ and $\mathbf{y}$ are in the same cycle. Let $a = \ell_{\mathbf{x},\mathbf{y}}(\pi)$ and $b = \ell_{\mathbf{x},\mathbf{y}}(\pi)$ (see definition 3.2.2). (These numbers are both positive since the swap is non-trivial, i.e. $\mathbf{x} \neq \mathbf{y}$.) Then we have

$$\mathbf{x} \mapsto \pi(\mathbf{x}) \mapsto \pi^2(\mathbf{x}) \mapsto \pi^{a-1}(\mathbf{x}) \mapsto \mathbf{y} \mapsto \pi(\mathbf{y}) \mapsto \pi^2(\mathbf{y}) \mapsto \ldots \mapsto \pi^{b-1}(\mathbf{y}) \mapsto \mathbf{x}.$$

This is a single cycle of length $a+b$; all lattice sites listed are distinct. After the swap, we have

$$\mathbf{y} \mapsto \pi(\mathbf{x}) \mapsto \pi^2(\mathbf{x}) \mapsto \pi^{a-1}(\mathbf{x}) \mapsto \mathbf{y} \quad \text{and} \quad \mathbf{x} \mapsto \pi(\mathbf{y}) \mapsto \pi^2(\mathbf{y}) \mapsto \ldots \mapsto \pi^{b-1}(\mathbf{y}) \mapsto \mathbf{x}.$$

These are disjoint cycles of length a and b, respectively; the first contains $\mathbf{x}$ and the second contains $\mathbf{y}$. $\square$

Remark 5.3.10. If $\mathbf{x} \circ\!\!-\!\!\circ \mathbf{y}$, i.e. the swap splits their common cycle, the old cycle lengths $\ell_{\mathbf{x}}(\pi)$ and $\ell_{\mathbf{y}}(\pi)$ are equal, and the new cycle lengths after the swap are

$$\ell_{\mathbf{x}}(\pi') = \ell_{\mathbf{y},\mathbf{x}}(\pi) \qquad \text{and} \qquad \ell_{\mathbf{y}}(\pi') = \ell_{\mathbf{x},\mathbf{y}}(\pi).$$

Otherwise, i.e. the swap merges the disjoint cycles, we have

$$\ell_{\mathbf{x}}(\pi') = \ell_{\mathbf{y}}(\pi') = \ell_{\mathbf{x}}(\pi) + \ell_{\mathbf{y}}(\pi).$$

5.4 Winding numbers and the swap-and-reverse algorithm

The propositions of section 5.3 showed that the swap-only algorithm is correct, asymptotically in the number of Metropolis sweeps — in particular, any permutation is reachable from any other with non-zero probability. However, in practice some of these transition probabilities can be quite small. In particular, we observe empirically that the swap-only algorithm always generates permutations with zero winding number. This is readily proved.

Proposition 5.4.1. *In the short-jump-length regime (as discussed in sections 2.3, 3.1, and 3.6), the swap step of the swap-only algorithm preserves winding number.*

Remark. This means that a single jump of length on the order of $L/2$ — which happens with non-zero but very small probability — is required for the SO algorithm to change a winding number.

Proof. The permutations before and after the swap have winding numbers

$$\mathbf{W}(\pi) = \begin{pmatrix} W_x(\pi) \\ W_y(\pi) \\ W_z(\pi) \end{pmatrix} = \sum_{i=1}^{N} \mathbf{d}_\Lambda(\pi(\mathbf{x}_i), \mathbf{x}_i), \quad \mathbf{W}(\pi') = \begin{pmatrix} W_x(\pi') \\ W_y(\pi') \\ W_z(\pi') \end{pmatrix} = \sum_{i=1}^{N} \mathbf{d}_\Lambda(\pi'(\mathbf{x}_i), \mathbf{x}_i)$$

where the difference vector $\mathbf{d}_\Lambda$ is as defined in equation (3.1.4). Since we work in the regime of short jumps, and since $\pi'(\mathbf{x}_i)$ is a nearest neighbor of $\pi(\mathbf{x}_i)$, the Euclidean charts overlap and the $\mathbf{x}_i$'s cancel when we subtract $\mathbf{W}(\pi)$ from $\mathbf{W}(\pi')$. Also, π agrees with π' except at the two swap points $\mathbf{x}$ and $\mathbf{y}$; we have $\pi'(\mathbf{x}) = \pi(\mathbf{y})$ and vice versa. Thus the change in winding number is computed entirely in the same chart and we have

$$\begin{aligned}\mathbf{W}' - \mathbf{W} &= \frac{1}{L}\sum_{i=1}^{N} \mathbf{d}_\Lambda(\pi'(\mathbf{x}_i), \pi(\mathbf{x}_i)) = \frac{1}{L}\left[\mathbf{d}_\Lambda(\pi'(\mathbf{x}), \pi(\mathbf{x})) + \mathbf{d}_\Lambda(\pi'(\mathbf{y}), \pi(\mathbf{y}))\right] \\ &= \frac{1}{L}\left[\pi'(\mathbf{x}) - \pi(\mathbf{x}) + \pi'(\mathbf{y}) - \pi(\mathbf{y})\right] = \frac{1}{L}\left[\pi(\mathbf{y}) - \pi(\mathbf{x}) + \pi(\mathbf{x}) - \pi(\mathbf{y})\right] = 0.\end{aligned}$$

□

An example is shown in figure 5.4 in dimension $d = 2$ with $N = 8$ points on a lattice of width $L = 4$. The sites affected by the permutation swap are $\mathbf{x} = \mathbf{x}_3$ and $\mathbf{y} = \mathbf{x}_7$. The change in winding numbers is

$$\begin{aligned}\mathbf{W}(\pi') - \mathbf{W}(\pi) &= \frac{1}{4}\left[\mathbf{d}_\Lambda(\pi'(\mathbf{x}_3), \mathbf{x}_3) + \mathbf{d}_\Lambda(\pi'(\mathbf{x}_7), \mathbf{x}_7) - \mathbf{d}_\Lambda(\pi(\mathbf{x}_3), \mathbf{x}_3) - \mathbf{d}_\Lambda(\pi(\mathbf{x}_7), \mathbf{x}_7)\right] \\ &= \frac{1}{4}\left[\begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} -1 \\ 0 \end{pmatrix}\right] - \frac{1}{4}\left[\begin{pmatrix} 0 \\ -1 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix}\right] = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ &= \frac{1}{4}\left[\mathbf{d}_\Lambda(\pi'(\mathbf{x}), \pi(\mathbf{x}))\right] + \frac{1}{4}\left[\mathbf{d}_\Lambda(\pi'(\mathbf{y}), \pi(\mathbf{x}))\right] \\ &= \frac{1}{4}\left[\begin{pmatrix} 1 \\ 1 \end{pmatrix} + \begin{pmatrix} -1 \\ -1 \end{pmatrix}\right].\end{aligned}$$

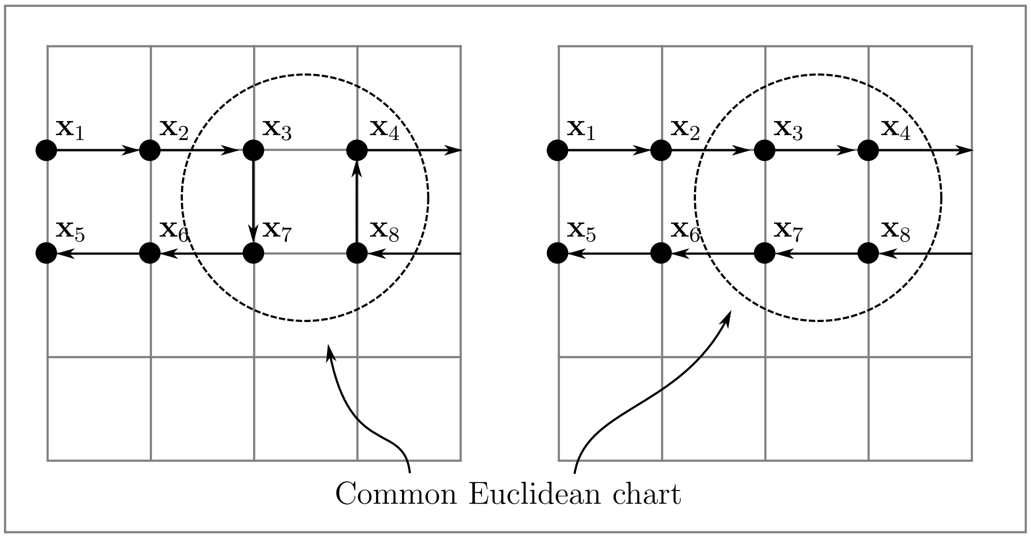


FIGURE 5.4. Example permutations π (left) and π' (right) illustrating winding-number conservation.

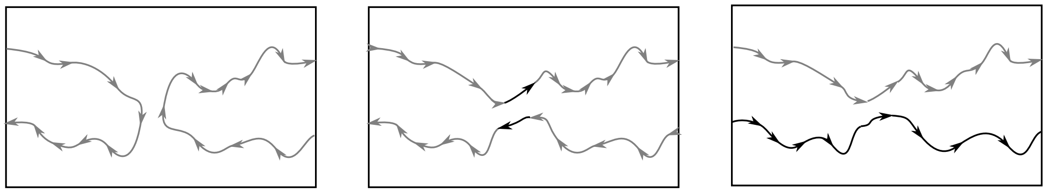


FIGURE 5.5. Conservation of winding number in the swap-only algorithm, and a partial solution provided by the swap-and-reverse algorithm.

A partial solution is explained intuitively by figure 5.5. Part 1 of the figure shows a permutation π with a long cycle on the torus which almost meets itself in the x direction. In part 2, after a Metropolis step sending π to π', one cycle winds by $+1$ and the other by -1. Metropolis steps create winding cycles only in opposite-direction pairs; the total $W_x(\pi)$ is still zero. Part 3 of the figure shows that if we reverse one cycle (which is a zero-energy move), $W_x(\pi)$ is now 2. In general, winding numbers of even parity can be generated. We are sampling from several, but not all, modes in a multimodal probability distribution on permutations which is indexed by the winding numbers W_x, W_y, and W_z.

The *swap-and-reverse* algorithm adds a second type of sweep to the swap-only algorithm. Namely: (1) In a swap-only sweep, for each lattice site one does a Metropolis step as above. (2) In a cycle-reversing sweep, for each cycle in the permutation, one reverses the direction of the cycle with probability 1/2. This permits winding numbers of even parity in each of the three axes. The correctness proof is unaffected, since cycle reversal is a zero-energy change. The time required to reach permutations with non-zero winding numbers, which the asymptotic correctness proof does not address, is reduced. The additional penalty in terms of CPU time consumed by cycle reversal is found to be negligible.

Other solutions exist to generate winding numbers of arbitrary parity: the band-update method of chapter 6 and the worm algorithm of chapter 7. As will be shown, they suffer from too-low acceptance rate and too-long stopping time, respectively. Therefore, the swap-and-reverse algorithm is our current best algorithm; it is used to generate all the results discussed in chapter 11. The order parameters f_S and f_W depend on winding phenomena, but the other three, $1/\xi$, f_I, and $f_{\max}$, do not; furthermore, results obtained in chapter 11 using each of the five order parameters are, for the most part, compatible. Yet, as we will see in chapter 11, f_S and f_W do not permit successful finite-size scaling.

Chapter 6
Band updates

Winding cycles are a global phenomenon, while the SO swaps of section 5.1 are local. The Swendsen-Wang algorithm for the planar Ising model addresses a global/local problem not unlike our winding-number problem, which was presented in detail in section 5.4. Motivated by Swendsen-Wang for the Ising model, we attempt to define a Metropolis step for our random-cycle model which changes one of the winding-number components W_x, W_y, or W_z by ± 1. This attempt at non-local updates has an unreasonably low acceptance rate, namely, on the order of e^{-L} where L is the box length. Nonetheless, this concept may provide fodder for better, future ideas.

6.1 The algorithm

As discussed in section 5.3, a Metropolis transition from π to π' may be thought of as composition with another permutation τ, i.e. $\pi' = \tau\pi$. We replace the swap operator $\tau = G_{\mathbf{a},\mathbf{b}}$ of section 5.3, which was a two-cycle, with a permutation τ which has a winding L-cycle. Without loss of generality, it suffices to discuss a permutation τ which sends every lattice point to itself except for one line of points with y and z coordinates equal to zero (figure 6.1). This τ will have winding number $(W_x, W_y, W_z) = (+1, 0, 0)$. If we can do that, then by reflection and rotation symmetries we can construct similar τ's with W_x, W_y, or W_z equal to ± 1.

Figure 6.1 displays the idea. The permutation τ is an L-cycle; we put $\pi' = \tau\pi$. It seems clear from the picture that the winding number is modified in the desired manner, and moreover it seems plausible to conjecture that winding numbers are additive with respect to permutation composition, i.e. that $\mathbf{W}(\pi') = \mathbf{W}(\tau) + \mathbf{W}(\pi)$. Proving either of these statements rigorously would be worthwhile if band updates were worth pursuing. However, as shown in the next section, they are not.

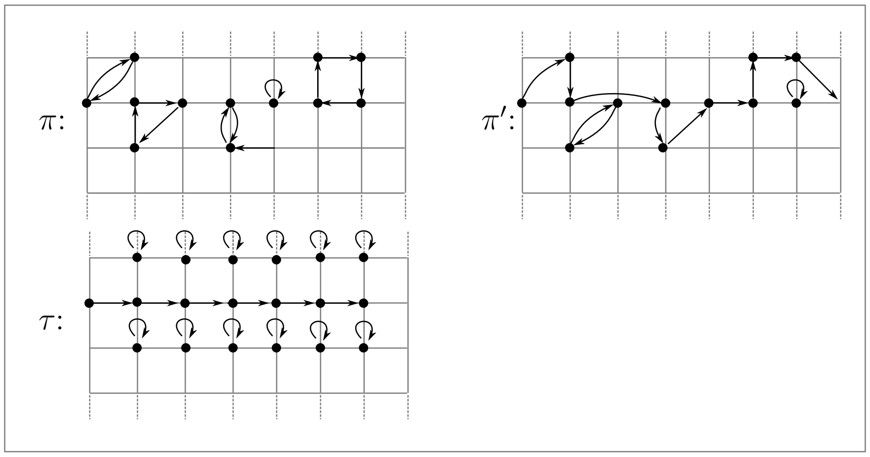


FIGURE 6.1. Example of band update as composition with an L-cycle: $\pi' = \tau\pi$. The winding number W_x increases by $+1$.

6.2 Acceptance rate

We examine the acceptance rate of the band-update algorithm using a semi-empirical method. Recall from section 3.3 that we define the random variable $j_{\mathbf{x}}(\pi)$ to be $\|\pi(\mathbf{x}) - \mathbf{x}\|_\Lambda$. By examination of histograms acquired over MCMC simulations, we see that this parameter is noncritical; its values change smoothly with T near T_c. For $T = 6$ and 7, respectively, we find[1] distributions for $j_{\mathbf{x}}$, as shown in table 6.1. Thus for $T = 6, 7$, respectively, mean jump lengths are 0.608 and 0.245, while mean squared jump lengths are 0.746 and 0.276. Roughly, the latter is of order 0.5. Recall in particular that it is the squared jump length which provides the main contribution to the system energy (equation (2.1.3)).

The left-hand side of figure 6.2 shows (in two dimensions only) various possibilities for a permutation arrow from a point $\mathbf{x}$. The right-hand side of the figure shows what

[1]We use the statistics convention wherein $\hat{P}_{\text{Gibbs}}$ is an experimental estimator for the exact (but unknown) value P_{Gibbs}.

j	$\hat{P}_{\text{Gibbs}}(J=j), T=6$	j	$\hat{P}_{\text{Gibbs}}(J=j), T=7$
0	0.4801	0.0000000	0.7745
1	0.3361	1.0000000	0.1822
$\sqrt{2}$	0.1545	1.4142136	0.0378
$\sqrt{3}$	0.0216	1.7320508	0.0042
4	0.0036	2.0000000	0.0012
$\sqrt{5}$	0.0032	2.2360680	0.0001
$\sqrt{6}$	0.0009		

TABLE 6.1. Empirical jump-length distribution for $T = 6, 7$.

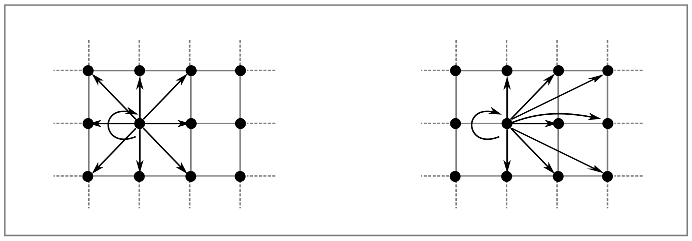


FIGURE 6.2. Change in jump lengths for a point affected by a band update.

happens to the jump lengths at $\mathbf{x}$ on a right shift. (All L points in the L-cycle will be affected similarly.) An up arrow of length 1 becomes a diagonal up-left arrow of length $\sqrt{2}$, a right arrow of length 1 becomes a right arrow of length 2, and so on. Given an attempted band update $\pi \to \pi'$, the typical value of the energy change ΔH at a single point (remembering that cycle-weight corrections are low-order perturbations, since α is small) is

$$\frac{T}{4}\left(\mathbb{E}[j_{\mathbf{x}}(\pi')^2] - \mathbb{E}[j_{\mathbf{x}}(\pi)^2]\right).$$

From the table just above, we can compute the expected values of $j_{\mathbf{x}}(\pi)^2$ and $j_{\mathbf{x}}(\pi')^2$. We find the energy difference per point to be on the order of $+0.8 \cdot T/4 \approx +1.2$. Since L points are involved in a band update, this means $\mathbb{E}[\Delta H] \approx 1.2L \approx L$. Transitions are accepted with probability $\min\{1, e^{-\Delta H}\}$ (section 4.5) which is approximately e^{-L}. We need to consider L from approximately 30 and upward to get past the most severe finite-size effects; $e^{-30} \approx 10^{-14}$ is effectively zero, and thus proposed band updates are effectively never accepted. This calculation matches with simulation tests performed in software.

6.3 Band updates with compensation

At the 2010 Workshop of David Landau's Center for Simulational Physics at the University of Georgia, Friederike Schmid of the University of Mainz suggested a solution to the acceptance-rate problem with the band-update algorithm. This idea comes too late to include in the large-scale computational runs done for this dissertation; however, it should work, and might be used in subsequent computational work on this problem.

Specifically, the problem with the band update is that the change in energy is of order L. To counteract this, in the same Metropolis step in which one proposes a band update as described above, one should also find another cycle of π, not intersecting the band, with cycle length approximately L. The proposed change will do the band shift, while also replacing this second cycle with one-cycles at each of its points — the latter reducing the energy by L or so. The total energy change will be approximately zero, and thus the acceptance rate should be usably high.

Chapter 7
The worm algorithm

This chapter, like chapter 5, presents a Markov chain for MCMC sampling of the model of random spatial permutations (chapter 2) within the framework of chapter 4. The worm algorithm solves the problem of winding-number-parity conservation within the SAR algorithm (section 5.4). However, a stopping time which is poorly bounded in the lattice size prevents the worm algorithm from being our algorithm of choice.

7.1 The idea of the worm algorithm

Worm algorithms have been used heavily in path-integral Monte Carlo (PIMC) simulations: see [BPS06, PST98, GCL97, KPS, NL04, PC87]. The context is that interparticle interactions are modeled using Brownian bridges in the Feynman-Kac approach. A naive, pre-PIMC sampling approach involves generating separate Brownian bridges from point A to point B. The PIMC idea is to generate a single Brownian bridge, then modify a bit at a time using MCMC methods.

For the random-cycle model with true Bose interactions, the Brownian bridges implicit in the V terms of equation (2.1.1) have been integrated out in equation (2.1.3). In fact, this is the key selling point of the random-cycle model in the larger context of the Bose gas (larger meaning beyond the scope of this dissertation). Most of the complexity of PIMC simulations, which are an efficient approach to Brownian bridges, goes away. If one were to adapt a PIMC worm algorithm to the RCM, one would need to spend significant time learning about PIMC. Yet it is likely that most of the complexity will also go away. Instead, it is simpler to ask: If one were to have a worm algorithm for the random-cycle model, what properties would it have? We require the following:

- We have a lattice with a fixed number N of points. There is no desire to work in the grand-canonical ensemble.
- We want the ability to open and close permutation cycles. (An open cycle is a "worm".)
- Given that, tips of open cycles may wander around the 3-torus before closing, permitting arbitrary winding numbers.

Thus, we want to sometimes open a cycle, then modify it with SO-like steps, then close it again. Following PIMC worm algorithms, all our Metropolis steps will involve the worm. This does touch all lattice points: a worm is opened at a site, then modified, then closed. Then, a worm is opened somewhere else, and so on.

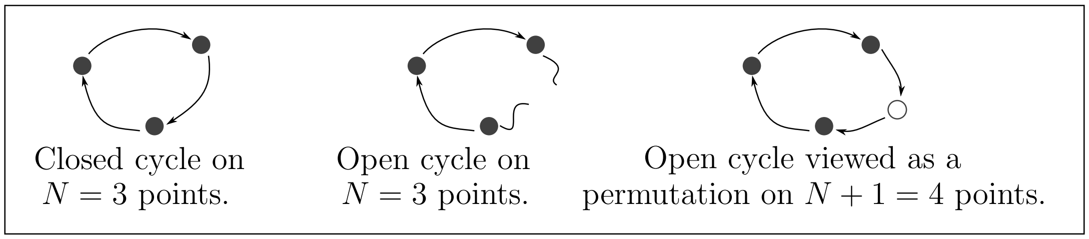

Closed cycle on $N = 3$ points.
Open cycle on $N = 3$ points.
Open cycle viewed as a permutation on $N + 1 = 4$ points.

FIGURE 7.1. Open cycles as permutations on $N + 1$ points.

Question: Can we leverage our knowledge of permutations? To see how, consider a closed cycle and an open cycle on $N = 3$ points (see also figure 7.1):

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}, \qquad \begin{pmatrix} 1 & 2 & 3 & \\ 2 & 3 & & 1 \end{pmatrix}$$

In the open cycle, $1 \mapsto 2$, $2 \mapsto 3$, $3 \mapsto$ nothing, and nothing $\to 1$. Call that nothing *something* — the *wormhole point*. It is an $(N+1)$st point, w:

$$\begin{pmatrix} 1 & 2 & 3 & w \\ 2 & 3 & w & 1 \end{pmatrix}$$

Henceforth, the wormhole point will be written as w or $N+1$. In diagrams, it will be an open dot while the other N points will be written with filled dots (as in figure 7.1). Now we have permutations on $\mathcal{S}_{N+1}$. Given $\pi \in \mathcal{S}_N$, inject π into $\mathcal{S}_{N+1}$ via $\pi(w) = w$.

Definition 7.1.1. For $\pi \in \mathcal{S}_{N+1}$, we say π is a *closed permutation* if $\pi(w) = w$. We say π is an *open permutation* if $\pi(w) \neq w$. Likewise, a cycle of π is said to be open or closed, respectively, if it does or does not contain w.

Remark. The PIMC jargon is that closed permutations are in the *Z sector* (for partition function), while open permutations are in the *G sector* (for Matsubara Green's function).

The goal is to invent an energy function, Gibbs distribution, and Metropolis algorithm for these extended permutations in $\mathcal{S}_{N+1}$ such that the marginal distribution on $\mathcal{S}_{N+1}$, conditioned on closed permutations, matches the RCM Gibbs distribution (equation (2.1.4)). Then, random variables will be sampled only at closed permutations.

7.2 Extended random-cycle model

Recall that we inject $\pi \in \mathcal{S}_N$ into $\mathcal{S}_{N+1}$ via $\pi(w) = w$. The $(N+1)$st point w is *non-spatial*: it has no distance associated with it.

Definition 7.2.1. The *extended lattice* is

$$\Lambda' := \Lambda \cup \{w\}$$

Definition 7.2.2. For $\pi \in \mathcal{S}_{N+1}$, define

$$H'(\pi) = \frac{T}{4} \sum_{\substack{i=1 \\ \pi(\mathbf{x}_i) \neq w}}^{N} \|\mathbf{x}_i - \mathbf{x}_{\pi(i)}\|_\Lambda^2 + \sum_{\ell=1}^{N} \alpha_\ell r_\ell(\pi) + \gamma \, 1_{\mathcal{S}_{N+1} \setminus \mathcal{S}_N}(\pi). \tag{7.2.3}$$

That is, we add γ to the energy if the permutation is open. Note that this *extended energy* agrees with the RCM energy (equation (2.1.3)) on closed permutations. (The γ term is only one particular choice; one might develop a better choice.) This is used to prove the marginality condition below. The *extended Gibbs distribution* and *extended partition function* are defined in the obvious way, as follows.

Definition 7.2.4. Let

$$P'_{\text{Gibbs}}(\pi) = \frac{e^{-H'(\pi)}}{Z'} \tag{7.2.5}$$

where the partition function is

$$Z' = \sum_{\pi \in \mathcal{S}_{N+1}} e^{-H'(\pi)}. \tag{7.2.6}$$

7.3 Proof of marginality

As long as the energy function for the ERCM and the RCM agree on closed permutations, the following desired marginality condition holds. This means that the interaction in section 2.1 — or any other to-be-invented interaction model — may use the worm algorithm as long as it agrees on closed permutations.

Proposition 7.3.1 (Marginality condition)**.** *Let $\mathcal{S}_N \hookrightarrow \mathcal{S}_{N+1}$ by taking $\pi(w) = w$. Let H, H' be energy functions on $\mathcal{S}_N$ and $\mathcal{S}_{N+1}$, respectively, such that for all $\pi \in \mathcal{S}_N$,*

$$H(\pi) = H'(\pi). \tag{7.3.2}$$

Let $P_{\text{Gibbs}}, P'_{\text{Gibbs}}, Z, Z'$ be as above. Then for $\pi \in \mathcal{S}_N$,

$$P'_{\text{Gibbs}}(\pi \mid \pi \in \mathcal{S}_N) = P_{\text{Gibbs}}(\pi). \tag{7.3.3}$$

Proof. Let $\pi \in \mathcal{S}_N$. The left-hand side of equation (7.3.3) is, by definition of conditional expectation,

$$P'_{\text{Gibbs}}(\pi \mid \pi \in \mathcal{S}_N) = \frac{P'_{\text{Gibbs}}(\pi)\, 1_{\mathcal{S}_N}(\pi)}{P'_{\text{Gibbs}}(\mathcal{S}_N)}.$$

The numerator is the Gibbs probability for closed permutations, or zero for open ones:

$$P'_{\text{Gibbs}}(\pi)\, 1_{\mathcal{S}_N}(\pi) = \frac{1}{Z'} e^{-H'(\pi)}\, 1_{\mathcal{S}_N}(\pi) = \frac{1}{Z'} e^{-H(\pi)}\, 1_{\mathcal{S}_N}(\pi)$$

since H and H' agree on closed permutations. The denominator is the total probability of closed permutations:

$$P'_{\text{Gibbs}}(\mathcal{S}_N) = \frac{1}{Z'} \sum_{\pi \in \mathcal{S}_N} e^{-H'(\pi)} = \frac{1}{Z'} \sum_{\pi \in \mathcal{S}_N} e^{-H(\pi)}.$$

Since $\pi \in \mathcal{S}_N$, the ratio is

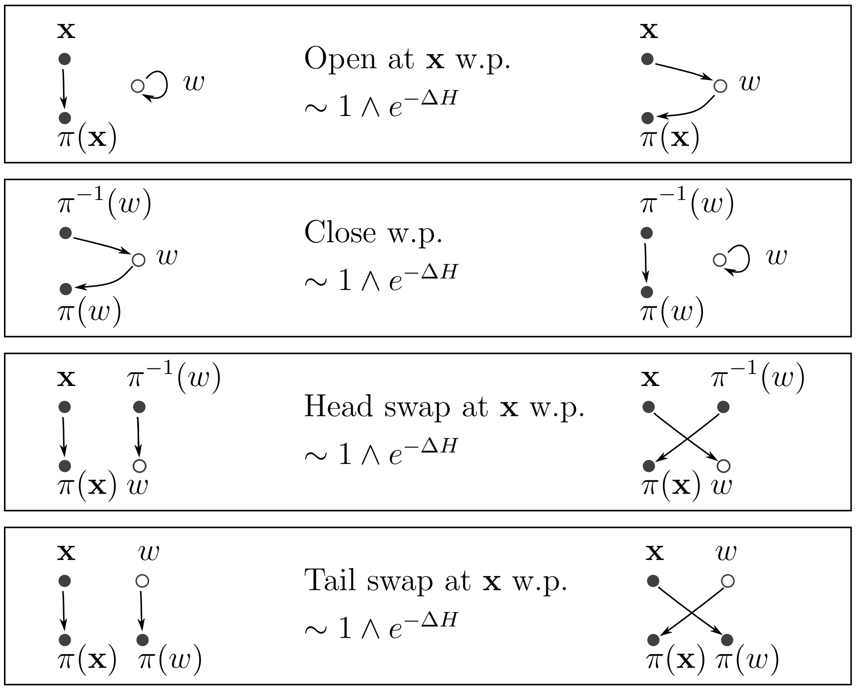
$$\frac{\frac{1}{Z'} e^{-H(\pi)}\, 1_{\mathcal{S}_N}(\pi)}{\frac{1}{Z'} \sum_{\pi \in \mathcal{S}_N} e^{-H(\pi)}} = \frac{e^{-H(\pi)}\, 1_{\mathcal{S}_N}(\pi)}{\sum_{\pi \in \mathcal{S}_N} e^{-H(\pi)}} = \frac{e^{-H(\pi)}\, 1_{\mathcal{S}_N}(\pi)}{Z} = P_{\text{Gibbs}}(\pi).$$

□

7.4 The worm algorithm

Now that we have the correct Gibbs distribution for the ERCM, the next step is to devise a Metropolis algorithm to sample from it. Below, we will prove correctness. The *worm algorithm*, within the context of the recipe in section 4.5, is as follows:

- A sweep begins with a closed permutation π.
- The permutation is now closed, so $\pi(w) = w$. Select a lattice site $\mathbf{x}$ at uniform random. With probability proportional to $1 \wedge e^{-\Delta H}$, open the permutation by swapping the arrows of $\mathbf{x}$ and w. This is called an *open* move. (See figure 7.2.)
- Now that the permutation is open, do a *head swap*, *tail swap*, or *close*.
- Head swap: Pick a lattice site $\mathbf{x}$ nearest-neighbor to the lattice site $\pi^{-1}(w)$. With probability proportional to $1 \wedge e^{-\Delta H}$, swap arrows as in figure 7.2. The head swap is trivial if $\mathbf{x} = \pi^{-1}(w)$, which happens only if the head swap is rejected. The head swap would be a close if $\mathbf{x} = w$, but we choose $\mathbf{x}$ to be a lattice site. Thus, the permutation remains open on a head swap.



$\mathbf{x}$
$\pi(\mathbf{x})$
w
Open at $\mathbf{x}$ w.p.
$\sim 1 \wedge e^{-\Delta H}$
$\pi^{-1}(w)$
$\pi(w)$
Close w.p.
$\sim 1 \wedge e^{-\Delta H}$
Head swap at $\mathbf{x}$ w.p.
$\sim 1 \wedge e^{-\Delta H}$
Tail swap at $\mathbf{x}$ w.p.
$\sim 1 \wedge e^{-\Delta H}$

FIGURE 7.2. Metropolis moves for the worm algorithm.

- Tail swap: Pick a lattice site $\pi(\mathbf{x})$ nearest-neighbor to the lattice site $\pi(w)$. With probability proportional to $1 \wedge e^{-\Delta H}$, swap arrows as in figure 7.2. The tail swap is trivial if $\pi(\mathbf{x}) = \pi(w)$, which happens only if the tail swap is rejected. The tail swap would be a close if $\pi(\mathbf{x}) = w$, but we choose $\pi(\mathbf{x})$ to be a lattice site. Thus, the permutation remains open on a tail swap.

- Close: with probability proportional to $1 \wedge e^{-\Delta H}$, swap arrows as in figure 7.2. The permutation is now closed.

- Once the permutation is closed — after an open, some number of head/tail swaps, and a close, or after a rejected open — a *worm sweep* has been completed. At every sweep, one may obtain a value of any desired random variables for inclusion in computation of their sample means.

Definition 7.4.1. A head swap at $\mathbf{x}$ is *trivial* if $\mathbf{x} = \pi^{-1}(w)$; a tail swap at $\mathbf{x}$ is trivial if $\pi(\mathbf{x}) = \pi(w)$.

7.5 Fibration of $\mathcal{S}_{N+1}$ over $\mathcal{S}_N$

The definitions and lemmas in this section facilitate explicit construction of the Markov matrix, and are necessary for proving correctness of the worm algorithm. As suggested by figure 7.3, we may separate all of $\mathcal{S}_{N+1}$ into the closed permutations $\mathcal{S}_N$ and the open permutations $\mathcal{S}_{N+1} \setminus \mathcal{S}_N$. Furthermore, for each of the $N!$ closed permutations π, we may open π at any of the N sites $\mathbf{x}_1, \ldots, \mathbf{x}_N$. Collecting each of the N open permutations obtained from each closed permutation creates a *fibration* of $\mathcal{S}_{N+1}$. The key points about the structure of this fibration, formalized by the lemmas below, are as follows.

- Each open permutation is one opener move away from a base closed permutation. The N open permutations above a base closed permutation π are the fiber over π.
- This induces a disjoint partition of the open permutations $\mathcal{S}_{N+1} \setminus \mathcal{S}_N$.
- Opens and closes, as defined in section 7.4, stay within fibers; non-trivial head swaps and tail swaps cross fibers.
- For each open permutation, the six non-trivial head swaps and six tail swaps result in twelve distinct permutations.
- Head swaps and tail swaps are transitive on fibers.

We first define maps corresponding to worm Metropolis moves.

Definition 7.5.1. The four worm Metropolis moves of figure 7.2 may be viewed in terms of maps. Throughout, $\mathbf{z} \in \Lambda \cup \{w\}$.

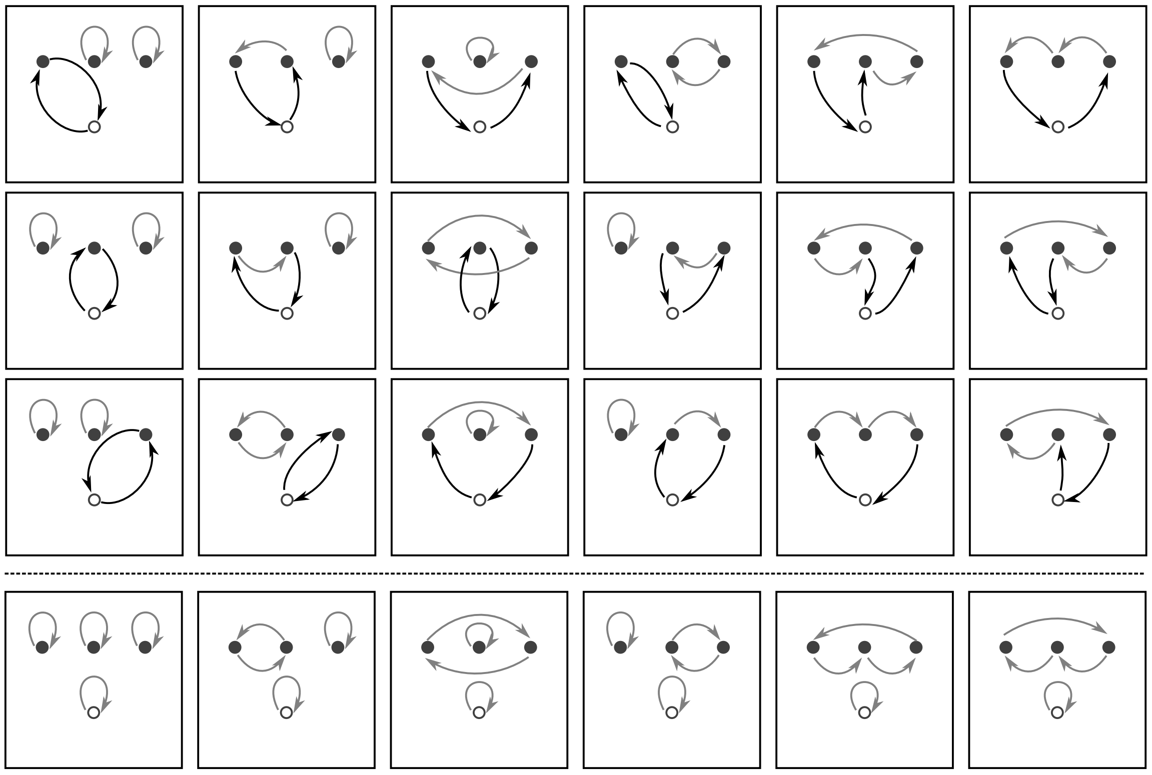


FIGURE 7.3. Fibration of $\mathcal{S}_4$ over $\mathcal{S}_3$. Closed permutations (i.e. $\mathcal{S}_3$) are along the bottom row; open permutations (i.e. $\mathcal{S}_4 \setminus \mathcal{S}_3$) are above the bottom row. The column, or fiber, above each closed permutation π contains the open permutations obtained from π by an opener move. Arrows modified by opener moves are shown in black.

Let $O:\ \mathcal{S}_N \times \Lambda \to \mathcal{S}_{N+1} \setminus \mathcal{S}_N$ send $O(\pi, \mathbf{x}) = \pi'$ such that

$$\pi'(\mathbf{x}) = w,$$
$$\pi'(w) = \pi(\mathbf{x}),$$
$$\pi'(\mathbf{z}) = \pi(\mathbf{z}), \quad \mathbf{z} \neq \mathbf{x}, w.$$

Let $C:\ \mathcal{S}_{N+1} \setminus \mathcal{S}_N \to \mathcal{S}_N$ send $C(\pi) = \pi'$ such that

$$\pi'(\pi^{-1}(w)) = \pi(w),$$
$$\pi'(w) = w,$$
$$\pi'(\mathbf{z}) = \pi(\mathbf{z}), \quad \mathbf{z} \neq \pi^{-1}(w), w.$$

Let $S:\ \mathcal{S}_{N+1} \setminus \mathcal{S}_N \times \Lambda \rightarrow \mathcal{S}_{N+1} \setminus \mathcal{S}_N$ send $S(\pi, \mathbf{x}) = \pi'$ such that

$$\begin{aligned} \pi'(\mathbf{x}) &= w, \\ \pi'(\pi^{-1}(w)) &= \pi(\mathbf{x}), \\ \pi'(\mathbf{z}) &= \pi(\mathbf{z}), \quad \mathbf{z} \neq \mathbf{x}, \pi^{-1}(w). \end{aligned}$$

Let $T:\ \mathcal{S}_{N+1} \setminus \mathcal{S}_N \times \Lambda \rightarrow \mathcal{S}_{N+1} \setminus \mathcal{S}_N$ send $T(\pi, \mathbf{x}) = \pi'$ such that

$$\begin{aligned} \pi'(\mathbf{x}) &= \pi(w), \\ \pi'(w) &= \pi(\mathbf{x}), \\ \pi'(\mathbf{z}) &= \pi(\mathbf{z}), \quad \mathbf{z} \neq \mathbf{x}, w. \end{aligned}$$

Throughout the proofs of the fibration-structure lemmas, we will use the following fact.

Lemma 7.5.2. *If $\mathbf{x} \neq \mathbf{y}$, then $\pi(\mathbf{x}) \neq \pi(\mathbf{y})$ and $\pi^{-1}(\mathbf{x}) \neq \pi^{-1}(\mathbf{y})$.*

Proof. If $\mathbf{x} \neq \mathbf{y}$ and $\pi(\mathbf{x}) = \pi(\mathbf{y})$, then π is not 1-1 which is a contradiction since π is a permutation. This applies to π^{-1} as well, since π^{-1} is also a permutation. $\square$

Now we may prove the fibration-structure lemmas.

Lemma 7.5.3. *Each open permutation π is one opener move away from a base closed permutation π'. That is, for all $\pi \in \mathcal{S}_{N+1} \setminus \mathcal{S}_N$, there exists $\pi' \in \mathcal{S}_N$ such that $C(\pi) = \pi'$.*

Proof. Let $\pi \in \mathcal{S}_{N+1}$. Since π is open, $\pi(w) \neq w$ and $\pi^{-1}(w) \neq w$. Let $\mathbf{a} = \pi^{-1}(w)$ and $\mathbf{b} = \pi(w)$. Both are lattice points. Applying C, we have $C(\pi) = \pi'$ where $\pi'(\mathbf{a}) = \mathbf{b}$, $\pi'(w) = w$, and $\pi'(\mathbf{z}) = \pi(\mathbf{z})$ for all remaining lattice points $\mathbf{z} \neq \mathbf{a}, \mathbf{b}$. Since $\pi'(w) = w$, π' is closed. $\square$

Definition 7.5.4. For $\pi \in \mathcal{S}_N$, $C^{-1}(\pi) \subset \mathcal{S}_{N+1} \setminus \mathcal{S}_N$ is the **fiber** of open permutations over π.

Lemma 7.5.5. *Opens and closes stay within fibers, and each fiber has N elements.*

Proof. Closes stay within fibers by definition of fiber. Next, fix $\pi \in \mathcal{S}_N$ and let $\mathbf{x}_1, \mathbf{x}_2 \in \Lambda$. (These are two different ways to open the same closed permutation.) Let

$$\pi_1' = O(\pi, \mathbf{x}_1), \quad \pi_2' = O(\pi, \mathbf{x}_2).$$

Then π_1' and π_2' have

$$\mathbf{x}_1 \mapsto w \mapsto \pi(\mathbf{x}_1), \qquad \mathbf{x}_2 \mapsto w \mapsto \pi(\mathbf{x}_2),$$

respectively, agreeing with π at all other lattice points $\mathbf{z}$. Now, $C(\pi_1')$ and $C(\pi_2')$ have

$$\mathbf{x}_1 \mapsto \pi(\mathbf{x}_1), w \mapsto w, \qquad \mathbf{x}_2 \mapsto \pi(\mathbf{x}_2), w \mapsto w$$

respectively, agreeing with π at all other lattice points $\mathbf{z}$. But this means $C(\pi_1')$ agrees with $C(\pi_2')$ agree at all points of Λ', so $C(\pi_1') = C(\pi_2')$. Thus, π_1' and π_2' are in the same fiber.

For the last claim, fix $\pi \in \mathcal{S}_N$ and enumerate the N lattice points of Λ as $\mathbf{x}_1, \ldots, \mathbf{x}_N$. We claim that the N permutations

$$\pi_1' = O(\pi, \mathbf{x}_1), \ldots, \pi_N' = O(\pi, \mathbf{x}_N),$$

which are all now known to be in the same fiber, are all distinct. To see this, fix $i \neq j$ from out of $\{1, 2, \ldots, N\}$. Then π_i' and π_j' have

$$\mathbf{x}_i \mapsto w \mapsto \pi(\mathbf{x}_i), \qquad \mathbf{x}_j \mapsto w \mapsto \pi(\mathbf{x}_j).$$

Since $\mathbf{x}_i \neq \mathbf{x}_j$, by lemma 7.5.2 $\pi(\mathbf{x}_i) \neq \pi(\mathbf{x}_j)$. Since

$$\pi_i'(w) = \pi(\mathbf{x}_i) \neq \pi(\mathbf{x}_j) = \pi_j'(w),$$

π_i' and π_j' send w to different points. Therefore, the permutations π_i' and π_j' are distinct. $\square$

Lemma 7.5.6. *The above fibration induces a disjoint partition of the open permutations $\mathcal{S}_{N+1} \setminus \mathcal{S}_N$. That is, for $\pi'_1, \pi'_2 \in \mathcal{S}_N$,*

$$\pi'_1 \neq \pi'_2 \implies C^{-1}(\pi'_1) \cap C^{-1}(\pi'_2) = \emptyset \qquad \text{and} \qquad \bigcup_{\pi \in \mathcal{S}_N} C^{-1}(\pi) = \mathcal{S}_{N+1} \setminus \mathcal{S}_N.$$

Proof. For the first claim, suppose the intersection is non-empty. Let $\pi \in \mathcal{S}_{N+1} \setminus \mathcal{S}_N$ be such that $\pi \in C^{-1}(\pi'_1)$ and $\pi \in C^{-1}(\pi'_2)$. This means $C(\pi) = \pi'_1$ and $C(\pi) = \pi'_2$ with $\pi'_1 \neq \pi'_2$, which is a contradiction since the map C is uniquely defined for all $\pi \in \mathcal{S}_{N+1} \setminus \mathcal{S}_N$.

For the second claim: there are $N!$ closed permutations. We know from the first claim that the $N!$ fibers, one above each closed permutation, are all disjoint. From lemma 7.5.5, we know that each fiber has N elements. We have accounted for all $N \cdot N! = (N+1)! - N!$ open permutations, so we must have all of $\mathcal{S}_{N+1} \setminus \mathcal{S}_N$. $\square$

Lemma 7.5.7. *Non-trivial head swaps and tail swaps (definition 7.4.1) cross fibers.*

Proof. First consider head swaps. Let $\pi, \pi' \in \mathcal{S}_{N+1} \setminus \mathcal{S}_N$ differ by a non-trivial head swap, namely, there is $\mathbf{x} \neq \pi^{-1}(w)$ such that $\pi' = S(\pi, \mathbf{x})$. Then π and π' have

$$\begin{array}{rccccccccccc}
\pi: & \mathbf{x} & \mapsto & \pi(\mathbf{x}) & \mapsto & \pi^2(\mathbf{x}), & & \pi^{-1}(w) & \mapsto & w & \mapsto & \pi(w), \\
\pi': & \pi^{-1}(w) & \mapsto & \pi(\mathbf{x}) & \mapsto & \pi^2(\mathbf{x}), & & \mathbf{x} & \mapsto & w & \mapsto & \pi(w),
\end{array}$$

respectively. Now apply C to each: $C(\pi)$ and $C(\pi')$ have

$$\begin{array}{rcccccccccccc}
C(\pi): & \mathbf{x} & \mapsto & \pi(\mathbf{x}) & \mapsto & \pi^2(\mathbf{x}), & & \pi^{-1}(w) & \mapsto & \pi(w), & w & \mapsto & w, \\
C(\pi'): & \pi^{-1}(w) & \mapsto & \pi(\mathbf{x}) & \mapsto & \pi^2(\mathbf{x}), & & \mathbf{x} & \mapsto & \pi(w), & w & \mapsto & w,
\end{array}$$

respectively. Since $\mathbf{x} \neq \pi^{-1}(w)$, $C(\pi) \neq C(\pi')$.

Next, consider tail swaps. Let $\pi, \pi' \in \mathcal{S}_{N+1} \setminus \mathcal{S}_N$ differ by a non-trivial tail swap, namely, there is $\pi(\mathbf{x}) \neq \pi(w)$ such that $\pi' = T(\pi, \mathbf{x})$. Then π and π' have

$$\begin{array}{rccccccccccc}
\pi: & \pi^{-1}(\mathbf{x}) & \mapsto & \mathbf{x} & \mapsto & \pi(\mathbf{x}), & & \pi^{-1}(w) & \mapsto & w & \mapsto & \pi(w), \\
\pi': & \pi^{-1}(\mathbf{x}) & \mapsto & \mathbf{x} & \mapsto & \pi(w), & & \pi^{-1}(w) & \mapsto & w & \mapsto & \pi(\mathbf{x}),
\end{array}$$

respectively. Now apply C to each: $C(\pi)$ and $C(\pi')$ have

$$\begin{array}{rcccccccccccc}
C(\pi): & \pi^{-1}(\mathbf{x}) & \mapsto & \mathbf{x} & \mapsto & \pi(\mathbf{x}), & & \pi^{-1}(w) & \mapsto & \pi(w), & w & \mapsto & w, \\
C(\pi'): & \pi^{-1}(w) & \mapsto & \mathbf{x} & \mapsto & \pi(w), & & \pi^{-1}(w) & \mapsto & \pi(\mathbf{x}), & w & \mapsto & w,
\end{array}$$

respectively. Since $\pi(\mathbf{x}) \neq \pi(w)$, $C(\pi) \neq C(\pi')$. $\square$

Lemma 7.5.8. *For each open permutation, the six non-trivial head swaps and six non-trivial tail swaps result in twelve distinct permutations.*

Proof. Fix $\pi \in \mathcal{S}_{N+1} \setminus \mathcal{S}_N$. Let $\mathbf{x}_1, \ldots, \mathbf{x}_6$ be the six nearest-neighbor lattice sites to the lattice site $\pi^{-1}(w)$; let $\mathbf{y}_1, \ldots, \mathbf{y}_6$ be the six lattice sites such that $\pi(\mathbf{y}_1), \ldots, \pi(\mathbf{y}_6)$ are nearest-neighbor lattices site to the lattice site $\pi(w)$. (See figure 7.2.)

First, we show that the six permutations $S(\pi, \mathbf{x}_1), \ldots, S(\pi, \mathbf{x}_6)$ are distinct. Let $i \neq j$ for $i, j = 1, \ldots, 6$; let $\pi_i = S(\pi, \mathbf{x}_i)$ and $\pi_j = S(\pi, \mathbf{x}_j)$. Then π has

$$\pi: \quad \mathbf{x}_i \mapsto \pi(\mathbf{x}_i), \qquad \mathbf{x}_j \mapsto \pi(\mathbf{x}_j), \qquad \pi^{-1}(w) \mapsto w;$$

π_i, and π_j have

$$\begin{array}{lllll} \pi_i: & \mathbf{x}_i \mapsto w, & & \pi^{-1}(w) \mapsto \pi(\mathbf{x}_i), \\ \pi_j: & \mathbf{x}_j \mapsto w, & & \pi^{-1}(w) \mapsto \pi(\mathbf{x}_j), \end{array}$$

respectively. Since $\mathbf{x}_i \neq \mathbf{x}_j$, $\pi_i \neq \pi_j$.

Second, we show that the six permutations $T(\pi, \mathbf{y}_1), \ldots, T(\pi, \mathbf{y}_6)$ are distinct. Let $i \neq j$ for $i, j = 1, \ldots, 6$; let $\pi_i = T(\pi, \mathbf{y}_i)$ and $\pi_j = T(\pi, \mathbf{y}_j)$. Then π has

$$\pi: \quad \mathbf{y}_i \mapsto \pi(\mathbf{y}_i), \qquad \mathbf{y}_j \mapsto \pi(\mathbf{y}_j), \qquad w \mapsto \pi(w);$$

π_i, and π_j have

$$\begin{array}{lllll} \pi_i: & \mathbf{y}_i \mapsto \pi(w), & & w \mapsto \pi(\mathbf{y}_i), \\ \pi_j: & \mathbf{y}_j \mapsto \pi(w), & & w \mapsto \pi(\mathbf{y}_j), \end{array}$$

respectively. Since $\mathbf{y}_i \neq \mathbf{y}_j$, by lemma 7.5.2 $\pi(\mathbf{y}_i) \neq \pi(\mathbf{y}_j)$. Since π_i, π_j send w to to different sites, $\pi_i \neq \pi_j$.

Third, we show that the head-swaps of π are distinct from the tail-swaps of π. Fix $\pi \in \mathcal{S}_{N+1} \setminus \mathcal{S}_N$ and let $i, j \in \{1, \ldots, 6\}$. Then π has

$$\pi: \quad \mathbf{x}_i \mapsto \pi(\mathbf{x}_i), \qquad \mathbf{y}_j \mapsto \pi(\mathbf{y}_j), \qquad \pi^{-1}(w) \mapsto w \mapsto \pi(w);$$

$S(\pi, \mathbf{x}_i)$ and $T(\pi, \mathbf{y}_j)$ have

$$\begin{array}{llllllll} S(\pi, \mathbf{x}_i): & \pi^{-1}(w) & \mapsto & \pi(\mathbf{x}_i), & \mathbf{x}_i & \mapsto & w & \mapsto \pi(w); \\ T(\pi, \mathbf{y}_j): & \mathbf{y}_j & \mapsto & \pi(w), & \pi^{-1}(w) & \mapsto & w & \mapsto \mathbf{y}_j; \end{array}$$

respectively. Under these two permutations, w has images $\pi(w)$ and $\mathbf{y}_j$, respectively, and preimages $\mathbf{x}_i$ and $\pi^{-1}(w)$. By definition 7.4.1, the non-trivial head swap $S(\pi, \mathbf{x}_i)$ has $\mathbf{x}_i \neq \pi^{-1}(w)$ and the non-trivial tail swap $T(\pi, \mathbf{y}_j)$ has $\pi(w) \neq \mathbf{y}_j$. Thus, $S(\pi, \mathbf{x}_i)$ and $T(\pi, \mathbf{y}_j)$ are distinct permutations. □

7.6 Explicit construction of the Markov matrix

Transition probabilities were described in section 7.4 as being proportional to $1 \wedge e^{-\Delta H}$. We put the constants of proportionality to be the following:

- a for head swaps and tail swaps;
- b for closer moves;
- c for opener moves.

For SO/SAR, we chose the normalizing factor easily. Here, with a more complicated algorithm, we will choose the normalizing factors to satisfy detailed balance. In particular, in this section we will obtain $c = b = 1/N$ and $a = (1-b)/12$.

The Markov matrix at each Metropolis step is now $(N+1)! \times (N+1)!$:

- A closed permutation transitions only to itself, or to any of the N open permutations in the fiber above it. Thus, there are $N+1$ non-zero entries in π's row of A'.
- An open permutation transitions to any of the 12 open permutations available by head-swapping or tail-swapping, or to itself, or to the closed permutation at the base of its fiber. Thus, there are 14 non-zero entries in π's row of A'.

Definition 7.6.1. For open π, let

$$\{\mathbf{x}_1, \ldots, \mathbf{x}_6\} = \{\mathbf{x} \in \Lambda : \|\mathbf{x}, \pi^{-1}(w)\|_\Lambda = 1\}$$

and

$$\{\mathbf{y}_1, \ldots, \mathbf{y}_6\} = \{\mathbf{y} \in \Lambda : \|\pi(\mathbf{y}), \pi(w)\|_\Lambda = 1\}.$$

Then define

$$\begin{aligned} R_S(\pi) &= \{S(\pi, \mathbf{x}_1), \ldots, S(\pi, \mathbf{x}_6)\}, \\ R_T(\pi) &= \{T(\pi, \mathbf{y}_1), \ldots, T(\pi, \mathbf{y}_6)\}. \end{aligned}$$

These are the twelve open permutations reachable from π via head swaps and tail swaps, respectively (lemma 7.5.8). For closed π, define

$$R_O(\pi) = \{O(\pi, \mathbf{x}_1), \ldots, O(\pi, \mathbf{x}_N)\}.$$

These are the N open permutations reachable from π via opener moves.

With these definitions, the entries of the transition matrix are as follows. In analogy with H, P_{Gibbs}, Z, etc. for the random-cycle model and H', P'_{Gibbs}, Z', etc. for the extended random-cycle model, we call this worm-algorithm transition matrix A' to distinguish it from the matrices A and $A_{\mathbf{x}}$ (equations (5.2.5) and (5.2.6)) for the swap-only algorithm.

If π is closed:

$$A'(\pi, \pi') = \begin{cases} c\left(1 \wedge e^{-H(\pi')+H(\pi)}\right), & \pi' \in R_O(\pi); \\ 1 - \displaystyle\sum_{\pi' \in R_O(\pi)} c\left(1 \wedge e^{-H(\pi')+H(\pi)}\right), & \pi' = \pi; \\ 0, & \text{otherwise.} \end{cases}$$

If π is open:

$$A'(\pi, \pi') = \begin{cases} a\left(1 \wedge e^{-H(\pi')+H(\pi)}\right), & \pi' \in R_S(\pi); \\ a\left(1 \wedge e^{-H(\pi')+H(\pi)}\right), & \pi' \in R_T(\pi); \\ b\left(1 \wedge e^{-H(\pi')+H(\pi)}\right), & \pi' = C(\pi); \\ 1 - t(\pi), & \pi' = \pi; \\ 0, & \text{otherwise} \end{cases}$$

where

$$t(\pi) = \left(\sum_{\pi'' \in R_S(\pi) \cup R_T(\pi)} a \left(1 \wedge e^{-H(\pi'')+H(\pi)}\right) \right) - b \left(1 \wedge e^{-H(C(\pi))+H(\pi)}\right)$$

For row normalization for closed π, note that $c(1 \wedge e^{-\Delta H})$ is between 0 and c so $\sum$ rest is between 0 and cN. Take

$$c = 1/N. \tag{7.6.2}$$

Row normalization for open π then gives

$$12a + b \leq 1. \tag{7.6.3}$$

In practice, we set $12a + b = 1$. That is, we do the following on open permutations: with probability $1/N$, propose a close; else, propose a head or tail swap with equal probability $\frac{1}{2}(1 - \frac{1}{N})$.

The Markov chain for worm Metropolis steps is homogeneous: we use the same transition matrix A' at each step. The correctness proofs of the following section will then imply (by the machinery of chapter 4) that we sample from the extended Gibbs distribution for $\mathcal{S}_{N+1}$. Then by proposition 7.3.1 we will sample from the Gibbs distribution for $\mathcal{S}_N$ whenever the permutation closes.

7.7 Correctness

As discussed in sections 4.5 and 5.3, we need to prove irreducibility, aperiodicity, and detailed balance for the worm Markov chain.

Proposition 7.7.1 (Irreducibility). *The worm algorithm's Markov chain is irreducible.*

Proof. This follows immediately from propositions 5.3.1 and 7.7.2: namely, the worm's chain is irreducibility if the SO's chain is, and moreover the SO's chain is irreducible. $\square$

Proposition 7.7.2. *The worm algorithm's Markov chain is irreducible if the SO algorithm's Markov chain is irreducible.*

Proof. The key point is that the composition of an open, head swap, and close are precisely an SO swap. Let $\mathbf{x}$ and $\mathbf{y}$ be lattice points such that $\pi(\mathbf{x})$ and $\pi(\mathbf{y})$ are nearest neighbors. Starting with π, then applying an open at $\mathbf{x}$, a head swap at $\mathbf{y}$, and a close, we have

$$\begin{array}{lcclcclccl}
\pi: & \mathbf{x} & \mapsto & \pi(\mathbf{x}), & \mathbf{y} & \mapsto & \pi(\mathbf{y}), & w & \mapsto & w; \\
\pi' = O(\pi, \mathbf{x}): & \mathbf{x} & \mapsto & w, & \mathbf{y} & \mapsto & \pi(\mathbf{y}), & w & \mapsto & \pi(\mathbf{x}); \\
\pi'' = S(\pi', \mathbf{y}): & \mathbf{x} & \mapsto & \pi(\mathbf{y}), & \mathbf{y} & \mapsto & w, & w & \mapsto & \pi(\mathbf{x}); \\
\pi''' = C(\pi''): & \mathbf{x} & \mapsto & \pi(\mathbf{y}), & \mathbf{y} & \mapsto & \pi(\mathbf{x}), & w & \mapsto & w.
\end{array}$$

This shows that, if the SO algorithm is irreducible on $\mathcal{S}_N$, the worm algorithm is irreducible on $\mathcal{S}_N$. But then the worm algorithm is also irreducible on $\mathcal{S}_{N+1}$: fix an initial and final permutation; close the initial permutation, if it is open, to obtain a closed permutation; use the preceding argument to reach the closed permutation which lies under the fiber of the desired final open permutation; do an open move (see lemma 7.5.5) if the final permutation is open. $\square$

Remark 7.7.3. The worm algorithm has an additional degree of freedom. If $\mathbf{x}$ and $\mathbf{y}$ are nearest-neighbor lattice sites, then the composition of an open at $\mathbf{x}$, a tail swap at $\mathbf{y}$, and a close results in a similar swap of the jump targets of $\mathbf{x}$ and $\mathbf{y}$:

$$\begin{array}{lcclcclccl}
\pi: & \mathbf{x} & \mapsto & \pi(\mathbf{x}), & \mathbf{y} & \mapsto & \pi(\mathbf{y}), & w & \mapsto & w; \\
\pi' = O(\pi, \mathbf{x}): & \mathbf{x} & \mapsto & w, & \mathbf{y} & \mapsto & \pi(\mathbf{y}), & w & \mapsto & \pi(\mathbf{x}); \\
\pi'' = T(\pi', \mathbf{y}): & \mathbf{x} & \mapsto & w, & \mathbf{y} & \mapsto & \pi(\mathbf{x}) & w & \mapsto & \pi(\mathbf{y}); \\
\pi''' = C(\pi''): & \mathbf{x} & \mapsto & \pi(\mathbf{y}), & \mathbf{y} & \mapsto & \pi(\mathbf{x}), & w & \mapsto & w.
\end{array}$$

Proposition 7.7.4 (Aperiodicity)**.** *The worm algorithm's Markov chain is aperiodic.*

Proof. The proof is the same as in the SO case, proposition 5.3.5. $\square$

Proposition 7.7.5 (Detailed balance)**.** *The Markov chain of the worm algorithm satisfies detailed balance with $b = c$.*

Proof. We need

$$P'_{\text{Gibbs}}(\pi)A'(\pi,\pi') = P'_{\text{Gibbs}}(\pi')A'(\pi',\pi).$$

For closed π to closed π': If $\pi = \pi'$ then we have detailed balance trivially. If $\pi \neq \pi'$ then $A'(\pi,\pi') = A'(\pi',\pi) = 0$ since there are no transitions between distinct closed permutations.

For closed π to open π': If π' is not in the fiber above π, then $A'(\pi,\pi') = A'(\pi',\pi) = 0$ since opens and closes respect fibers (lemma 7.5.5). Now suppose π' is in the fiber above π. As in the SO algorithm (proposition 5.3.6), do cases on ΔH positive or negative. If $H'(\pi') \leq H'(\pi)$, then

$$e^{-H'(\pi)}c = e^{-H'(\pi')}be^{-H'(\pi)}e^{H'(\pi')}.$$

Choose

$$b = c \tag{7.7.6}$$

to satisfy detailed balance. The case $H'(\pi') > H'(\pi)$ results in the same $b = c$ condition.

For open π to closed π': If π is not in the fiber above π', then $A'(\pi,\pi') = A'(\pi',\pi) = 0$ (lemma 7.5.5). If π is in the fiber above π', then we recover the $b = c$ condition.

It now remains to consider open π transitioning to open π'. We assume this to be the case for the rest of the proof.

If $A'(\pi,\pi') = 0$ then we claim $A'(\pi',\pi) = 0$, as in lemma 5.3.8. We have $\pi' \neq \pi$, $\pi' \notin R_S(\pi)$, and $\pi' \notin R_T(\pi)$. We need to show $\pi \neq \pi'$ (which is obvious), $\pi \notin R_S(\pi')$, and $\pi \notin R_T(\pi')$. We prove the contrapositive:

$$\pi \in \{\pi'\} \cup R_S(\pi') \cup R_T(\pi') \quad \Longrightarrow \quad \pi' \in \{\pi\} \cup R_S(\pi) \cup R_T(\pi).$$

If $\pi = \pi'$ then detailed balance is trivially satisfied. Suppose $\pi \in R_S(\pi')$. Then for some $\mathbf{x}_i$, $i = 1, \ldots, 6$, π' and π have

$$\begin{array}{lcccclcccc} \pi': & \mathbf{x}_i & \mapsto & \pi'(\mathbf{x}_i), & & \pi^{-1}(w) & \mapsto & w & \mapsto & w, \\ \pi: & \mathbf{x}_i & \mapsto & w, & & \pi^{-1}(w) & \mapsto & w & \mapsto & \pi'(\mathbf{x}_i). \end{array}$$

The lattice sites $\mathbf{x}_i$ and w are nearest neighbors and π', π agree at all other sites, so there is a head swap sending π to π'. The case $\pi \in \mathbb{R}_T(\pi')$ is completely analogous. This completes the proof of the claim that $A'(\pi, \pi') = 0 \implies A'(\pi', \pi) = 0$.

If $A'(\pi, \pi') \neq 0$ then we claim $A'(\pi', \pi) \neq 0$, again as in lemma 5.3.8. The logic is the same as in the contrapositive argument which was just completed.

The last step is to show detailed balance for open π, π' where $A'(\pi, \pi') \neq 0$. Again we do cases on whether the energy decreases or increases. If $H'(\pi') \leq H'(\pi)$, then equation (7.7.6) is

$$ae^{-H'(\pi)}\,(1) = ae^{-H'(\pi')}\left(e^{-H'(\pi)}e^{H'(\pi')}\right).$$

If $H'(\pi') > H'(\pi)$, then we have

$$ae^{-H'(\pi)}\left(e^{-H'(\pi')}e^{H'(\pi)}\right) = ae^{-H'(\pi')}\,(1).$$

In either case, detailed balance holds. □

Remark. Note that for closed π, there are N choices of open π'; for open π, there is one choice of closed π'. In the software implementation, the $1/N$ for opens comes in through uniform-random choice of $\mathbf{x} \in \Lambda$. The result is that, for closed π, one may only attempt an open. For open π, one attempts a close $1/N$ of the time, and head or tail swaps each half the rest of the time, respectively.

As a sanity check, we point out that cycles may grow or shrink upon worm moves.

Proposition 7.7.7. *Non-trivial worm head swaps and tail swaps either split one cycle into two, or join two cycles into one.*

Proof. This is the same as for the SO case (proposition 7.7.7), which is strictly an algebraic result involving permutations: the non-spatiality of the w point plays no role. □

7.8 Stopping time and modification ideas

The essence of the winding-number problem, as discussed in section 5.4, is that the configuration space has multiple energy minima (which are equivalent to probability maxima), indexed by winding numbers W_x, W_y, and W_z. One might also say that the probability distribution for random spatial permutations is *multimodal*. The swap-only algorithm creates only permutations with winding numbers equal to 0. The swap-and-reverse algorithm creates permutations with even winding numbers: the cycle-reversal move has zero energy change and allows subsequent permutations to hop across a low double-winding-number barrier.

The worm algorithm was designed to permit cycles with winding numbers of both parities to be created: a cycle is opened, its tips wander around (perhaps around the torus), and then it recloses — all of these steps happening with low-energy changes afforded by worm tunneling through the energy barrier. The only problem is that the open worm tips wander around randomly within the L box, and fail to reconnect as L increases. This is the *stopping-time problem*. Specifically, histograms show that the distribution of the wormspan $\|\pi(w) - \pi^{-1}(w)\|_\Lambda$ peaks around $L/2$.

Recall from section 4.2 that the correctness proofs of sections 5.3 and 7.7 only address the limit $M \to \infty$; they do not address rate of convergence. The worm algorithm is correct, but we are not willing to wait long enough for it to produce its correct results. Figure 7.4 shows the problem. CPU time is plotted for 10^4 SAR sweeps, at $T = 6.0$, as a function of $N = L^3$ for $L = 5$ to 12. For the SAR algorithm, CPU time is nearly linear in N. (In fact, it has an N^2 dependency, but with a low constant of proportionality, as discussed in section 9.5.) For the worm algorithm, CPU time is not linear in N; we cannot complete a computation for L large enough to be interesting, namely, 40 to 80. Specifically, a log-log plot and regression on the data of figure 7.4 show that the worm algorithm is strongly quadratic in N. Fortunately, examination of random-variable plots for $L = 10$, comparing SAR to worm, show that similar results are produced — other than, of course, the winding-number histogram itself.

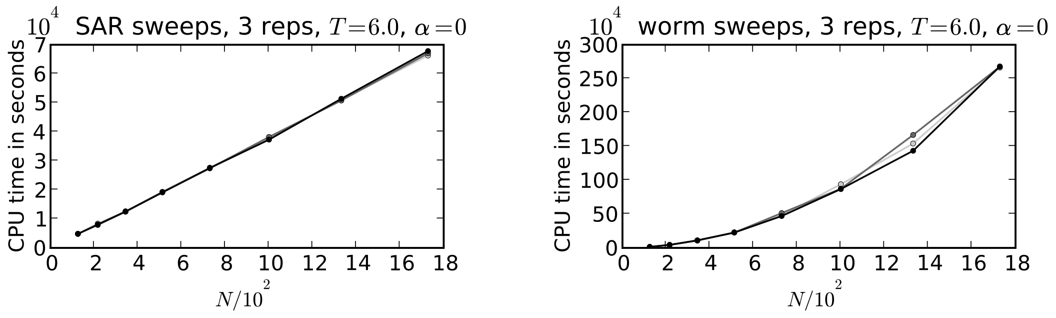


FIGURE 7.4. Scalability of SAR and worm algorithms. CPU times for 10^4 SAR sweeps and 10^3 worm sweeps are shown as a function of $N = L^3$ for $L = 5$ to 12. SAR time is nearly linear in N; worm time is strongly quadratic in N. Interesting L (40-80) are unattainable.

Other ideas for addressing the winding-number problem include the following:

- In analogy with cluster updates for the Ising model, form a band around the torus and do an L-cycle transformation. Couple the SAR algorithm with occasional band updates. However, band updates have a too-low acceptance rate, as was shown in chapter 6.
- Temporarily pinch the torus geometry somehow in the SAR algorithm, such that the distance penalty for wrapping around the torus is decreased.
- Temporarily reduce and restore the temperature T in the SAR algorithm — this is an annealing method. This approach brings with it a performance problem: re-thermalization (section 9.6) would need to be performed after each annealing step.
- Modify the worm algorithm to direct the worm somehow. At the time the worm is opened, add a distance weight of $\pm L$ in the x, y, or z direction which will

be removed by a wrap around the torus, increasing or decreasing that winding-number component by 1. Our attempts to do this have not satisfied detailed balance.

- Review the PIMC literature again and seek other inspiration.

The worm algorithm, even though it is effectively unusable as currently designed, is the only way we currently have of sampling from the full winding-number configuration space, i.e. odd as well as even winding numbers. Thus, it will be worth the future effort to solve the stopping-time problem.

Chapter 8
ΔH computations

When computing ΔH for the swap-only, swap-and-reverse, or worm algorithms, it is inefficient to find $H(\pi')$ and $H(\pi)$ separately, then compute their difference: swap and worm moves are local, and most of the energy terms are unchanged from π to π'. Instead (this is true for Metropolis simulations in general), one discovers a formula for the energy change in a proposed Metropolis move. Even though these minimal energy-change formulas are a software-optimization detail, they need to be considered carefully lest errors intrude.

We write the energy of equation (7.2.3) as

$$H = D + V + W \tag{8.0.1}$$

where H is total energy, D is the distance-related single-jump terms, V is the jump-pair-interaction terms, and W is the worm-dependent terms. (Note that V is identically zero if there are no interactions, and W is identically zero for the SO, SAR, and band-update algorithms.)

8.1 Swap and worm with no interactions

Recall that the wormhole point is non-spatial and thus does not participate in distance computations. As is clear from figures 5.1 and 7.2 (on pages 61 and 82, respectively), the change in distance-related terms is

$$\Delta D = \|\mathbf{x} - \pi(\mathbf{y})\|_\Lambda^2 + \|\mathbf{y} - \pi(\mathbf{x})\|_\Lambda^2 - \|\mathbf{x} - \pi(\mathbf{x})\|_\Lambda^2 - \|\mathbf{y} - \pi(\mathbf{y})\|_\Lambda^2 \quad \text{(swap-only)}$$

$$\Delta D = -\|\mathbf{x} - \pi(\mathbf{x})\|_\Lambda^2 \quad \text{(worm open)}$$

$$\Delta D = \|\pi^{-1}(w) - \pi(w)\|_\Lambda^2 \quad \text{(worm close)}$$

$$\Delta D = \|\pi^{-1}(w) - \pi(\mathbf{x})\|_\Lambda^2 - \|\mathbf{x} - \pi(\mathbf{x})\|_\Lambda^2 \quad \text{(worm head swap)}$$

$$\Delta D = \|\mathbf{x} - \pi(w)\|_\Lambda^2 - \|\mathbf{x} - \pi(\mathbf{x})\|_\Lambda^2 \quad \text{(worm tail swap).}$$

8.2 Swap and worm with two-cycle interactions

The non-spatiality of the wormhole point plays no role in the algebraic notion of cycle lengths. Thus, the same Δr_2 formulas apply to both algorithms. The ΔD is the same as in section 8.1; he we describe only the ΔV.

Recall the definition of a swap from section 5.1. The simplicity of figure 8.1 masks a bit of detail: namely, the four points may not all be distinct. Thus, there are several cases. (See figure 8.2.)

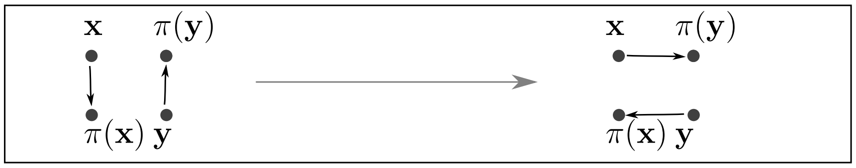


FIGURE 8.1. A swap.

- Case 0: $\pi(\mathbf{x}) = \pi(\mathbf{y})$ (and so also $\mathbf{x} = \mathbf{y}$): this is a trivial move; $\pi' = \pi$. $\Delta r_2 = 0$.
- Case 1: $\mathbf{x} = \pi(\mathbf{x})$.
  - Case 1a: $\mathbf{y} = \pi(\mathbf{y})$. $\Delta r_2 = +1$.
  - Case 1b: $\mathbf{y} \neq \pi(\mathbf{y})$ but $\mathbf{y} = \pi^2(\mathbf{y})$. $\Delta r_2 = -1$.

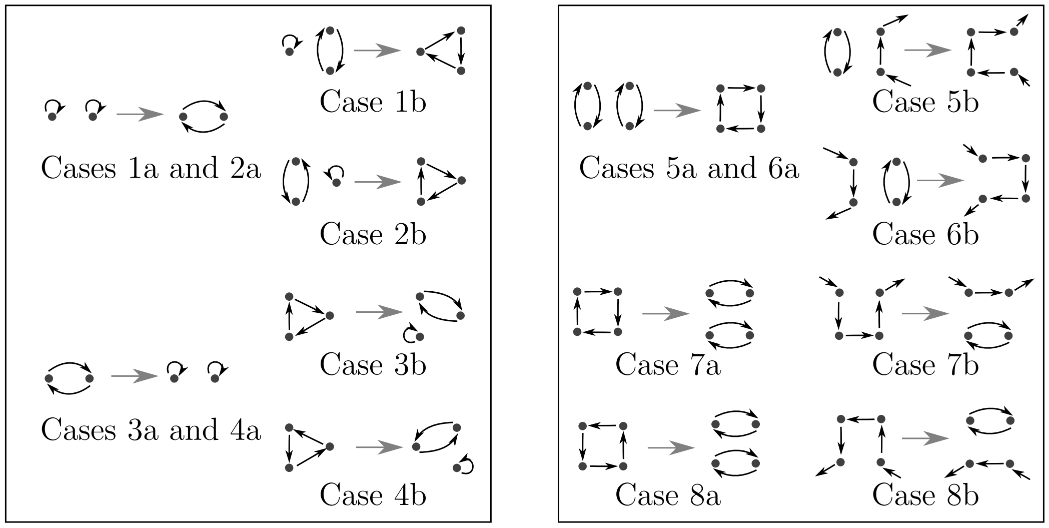


FIGURE 8.2. Cases for Δr_2.

- Case 1c: $\mathbf{y} \neq \pi(\mathbf{y}), \pi^2(\mathbf{y})$. $\Delta r_2 = 0$.

- Case 2: $\mathbf{y} = \pi(\mathbf{y})$.
  - Case 2a: $\mathbf{x} = \pi(\mathbf{x})$. Same as case 1a. $\Delta r_2 = +1$.
  - Case 2b: $\mathbf{x} \neq \pi(\mathbf{x})$ but $\mathbf{x} = \pi^2(\mathbf{x})$. $\Delta r_2 = -1$.
  - Case 2c: $\mathbf{x} \neq \pi(\mathbf{x}), \pi^2(\mathbf{x})$. $\Delta r_2 = 0$.
- Case 3: $\mathbf{x} = \pi(\mathbf{y})$.
  - Case 3a: $\pi(\mathbf{x}) = \mathbf{y}$. $\Delta r_2 = -1$.
  - Case 3b: $\pi^2(\mathbf{x}) = \mathbf{y}$. $\Delta r_2 = +1$.
  - Case 3c: $\mathbf{y} \neq \pi(\mathbf{x}), \pi^2(\mathbf{x})$. $\Delta r_2 = 0$.
- Case 4: $\pi(\mathbf{x}) = \mathbf{y}$.
  - Case 4a: $\pi(\mathbf{y}) = \mathbf{x}$. Same as case 3a. $\Delta r_2 = -1$.
  - Case 4b: $\pi^2(\mathbf{y}) = \mathbf{x}$. $\Delta r_2 = +1$.

  - Case 4c: $\mathbf{x} \neq \pi(\mathbf{y}), \pi^2(\mathbf{y})$. $\Delta r_2 = 0$.

- Case 5: $\pi^2(\mathbf{x}) = \mathbf{x}$.
  - Case 5a: $\pi^2(\mathbf{y}) = \mathbf{y}$. $\Delta r_2 = -2$.
  - Case 5b: $\pi^2(\mathbf{y}) \neq \mathbf{y}$. $\Delta r_2 = -1$.
- Case 6: $\pi^2(\mathbf{y}) = \mathbf{y}$.
  - Case 6a: $\pi^2(\mathbf{x}) = \mathbf{x}$. Same as 5a. $\Delta r_2 = -2$.
  - Case 6b: $\pi^2(\mathbf{x}) \neq \mathbf{x}$. $\Delta r_2 = -1$.
- Case 7: $\pi^2(\mathbf{x}) = \mathbf{y}$.
  - Case 7a: $\pi^2(\mathbf{y}) = \mathbf{x}$. $\Delta r_2 = +2$.
  - Case 7b: $\pi^2(\mathbf{y}) \neq \mathbf{x}$. $\Delta r_2 = +1$.
- Case 8: $\pi^2(\mathbf{y}) = \mathbf{x}$.
  - Case 8a: $\pi^2(\mathbf{x}) = \mathbf{y}$. $\Delta r_2 = +2$.
  - Case 8b: $\pi^2(\mathbf{x}) \neq \mathbf{y}$. $\Delta r_2 = +1$.
- All other cases: $\Delta r_2 = 0$.

8.3 Swap and worm with r_ℓ interactions

The non-spatiality of the wormhole point plays no role in the algebraic notion of cycle lengths. Thus, the same Δr_2 formulas apply to both algorithms.

Recall proposition 5.3.9 and remark 5.3.10: if $\mathbf{x}$ and $\mathbf{y}$ are in separate cycles before the swap, they are in the same cycle afterward, and vice versa. In the former case, the new common cycle length is the sum of the old separate cycle lengths; in the latter case, the new cycle lengths are taken from the number of permutation jumps from one site to the other. (Throughout this section, please consult figure 8.3 for illumination.) Given that general pair of facts, we split out subcases which are convenient as a software-optimization detail:

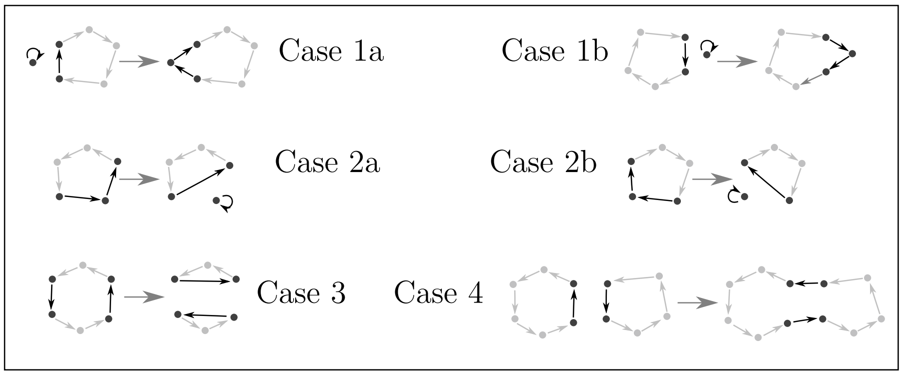


FIGURE 8.3. Cases for Δr_ℓ. Sites and arrows not participating in changes are shown in grey.

- Case 0: $\pi(\mathbf{x}) = \pi(\mathbf{y})$ (and so also $\mathbf{x} = \mathbf{y}$): this is a trivial move; $\pi' = \pi$. $\Delta r_\ell = 0$ for all $\ell = 1, \ldots, N$.
- Case 1: $\mathbf{x}$ and $\mathbf{y}$ are in different cycles, but one of them is in a one-cycle.
  - Case 1a: $\mathbf{x} = \pi(\mathbf{x})$: $\Delta r_1 = -1$, $\Delta r_{\ell_{\mathbf{y}}(\pi)} = -1$, $\Delta r_{\ell_{\mathbf{y}}(\pi)+1} = +1$.
  - Case 1b: $\mathbf{y} = \pi(\mathbf{y})$: $\Delta r_1 = -1$, $\Delta r_{\ell_{\mathbf{x}}(\pi)} = -1$, $\Delta r_{\ell_{\mathbf{x}}(\pi)+1} = +1$.
- Case 2: $\mathbf{x}$ and $\mathbf{y}$ are in the same cycle, but one is the jump target of the other.
  - Case 2a: $\mathbf{y} = \pi(\mathbf{x})$. $\Delta r_{\ell_{\mathbf{x}}(\pi)} = -1$, $\Delta r_{\ell_{\mathbf{x}}(\pi)-1} = +1$, $\Delta r_1 = +1$.
  - Case 2a: $\mathbf{x} = \pi(\mathbf{y})$. $\Delta r_{\ell_{\mathbf{y}}(\pi)} = -1$, $\Delta r_{\ell_{\mathbf{y}}(\pi)-1} = +1$, $\Delta r_1 = +1$.
- Case 3: $\mathbf{x}$ and $\mathbf{y}$ are in the same cycle, and neither is the jump target of the other. Let $a = \ell_{\mathbf{x},\mathbf{y}}(\pi)$ and $b = \ell_{\mathbf{y},\mathbf{x}}(\pi)$. Then $\Delta r_{a+b} = -1$, $\Delta r_a = +1$, $\Delta r_b = +1$.

- Case 4: $\mathbf{x}$ and $\mathbf{y}$ are in separate cycles. $\Delta r_{\ell_{\mathbf{x}(\pi)}} = -1$, $\Delta r_{\ell_{\mathbf{y}(\pi)}} = -1$, $\Delta r_{\ell_{\mathbf{x}(\pi)}+\ell_{\mathbf{y}(\pi)}} = +1$.

8.4 Swap with V interactions

Recall from proposition 7.3.1 that as long as the extended energy function H' agrees with the energy function H on closed cycles, P'_{Gibbs} has the correct marginal distribution on closed cycles. Thus, when writing energy terms for open cycles, we can choose how to define the energy. For r_2 and r_ℓ (the previous two sections), it is simplest to say that the non-spatial point w can participate in permutation cycles. For other interactions that depend on the spatiality of points, it is simplest to say that w does not participate. Thus, here we split out swap and worm cases.

The change in energy is simply the contributions from the old arrows $\mathbf{x} \mapsto \pi(\mathbf{x})$ and $\mathbf{y} \mapsto \pi(\mathbf{y})$ to all other arrows, along with their mutual interaction, subtracted from the contributions from the new arrows $\mathbf{x} \mapsto \pi(\mathbf{y})$ and $\mathbf{y} \mapsto \pi(\mathbf{x})$ to all other arrows, along with their mutual interaction:

$$\begin{aligned}\Delta V = &\sum_{\mathbf{v}\neq\mathbf{x},\mathbf{y}} V(\mathbf{x},\pi(\mathbf{y}),\mathbf{v},\pi(\mathbf{v})) + \sum_{\mathbf{v}\neq\mathbf{x},\mathbf{y}} V(\mathbf{y},\pi(\mathbf{x}),\mathbf{v},\pi(\mathbf{v})) + V(\mathbf{x},\pi(\mathbf{y}),\mathbf{y},\pi(\mathbf{x})) \\ &- \sum_{\mathbf{v}\neq\mathbf{x},\mathbf{y}} V(\mathbf{x},\pi(\mathbf{x}),\mathbf{v},\pi(\mathbf{v})) - \sum_{\mathbf{v}\neq\mathbf{x},\mathbf{y}} V(\mathbf{y},\pi(\mathbf{y}),\mathbf{v},\pi(\mathbf{v})) - V(\mathbf{x},\pi(\mathbf{x}),\mathbf{y},\pi(\mathbf{y})).\end{aligned}$$

8.5 Worm with V interactions

The non-spatial point has no interactions, so we simply track the creation and destruction of spatial-to-spatial arrows for the four types of worm move. (See figure 8.4.)

Open:

$$-\sum_{\mathbf{v}\neq\mathbf{x},w} V(\mathbf{x},\pi(\mathbf{x}),\mathbf{v},\pi(\mathbf{v})).$$

Close:

$$\sum_{\mathbf{v}\neq\pi^{-1}(w),w} V(\pi^{-1}(w),\pi(w),\mathbf{v},\pi(\mathbf{v})).$$

Head swap:

$$\sum_{\mathbf{v}\neq\mathbf{x},\pi^{-1}(w)} V(\pi^{-1}(w),\pi(\mathbf{x}),\mathbf{v},\pi(\mathbf{v})) - \sum_{\mathbf{v}\neq\mathbf{x},\pi^{-1}(w)} V(\mathbf{x},\pi(\mathbf{x}),\mathbf{v},\pi(\mathbf{v})).$$

Tail swap:

$$\sum_{\mathbf{v}\neq\mathbf{x},w,\pi^{-1}(w)} V(\mathbf{x},\pi(w),\mathbf{v},\pi(\mathbf{v})) - \sum_{\mathbf{v}\neq\mathbf{x},w,\pi^{-1}(w)} V(\mathbf{x},\pi(\mathbf{x}),\mathbf{v},\pi(\mathbf{v})).$$

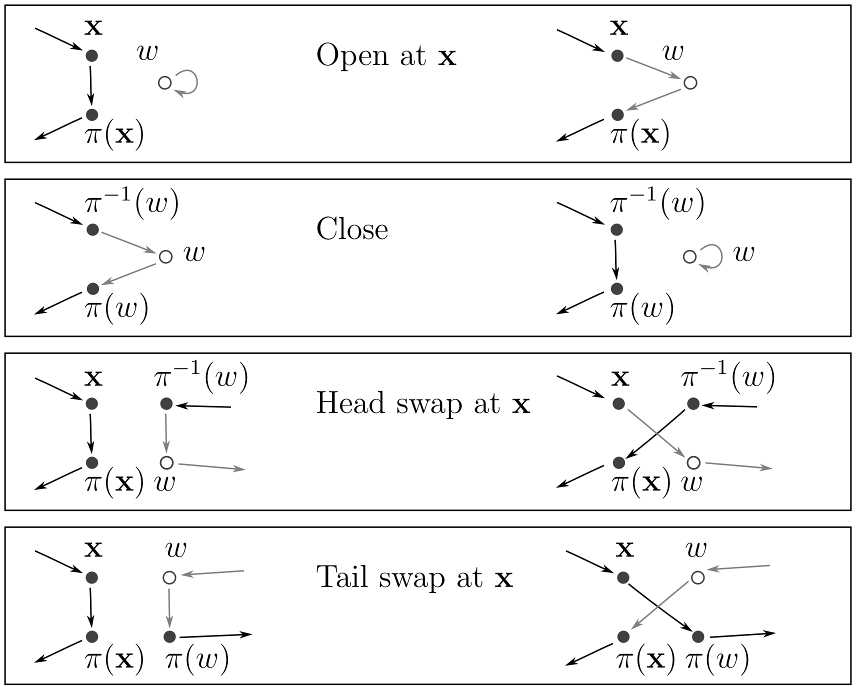


FIGURE 8.4. Cases for worm ΔV. Non-spatial arrows (i.e. those starting or ending at w) are shown in grey.

Chapter 9
Algorithms for single MCMC runs

Chapters 5, 7, and 8 describe the swap-only, swap-and-reverse, band-update, and worm algorithms. The content there focuses on algorithm steps and algorithm correctness, without reference to a specific programming language or a specific software implementation. This chapter, by contrast, focuses on particular software-design choices which were made by the author.

The software program (`mcrcm`) which does a single MCMC run is written in the C language. Execution of multiple MCMC runs, including parallel processing, is done using a scripting language such as Bash or Python; this is described in the next chapter. Throughout this chapter and the next, names of data structures and subroutines from the program code are written in `typewriter font`.

The methods here are applicable for any of the algorithms of chapters 5 and 7 — swap-only, swap-and-reverse, and worm — or any other to-be-invented algorithm which satisfies the hypotheses listed in section 4.5 on recipes for MCMC algorithms.

9.1 Data structures

There is one main data structure, of C type `points_t`, containing *points* and a *cycle list* (see figure 9.1):

- `dims` contains the lattice dimensions L, L, L. (For 1-dimensional or 2-dimensional use, not described in this thesis, these would be $L, 1, 1$ or $L, L, 1$, respectively.)

- `N` is the product of the three dimensions. It is used so frequently in the program code that it is worth computing this product once at the start of the program and keeping it here.

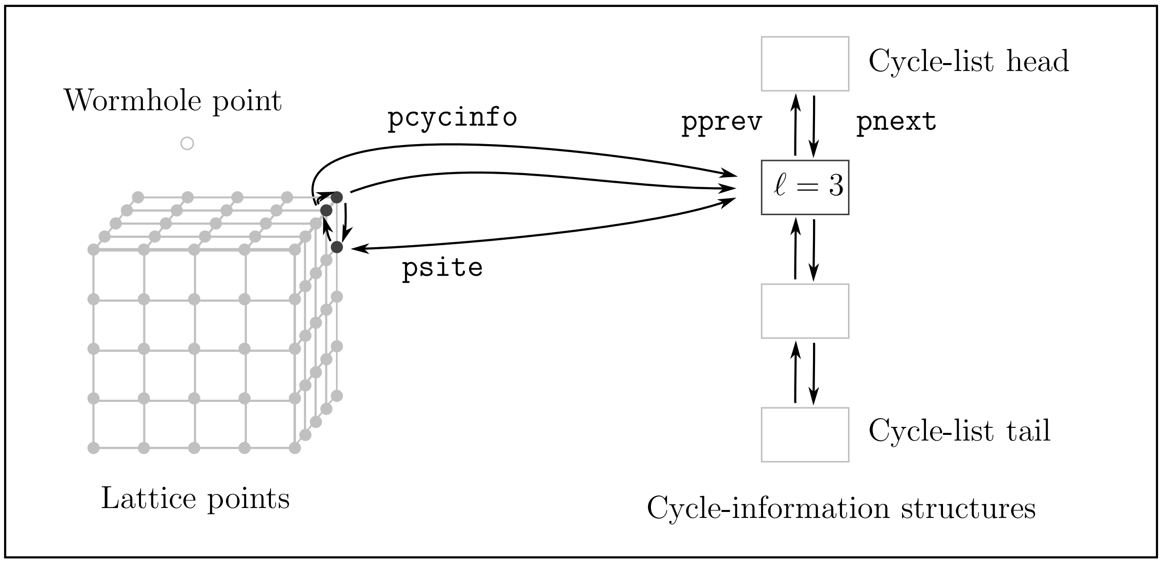


FIGURE 9.1. Lattice sites and the wormhole point are shown in grey; three lattice sites participating in a 3-cycle are shown in black. Every point (including the grey ones for which these arrows are not shown) contains the location (`pcycinfo`) of a cycle-information structure which caches the length of the cycle (`cyclen`); each cycle-information structure contains the location (`psite`) of one point in the cycle. Cycle-information structures are stored in a doubly linked list (`pprev` and `pnext`).

- `lattice`: $L \times L \times L$ array of points (data type `point_t`).
- `wormhole`: an $(N+1)$st point (data type `point_t`) called the wormhole point (chapter 7). (For the swap-only and swap-and-reverse algorithms, all permutations send the wormhole point to itself.)
- `pcycinfo_list_head` and `pcycinfo_list_tail` are the locations of the first and last cycle-information structure in the doubly linked list of cycle-information structures.

Each point (`point_t` data type) **x** contains:

- `selfi`, `selfj`, `selfk`: lattice coordinates of each point **x**, with x_i, x_j, x_k from 0 to $L-1$. For the wormhole point, these coordinates are set to an undefined value.

- `pfwd`: location of permutation image $\pi(\mathbf{x})$.
- `bfwd`: location of permutation preimage $\pi^{-1}(\mathbf{x})$. This is used for the reason described in section 5.1. Namely, in a swap, one
  - selects a site $\mathbf{x}$,
  - follows the forward permutation pointer to find $\pi(\mathbf{x})$,
  - uses the lattice structure to find a site $\pi(\mathbf{y})$ which is a nearest neighbor of $\pi(\mathbf{x})$, and finally
  - follows the backward permutation pointer to find $\mathbf{y} = \pi^{-1}(\pi(\mathbf{y}))$.
- `fwd_d` and `fwd_dsq`: distance $\|\pi(\mathbf{x}) - \mathbf{x}\|_\Lambda$ and squared distance $\|\pi(\mathbf{x}) - \mathbf{x}\|_\Lambda^2$. These are cached for performance reasons. It is found empirically that approximately 10-20% of proposed Metropolis changes are accepted. Thus, without caching of these distances, most of the time the forward-distance values would be computed redundantly.
- `pcycinfo`: location of cycle-information structure (see below).
- `mark`: this is a single integer stored at each lattice site. When the cycle-information list is set up (section 9.3) or sanity-checked (section 9.16), it is necessary to sweep through all lattice sites, following permutation cycles, remembering which sites have already been visited. One could allocate a list of marks (one for each lattice site) at the beginning of these routines, and free that list at the end. Instead, the marks are stored in the lattice structure so that this scratch space is available when needed.

Cycle information is stored in a doubly linked list of cycle-information structures (data type `cycinfo_t`). Each contains:

- `psite`: Location of one site in the cycle.

- `cyclen`: Length of the cycle.
- `pprev` and `pnext`: Location of the previous and next cycle-information structure in the doubly linked list.

9.2 Overview

A true outline of the program (routine `main` in file `mcrcm.c`) embodies the outline given in section 4.5. Namely:

- Determine the input parameters L, T, interaction type (non-interacting, r_2, r_ℓ) and parameter α, algorithm type (swap-only, swap-and-reverse, worm), and `number_of_sweeps`. (These are passed into the program via the command line.)
- Print all control parameters (see section 9.18).
- Initialize (see section 9.3).
- Thermalization phase:

  ```
  Loop until thermalized:
      Do one swap-only, swap-and-reverse, or worm sweep (section 9.4).
      See if thermalization is complete (section 9.6).
      Optionally sanity-check $H$ and cached cycle information (section 9.16).
      Optionally display random-variable instances (section 9.18).
  ```

- Accumulation phase:

  ```
  For sweep number from 0 to number_of_sweeps−1:
      Do one swap-only, swap-and-reverse, or worm sweep (section 9.4).
      Optionally write $\pi$ to disk (section 9.17).
      Optionally sanity-check $H$ and cached cycle information (section 9.16).
  ```

Optionally print realizations of user-specified random variables (section 9.18).

Remember random-variable instances, for statistical use.

Compute statistics of random variables (theorem 4.2.9 and section 4.3).

Display statistics of random variables (section 9.18).

9.3 Initialization

Software initialization consists of four main steps:

- Allocate memory for the lattice points: subroutine `get_cubic_lattice_points`. Set the initial permutation to one of the following:
  - The identity permutation. (This is the default, set up by the subroutine `get_cubic_lattice_points`.)
  - A uniform-random permutation on $\mathcal{S}_N$, with the wormhole point sent to itself: subroutine `set_unif_rand_pmt`.
- Allocate time-series arrays for each random variable: `allocate_rvs`.
- Initialize the cycle-information list: subroutine `set_up_cycinfo_list`.
- Find the initial system energy H, separated into D and V terms: subroutine `get_H_of_pi`. This is a straightforward implementation of equation (2.1.3).

9.4 Metropolis sweeps

The swap-only algorithm uses a swap-only sweep; the swap-and-reverse algorithm uses a swap-only sweep followed by a cycle-reverse sweep. A swap-only sweep (subroutine `SO_sweep`) is as follows:

- Loop through lattice sites $\mathbf{x} = (x, y, z)$ lexically, i.e. x from 0 to $L-1$, y from 0 to $L-1$, z from 0 to $L-1$.

- For the site $\mathbf{x}$, follow the forward permutation pointer to find $\pi(\mathbf{x})$.
- Use the lattice structure to select $\pi(\mathbf{y})$ which is one of the six (i.e. $2d$) nearest-neighbor sites to $\pi(\mathbf{x})$.
- Follow the backward permutation pointer to find the point $\mathbf{y}$.
- In subroutine `try_SO_swap`, propose and perhaps accept a modification of the permutation. This is a Metropolis step, described in section 9.5.

(A slight modification of this algorithm would choose $\mathbf{x}$ at uniform-random location on the lattice, N times, rather than looping sequentially through all N lattice sites.) After a swap-only sweep has been performed, the swap-and-reverse algorithm then does a reverse sweep (subroutine `reverse_sweep`):

- For each cycle in the permutation (i.e. one follows the doubly-linked list of cycle-information structures), with probability $1/2$ reverse all the arrows in that cycle. This is done for the reason described in section 5.4, namely, it permits non-zero (but only even) winding numbers.
- At each point, the forward and backward permutation pointers must be updated, and the cached forward distance and forward squared distance must be copied from one point to another. The cycle-information structure is not affected. Also, the system energy is not affected.

A worm sweep (subroutine `worm_sweep`) is done as in section 7.4:

- One attempts to open the permutation at a uniform-random lattice site $\mathbf{x}$.
- If the open is not accepted, the sweep is complete.
- Otherwise, some number of head swaps and/or tail swaps are proposed and perhaps accepted. Eventually, a close is proposed and accepted. The sweep is then complete.

Notice that the swap-only and reverse sweeps are of deterministic length: the former processes all N lattice sites; the latter processes all cycles. (Of course, it takes more CPU time for an accepted proposal then a rejected proposal. Nonetheless, a fixed number of proposals is always made.) For the worm sweep, though, the stopping time is random, depending on the time for the worm head and tail to approach one another on the lattice. (See section 7.8 for more details.)

9.5 Metropolis steps

See chapter 8, and in particular section 8.3 (Δr_ℓ), for background information on ΔH and Δr_ℓ. A *Metropolis proposal*, for a swap as described in section 5.1, is as follows:

- To find ΔD as described in section 8.1: compute $\|\mathbf{x}-\pi(\mathbf{y})\|_\Lambda^2$ and $\|\mathbf{y}-\pi(\mathbf{x})\|_\Lambda^2$. These will be D terms for the proposed new permutation π', if it is accepted. The corresponding D terms for the current permutation π, namely, $\|\mathbf{x}-\pi(\mathbf{x})\|_\Lambda^2$ and $\|\mathbf{y}-\pi(\mathbf{y})\|_\Lambda^2$, are already cached at the points $\mathbf{x}$ and $\mathbf{y}$. Then

$$\Delta D = \frac{T}{4}\left(\|\mathbf{x}-\pi(\mathbf{y})\|_\Lambda|^2 + \|\mathbf{y}-\pi(\mathbf{x})\|_\Lambda|^2 - \|\mathbf{x}-\pi(\mathbf{x})\|_\Lambda|^2 - \|\mathbf{y}-\pi(\mathbf{y})\|_\Lambda|^2\right).$$

- The subroutine `get_Delta_V_SO` performs the ΔV computations described in chapter 8. It also computes $\mathbf{x} \circ\!\!-\!\!\circ \mathbf{y}$ (namely, whether the points $\mathbf{x}$ and $\mathbf{y}$ are in the same cycle or not), along with $\ell_{\mathbf{x},\mathbf{y}}(\pi)$ and $\ell_{\mathbf{y},\mathbf{x}}(\pi)$. (If $\mathbf{x}$ and $\mathbf{y}$ are in the same cycle, then the cycle lengths are equal; if they are in different cycles, $\ell_{\mathbf{x},\mathbf{y}}(\pi)$ and $\ell_{\mathbf{y},\mathbf{x}}(\pi)$ are undefined.)

- The total change in system energy for the proposed change is $\Delta H = \Delta D + \Delta V$.

The Metropolis proposal is accepted with probability $\min\{1, \exp(-\Delta H)\}$. That is, if a pseudorandom number (section 9.19) uniformly distributed between 0 and 1 is less than $\exp(-\Delta H)$, then π is replaced by π'.

A *Metropolis update* consists of the following:

- The new system energy H' is set to $H + \Delta H$.
- The forward and backward permutation pointers `pfwd` and `pbwd` at points $\mathbf{x}$ and $\mathbf{y}$ are updated so that $\pi'(\mathbf{x}) = \pi(\mathbf{y})$ and $\pi'(\mathbf{x}) = \pi(\mathbf{y})$.
- The cached forward squared distances $\|\mathbf{x} - \pi'(\mathbf{x})\|_\Lambda|^2$ and $\|\mathbf{y} - \pi'(\mathbf{y})\|_\Lambda|^2$ are stored in the `fwd_dsq` slots of the `point_t` data structures for points $\mathbf{x}$ and $\mathbf{y}$. Respective square roots are stored in the `fwd_d` slots.
- Cycle-information structures are updated by the subroutine `update_cycinfo`, which is discussed next.

Without cycle-length caching: Recall from section 8.3 that computing Δr_ℓ requires the following steps:

- See if $\mathbf{x}$ and $\mathbf{y}$ are in the same cycle.
- If so, find $\ell_{\mathbf{x},\mathbf{y}}(\pi)$ and $\ell_{\mathbf{x},\mathbf{y}}(\pi)$; if not, find $\ell_{\mathbf{x}}(\pi)$ and $\ell_{\mathbf{y}}(\pi)$.

To find these values, one may start at site $\mathbf{x}$, moving forward one permutation jump at a time. If one reaches $\mathbf{y}$ before returning to $\mathbf{x}$, then $\mathbf{x}$ and $\mathbf{y}$ are in the same cycle, and $\ell_{\mathbf{x},\mathbf{y}}(\pi)$ has already been found. Continuing to count hops back to $\mathbf{x}$ yields $\ell_{\mathbf{y},\mathbf{x}}(\pi)$. If, on the other hand, one returns to $\mathbf{x}$ without having encountered $\mathbf{y}$, then $\mathbf{x}$ and $\mathbf{y}$ are in different cycles, and $\ell_{\mathbf{x}}(\pi)$ has already been computed. Counting hops from $\mathbf{y}$ back to itself yields $\ell_{\mathbf{y}}(\pi)$.

Notice that each of these cycle-following steps requires $O(\ell)$ machine operations where ℓ is the mean cycle length. For subcritical temperatures T where cycles become long (of length at most N), these cycle-following steps become unacceptably time-consuming. Caching of cycle lengths has been found to reduce simulation time, for $L = 40$ lattices, by a factor of 15. The improvement is even more pronounced as L is increased.

With cycle-length caching: Keeping a list of cycles means that the following Metropolis-proposal steps take $O(1)$ machine operations:

- To see if $\mathbf{x}$ and $\mathbf{y}$ are in the same cycle, check the two points' `pcycinfo` slots and see whether they are equal or not.
- The cycle lengths $\ell_{\mathbf{x}}(\pi)$ and $\ell_{\mathbf{y}}(\pi)$ are immediately found by consulting the `cyclen` slots of the `pcycinfo` data structures.

Now, if $\mathbf{x}$ and $\mathbf{y}$ are in the same cycle, one must additionally find either $\ell_{\mathbf{x},\mathbf{y}}(\pi)$ or $\ell_{\mathbf{y},\mathbf{x}}(\pi)$. (Note that $\ell_{\mathbf{x},\mathbf{y}}(\pi) + \ell_{\mathbf{y},\mathbf{x}}(\pi) = \ell_{\mathbf{x}}(\pi) = \ell_{\mathbf{y}}(\pi)$ so it suffices to find either one or the other.) This is, *a priori*, is of complexity $O(\ell)$. However, it has been found empirically that in the MCMC simulations described by this thesis, one of the two of $\ell_{\mathbf{x},\mathbf{y}}(\pi)$ or $\ell_{\mathbf{y},\mathbf{x}}(\pi)$ is almost always small. That is, a split of a large cycle usually pinches off a small cycle; rarely is a large cycle split evenly. It is likewise found empirically that on merges of disjoint cycles, usually one of the two cycles is small. Thus, if $\mathbf{x}$ and $\mathbf{y}$ are in the same cycle, it suffices to start at $\mathbf{x}$, searching forward and backward one jump at a time. If one encounters $\mathbf{y}$ on forward jumps, $\ell_{\mathbf{x},\mathbf{y}}(\pi)$ has been found; if one encounters $\mathbf{x}$ on backward jumps, $\ell_{\mathbf{y},\mathbf{x}}(\pi)$ has been found. Thus the complexity is of order

$$O(\min\{\ell_{\mathbf{x}}, \ell_{\mathbf{y}}\}).$$

A result of this is that all other computations done in our MCMC simulations are $O(N)$. This bit, however, is necessarily $O(N^2)$. Yet, it is $O(N^2)$ with a low constant of proportionality, since one of $\ell_{\mathbf{x}}$ and $\ell_{\mathbf{y}}$ is almost always small. Plots of CPU time as a function of N are shown in section 9.21.

It has just been demonstrated that keeping cached cycle lengths makes Δr_ℓ computations for a Metropolis proposal quicker. Of course, one pays for this by needing to maintain cached cycle lengths after Metropolis updates. The following steps are performed in the subroutine `update_cycinfo`.

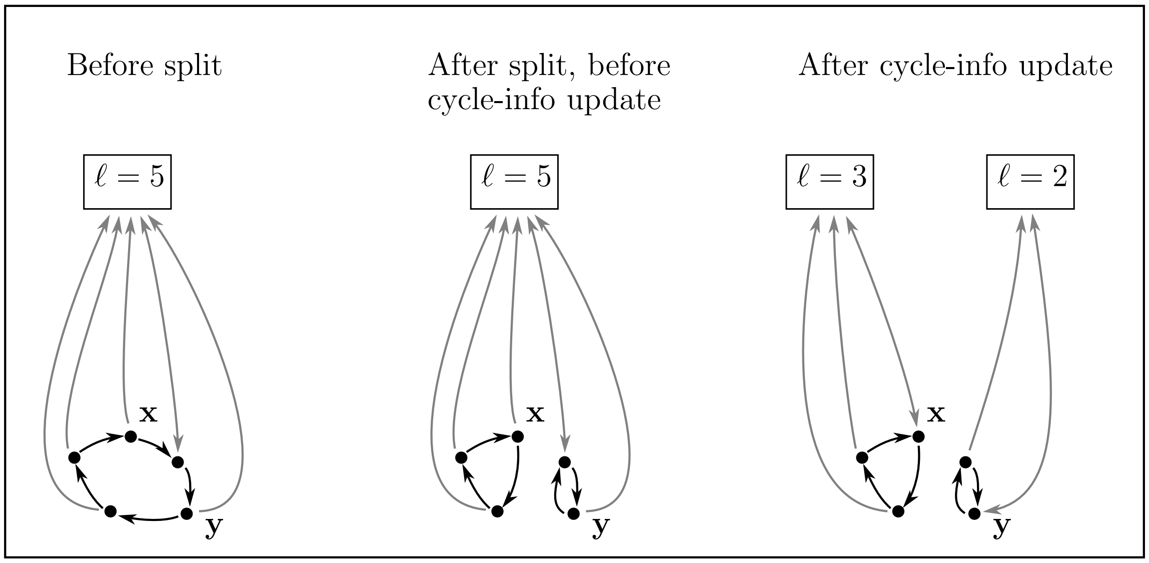


FIGURE 9.2. Update of permutation and cycle information on a split swap. Grey arrows represent pointers between sites and their cycle-information structures; black arrows represent forward permutation pointers.

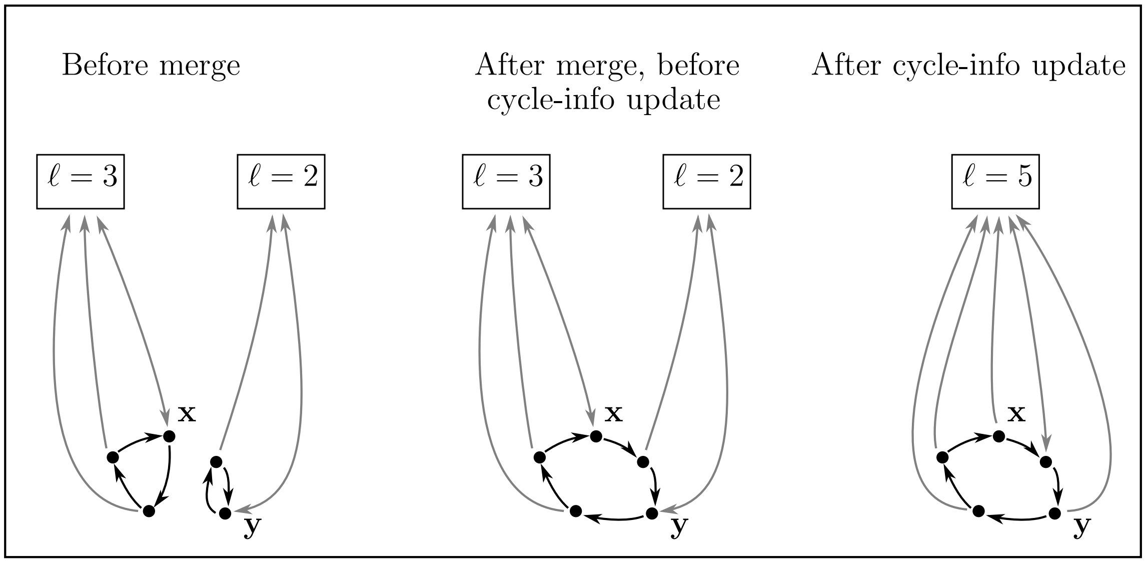


FIGURE 9.3. Update of permutation and cycle information on a merge swap. Grey arrows represent pointers between sites and their cycle-information structures; black arrows represent forward permutation pointers.

If the swap has split a cycle into two (figure 9.2):

- To minimize the number of computations, as described above for Metropolis proposals, find the shorter new cycle. Suppose $\mathbf{x}$'s new cycle is longer than $\mathbf{y}$'s. (If not, swap local variables in the subroutine to make this so.)
- All points in $\mathbf{y}$'s new cycle must now point to a new cycle-information structure. The cycle has been split, so follow from the new $\pi'(\mathbf{y})$ (which was $\pi(\mathbf{x})$ before the merge) around to and including $\mathbf{y}$. As above, the number of sites that must be visited is $\min\{\ell_{\mathbf{x}}(\pi'), \ell_{\mathbf{y}}(\pi')\}$.
- The cycle-information structures for both split cycles need to have their cycle lengths updated: $\ell_{\mathbf{x}}(\pi') = \ell_{\mathbf{y},\mathbf{x}}(\pi)$ and $\ell_{\mathbf{y}}(\pi') = \ell_{\mathbf{x},\mathbf{y}}(\pi)$.
- The cycle-information structures for the two cycles need to have their site pointers point to $\mathbf{x}$ and $\mathbf{y}$, respectively.
- The new cycle-information structure for the $\mathbf{y}$ cycle must be added to the doubly linked list of cycle-information structures.

If the swap has merged two cycles into one (figure 9.3):

- To minimize the number of computations, find the shorter old cycle. Suppose $\mathbf{x}$'s old cycle is longer than $\mathbf{y}$'s. (If not, swap local variables in the subroutine to make this so.)
- The cycle-information structure for the merged cycle needs to have its cycle length updated: $\ell_{\mathbf{x}}(\pi') = \ell_{\mathbf{x}}(\pi) + \ell_{\mathbf{y}}(\pi)$.
- All points in $\mathbf{y}$'s old cycle must have their cycle-information structures now point to $\mathbf{x}$'s cycle-information structure. The two cycles have been merged, though, so follow from the new $\pi'(\mathbf{x})$ (which was $\pi(\mathbf{y})$ before the merge) around

to and including $\mathbf{y}$. As above, the number of sites that must be visited is $\min\{\ell_{\mathbf{x}}(\pi), \ell_{\mathbf{y}}(\pi)\}$.

- The cycle-information structure for the old $\mathbf{y}$ cycle must be removed from the cycle-information list and freed.

9.6 Thermalization detection

As noted in section 9.2, the initial permutation selected in an MCMC sequence is atypical: the identity permutation has zero energy, lower than the mean energy for the stationary distribution, whereas a uniform-random permutation almost always has energy higher than the mean due to its long jump lengths. (See equation (2.1.3)and figure 9.4.)

As always in MCMC simulations [Berg, LB], one must run through some number of Metropolis steps until the system has thermalized, i.e. when the stationary distribution has been reached. In the MCMC discipline, practitioners use various techniques. The thermalization-detection algorithm chosen for the work described by this dissertation counts turning points of smoothed system energy. This algorithm may be justified using conditional expectation of ΔH, as well as visually.

Recall from chapters 2 and 5.2 that permutations have energies $H(\pi)$ and $H(\pi')$, probabilities $P_{\text{Gibbs}}(\pi) = e^{-H(\pi)}/Z$ and $P_{\text{Gibbs}}(\pi') = e^{-H(\pi')}/Z$, and Metropolis transition probabilities

$$A(\pi, \pi') = C(1 \wedge e^{-(H(\pi')-H(\pi))}).$$

The premise is that a permutation π is taken from the stationary distribution if the subsequent permutation π' is equally likely to have higher or lower energy. Stated probabilistically, we say that the expected value of $H(\pi')$, conditioned on transitioning from π, should be zero. Recall that for a random variable X and an event A, with

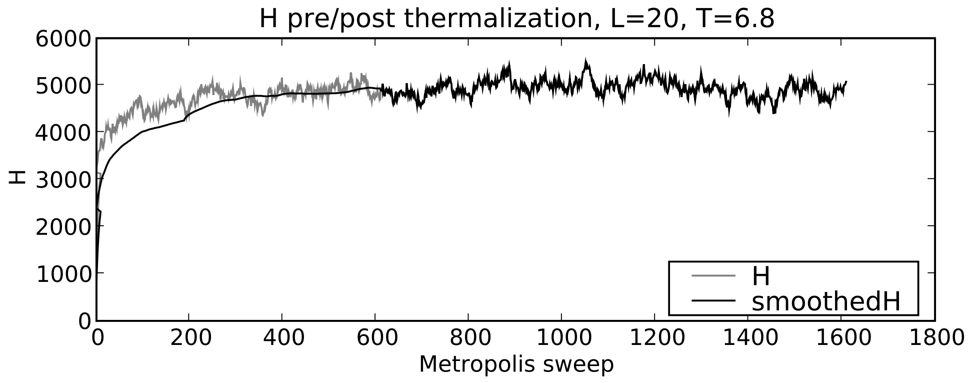


FIGURE 9.4. Plot of system energy versus Metropolis sweep number. The transition from grey to black indicates thermalization was detected, via 40 turning points of energy smoothed over a sliding window of 200 sweeps.

outcomes x, we have

$$\mathbb{E}[f(X) \mid X \in A] = \frac{\displaystyle\sum_{x\in A} P(x) f(x)}{\displaystyle\sum_{x\in A} P(x)} = \frac{\displaystyle\sum_{x\in A} P(x) f(x)}{P(A)}.$$

Here, this translates to

$$\mathbb{E}[H(\pi') \mid \pi] = \frac{\displaystyle\sum_{\pi' \circ\!\!-\!\!\circ\, \pi} P_{\text{Gibbs}}(\pi') H(\pi')}{\displaystyle\sum_{\pi' \circ\!\!-\!\!\circ\, \pi} P_{\text{Gibbs}}(\pi')} = \frac{\displaystyle\sum_{\pi' \circ\!\!-\!\!\circ\, \pi} P_{\text{Gibbs}}(\pi) A(\pi,\pi') H(\pi')}{\displaystyle\sum_{\pi' \circ\!\!-\!\!\circ\, \pi} P_{\text{Gibbs}}(\pi) A(\pi,\pi')}$$

$$= \frac{\displaystyle\sum_{\pi' \circ\!\!-\!\!\circ\, \pi} P_{\text{Gibbs}}(\pi) A(\pi,\pi') (H(\pi) + \Delta H)}{\displaystyle\sum_{\pi' \circ\!\!-\!\!\circ\, \pi} P_{\text{Gibbs}}(\pi) A(\pi,\pi')}$$

$$= \frac{\displaystyle\sum_{\pi' \circ\!\!-\!\!\circ\, \pi} P_{\text{Gibbs}}(\pi) A(\pi,\pi') H(\pi) + \sum_{\pi' \circ\!\!-\!\!\circ\, \pi} P_{\text{Gibbs}}(\pi) A(\pi,\pi') \Delta H}{\displaystyle\sum_{\pi' \circ\!\!-\!\!\circ\, \pi} P_{\text{Gibbs}}(\pi) A(\pi,\pi')}$$

$$= H(\pi) + \frac{\sum\limits_{\pi' \circ\!\!-\!\!\circ\, \pi} P_{\text{Gibbs}}(\pi) A(\pi, \pi') \Delta H}{\sum\limits_{\pi' \circ\!\!-\!\!\circ\, \pi} P_{\text{Gibbs}}(\pi) A(\pi, \pi')}.$$

(Recall from definition 5.2.4 that the sum over $\pi' \circ\!\!-\!\!\circ\, \pi$ includes all permutations π', including π itself, reachable from an accepted or rejected swap.) For pre-thermalization π, this expectation should be highly positive (when the initial π_1 is the identity), or highly negative (when the initial π_1 is uniform-random). For post-thermalization π, this expectation should be approximately zero. Alternatively, for various post-thermalization π, this expectation should be equally likely positive or negative. As discussed in section 5.2, given π, there are $3N + 1$ choices of π' (including π itself) which could be reached on a swap. Yet, in the MCMC sequence, only one π' is chosen. Thus, this conditional expectation is CPU-intensive to compute, and moreover does not take advantage of the MCMC sequence itself.

We estimate the above conditional expectation by first defining

$$H_S(t) = \frac{1}{S} \sum_{i=0}^{S-1} H(\pi_{t-i}),$$

where S is the *smoothing window size* and t denotes a Metropolis sweep counter with $t \geq S$. That is, the system energy at Metropolis sweep t is averaged over the last S sweeps. The system is deemed to be thermalized when $H_S(t)$ has changed sign sufficiently many times, i.e. when $H_S(t)$ has had more than a threshold number of turning points.

As is typically the case in such matters, rigorously determining the spectral gap in the Markov transition matrix is computationally intractable. One relies instead on heuristics which are justified by the practitioner's experience and all available evidence. Visually examining the plot in figure 9.4, one may decide that thermalization has occurred by, say, Metropolis sweep 300. Examining a plot for different L, T, and interaction strength, one might choose a different Metropolis sweep count. I have examined such plots over a broad range of parameter values; I have chosen $S = 200$

and threshold number of turning points equal to 40, ensuring that the automated thermalization detection (when grey turns to black in figure 9.4) agrees with my visual judgment. Thermalization takes one percent or less of total CPU time, as shown in the caption of figure 9.5.

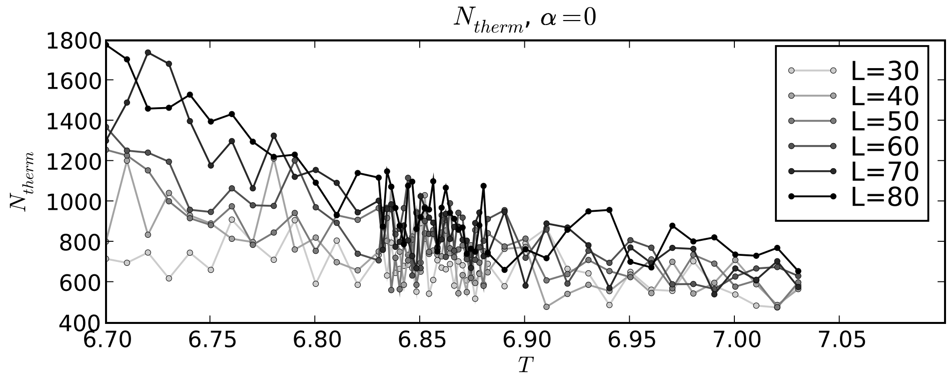


FIGURE 9.5. Thermalization time as a function of L and T, for $\alpha = 0$. Plots for other α are similar. Data accumulation takes 10^5 or 10^6 sweeps; thermalization is here seen to take on the order of 10^3 sweeps, i.e. a fraction of a percent of CPU time.

9.7 Computation of random variables

All the random variables described in chapter 3 are computed in `mcrcm`. Details of each are described in the sections following this one. The software architecture of `mcrcm` is such that it is easy to add a newly invented random variable to the code.

9.8 Computation of system energy

The system energy H, with non-interacting terms D and interacting terms V (as defined in chapter 2), is tracked; one easily scales to obtain the energy per site (or energy density) $h = H/N$, $d = D/N$, and $v = V/N$.

As described in section 9.3, system energy H is computed for the initial permutation. As described in section 9.4, ΔH (split out into ΔD and ΔV) is computed for a proposed new permutation; if the proposal is accepted, H is replaced by $H + \Delta H$. As described in section 9.16, optional automated checks verify that the accumulated ΔH computations (chapter 8) correctly track the true system energy.

A plot of H as a function of Metropolis sweep, within a single invocation of `mcrcm`, is shown in figure 9.4 on page 117. Dependence of H on L, T, and α is shown in figure 9.6. Note that the system energy, as centrally important as it is, does not function as an order parameter (section 3.7): it exhibits no sharp transition near the critical temperature.

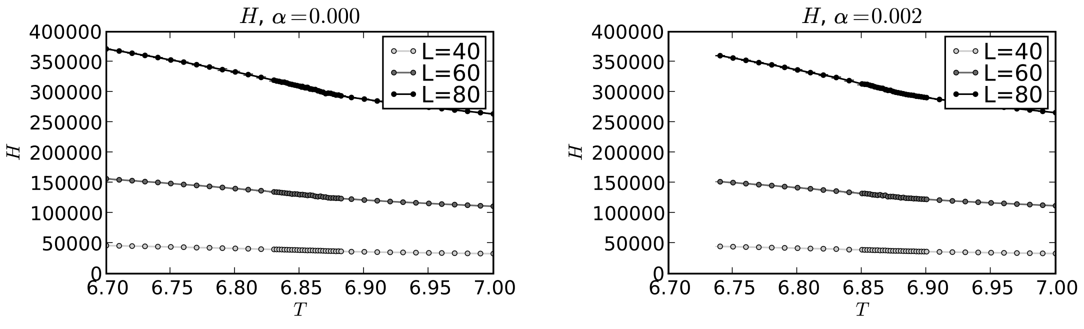


FIGURE 9.6. Behavior of system energy H as function of L and T, for $\alpha = 0$, 0.002.

9.9 Computation of $r_\ell(\pi)$

Given the cycle-information list as described in sections 9.1 and 9.5, it is trivial to compute the cycle-length occupation numbers $r_\ell(\pi)$ for $\ell = 1$ to N. Namely: Set $r_1, \ldots, r_N = 0$. For each cycle in the cycle list, increment r_ℓ by 1 where ℓ is the length of the current cycle. Dependence of the sample mean of r_2 as a function of L, T, and α is shown in figure 9.7.

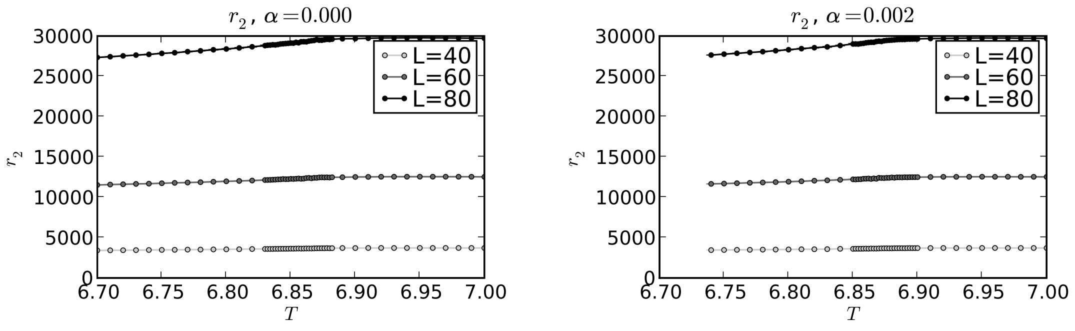


FIGURE 9.7. Behavior of r_2 as function of L and T, for $\alpha = 0$, 0.002.

9.10 Computation of cycle lengths and correlation length

Given the lattice data structure as described in section 9.1, one easily computes the the spatial cycle lengths $s_{\mathbf{x}}(\pi)$ defined in section 3.2. For each site $\mathbf{x}$ in the lattice Λ, one examines the cached `fwd_d` attribute, which is precisely $s_{\mathbf{x}}(\pi)$. As described in section 3.2, the correlation length ξ is the spatial cycle length averaged over all N lattice points. Plots and analysis of the order parameter $1/\xi$ may be found in chapter 11.

9.11 Computation of mean and maximum jump length

Mean jump length is as defined in section 3.3. The maximum jump length is the largest jump length encountered at any of the N lattice sites, for any of the M permutations in the MCMC sequence. This confirms the hypothesis of short jump lengths as mentioned in section 3.6. See figure 9.8. As discussed in section 3.2, the jump length is averaged not only over all permutations π generated in the MCMC sequence, but moreover is averaged over all N lattice points for each π.

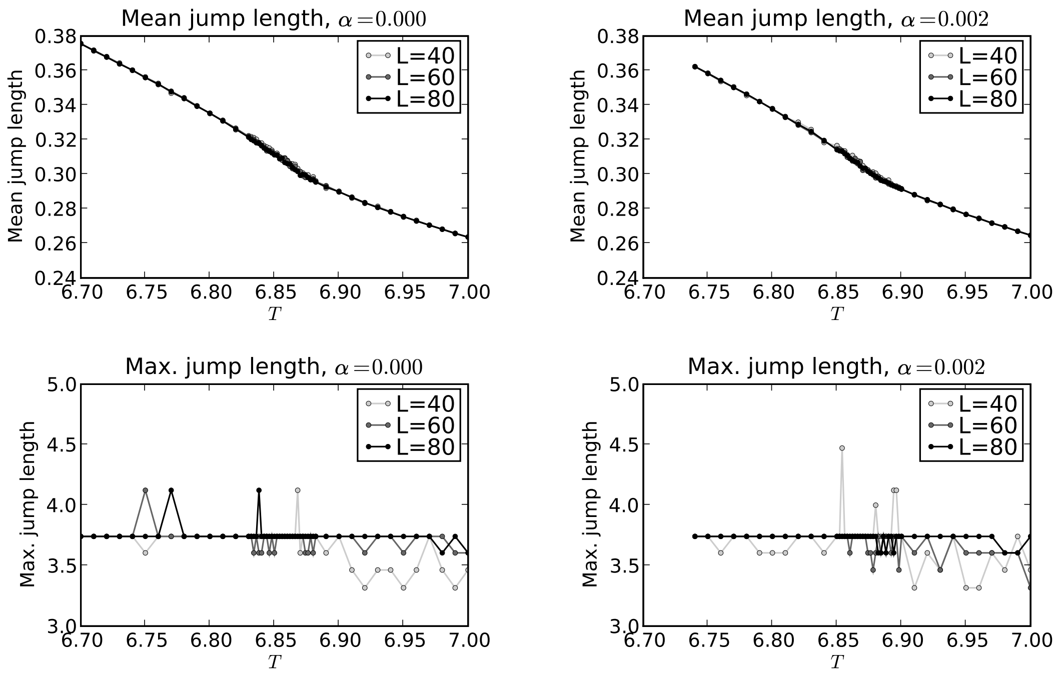


FIGURE 9.8. Mean and maximum jump length as function of L and T, for $\alpha = 0$, 0.002.

9.12 Computation of f_I

As discussed in section 3.4, f_I is computed using the vertical intercept of a tangent-line approximation to $\mathbb{E}[f_{1,k}]$ as a function of k/N.

Computation of $\mathbb{E}[f_{1,k}]$ as a function of k is straightforward: we maintain an array indexed from 1 to N of counters, all initially set to zero. For each permutation π, we count the number of sites participating in cycles of length k. This is directly obtained from the cycle-information structure (see section 9.1). Scaling this array by $1/N$ yields a finite-sample approximation to $\mathbb{E}[f_{k,k}]$. Cumulatively summing the array gives an estimator of $\mathbb{E}[f_{1,k}]$. See figure 9.9.

Given that, the tangent line is found, in the presence of statistical variability, as follows: subtract off the diagonal line $(k/N, k/N)$ which runs from the lower left corner to the upper right corner. The peak of the difference (the dashed line in figure

9.9) shows the outermost point of the original $\mathbb{E}[f_{1,k}]$ estimator. Then the tangent line is taken to have slope 1 running through this point. As the number M of permutations increases, this outermost point adheres closely to the visible diagonal; its location is defined by its peers. Thus, we are not computing f_I based on a single, noisy data point, but rather using much of the available $\mathbb{E}[f_{1,k}]$ data.

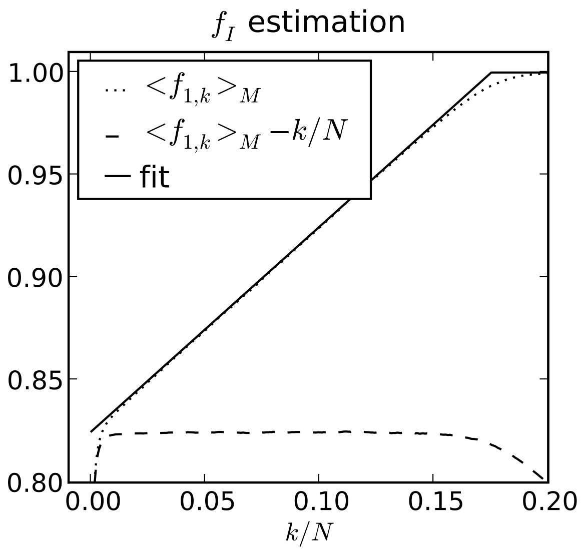


FIGURE 9.9. Estimation of f_I. The dotted line shows raw $\langle f_{1,k}\rangle_M$ data obtained by cumulatively summing an array of sample averages of $f_{k,k}(\pi)$. The dashed line is the dotted line minus the diagonal $(k/N, k/N)$. The abscissa of the peak of this difference is the abscissa of the outermost point of the dotted line. The tangent line is drawn through there, with slope 1. Then f_I is one minus the vertical intercept of that tangent line: in this case, $f_I = 1 - 0.825 = 0.175$. The simulation used an MCMC run of 10^4 permutations on $L = 20$ at $T = 6.5$.

9.13 Computation of $\ell_{\max}$, $f_{\max}$, and macroscopic-cycle quotient

The quantities $\ell_{\max}$, $f_{\max}$, f_I, and macroscopic-cycle quotient $f_{\max}/f_I$ were defined in sections 3.4 and 3.5. It is easy to compute $\ell_{\max}$: find the longest cycle in the cycle-information list (section 9.1). Computation of f_I is found as described in section 9.12.

Figure 9.10 makes clear the difference between subcritical and supercritical behavior which was alluded to in section 2.3. Below T_c, r_1, r_2, etc. are smaller than above T_c, since there is occasionally a long cycle: $\ell_{\max}$ is markedly higher below T_c. Plots and analysis of the order parameter $f_{\max}$ may be found in chapter 11. The macroscopic-cycle quotient is analyzed in section 11.8.

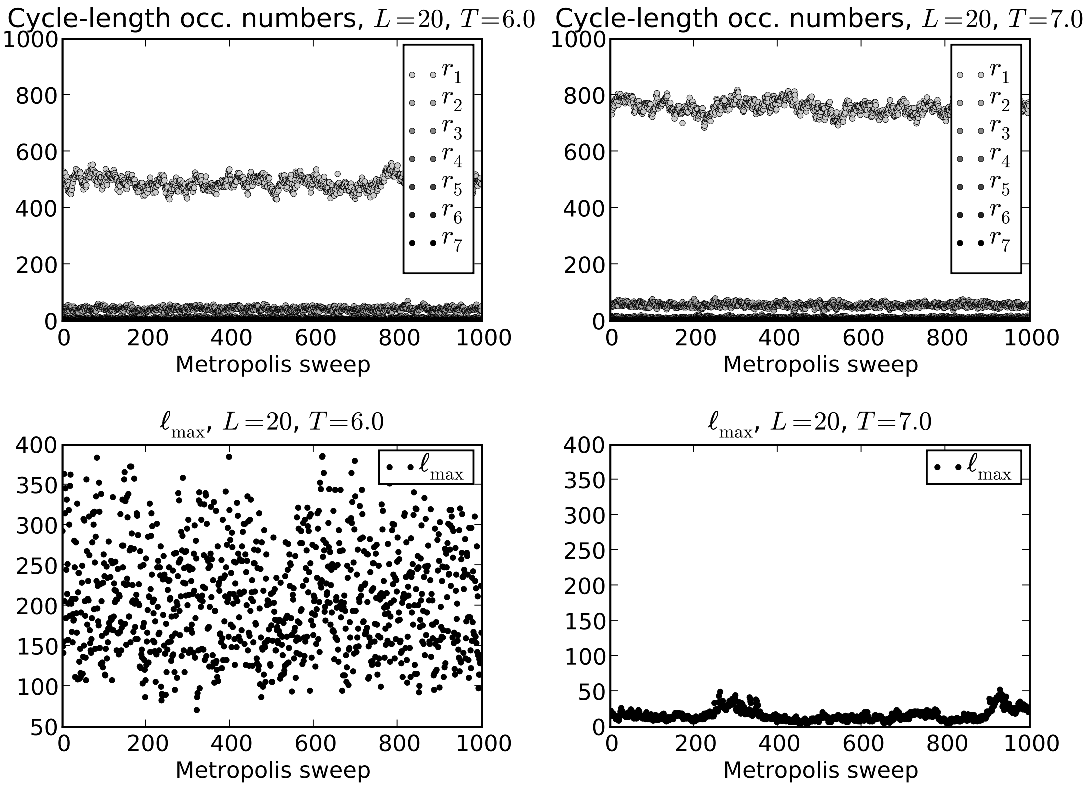


FIGURE 9.10. Per-realization values of r_ℓ and $\ell_{\max}$ with $L = 20$, $T = 6.0, 7.0$.

9.14 Computation of winding numbers, f_S, and f_W

The winding-number triple $\mathbf{W} = (W_x, W_y, W_z)$ is as defined in section 3.6. Straightforward application of equations (3.6.2) and (3.6.3), for $\mathbf{W}$ and $\mathbf{W}^2$, respectively, are sufficient as long as the difference vectors $\mathbf{d}_\Lambda(\mathbf{x}, \mathbf{y})$ are obtained. For these (see also section 3.1) one first computes the difference $\mathbf{x} - \mathbf{y}$. Then, for each of the three coordinate slots, one adds or subtracts multiples of L until the result is between $-L/2$ and $L/2$. In figure 9.11 one observes the even parity of winding-number components, as discussed in section 5.4. As well, the temperature dependence in the histograms shows the subcritical transition to winding cycles. Plots and analyses of the order parameters f_S and f_W may be found in chapter 11.

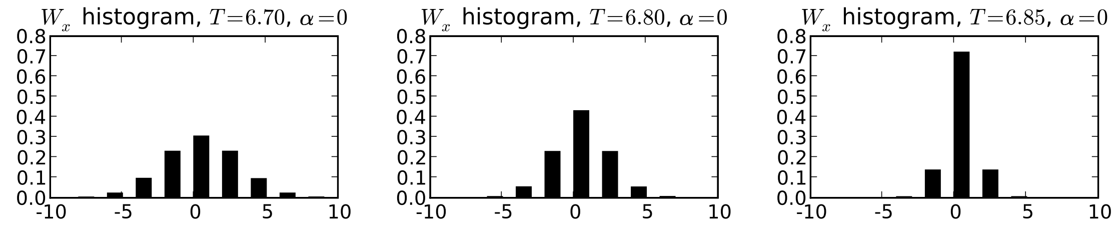


FIGURE 9.11. Histograms of W_x for $\alpha = 0$ and $T = 6.70$, 6.80, 6.85.

9.15 Computation of integrated autocorrelation times

Integrated autocorrelation times are estimated precisely as described in sections B.7 through B.10 of appendix B. Error bars in order-parameter plots in this dissertation are the τ_{int}-corrected sample standard deviations of the sample means.

Some estimated integrated autocorrelation times are shown in figure3 9.12. The key point is that uncertainty increases in the critical region. For this reason, simulations in the critical region were run with 10^6 Metropolis sweeps for $L = 30, 40, 50$, and with 10^5 sweeps otherwise. Similarly, figure 9.13 shows estimates of the correlation factors η , in the context of appendix B. Namely, $\eta = 0$ yields an IID sequence; η for

MCMC simulations done in this dissertation are typically 0.99 and above, indicating highly autocorrelated MCMC sequences.

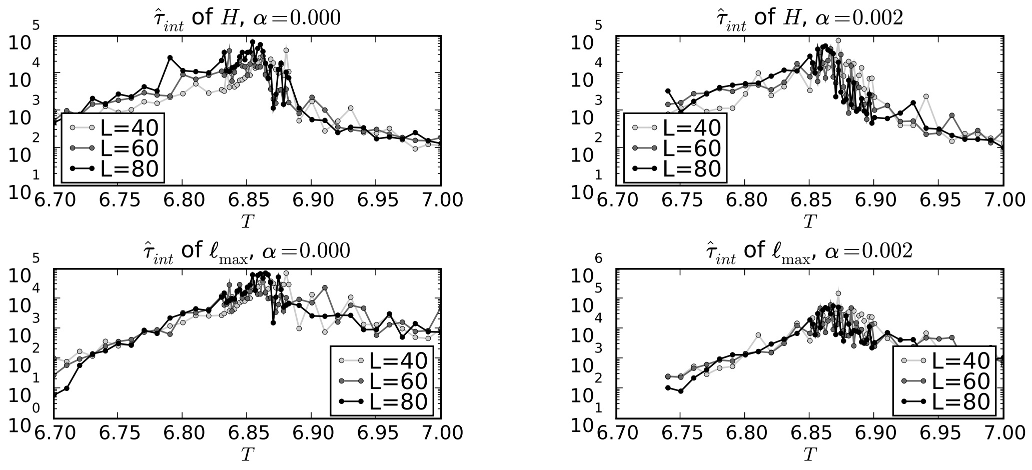


FIGURE 9.12. Estimated integrated autocorrelation times of H and $\ell_{\max}$ as functions of L and T, for $\alpha = 0$ and 0.002.

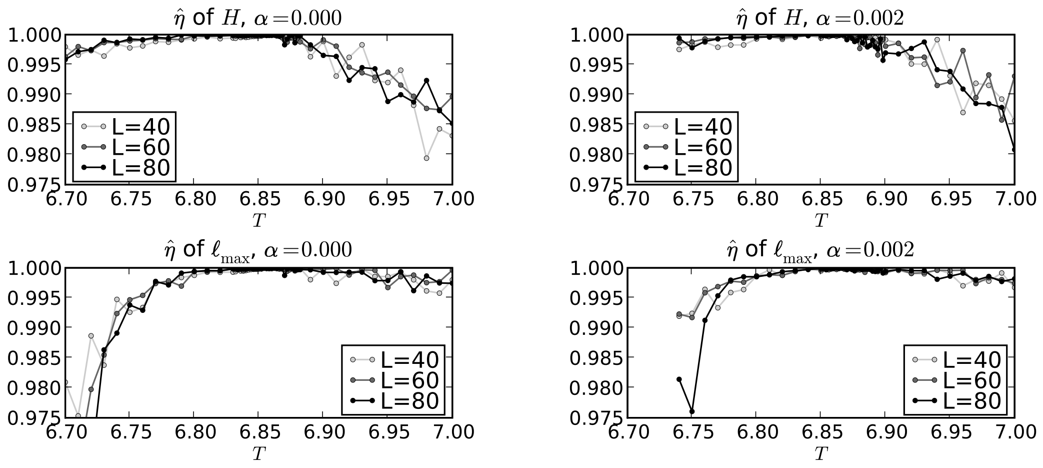


FIGURE 9.13. Estimated autocorrelation factor of H and $\ell_{\max}$ as functions of L and T, for $\alpha = 0$ and 0.002.

9.16 Consistency checks

The following checks are run for every MCMC simulation described in this dissertation:

- Interaction type is one of none, two-cycle interactions, r_ℓ (Ewens) interactions. This protects against uninitialized variables.
- Worm move is one of open, close, head swap, or tail swap. Again, this protects against uninitialized variables.
- $L \geq 1$; $d = 1, 2, 3$.
- The cycle-information list is never empty when a cycle-information structure is removed (e.g. on a merge).

The following checks are omitted from production runs due to their CPU-time expense, but were run during code development and testing:

- `check_H`: As described in sections 9.3 and 9.5, the system energy is initially computed by `get_H_of_pi`, then updated by ΔH as computed in Metropolis proposals/updates. If H-checking is enabled (in the header file `checks.h`), then after every Metropolis sweep, the system energy is computed by brute force, and is verified to be within roundoff error of the system energy which has been tracked by ΔH computations.
- `sanity_check_cycinfo_list`: This is also enabled in the header file `checks.h`. It consists of three checks, which are run after every Metropolis sweep: (1) The cycle list partitions the lattice sites. All sites are marked unvisited; all cycles in the cycle-information list are followed, with visits to each site counted; each site is checked to have been visited no more and no less than once. (2) The cached cycle lengths are correct: These are verified by following permutation pointers

around each cycle. (3) For each cycle, all points in the cycle are verified to point to the same cycle-information structure.

The following visual checks were done for selected single runs:

- H plots, such as figure 9.4 of section 9.6 should show system energy increasing from 0 up to equilibrium for initial identity; decreasing down to equilibrium for initial uniform-random.

Visual checks for single runs:

- Output from `mcrcm`, as shown in section 9.18, looks sane; numbers appear to be in their normal ranges.
- $\langle f_{1,k} \rangle_M$ plots, as shown in section 9.12, are as usual.
- f_W, $f_{\max}$, and f_I are between 0 and 1.
- Sample mean of $\sum_{\ell=1}^{N} \ell r_\ell$ is equal to N. (This is `k*counts sum` in the MCMC output, as described in section 9.18.)
- Sample mean of energy density h is of order 1.
- Mean jump length is of order 1; maximum jump length is approximately 3-4.
- Winding-number histogram peaks at $W_x = 0$, with population out to perhaps ± 6 depending on system temperature.
- Metropolis acceptance rate is 10 to 20 percent.

Visual checks for runs over parameter values (chapter 10):

- A 30-page file containing all plots of the form shown in this chapter — not just a representative selection — is used to compare random-variable output to results from before a software change.

Automated software checking:

- All source code is compiled with `gcc -Wall -Werror`. This enables all warnings (e.g. unused variables, reading variables before they are initialized, missing arguments to `printf`, and so on), and treats warnings as fatal compilation errors.

- The open-source `valgrind` tool finds, at run time, many (but certainly not all) common programming errors. These include reading or writing off the end of arrays, continuing to use dynamically allocated memory after it is freed, dynamically allocated memory which is not freed, and reading of uninitialized variables.

- The open-source `gprof` tool tabulates in which subroutines CPU time is being spent. This helps to identify inefficient programming. See also the `gprof` output in section 9.20.

9.17 Saved realizations

The computer program described up to this point is called `mcrcm`, since it does Markov chain Monte Carlo simulations for the random-cycle model. As described in section 9.2, it uses MCMC methods to generate a sequence of permutations, accumulating sums of various random variables along the way, in order to display statistics of those random variables at the end.

A key software-optimization insight was offered by Volker Betz. Namely, the MCMC steps are time-consuming, so one might optionally wish to have `mcrcm` write the sequence of realized permutations to a disk file. Then, another program can simply read those permutations — quicker than they were generated using MCMC methods — and compute random variables. This is particularly advantageous in a shoulder-to-shoulder collaboration environment: when one invents a new random variable to compute, one need not re-realize another sequence of permutations.

The trade-off is that a single realization file can be quite large. Specifically, M permutations are stored. For each permtuation, 3 bytes are stored for each of N lattice sites $\mathbf{x}_i$: these are the x, y, and z coordinates of $\pi(\mathbf{x}_i)$. The realization file contains a header and a footer of negligible size (a few bytes each). Thus, it totals $3ML^3$ in size — for example, 240 MB for $L = 20$ and $M = 10^4$, and one would likely want several such files for different values of L, T, and/or α. One might, of course, develop more clever storage representations which reduce the necessary file size.

This second program is called `rvrcm`, since it computes random variables for the random-cycle model. By default, `mcrcm` does not store realization files: they occupy several megabytes for each run, totalling several gigabytes for a run through a set of parameters (chapter 10). If desired, however, one invokes `mcrcm` with an extra command-line argument `rzn=`*myfile*`.rzn`. Later, one invokes simply `rvrcm rzn=`*myfile*`.rzn`. The `.rzn` file contains a header with all control parameters which were initially supplied to `mcrcm`. The `rvrcm` program prints the same information as described in section 9.18.

For example:

```
mcrcm L=20 T=6.4 rell alpha0=0.2 rzn=experiment.rzn
rvrcm rzn=experiment.rzn
```

A `.rzn` file consists of three parts: The header contains all control parameters, e.g. L, T, interaction type, etc. The footer contains elapsed time spent in `mcrcm` as well as Metropolis statistics, i.e. acceptance rate for Metropolis proposals. In between is a sequence of permutation realizations.

9.18 MCMC output

Outputs from `mcrcm` fall into two categories: (1) sample statistics of random variables, which are always printed; (2) per-realization values of specified random variables, which are optionally printed.

Each invocation of `mcrcm` prints information such as the following. For interactive use, one views these data on-screen, or may send them to a printer. When running through a set of parameters, typically this output is redirected to a self-descriptive filename, e.g. `L_80_T_6.7_alpha_0.000.txt`, as described in chapter 10.

```
# RNG = Mersenne twister
# L = 80 d = 3 N = 512000
# Boundary conditions: periodic.
# T =   6.7000000 beta =   0.1492537 alpha0 =   0.0000000
# gamma =   0.0500000
# Interactions:  constant r_ell.
# Initial permutation: identity.
# Site selection for Metropolis sweeps: sequential.
# Thermalization is detected by 40 turning points of system energy
# smoothed over a 200-point window.
# Terminate after 100000 accumulations:
# * 1 SAR  sweep per accumulation.
# * 0 worm sweeps per accumulation.
#
# Initial HDV =   0.0000000   0.0000000   0.0000000
# Sweep 0 HDV = 54809.3500000 54809.3500000   0.0000000
# Thermalization complete:  sweep   1778 H = 359840.2499997
# Ntherm =   1778
# Accumulation complete:  acc 100000 H = 373669.0499998
```

```
#
# rho_infty max dev =   0.9140418 at k = 21976
# fI                            =    0.0859582
#
# k=  1 <counts> 339457.4268600
# k=  2 <counts> 27347.3173400
# k=  3 <counts> 5740.3265700
# k=  4 <counts> 2631.4549600
# k=  5 <counts> 1342.3165500
# k=  6 <counts> 804.4376700
# k=  7 <counts> 512.2292200
# k=  8 <counts> 351.5231700
# k=  9 <counts> 252.2729800
# k= 10 <counts> 188.1364500
# k*counts sum                  = 512000.0000000
#
# fM                            =    0.0537150
# fM/fI                         =    0.6248961
#
# mean_H                        = 371597.3002873
# stddev_H                      = 1701.6752760
# tauint_H                      =  481.1931762
# eta_H                         =    0.9958523
# cssm_H                        =  118.0419619
#
# mean_D                        = 371597.3002873
# mean_V                        =    0.0000000
# mean_h                        =    0.7257760
# mean_d                        =    0.7257760
# mean_v                        =    0.0000000
#
# mean_r2                       = 27347.3173400
# stddev_r2                     =  172.6096035
# tauint_r2                     =  163.4730368
# eta_r2                        =    0.9878400
# cssm_r2                       =    6.9789168
#
# mean_lmax                     = 27502.0656900
# stddev_lmax                   = 8577.6258591
# tauint_lmax                   =    6.0556505
# eta_lmax                      =    0.7165392
# cssm_lmax                     =   66.7494207
#
# mean_jumplenbar               =    0.3754422
# stddev_jumplenbar             =    0.0015945
# maxjumplen                    =    3.7416574
#
# mean_ellbar                   = 1893.6137332
# stddev_ellbar                 =  821.5275072
# tauint_ellbar                 =   12.1998395
# eta_ellbar                    =    0.8484830
# cssm_ellbar                   =    9.0740082
#
# mean_recipmeanspatlen         =    0.0005406
# stddev_recipmeanspatlen       =    0.0002356
# tauint_recipmeanspatlen       =   38.0712869
# eta_recipmeanspatlen          =    0.9488115
# cssm_recipmeanspatlen         =    0.0000046
#
# mean_wno                      =   20.3278800
# stddev_wno                    =   16.5433065
#
# mean_fS                       =    0.5674866
# stddev_fS                     =    0.4618340
# tauint_fS                     =    3.5586073
# eta_fS                        =    0.5612695
# cssm_fS                       =    0.0027550
# recip_fS                      =    1.7621560
#
# mean_fW                       =    0.0810994
# stddev_fW                     =    0.0082444
# tauint_fW                     =  155.0150517
# eta_fW                        =    0.9871807
# cssm_fW                       =    0.0003246
# recip_fW                      =   12.3305479
#
# Wx < -10:         1 /  100000 = 0.00001000
# Wx = -10:        14 /  100000 = 0.00014000
# Wx =  -9:         0 /  100000 = 0.00000000
# Wx =  -8:       233 /  100000 = 0.00233000
# Wx =  -7:         0 /  100000 = 0.00000000
# Wx =  -6:      2195 /  100000 = 0.02195000
# Wx =  -5:         0 /  100000 = 0.00000000
# Wx =  -4:      9496 /  100000 = 0.09496000
# Wx =  -3:         0 /  100000 = 0.00000000
# Wx =  -2:     22900 /  100000 = 0.22900000
# Wx =  -1:         0 /  100000 = 0.00000000
# Wx =   0:     30423 /  100000 = 0.30423000
# Wx =   1:         0 /  100000 = 0.00000000
# Wx =   2:     22919 /  100000 = 0.22919000
# Wx =   3:         0 /  100000 = 0.00000000
# Wx =   4:      9341 /  100000 = 0.09341000
# Wx =   5:         0 /  100000 = 0.00000000
# Wx =   6:      2196 /  100000 = 0.02196000
# Wx =   7:         0 /  100000 = 0.00000000
# Wx =   8:       261 /  100000 = 0.00261000
# Wx =   9:         0 /  100000 = 0.00000000
# Wx =  10:        19 /  100000 = 0.00019000
# Wx >  10:         2 /  100000 = 0.00002000
# Wx_min    =     -12
# Wx_max    =      12
#
# OPENS:
#   None.
# CLOSES:
#   None.
# HS:
#   None.
# TS:
#   None.
# GK:
#   Metropolis keeps:   44596819174 / 51200000000 ( 87.103%)
#   Metropolis changes: 6603180826 / 51200000000 ( 12.897%)
# Elapsed thermalization seconds: 302.841613
# Elapsed accumulation   seconds: 136966.313018
# Elapsed total seconds:          137269.154631
```

In addition to the above sample statistics, per-realization values of specified random variables may also be printed. A full list of options may be found by invoking `mcrcm --help`. For example, `mcrcm hv=1` (i.e. H verbosity is set to 1, rather than its default value of 0) results in the following output:

```
# RNG = Mersenne twister
# L = 10 d = 3 N = 1000
... (header information is similar to the previous example)
# Initial HDV =   0.00000   0.00000   0.00000
# Sweep 0 HDV = 132.60000 132.60000   0.00000
    0 132.60000 132.60000   0.00000 132.60000   0.00000 # 0 H D V
    1 190.40000 190.40000   0.00000 161.50000   0.00000 # 0 H D V
   ... (many per-realization values omitted for brevity)
  597 629.00000 629.00000   0.00000 603.73065   0.00000 # 0 H D V
  598 632.40000 632.40000   0.00000 603.73065   0.00000 # 0 H D V
  599 642.60000 642.60000   0.00000 603.73065   0.00000 # 0 H D V
# Thermalization complete:  sweep    600 H = 642.6000000
  600 598.40000 598.40000   0.00000 598.40000 598.40000 # 1 H
  601 591.60000 591.60000   0.00000 591.60000 591.60000 # 1 H
  602 581.40000 581.40000   0.00000 581.40000 581.40000 # 1 H
   ... (many per-realization values omitted for brevity)
 1597 581.40000 581.40000   0.00000 581.40000 581.40000 # 1 H
 1598 578.00000 578.00000   0.00000 578.00000 578.00000 # 1 H
 1599 595.00000 595.00000   0.00000 595.00000 595.00000 # 1 H
```

```
# Accumulation complete:  acc 1000 H = 372745.0766667
#
# rho_infty max dev =   0.9471790 at k = 27
# fI                           =    0.0528210
... (statistical output as in the previous example)
# Elapsed thermalization seconds: 0.241671
# Elapsed accumulation   seconds: 0.477683
# Elapsed total seconds:          0.719354
```

Notice that all the sample statistics are printed as before, on lines which start with a pound sign. In addition, since H verbosity was requested, non-pound-sign lines show Metropolis sweep number, H, D, V, smoothed H (see section 9.6), and thermalization flag times smoothed H. With pound-sign lines stripped out or ignored, a graphing utility can then be used to produce a plot such as that shown in figure 9.4 on page 117.

9.19 Pseudorandom numbers

The default pseudorandom-number generator ("RNG") is the Mersenne Twister [MN]. One may instead select, at compile time via `rcmrand.h`, `rand48` (of lower quality than Mersenne twister), Linux `/dev/urandom` (slower than Mersenne twister), or `psdes` from Numerical Recipes [NR].

9.20 Tools

- Linux environment, although: in principle everything other than use of `/dev/urandom` should be portable to other operating systems.
- Optimizing compiler with full warnings enabled: `gcc -O3 -Wall -Werror`.
- Build tool: `make` and automatic makefile generation.

- Performance analyzer: `gprof`. This shows where a program is spending most of its time.
- Error detector: `valgrind`. Finds many (but not all) common errors, e.g. `malloc` without `free`, or double `free`.
- Code navigation: `ctags`. Allows a smart text editor (e.g. `vim`, `emacs`) to jump directly to a subroutine body.
- Graphing utility, used for all plots in this dissertation: `pgr`, which is the author's Python script wrapped around `pylab.plot()`.

Sample `gprof` output:

```
%     cumul.self         self    total
time  secs. secs.  calls us/call us/call name
32.52 1.20  1.20      1500 0.80   1.41  SO_sweep
25.47 2.14  0.94 21597444 0.00   0.00  get_mtrand_double
 9.49 2.49  0.35      1500 0.23   0.51  U_sweep
 7.32 2.76  0.27      1000 0.27   0.27  get_pmt_winding_numbers
 5.15 2.95  0.19      1000 0.19   0.19  get_mean_cycle_length
 4.61 3.12  0.17 12000000 0.00   0.00  get_Delta_V_SO
 2.71 3.22  0.10 12000000 0.00   0.00  get_mtrand_int32
 2.71 3.32  0.10      1000 0.10   0.10  get_mean_jump_length
 2.44 3.41  0.09 12000000 0.00   0.00  find_dxy_dyx_xoy
 2.17 3.49  0.08      1000 0.08   0.14  get_rho_L_pi
 1.63 3.55  0.06      1000 0.06   0.06  pmt_get_cycle_counts_and_lmax
 ...
 ...
 0.00 3.69  0.00         1 0.00   0.00  report_metro_stats
 0.00 3.69  0.00         1 0.00   0.00  set_default_mcmc_params
```

```
0.00 3.69  0.00         1 0.00   0.12  set_up_cycinfo_list
0.00 3.69  0.00         1 0.00   0.00  therm_ctl_free
0.00 3.69  0.00         1 0.00   0.00  therm_ctl_init
```

What is being seen here:

- The `name` column shows the names of all subroutines invoked during the execution of the program.
- The `% time` column shows the percent of total CPU time spent in a given subroutine. The output is sorted by decreasing order of CPU time.
- The self-seconds column shows the total wall time spent in the given subroutine; the cumulative-seconds column shows total wall time spent in the given subroutine and all those listed above it.
- The `calls` column counts the number of invocations of the subroutine. This helps the programmer distinguish between a routine which is time-consuming on each call, and a routine which is quick but perhaps overused.
- The `self us/call` and `total us/call` columns displays the mean and total number of microseconds spent in invocations to the subroutine.

9.21 Performance

Memory requirements with $N = L^3$ points and $M = 10^5$ sweeps, taken from the `VSZ` field of a Linux `ps aux` listing, are shown in table 9.1. A few hundred bytes are needed for each lattice point; this scales linearly with N. As well, for each random variable, the full time series over all M permutations in the MCMC sequence is saved. This scales linearly with M.

Figure 9.14 shows CPU times as a function of L, T, and α. For T near T_c and $L = 30, 40, 50$, in order to reduce variance in the critical-slowdown regime, 10^6 sweeps were performed. The sawtooth effect is due to the fact that simulations for a few initial values of T, e.g. $T = 6.74$, 6.76, were performed in a different computing environment than later simulations for more values of T including $T = 6.75$, 6.834, etc.

L	10	20	30	40	50	80	100	200
Megabytes	25	26	29	34	44	103	177	1243

TABLE 9.1. Memory requirements for `mcrcm` with $M = 10^5$ and varying L.

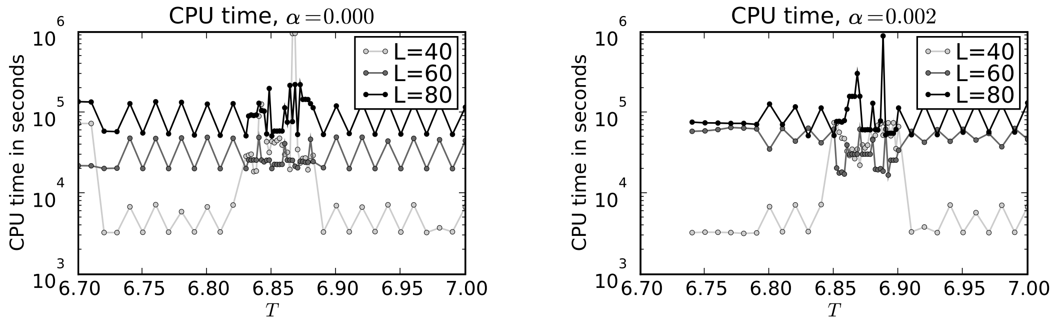


FIGURE 9.14. CPU time in seconds as function of L and T, for $\alpha = 0$, 0.002.

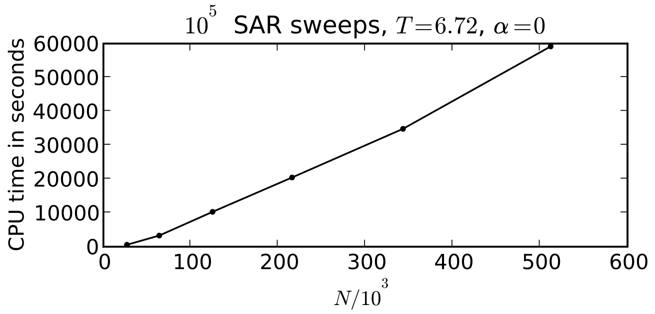


FIGURE 9.15. Scalability of the SAR algorithm. CPU times for 10^5 sweeps are shown as a function of $N = L^3$ for $L = 30$, 40, 50, 60, 70, and 80. SAR time is nearly linear in N.

As was discussed in section 9.5, most computations in our simulations are $O(N)$, with an $O(N^2)$ component that has a small constant of proportionality. See figure

9.15 which substantiates this claim. See also section 7.8 for a comparison of the SAR algorithm with the worm algorithm.

Chapter 10
Batching of MCMC runs

As described in chapter 9, the C program `mcrcm` is given a set of parameters as follows: L, T, interaction type and interaction parameter α, algorithm type (i.e. SO, SAR, band-update, or worm) and number of sweeps. Then a sequence of random permutations is generated, and sample means of random variables are computed over that sequence. The result is, for example, that the system energy H had sample mean 4939.7 with sample standard deviation 155.6.

One wishes, however, to find patterns in such data. In particular, as described in chapter 11: for various interactions α, for larger and larger L permitting extrapolation to the thermodynamic limit, one wishes to estimate the critical temperature T_c at which various order parameters (section 3.7) have a point of non-analyticity in the infinite-volume limit.

The C language was chosen for `mcrcm`, due to its efficiency for large-scale computations. (Data sets discussed in this dissertation have taken approximately 5.5 CPU-years; the choice of C has proved worthwhile, as an early Python implementation ran a factor of 40 times slower.) For the relatively lightweight task of scheduling parallel-processing tasks over multiple parameter values, extracting and collating sample statistics of random variables, and viewing the results — tasks for which the CPU time is measured in minutes, at most — it suffices to use easier-to-code scripting languages such as Bash or Python. Examples are shown in subsequent sections. The author has developed a flexible Python module, `taskutil.py`, for automating most of the tasks described in this chapter. However, such content is non-mathematical, of dubious value to an already lengthy mathematics dissertation. Equivalent, but briefer and simpler, scripting snippets in Bash will be shown instead.

10.1 Collecting data over multiple parameter values

Given a choice of parameter `nacc` (number of sweeps), and choices of parameters L, T, and α as described in chapter 2, a simple example of running `mcrcm` programs to compute sample statistics of random variables is as follows:

```
nacc=100000
datadir=sar_nacc_${nacc}
Ls="40 60 80"
Ts="6.40 6.50 6.60 6.70 6.80 6.90 7.00 7.10 7.20"
alphas="0.000 0.001 0.002"
mkdir -p $datadir
for L in $Ls; do
    for T in $Ts; do
        for alpha in $alphas; do
            file=$datadir/L_${L}_T_${T}_rell_alpha_${alpha}.txt
            mcrcm L=$L T=$T rell alpha0=$alpha nacc=$nacc > $file
        done
    done
done
```

For example, one of the loop iterations will execute the command

```
mcrcm L=40 T=6.70 rell alpha0=0.001 nacc=100000
```

and direct the output (as described in section 9.18) to the file

```
sar_nacc_100000/L_40_T_6.70_rell_alpha_0.001.txt,
```

the contents of which were shown in section 9.18. In total, 81 such files will be produced. Next one may wish to, say, select out only the values of `mean_fS` — the sample mean of f_S (sections 3.6 and 9.14) — from all 81 files.

The author's `taskutil.py` module implements this basic idea, with various elaborations. For example, one might divide the 81 tasks into 3 tasks for 27 processors each, with processors running in parallel. One might also wish to implement restart logic in case of unexpected downtime (e.g. due to a thunderstorm), wherein the script sees if a given file has already been completed rather than launching an already-completed task.

10.2 Extracting data over multiple parameter values

Given a list of data files as described in the previous section, one may wish to extract out a specified sample statistic for a specified random variable. A simple sample script which does this is

```
RV_name="mean_fS"
nacc=100000
datadir=sar_nacc_${nacc}
Ls="40 60 80"
Ts="6.40 6.50 6.60 6.70 6.80 6.90 7.00 7.10 7.20"
alphas="0.000 0.001 0.002"
for alpha in $alphas; do
    echo "alpha = $alpha"
    for L in $Ls; do
        for T in $Ts; do
            file=$datadir/L_${L}_T_${T}_rell_alpha_${alpha}.txt
            RV=‘grep $RV_name $file | awk ’{print $NF}’‘
            echo $L $T $RV
        done
    done
    echo ""
```

```
done

alpha = 0.000
40 6.40 1.0506240
40 6.50 0.8573933
...

alpha = 0.1
40 6.40 1.3394773
40 6.50 1.2505133
...
```

Such output may be analyzed or plotted as desired. Examples were shown throughout chapter 9.

10.3 Parallel processing

As discussed in section 10.1, when one is examining a set of (L, T, α) parameter values, one invokes an `mcrcm` executable for each particular triple of (L, T, α). If sufficiently many processors are available, there is no reason one cannot run, say, $(L = 40, T = 6.5, \alpha = 0.1)$ at the same time as $(L = 40, T = 6.5, \alpha = 0.2)$. In high-performance computing jargon, such parallelism is called *trivially parallel* or *embarrassingly parallel*. The author uses three such paradigms:

- On a single-processor laptop, `mcrcm` programs are launched in sequence, as in section 10.1.
- On the University of Arizona Department of Mathematics chivo cluster, which is four hosts with two CPUs each, one might (to be civil to other users) pick three hosts, running one parameter set on each, further prefixing with Unix `nice -10`.

- On the University of Arizona High Performance Computing Center's ICE cluster, which is a single host with over 1000 CPUs shared by dozens of on-campus researchers, one might request, say, 32 CPUs and divide the parameter set among those CPUs.

What has not been implemented is parallelization of `mcrcm` itself. As of this writing, there is no need to do so.

Chapter 11
Results

11.1 Finite-size scaling methodology

Finite-size scaling takes the form of a hypothesis, or rather a set of hypotheses, which is tested against the data. See [PV] for an excellent survey of techniques; see section 11.2 for a derivation of the formulas.

We have an infinite-volume random variable $S(T)$, e.g. any of the order parameters defined in section 3.7. The finite-volume quantity is $S_L(T)$. Define $t = (T - T_c)/T_c$. Examine, say, $0.99 < t < 1.01$. The first hypothesis is that the correlation length $\xi(T)$ follows a power law

$$\xi(T) \sim |t|^{-\nu}, \quad T \to T_c$$

For the infinite-volume quantity, we also expect a power-law behavior

$$S(T) \sim t^{\rho}, (-t)^{\rho}, \quad \text{or} \quad |t|^{\rho}, \quad \text{i.e.} \quad \xi^{-\rho/\nu}.$$

(The domain of validity is $t < 0$ or $t > 0$ depending on whether the order parameter S is left-sided or right-sided, respectively.) One moreover hypothesizes that for T near T_c, $S_L(T)$ and $S(T)$ are related by a *universal function* Q_S which depends on T only through the ratio L/ξ:

$$S_L(T) = L^{-\rho/\nu} Q_S(L^{1/\nu} t) \sim L^{-\rho/\nu} Q_S((L/\xi)^{1/\nu}). \tag{11.1.1}$$

The flow of data and respective uncertainties are as follows:

- Markov chain Monte Carlo simulations, with error bars determined using the method of integrated autocorrelation time (see [Berg] and our appendix B), yield $S_L(T, \alpha)$ data points. There are five order parameters S, six values of L

(30, 40, 50, 60, 70, 80), nine values of α, and a few dozen values of T for each α.

- For each S, L, and α, we use $S_L(T,\alpha)$ values for all available values of T and α to estimate [1] $\hat{\rho}_S(L)$. (Critical exponents are assumed to be independent of α for small α, or with weak enough dependence on α that that dependence is lost in the noise.) Error bars may be propagated from the MCMC simulations, or computed from regression uncertainties.

- Extrapolating $\hat{\rho}_S(L)$ in $L \to \infty$ results in the five estimated critical exponents $\hat{\rho}_S$. Uncertainties are computed from the regression analysis.

- Once the critical exponents have been estimated, we obtain $\hat{T}_{c,S}(\alpha)$ for each of the five order parameters S and for each α. Uncertainties are computed by visual inspection of the crossing plots discussed in section 11.5.

- Once the critical exponents and T_c have been estimated, one should be able to obtain plots of the universal function Q_S which is, up to sampling variability, independent of L, T, and α. This verifies that the finite-size-scaling hypothesis was the correct approach to use.

- The shift in reduced critical temperature is as in equation (2.3.1). Error bars are computed from regression uncertainties.

11.2 Derivation of finite-size scaling

A clear explanation is found in on-line notes of Claudia Brüns of the Argelander Institute for Astronomy of the University of Bonn. (We are unwilling to provide a bibliographic reference to an internet address, which may change in the future. Nonetheless, we feel compelled to acknowledge the author to whom this expalanation

[1]We use the statistics convention wherein $\hat{\rho}$ is an experimental estimator for the exact (but unknown) value ρ.

is due.) Those notes are are reproduced essentially verbatim in this section, except for change of notation.

For T away from T_c, $\xi \ll L$ and so $S_L(T)$ is not affected by lattice size. The finite-volume quantity $S_L(T)$ corresponds to the infinite-volume quantity $S_\infty(T)$, and we know $S_\infty(T) \propto \xi^{-\rho/\nu}$ as discussed in the previous section. For T near T_c, on the other hand, the correlation length ξ approaches the system size L so $S_L(T) \propto \xi^{-\rho/\nu} \approx L^{-\rho/\nu}$. As well, $S_L(T)$ differs significantly from $S_\infty(T)$ via the constant of proportionality. Combining these two regimes into a single expression gives

$$S_L(T) = \xi^{-\rho/\nu} R_S(L/\xi) \tag{11.2.1}$$

where

$$R_S(L/\xi) \propto \begin{cases} \text{constant}, & \xi \ll L \\ (L/\xi)^{-\rho/\nu}, & \xi \to L. \end{cases} \tag{11.2.2}$$

The infinite-volume correlation length $\xi = \xi_\infty(T)$ is unknown, so we define a scaling function Q_S to get rid of it:

$$Q_S(L/\xi) = (L/\xi)^\rho R_S((L/\xi)^\nu), \tag{11.2.3}$$

i.e.

$$R_S(L/\xi) = (L/\xi)^{-\rho/\nu} Q_S((L/\xi)^{1/\nu}). \tag{11.2.4}$$

The scaling function Q_S is finite for $\xi \to L$:

$$Q_S(L/\xi) = (L/\xi)^\rho R_S((L/\xi)^\nu) \tag{11.2.5}$$

$$Q_S(L/\xi) \propto (L/\xi)^\rho \cdot ((L/\xi)^\nu)^{-\rho/\nu} = 1. \tag{11.2.6}$$

Placing equation (11.2.4) into equation (11.2.1) yields

$$\begin{aligned} S_L(T) &= \xi^{-\rho/\nu}(L/\xi)^{-\rho/\nu} \cdot Q_S((L/\xi)^{1/\nu}) && (11.2.7) \\ &= L^{-\rho/\nu} Q_S(L^{1/\nu}\xi^{-1/\nu}) && (11.2.8) \\ &= L^{-\rho/\nu} Q_S(L^{1/\nu} t). && (11.2.9) \end{aligned}$$

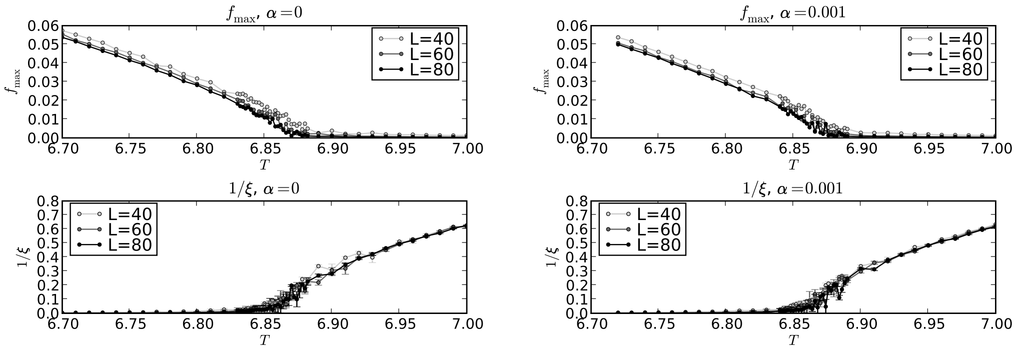


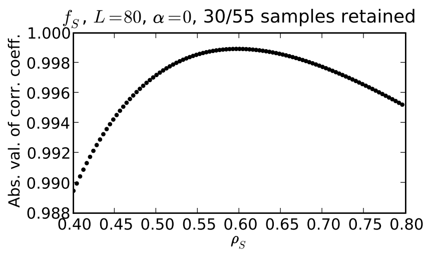
FIGURE 11.1. Order parameters $f_{\max}$ and $1/\xi$ for $L = 40, 60, 80$ and $\alpha = 0$ and 0.001. The remaining order parameters f_S, f_W, and f_I behave similarly to $f_{\max}$ but with not all with the same critical exponents.

11.3 Determination of L-dependent critical exponents

For each of order parameter S, interaction parameter α, and box length L, we examine all $S(L, T, \alpha)$ data for which $S > \varepsilon$, with ε taken from the plots to ensure that we examine the portions of the curves corresponding to non-zero order parameter in the infinite limit (see figure 11.1). For $1/\xi$, this means $T > T_c$; for the other four order parameters, this means $T < T_c$. From plots such as those in figure 11.1, we choose ε to be 0.1 for $1/\xi$, 0.01 for $f_{\max}$, 0.01 for f_I, 0.05 for f_S, and 0.01 for f_W. For each S, α, and L, we then apply linear regression to $S(L, T)^{1/\rho_S}$ for varying ρ_S. We find $\hat{\rho}_S(L)$ which optimizes the correlation coefficient [Young] of the linear regression. Results are shown in figure 11.2. Given $\hat{\rho}_S(L)$ along with its corresponding linear-regression parameters $\hat{m}$ and $\hat{b}$, we may plot a power-law fit to the simulational data. One such comparison plot is shown in figure 11.3.

11.4 Extrapolation of critical exponents for the infinite-volume limit

Next, for each S, given estimates $\hat{\rho}_S(L)$ for increasing values of L, we plot $\hat{\rho}_S(L)$ versus $1/L$. The vertical intercept of this plot estimates the infinite-volume exponent



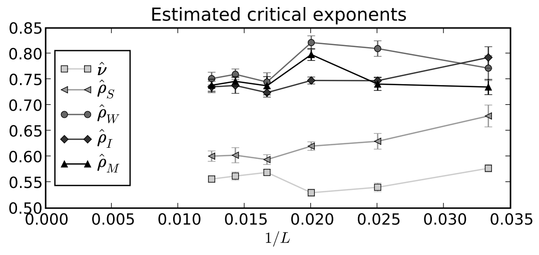


FIGURE 11.2. On the left: determination of critical exponent $\hat{\rho}_S(L, \alpha)$ for order parameter f_S, as the value which minimizes linear-regression error for $S_L(T, \alpha)^{1/\rho}$. Visually, one sees $\hat{\rho}_S(L = 80, \alpha = 0.0) \approx 0.59$. On the right: estimated critical exponents for $L = 30, 40, 50, 60, 70, 80$.

α	Mean	Std.err.	Count
0.000	0.6242981	0.0000897	78
0.0001	0.6243312	0.0001079	78
0.0002	0.6245691	0.0000921	72
0.0005	0.6245402	0.0001062	66
0.0008	0.6244347	0.0000856	72
0.001	0.6244779	0.0001020	60
0.002	0.6246345	0.0001154	42
0.003	0.6245906	0.0001559	48
0.004	0.6245966	0.0001964	42

TABLE 11.1. $f_{\max}/f_I$ as a function of α. An upward trend is visible, but it is not pronounced.

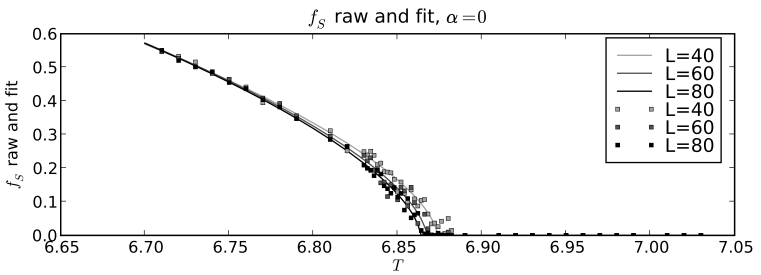


FIGURE 11.3. Power-law fit vs. raw simulational data for order parameter f_S, $\alpha = 0$.

$\hat{\nu}$	0.5559	$\pm$ 0.0037
$\hat{\rho}_S$	0.6201	$\pm$ 0.0065
$\hat{\rho}_W$	0.7750	$\pm$ 0.0073
$\hat{\rho}_I$	0.7451	$\pm$ 0.0052
$\hat{\rho}_M$	0.7486	$\pm$ 0.0059

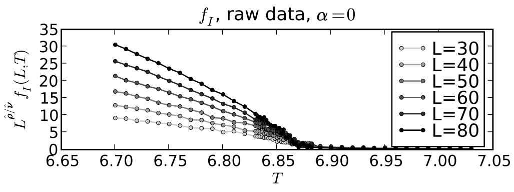
TABLE 11.2. Extrapolated estimates of the infinite-volume critical exponents, found from the vertical intercept of figure 11.2.

$\hat{\rho}_S(\alpha)$. (See figure 11.2.) Results are shown in table 11.2.

11.5 Determination of critical temperature

Given the above estimators of the critical exponents, the *crossing method* [PV] estimates $T_c(\alpha)$. Namely, we plot $L^{\hat{\rho}/\hat{\nu}} S_L(T)$ as a function of T. At $T = T_c$ we have $t = 0$ and $L^{\rho/\nu} S_L(T) = Q_S(0)$, regardless of L (equation (11.1.1)). Thus, these curves will cross (approximately, due to sampling variability) at $T = T_c$. If they do not, the finite-size-scaling hypothesis is not verified. (Note in particular that for order parameter $1/\xi$ whose critical exponent is ν, we apply the crossing method to $LS_L(T)$ as a function of T: thus, the $T_c(\alpha)$ estimate using $1/\xi$ is independent of $\hat{\nu}$.) See for example figure 11.4. Results are in table 11.3 and figure 11.6

Using order parameters f_S and f_W, which depend on winding phenomena, one does not see clear crossing behavior. We suggest that either this is related to the even-winding-number issue discussed in section 5.4, or f_S and f_W are not good order parameters for this model. We suspect the former; in every manner except this crossing issue, f_S and f_W behave as expected. (In the absence of clear crossing behavior for f_S and f_W, for the sake of discussion we nonetheless provide best visual estimates for $\hat{T}_c(\alpha)$ for f_S and f_W. These will not be used for further analysis toward our final result.)



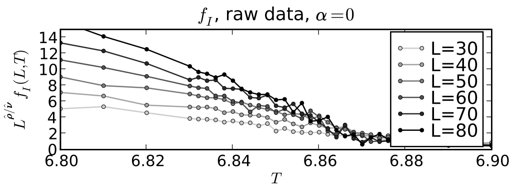


FIGURE 11.4. The crossing method to estimate $T_c(\alpha)$ for order parameter f_I, with $\hat\rho$ and $\hat\nu$ as above: $T_c(\alpha)$ corresponds to the horizontal coordinate of the intersection point of the plots. The upper-right-hand plot is a close-up of the upper-left-hand plot. Order parameters f_S and f_W, which depend on winding phenomena, do not exhibit clear crossing behavior.

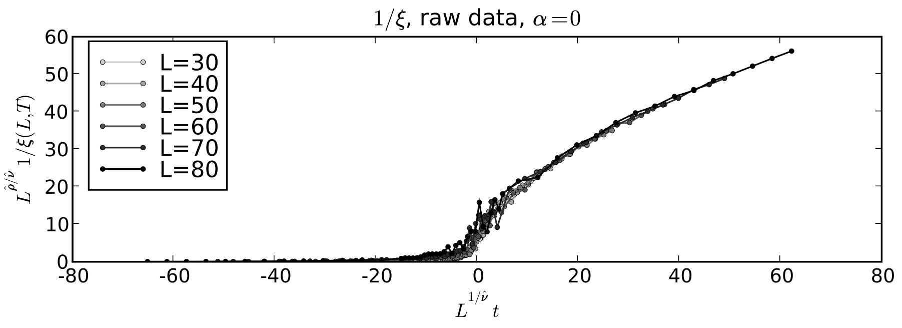


FIGURE 11.5. Collapse plot for order parameter $1/\xi$.

11.6 Verification of finite-size-scaling hypothesis

Now that we have estimated ρ_S, ν, and $T_c(\alpha)$ for each of the five order parameters S, we may plot $L^{\rho_S/\nu} S_L(T,\alpha)$ as a function of $L^{1/\nu}t$. This is a plot of the scaling function Q_S. If the hypothesis is correct, the curves for all L should coincide, or collapse, to within sampling error — which they do (e.g. figure 11.5).

11.7 Determination of the shift in critical temperature

As discussed in section 2.5, we are seeking a linear relationship between $\Delta T_c(\alpha)$ and α, with constant c. This can be visualized in figure 11.7, which is obtained from the $T_{c,S}(\alpha)$ data of figure 11.6 using equation (2.3.1). We start with all the $(\alpha, \Delta T_c(\alpha))$

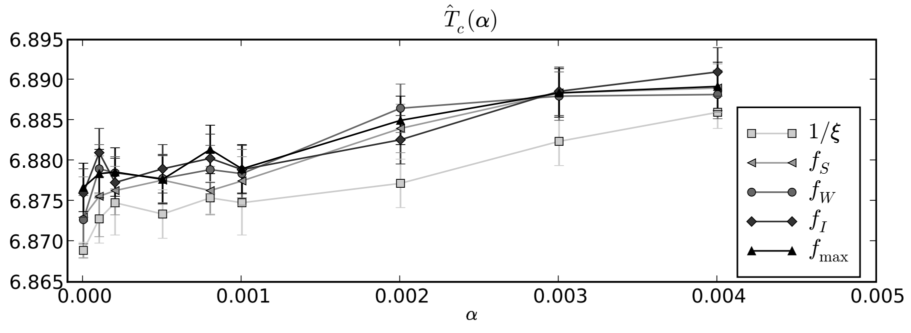


FIGURE 11.6. Critical temperature as function of α.

data points from section 11.5. We omit values obtained using f_S and f_W, due to the aforemention lack of crossing behavior. We also omit values obtained using $\alpha = 0.004$, since the critical-temperature plots of figure 11.6 suggests that this starts to exceed the domain of linear approximation. We perform a linear regression with error bars [Young] on the $(\alpha, \Delta T_c(\alpha))$ data points. We use a slope-only fit, rather than a slope-intercept fit, since $\Delta T_c(\alpha)$ has zero intercept by its very definition. We find

$$c = 0.618 \pm 0.086 \text{ (2 } \sigma \text{ error bar).}$$

Within experimental uncertainty, this result, for points on the lattice with Ewens cycle-weights, matches the c value of equation (2.5.3) for point positions varying on the continuum with decaying-cycle-weight interactions.

11.8 Constancy of the macroscopic-cycle quotient

As discussed in section 2.5, we hypothesize that the macroscopic-cycle quotient $f_{\max}/f_I$ in the infinite-volume limit is dependent on α but is constant in T where it is defined, i.e. for $T < T_c$ since $f_I = 0$ for $T > T_c$. This may be visualized by comparing figures such as 2.4: one sees that $f_{\max}$ and f_I appear to have the same critical exponent. Alternatively, one may plot the ratio $f_{\max}/f_I$ (figure 11.8). In the infinite-volume

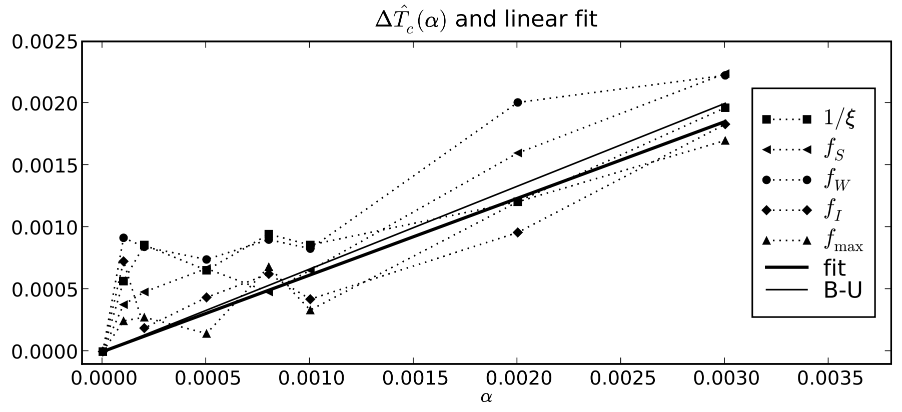


FIGURE 11.7. Shift in critical temperature, and linear fit, as function of α. Recall from equation (2.3.1) that $\Delta T_c(\alpha) = \frac{T_c(\alpha)-T_c(0)}{T_c(0)}$. Order parameters f_S and f_W were omitted from the fit, due to lack of crossing behavior; $\alpha = 0.004$ was omitted due to onset of curvature of $T_c(\alpha)$. The heavy solid line shows a linear fit with empirically determined constant of proportionality; the lighter solid line is the comparison value of Betz and Ueltschi (slope 2/3) for decaying cycle weights and continuum point positions.

α	Using $1/\xi$	Using f_S	Using f_W	Using f_I	Using $f_{\max}$
0.000	6.8689	6.8730	6.8727	6.8760	6.8767
0.0001	6.8728	6.8756	6.8790	6.8810	6.8784
0.0002	6.8748	6.8763	6.8785	6.8773	6.8786
0.0005	6.8734	6.8776	6.8778	6.8790	6.8777
0.0008	6.8754	6.8763	6.8789	6.8803	6.8814
0.001	6.8748	6.8775	6.8784	6.8789	6.8790
0.002	6.8772	6.8840	6.8865	6.8826	6.8850
0.003	6.8824	6.8884	6.8880	6.8886	6.8884
0.004	6.8860	6.8890	6.8882	6.8910	6.8892

Table 11.3. Critical temperature as a function of α. All values have error bars of approximately 0.003.

limit, f_I is zero for $T > T_c$ and so we are interested only in the values of the quotient for $T < T_c$. In that region, the quotient does indeed appear to be constant in T.

We test this constancy hypothesis as follows. The respective critical exponents are ρ_M and ρ_I. The estimators are $\hat{\rho}_M$ and $\hat{\rho}_I$, computed by averaging over several different values of L and α as described in section 11.4. Treating these estimators as normally distributed (as justified by the raw data), we obtain the standard deviations of the $\hat{\rho}_{M,I}(L,\alpha)$ samples, along with the standard deviations of the means $\hat{\rho}_{M,I}$:

$$\begin{aligned} \hat{\rho}_M &= 0.7482 & \hat{\rho}_I &= 0.7445 \\ s_M &= 0.0428 & s_I &= 0.0374 \\ n_M &= 50 & n_I &= 50 \\ s_M/\sqrt{n_M} &= 0.006059 & s_I/\sqrt{n_I} &= 0.005295. \end{aligned}$$

The difference $\hat{\rho}_M - \hat{\rho}_I$ is also normally distributed about the true mean $\rho_M - \rho_I$, but $\hat{\rho}_M$ and $\hat{\rho}_I$ are not independent since they are sample means of random variables computed from the same Markov chain Monte Carlo sequence of permutations. Thus we use

$$\mathrm{Var}(\hat{\rho}_M - \hat{\rho}_I) = \mathrm{Var}(\hat{\rho}_M) + \mathrm{Var}(\hat{\rho}_I) - 2\mathrm{Cov}(\hat{\rho}_M, \hat{\rho}_I).$$

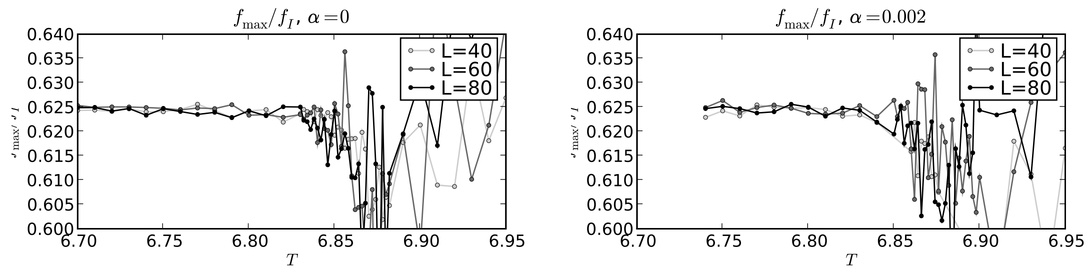


FIGURE 11.8. Macroscopic-cycle quotient $f_{\max}/f_I$ for $\alpha = 0, 0.002$.

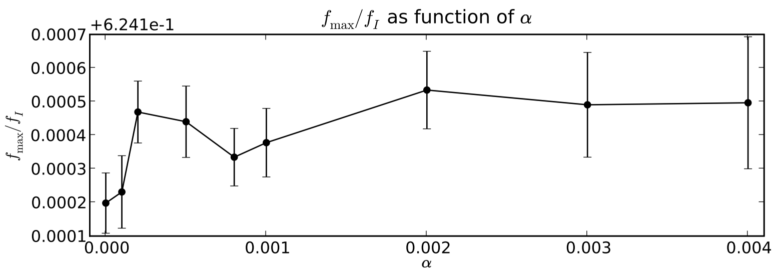


FIGURE 11.9. $f_{\max}/f_I$ as a function of α.

Computing the sample covariance of the $\hat{\rho}_M(L, \alpha)$ and $\hat{\rho}_I(L, \alpha)$ data series, we obtain the covariance and resulting standard error s_d of the difference

$$\mathrm{Cov}(\hat{\rho}_M, \hat{\rho}_I) = 0.0004 \qquad\qquad s_d/\sqrt{n} = 0.0070.$$

Normalizing, we find

$$\hat{\rho}_M - \hat{\rho}_I = 0.0037 \qquad\qquad \frac{\hat{\rho}_M - \hat{\rho}_I}{s_d/\sqrt{n}} = \frac{0.0037}{0.0070} = 0.5293.$$

We hypothesize $\rho_M - \rho_I = 0$; the estimated value $\hat{\rho}_M - \hat{\rho}_I$ lies comfortably within a standard deviation of this. We note, moreover, that the value of $f_{\max}/f_I$, while constant in T, trends upward with α (see table 11.1 and figure 11.9). This merits further investigation.

11.9 Conclusions

(1) For annealed point positions, equation (2.5.2) gives $T_c(0) \approx 6.625$. Our result $T_c(0) = 6.873 \pm 0.006$ (2 σ error bar) unambiguously shows that the lattice structure modifies the critical temperature, even in the non-interacting ($\alpha = 0$) case.

(2) As detailed in section 11.7, we find that the reduced shift in critical temperature as a function of interaction parameter α is

$$\Delta T_c(\alpha) \approx \frac{T_c(\alpha) - T_c(0)}{T_c(0)} = c\alpha$$

with

$$c = 0.618 \pm 0.086 \text{ (2 } \sigma \text{ error bar).}$$

This is compatible (section 2.5) with the related result of [BU08]. Even though the lattice structure changes the critical temperature (conclusion 1), the *shift* in critical temperature is unaffected.

(3) As described in section 2.4, Shepp and Lloyd [SL] find that $\mathbb{E}[\ell_{\max}]/N \approx 0.6243$ for uniform-random (non-spatial) permutations. For spatial permutations, we define a macroscopic-cycle quotient $\mathbb{E}[\ell_{\max}]/Nf_I$ which is the ratio of mean maximum cycle length as a fraction of the number of sites in long cycles. Our result (table 11.1 and figure 11.9) is compatible with that of Shepp and Lloyd for the non-interacting case, with an increase which appears to be linear as a function of interaction parameter α.

(4) We proved correctness for the pre-existing SO algorithm [GRU]; we invented the SAR, band-update, and worm algorithms, and proved them correct. The band-update algorithm suffers from a too-low acceptance probability; the worm algorithm suffers from a too-long stopping time; the SO algorithm prohibits (with very high probability) non-zero winding numbers. The SAR algorithm is our current best option, even though it only permits even winding numbers. Solving the deficiencies of the band-update or worm algorithms would be worth the effort.

(5) The order parameter $1/\xi$ is the most convenient to use for our problem: in the finite-size-scaling analysis, the crossings are independent of estimated critical exponent $\hat{\nu}$. The order parameters f_I and $f_{\max}$ are second-most convenient; one must estimate their critical exponents, but they are usable. The order parameters f_S and f_W, which depend on winding phenomena, do not pass the finite-size-scaling hypothesis. Either they are not good order parameters for the model of random spatial permutations, or they would benefit from a full-winding-number algorithm as discussed in the previous paragraph.

Chapter 12
Future work

Macroscopic-cycle quotient: Now that the α-dependence of the macroscopic-cycle quotient's constant upon α has been found empirically, one would next like to explain that dependence analytically.

Non-asymptotic algorithm correctness: The detailed-balance correctness proof of the swap-and-reverse algorithm shows that there is a non-zero transition probability between all pairs of permutations. However, those non-zero transition probabilities can be quite small. As the number of Metropolis steps goes to infinity, asymptotically all permutations can be reached; more interesting is the question of which permutations are actually reachable in reasonable simulation time. One answers this question empirically simply by running simulations; perhaps this suffices. For the system discussed in this paper, as well as for other systems studied using MCMC methods, it would be useful to have a non-asymptotic correctness theory.

Winding numbers of all parities: Ideally, one would have an algorithm to permit odd winding numbers, as discussed in section 5.4.

Bose-gas Hamiltonian: Sampling from the true Bose-gas distribution using the random-cycle model requires three changes. First, one needs to conduct simulations using the Bose-gas interaction (equation (2.1.1)) rather than the cycle-weight interaction (equation (2.1.3)). The interaction term V is a CPU-intensive Brownian-bridge computation [BU07]; unpublished work of Ueltschi and Betz shows that it may be approximated in the weak-interaction case by a simpler Riemann integral. Precomputed tables and interpolation may make use of this integral feasible.

Second, point positions must be allowed to vary on the continuum. This entails a second type of Metropolis step, in addition to that shown in section 5.1. Namely, one picks a point and moves it to a new position nearby, using the detailed-balance condition to choose the acceptance probability.

Third, software efficiency requires a hierarchical partitioning of Λ. The Metropolis step of section 5.1 relies on picking $\pi(\mathbf{y})$ near to $\pi(\mathbf{x})$. For points on the lattice, this is easy: each site has six nearest neighbors. For freely placed points, though, one must remember which sites are close to which. The most naive implementation involves computing the distances between all $N(N-1)/2$ pairs of points; the $O(N^2)$ computation time is overwhelming. Instead, the lattice may be partitioned into subcubes. Distances need to be computed only between each given point $\mathbf{x}$ and those in $\mathbf{x}$'s subcube and the 26 nearest-neighbor subcubes.

The second and third points simply require a software effort. Implementing them will be worthwhile only if the interaction terms can be simplified to the point that they are computationally feasible. This is a mathematical effort.

Appendix A
Bose-gas derivation of random permutations

In this sketch, we motivate the otherwise ab-initio construction of the model of random spatial permutations in chapter 2. More details may be found in [BU07, U07]. As above, we write $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$ for $\mathbf{x}_1, \dots, \mathbf{x}_N$ in a d-dimensional cube Λ of width L. The Hamiltonian for N pair-interacting particles is

$$\mathbf{H}(\mathbf{X}) = -\sum_{i=1}^{N} \nabla_i^2 + \sum_{1 \le i,j \le N} U(\mathbf{x}_i - \mathbf{x}_j). \tag{A.0.1}$$

The U considered here is either identically zero (for the non-interacting case), or a hard-core potential with radius a, i.e. $U(\mathbf{x}_i - \mathbf{x}_j)$ is infinite for $|\mathbf{x}_i - \mathbf{x}_j| \le a$ and zero for $|\mathbf{x}_i - \mathbf{x}_j| > a$. (This is an approximation to the true pair potential between helium atoms. See figure A.1 [Ceperley].) The hard-core radius a is also known as the scattering length.

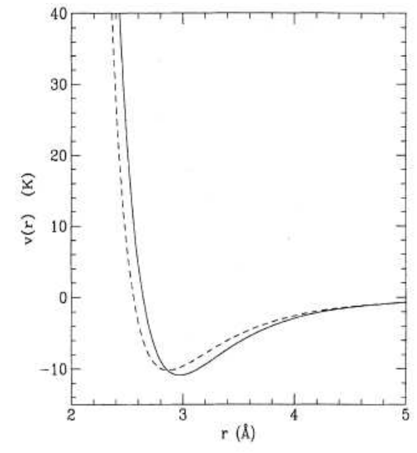


Figure A.1. Pair potential between helium atoms (Ceperley, 1995).

The partition function for N distinguishable particles[1] is $\mathrm{Tr}(e^{-\beta \mathbf{H}})$. Symmetrizing the partitition function, since our particles are bosons, the trace is

$$\mathrm{Tr}_{L^2_{\mathrm{sym}}}(e^{-\beta \mathbf{H}}) = \mathrm{Tr}_{L^2}\left(P_+ e^{-\beta \mathbf{H}}\right) = \mathrm{Tr}_{L^2}\left(e^{-\beta \mathbf{H}} P_+\right)$$

where

$$P_+ \, f(\mathbf{x}_1, \ldots, \mathbf{x}_N) := \frac{1}{N!} \sum_{\pi \in \mathcal{S}_n} M_\pi f(\mathbf{x}_1, \ldots, \mathbf{x}_N)$$

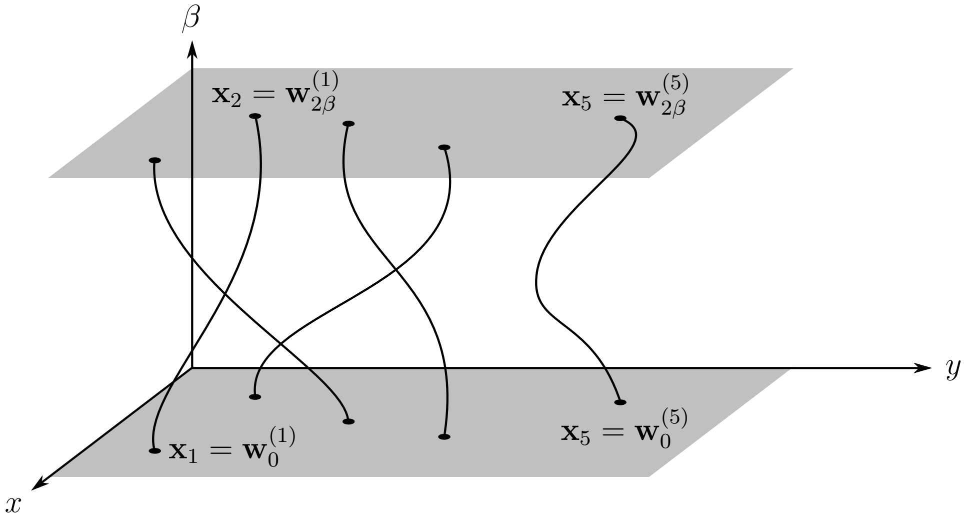
and

$$M_\pi(f\mathbf{x}_1, \ldots, \mathbf{x}_N) := f(\mathbf{x}_{\pi(1)}, \ldots, \mathbf{x}_{\pi(N)}).$$

That is,

$$\mathrm{Tr}_{L^2_{\mathrm{sym}}}(e^{-\beta \mathbf{H}}) = \frac{1}{N!} \sum_{\pi \in \mathcal{S}_N} \mathrm{Tr}_{L^2}\left(e^{-\beta \mathbf{H}} M_\pi\right).$$

(The operator $e^{-\beta \mathbf{H}}$ is bounded and compact, but this fact is not needed.)



β
$\mathbf{x}_2 = \mathbf{w}^{(1)}_{2\beta}$
$\mathbf{x}_5 = \mathbf{w}^{(5)}_{2\beta}$
y
$\mathbf{x}_1 = \mathbf{w}^{(1)}_0$
$\mathbf{x}_5 = \mathbf{w}^{(5)}_0$
x

FIGURE A.2. Feynman-Kac representation of a gas of 5 bosons. The horizontal plane represents the d spatial dimensions, and the vertical axis is the imaginary time dimension. The picture shows five particles and two cycles, of respective length 4 and 1.

[1]For a particle Hamiltonian, the $\beta = 1/T$ factor is in the expected place. This is in contrast to the permutation expression in chapter 2, where the β factor is, surprisingly, reciprocated. As discussed in [BU07, U07], the reciprocated β is correct for the permutation Hamiltonian.

The following steps are involved in developing a bosonic Feynman-Kac formula. The first three steps closely parallel the steps used to construct the familiar single-particle Feynman-Kac formula. (1) Interpret $e^{-\beta \mathbf{H}} M_\pi$ as an expectation over Brownian motions. (2) Write $e^{-\beta \mathbf{H}} M_\pi$ as an integral operator, and find the kernel. (3) Compute $\mathrm{Tr}(e^{-\beta \mathbf{H}} M_\pi)$ in terms of Brownian bridges. (4) Sum over $\pi \in \mathcal{S}_N$ to obtain $Z = \mathrm{Tr}_{L^2_{\mathrm{sym}}}(e^{-\beta \mathbf{H}})$. Importantly, one expresses Z as sum over permutations π of $e^{-H_P(\mathbf{X},\pi)}$, where this new H_P will be viewed as a Hamiltonian for permuations π. At this point, the permutation Hamiltonian is found inside $e^{-H_P(\mathbf{X},\pi)}$; one lacks an expression for its logarithm. (5) Decouple the non-interacting terms from the interacting terms in the permutation Hamiltonian, so that one may write $e^{-H_P^{(0)}(\mathbf{X},\pi)-H_P^{(1)}(\mathbf{X},\pi)}$. The bosonic Feynman-Kac formula now contains terms for two-jump interactions, three-jump interactions, and so on. (6) A cluster expansion allows one to drop all but two-jump interactions. The cluster expansion furthermore allows one to take the logarithm of $e^{-H_P(\mathbf{X},\pi)}$, with an explicit expression for $H_P(\mathbf{X},\pi)$. (7) One recognizes the random-cycle model from equation (2.1.1) of chapter 2, with an explicit two-jump interaction V. Specifically, given one permutation jump from $\mathbf{x}_i$ to $\mathbf{x}_{\pi(i)}$ and another permutation jump from $\mathbf{x}_j$ to $\mathbf{x}_{\pi(j)}$, the two-jump interaction $V(\mathbf{x}_i, \mathbf{x}_{\pi(i)}, \mathbf{x}_j, \mathbf{x}_{\pi(j)})$ involves the probability that two Brownian bridges, running in time 2β from $\mathbf{x}_i$ to $\mathbf{x}_{\pi(i)}$ and $\mathbf{x}_j$ to $\mathbf{x}_{\pi(j)}$, respectively, pass within distance $2a$ from one another.

Appendix B
Error bars, autocorrelation, and batched means

We make concrete various [CB, GS, Berg] ideas regarding autocorrelation of stationary Markov processes, with the particular goal of placing error bars on sample means. We focus on processes where the autocorrelation takes the form of a single exponential. We define a particular toy-model process, the *correlated-uniform Markov process*, which is exactly solvable. (This is in contrast to the typical Markov chain Monte Carlo process: in the MCMC field, one resorts to experimental methods only for systems which are *not* exactly solvable.) When a practitioner applies new methods to an MCMC process which is itself under examination, it can be difficult to identify computational problems which arise. Using this toy-model process, we elucidate strengths and shortcomings of autocorrelation and its estimators, clearly separating properties of the estimators themselves from the properties of the particular Markov process. The policies developed herein will be used to design and analyze MCMC experiments for the author's doctoral dissertation.

B.1 Problem statement

The following problem occurs throughout Markov chain Monte Carlo (MCMC) experiments. Let X_t be an identically distributed, but not necessarily independent, Markov process; let μ_X and σ_X^2 be the common mean and variance, respectively. (We will construct a specific process Y_t with the same properties. We reserve the notation X_t for a general process with these properties.) Given a time-series realization $X_0, \ldots, X_{N-1}$, the sole desired expressed in this paper is to to estimate μ_X, with an error bar on that estimate. The presence of correlations between the X_t's make this process more complicated than in the IID case.

The standard estimator for μ_X is the sample mean, $\overline{X}_N$. Given a time-series realization $X_0, \ldots, X_{N-1}$, we compute a single value of $\overline{X}_N$. Since the X_t's are random variables, $\overline{X}_N$ is itself a random variable. When we conduct M such experiments, we will get M different values of $\overline{X}_N$. (We will quantify below the dependence of the variance, or error bar, of $\overline{X}_N$, upon the autocorrelation of the process X_t.) Suppose for the sake of discussion that the autocorrelation is exponential: $\text{Corr}(X_i, X_j) = \eta^{|i-j|}$ for some $\eta \in [0, 1)$. Then $\eta = 0$ is the IID case, and higher η's correspond to more highly correlated processes. A few such realizations are shown in figure B.1.

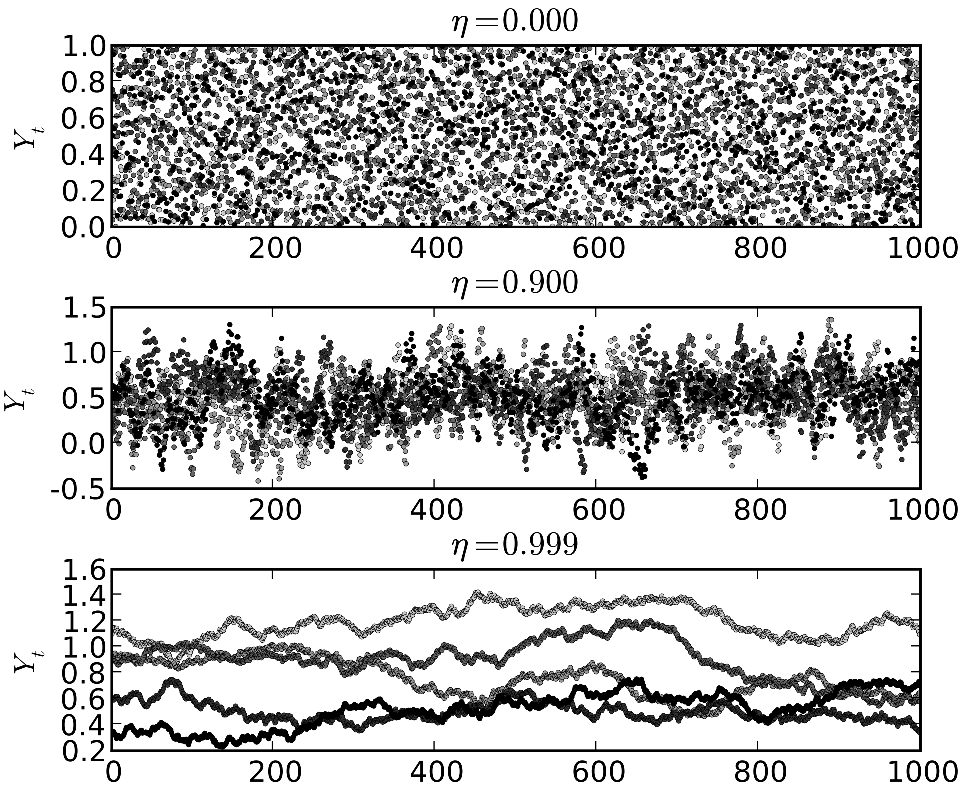


FIGURE B.1. Five realizations each of the correlated-uniform Markov process Y_t with $\eta = 0.0, 0.9, 0.999$.

For any process $W_1, \ldots, W_K$, write $m_K(W)$ for the sample mean and $s_K^2(W)$ for the sample variance, the unbiased estimator of $\text{Var}(W)$. Then:

- $m_N(X_t)$, which is $\overline{X}_N$, estimates μ_X. This is the *sample mean*, taken over N samples.

- $s_N^2(X_t)$ estimates σ_X^2. This is the *sample variance*, taken over N samples.

- $m_M(\overline{X}_N)$ estimates $\mu_{\overline{X}_N}$. This is also referred to as the sample mean; it is taken over MN samples.

- $s_M^2(\overline{X}_N)$ uses MN data points to estimate $\sigma^2_{\overline{X}_N}$, which is the *variance of the sample mean.*

- In the IID case, the true variance of the sample mean is $\sigma^2_{\overline{X}_N} = \sigma_X^2/N$; $t_N^2(X_t)$ $= s_N^2(X_t)/N$ is the *naive estimator* of the variance of the sample mean, using N data points. It is an unbiased estimator only in the IID case.

- $u_N^2(X_t)$ is the *corrected estimator* of $\sigma^2_{\overline{X}_N}$. The estimators $t_N^2(X_t)$ and $u_N^2(X_t)$ will be discussed graphically, numerically, and theoretically below. The estimated *integrated autocorrelation time* $\hat{\tau}_{\text{int}}$ will be used to compute $u_N^2(X_t)$ from $t_N^2(X_t)$.

- $\text{Var}(u_N^2(X_t))$ is the *error of the error bar.* It turns out that $u_N^2(X_t)$ is a rough estimator for $\text{Var}(\overline{X}_N)$, and $\text{Var}(u_N^2(X_t))$ increases with η. The very name "error of the error bar" sounds overwrought; yet, it is a necessary consideration in MCMC experiments, and must be thought through.

The processes Y_t of figure B.1, to be defined explicity in section B.4, have $\mu_Y = 1/2$ and $\sigma_Y^2 = 1/12$, regardless of the autocorrelation exponent η. (Note that $\sqrt{1/12} \approx 0.2887$.) We observe the following behavior from the aforementioned estimators. (See figures B.7 through B.10 starting on page 186, and table B.2 on page 189.)

- For all η, $\overline{Y}_N$ is unbiased for μ_Y. Its uncertainty widens visibly with autocorrelation exponent η. This uncertainty is the quantity of interest.

- Quantitatively, $s_M(\overline{Y}_N)$ gives a good idea of this increasing uncertainty. However, $s_M(\overline{Y}_N)$ requires M experiments, where M may be unacceptably large. If we were always willing to conduct such a large number of experiments, it would not be necessary to write this paper. We wish to estimate the variance of the sample mean *using only one experiment* $Y_0, \ldots, Y_{N-1}$. This is the rub.

- The corrected estimator $u_N^2(Y_t)$ corresponds roughly with $s_M(\overline{Y}_N)$, and moreover is computed from a single experiment $Y_0, \ldots, Y_{N-1}$. The roughness of the approximation of the error bar is acceptable: it is only an error bar.

To summarize, $s_M^2(\overline{X}_N)$ is a multi-experiment estimator for the variance of the sample mean; $u_N^2(X_t)$ is a single-experiment estimator. The former is of higher quality, but is more expensive to obtain; the latter carries its own uncertainty which worsens as the autocorrelation η increases.

Having motivated the problem, we now develop the notation and theory to make all of these ideas precise.

B.2 Autocovariance and autocorrelation

Definition B.2.1. A Markov process X_t, $t = 0, 1, 2, \ldots$, is *stationary* if the X_t's have a common mean $\mu_X = \mathbb{E}[X_t]$ and variance $\sigma_X^2 = \mathrm{Var}(X_t)$.

Definition B.2.2. Let X_t be a stationary Markov process with $\mathbb{E}[X_t] = \mu_X$ and $\mathrm{Var}(X_t) = \sigma_X^2$. The *autocovariance* and *autocorrelation* of X_t, respectively, are

$$\begin{aligned} C(t) &= \mathrm{Cov}(X_0, X_t) = \mathbb{E}[X_0 X_t] - \mathbb{E}[X_0]\mathbb{E}[X_t] = \mathbb{E}[X_0 X_t] - \mu_X^2 \\ c(t) &= \mathrm{Corr}(X_0, X_t) = \frac{\mathbb{E}[X_0 X_t] - \mathbb{E}[X_0]\mathbb{E}[X_t]}{\sigma_{X_0}\sigma_X} = \frac{\mathbb{E}[X_0 X_t] - \mu_X^2}{\sigma_X^2}. \end{aligned}$$

Remark B.2.3. In the literature, what we call the autocovariance is often referred to as autocorrelation. This incorrect and misleading terminology is, sadly, quite widespread.

Remark B.2.4. Recall that, as with all correlations, the autocorrelation takes values between -1 and 1.

B.3 The IID uniform process

Here we recall familiar [CB, GS] facts about random numbers U which are uniformly distributed on a closed interval $[a, b]$. These will be used as building blocks in section B.4. Writing the probability density function of U as $f_U(x)$, we have

$$\begin{aligned} f_U(x) &= \frac{1}{b-a} \cdot 1_{[a,b]}(x) & \mu_U &= \frac{1}{b-a}\int_a^b x\,dx = \frac{a+b}{2} \\ \mu_U^2 + \sigma_U^2 = \mathbb{E}[U^2] &= \frac{1}{b-a}\int_a^b x^2\,dx = \frac{a^2+ab+b^2}{3} & \sigma_U^2 &= \frac{(b-a)^2}{12}. \end{aligned}$$

Now consider an IID sequence $\{U_i\}$ of such random variables, indexed by the integers. We develop a particularly phrased formula which will simplify the calculations in section B.4. Note that if X_1, X_2 are IID with common mean μ_X and variance σ_X^2, then $\mathbb{E}[X_1^2] = \mu_X^2 + \sigma_X^2$ whereas $\mathbb{E}[X_1 X_2] = \mu_X^2$. For a sequence of IID X_i's, including our particular uniform U_i's, this means

$$\mathbb{E}[X_i X_j] = \mu_X^2 + \delta_{ij}\sigma_X^2. \tag{B.3.1}$$

B.4 The correlated-uniform Markov process

This paper addresses correlated Markov processes, focusing in particular on those with exponential autocorrelation. Here we construct a simple process for which the mean, variance, and autocorrelation are exactly solvable. In particular, the autocorrelation will be controlled by a parameter $\eta \in [0, 1]$, while the mean and variance will be the same as for IID $U(0, 1)$.

Definition B.4.1. Let U be uniformly distributed on $[a, b]$ as in the previous section, where $a < b$ are left variable for the moment. Let $0 \leq \eta \leq 1$ and $a < b$. The *correlated-uniform Markov process* Y_t is defined by $Y_0 \sim U(a, b)$, and for $t \geq 1$,

$$Y_t \quad = \quad \eta Y_{t-1} + (1-\eta)U_t \quad = \quad \eta^t U_0 + (1-\eta)\sum_{i=1}^{t} \eta^{t-i} U_i \tag{B.4.2}$$

where the first equality is a definition and the second equality follows by an easy induction argument.

Remark. Note that $\eta = 0$ is the IID case from the previous section; $\eta = 1$ would give a constant process with zero variance. The η parameter is the control knob with which we specify the autocorrelation of the process, as will be made precise in section B.5.

Definition B.4.3. Closely related to this is the *correlated-uniform stationary Markov process* (or asymptotic process)

$$Y_t = (1-\eta)\sum_{i=-\infty}^{t} \eta^{t-i} U_i. \tag{B.4.4}$$

In practice, we will run the original process for a number of time steps s until $\eta^s \approx 0$, such that the $\eta^s U_0$ term of equation (B.4.2) dies out, then consider the values of the process only from that time forward. In that regime, the process of definition B.4.3 is an approximation to that of definition B.4.1, but it is easier to manipulate algebraically.

We seek a, b such that the mean and variance of Y_t do not depend on η. The mean is immediate:

$$\mathbb{E}[Y_t] = (1-\eta)\sum_{i=-\infty}^{t} \eta^{t-i}\mathbb{E}[U_i] = \frac{a+b}{2}.$$

The variance $\mathrm{Var}(Y_t)$ is a special case of the covariance $\mathrm{Cov}(Y_t, Y_{t+k})$, which will be needed below. Equation (B.3.1) and expressions for geometric sums give us

$$\begin{aligned}
\mathbb{E}[Y_t Y_{t+k}] &= (1-\eta)^2 \eta^{2t+k} \sum_{i=-\infty}^{t} \eta^{-i} \sum_{j=-\infty}^{t+k} \eta^{-j}\ \mathbb{E}[U_i U_j] \\
&= (1-\eta)^2 \eta^{2t+k} \sum_{i=-\infty}^{t} \eta^{-i} \sum_{j=-\infty}^{t+k} \eta^{-j}\ (\mu_U^2 + \delta_{ij}\sigma_U^2) \\
&= \mu_U^2 (1-\eta)^2 \eta^{2t+k} \sum_{i=-\infty}^{t} \eta^{-i} \sum_{j=-\infty}^{t+k} \eta^{-j} + \sigma_U^2 (1-\eta)^2 \eta^{2t+k} \sum_{j=-\infty}^{t} \eta^{-2j} \\
&= \mu_U^2 + \sigma_U^2\ \eta^k \left(\frac{1-\eta}{1+\eta}\right).
\end{aligned}$$

Then

$$\mathrm{Var}(Y_t) = \sigma_U^2\ \left(\frac{1-\eta}{1+\eta}\right) = \frac{(b-a)^2}{12}\ \left(\frac{1-\eta}{1+\eta}\right).$$

Now we may solve for a and b such that μ_U and σ_U are the same as for IID $U(0,1)$, namely, 1/2 and 1/12 respectively. Solving the pair of equations

$$\frac{a+b}{2} = \frac{1}{2} \qquad \text{and} \qquad \frac{(b-a)^2}{12}\left(\frac{1-\eta}{1+\eta}\right) = \frac{1}{12},$$

we obtain

$$a = \frac{1}{2}\left(1 - \sqrt{\frac{1+\eta}{1-\eta}}\right) \qquad \text{and} \qquad b = \frac{1}{2}\left(1 + \sqrt{\frac{1+\eta}{1-\eta}}\right). \tag{B.4.5}$$

Note in particular that for $\eta = 0$, the IID case, we recover $a = 0, b = 1$ as expected. Figure B.2 shows some realizations for $\eta = 0.0, 0.5, 0.9$. For $\eta = 0.9$, correlations are clearly visible. Also note that there is a burn-in time required for the process to forget its initial state Y_0. In this figure, the asymptotic formula of definition B.4.3 appears valid for $t > 50$ or so, at which point $\eta^t = 0.9^{50} \approx 0.005 \approx 0$. This burn-in phenomenon is discussed in more detail in section B.8.

The following is pseudocode (technically, it is Python code, which is largely the same thing) for displaying N steps of Y_t, given the correlation-control parameter η and the number N_{therm} of burn-in iterates to be discarded:

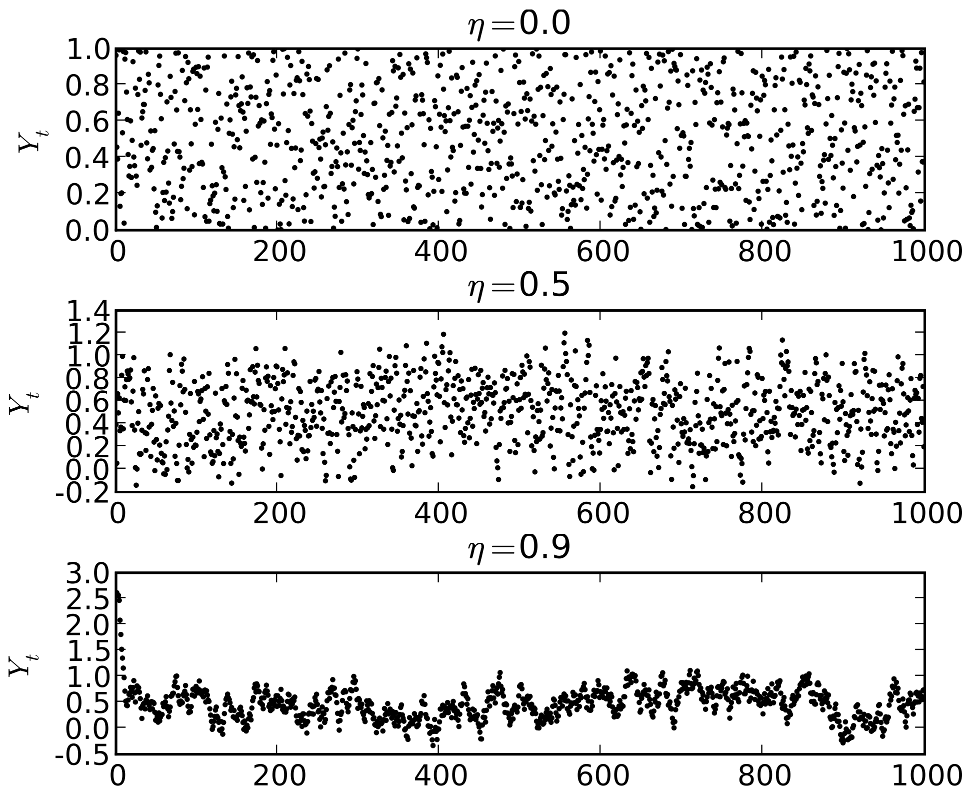


FIGURE B.2. Realizations of the correlated-uniform Markov process Y_t with $\eta = 0.0, 0.5, 0.9$. Burn-in iterates are included.

```
s = sqrt((1+eta)/(1-eta)); a = 0.5 * (1 - s); b = 0.5 * (1 + s)

Y = random.uniform(a, b) # Burn-in iterates
for k in range(0, Ntherm):
    U = random.uniform(a, b)
    Y = eta * Y + (1-eta) * U

for k in range(0, N):    # Iterates to be displayed
    U = random.uniform(a, b)
    Y = eta * Y + (1-eta) * U
    print Y
```

B.5 Statistics of the correlated-uniform Markov process

We now write all statistics of the correlated-uniform Markov process Y_t in terms of η. With a and b in terms of η (equation (B.4.5)), we have

$$\mu_U = \frac{a+b}{2} = \frac{1}{2}, \qquad \sigma_U^2 = \frac{(b-a)^2}{12} = \frac{1}{12}\left(\frac{1+\eta}{1-\eta}\right),$$
$$\mathbb{E}[Y_t] = \mu_Y = \frac{a+b}{2} = \frac{1}{2}, \qquad \operatorname{Var}(Y_t) = \sigma_Y^2 = \frac{(b-a)^2}{12}\left(\frac{1-\eta}{1+\eta}\right) = \frac{1}{12};$$

$$\begin{aligned}\mathbb{E}[Y_tY_{t+k}] &= \mu_U^2 + \sigma_U^2\eta^k\left(\frac{1-\eta}{1+\eta}\right) = \frac{1}{4} + \frac{\eta^k}{12},\\ \operatorname{Cov}(Y_t, Y_{t+k}) &= \mathbb{E}[Y_tY_{t+k}] - \mathbb{E}[Y_t]\mathbb{E}[Y_{t+k}] = \frac{\eta^k}{12},\\ \operatorname{Corr}(Y_t, Y_{t+k}) &= \frac{\operatorname{Cov}(Y_t, Y_{t+k})}{\sigma_Y\sigma_{Y_{t+k}}} = \eta^k.\end{aligned}$$

The remaining step needed to completely specify the correlated-uniform Markov process is to write down the PDF of Y_t. This could be done using convolutions, since Y_t is a weighted sum (weighted by powers of η) of IID uniform random variables. The algebra is messy, though, and an expression for the PDF is not needed in this work. It is sufficient to point out the following: (i) For $\eta = 0$, the density is uniform on $[0, 1]$. (ii) For η close to 1, which is the case of interest in this work, the density closely resembles a normal with mean 1/2 and variance 1/12. The support is compact, so the density cannot be Gaussian, but the support is wide enough to include substantial tail mass. See figure B.3 for empirical histograms.

B.6 The variance of the sample mean

When we use the data from an MCMC simulation to compute the sample mean of a random variable, the next order of business is to place an error bar on that sample mean.

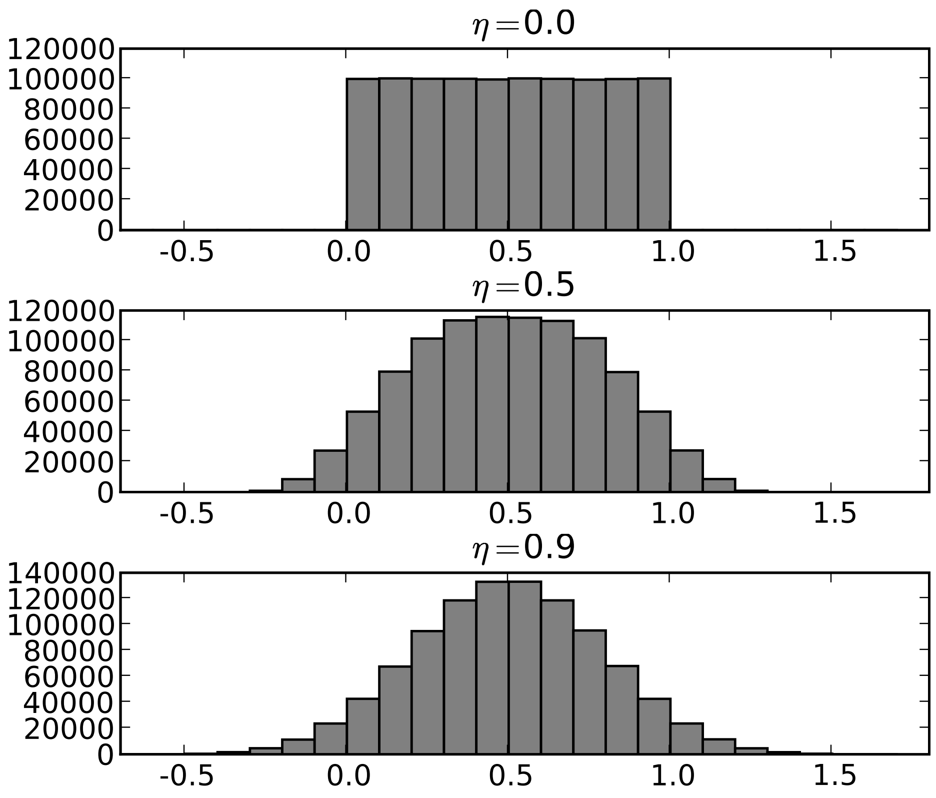


FIGURE B.3. Histograms of the correlated-uniform Markov process Y_t with $\eta = 0.0, 0.5, 0.9$: 10^6 iterates, bins of 0.1 from -0.7 to 1.7. Burn-in iterates have been discarded.

As before, let X_t be a stationary Markov process with common mean μ_X, variance σ_X, and autocorrelation $\mathrm{Corr}(X_t, X_{t+k}) = \eta^k$. Given $X_0, \ldots, X_{N-1}$, the sample mean $\overline{X}_N$ is an unbiased estimator of μ_X:

$$\overline{X}_N = \frac{1}{N} \sum_{i=0}^{N-1} X_i.$$

By linearity of expectation, $\mathbb{E}[\overline{X}_N] = \mu_X$. To find the variance of $\overline{X}_N$, we first need $\mathbb{E}[\overline{X}_N{}^2]$. This is

$$\mathbb{E}[\overline{X}_N{}^2] = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \mathbb{E}[X_i X_j].$$

Since

$$\mathrm{Corr}(X_i, X_j) = \eta^{|i-j|} = \frac{\mathbb{E}[X_i, X_j] - \mu_X^2}{\sigma_X^2},$$

we have

$$\mathbb{E}[X_i X_j] = \mu_X^2 + \sigma_X^2 \eta^{|i-j|}. \tag{B.6.1}$$

Then

$$\begin{aligned}\mathbb{E}[\overline{X}_N{}^2] &= \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(\mu_X^2 + \sigma_X^2\eta^{|i-j|}) = \mu_X^2 + \frac{\sigma_X^2}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\eta^{|i-j|} \\ &= \mu_X^2 + \frac{\sigma_X^2}{N^2}\left[\sum_{i=0}^{N-1} 1 + \sum_{i=0}^{N-2}\eta^{-i}\sum_{j=i+1}^{N-1}\eta^j + \sum_{i=1}^{N-1}\eta^i\sum_{j=0}^{i-1}\eta^{-j}\right].\end{aligned}$$

Applying geometric-sum formulas and several lines of algebra, we get

$$\mathbb{E}[\overline{X}_N{}^2] = \mu_X^2 + \frac{\sigma_X^2}{N} + \frac{2\sigma_X^2\eta}{N^2(1-\eta)}\left[(N-1) - \left(\frac{\eta - \eta^N}{1-\eta}\right)\right].$$

With $N \approx N-1$ we have

$$\mathbb{E}[\overline{X}_N{}^2] \approx \mu_X^2 + \frac{\sigma_X^2}{N}\left(\frac{1+\eta}{1-\eta}\right) - \frac{2\sigma_X^2\eta^2}{N^2(1-\eta)^2}(1-\eta^{N-1}).$$

With $\eta^N \approx 0$ and a bit more algebra we have

$$\mathbb{E}[\overline{X}_N{}^2] \approx \mu_X^2 + \frac{\sigma_X^2}{N}\left(\frac{1+\eta}{1-\eta}\right) \qquad \text{and} \qquad \mathrm{Var}(\overline{X}_N) \approx \frac{\sigma_X^2}{N}\left(\frac{1+\eta}{1-\eta}\right). \tag{B.6.2}$$

Recall that for the IID case ($\eta = 0$) we have $\mathrm{Var}(\overline{X}_N) = \sigma_X^2/N$. This expression recovers that; furthermore, correlations enlarge the error bar on the sample mean.

B.7 Estimates of autocorrelation

Throughout this section, let X_t be a stationary Markov process with $\mathbb{E}[X_t] = \mu_X$, $\mathrm{Var}(X_t) = \sigma_X^2$, and $\mathrm{Corr}(X_t, X_{t+k}) = \eta^k$. (Without loss of generality, take $k \geq 0$.) The simple correlated-uniform Markov process of section B.4 is one example of this;

moreover, an MCMC process on a finite state space may take this form. (As described in [Berg], η is related to the second dominant eigenvalue of the transition matrix of the Markov process. If the third dominant eigenvalue is comparable with the second, then the autocorrelation will not take the simple exponential form described here.)

Remark B.7.1. In the literature, one more often sees $\mathrm{Corr}(X_0, X_t) = \exp(-t/\tau_{\mathrm{exp}})$. Then τ_{exp} and η are put into one-to-one correspondence by

$$\tau_{\mathrm{exp}} = -1/\log \eta \qquad\qquad \text{and} \qquad\qquad \eta = \exp(-1/\tau_{\mathrm{exp}}).$$

For the correlated-uniform process, the autocorrelation is already known; for a general MCMC process, one wishes to estimate η (or τ_{exp}) from realization data. Recall that

$$\mathrm{Corr}(X_t, X_{t+k}) = \frac{\mathbb{E}[X_t X_{t+k}] - \mathbb{E}[X_t]\mathbb{E}[X_{t+k}]}{\sigma_{X_t}\sigma_{X_{t+k}}} = \frac{\mathbb{E}[X_0 X_k] - \mu_X^2}{\sigma_X^2} \tag{B.7.2}$$

where the second equality holds by the stationarity of the process, and that we always have

$$-1 \le \mathrm{Corr}(X_t, X_{t+k}) \le 1. \tag{B.7.3}$$

(This holds for the correlation of any pair of random variables.) Also recall that

$$\sigma_X^2 = \mathbb{E}[X_t^2] - \mathbb{E}[X_t]^2. \tag{B.7.4}$$

Recall as well [CB] that, for M realizations $X_t^{(0)}, \ldots, X_t^{(M-1)}$ of X_t, the unbiased estimator for the variance of X_t is

$$s_{X_t}^2 = \frac{1}{M-1}\left[\sum_{i=0}^{M-1}(X_t^{(i)})^2 - \frac{1}{M}\left(\sum_{i=0}^{M-1} X_t^{(i)}\right)^2\right]. \tag{B.7.5}$$

Definition B.7.6. Fix t and k. The *multi-realization estimator* of the autocorrelation $\mathrm{Corr}(X_t, X_{t+k})$, requiring M realizations $X_t^{(0)}, \ldots X_t^{(M-1)}$ of the process, is a straightforward combination of equations (B.7.2), (B.7.4), and (B.7.5). Namely,

$$\hat{c}_m(t,k) = \frac{\frac{1}{M}\sum_{i=0}^{M-1}\left(X_t^{(i)}X_{t+k}^{(i)}\right) - \frac{1}{M^2}\left(\sum_{i=0}^{M-1}X_t^{(i)}\right)\left(\sum_{j=0}^{M-1}X_{t+k}^{(j)}\right)}{\frac{1}{M-1}\left[\sum_{i=0}^{M-1}(X_t^{(i)})^2 - \frac{\left(\sum_{i=0}^{M-1}X_t^{(i)}\right)^2}{M}\right]^{1/2}\left[\sum_{j=0}^{M-1}(X_{t+k}^{(j)})^2 - \frac{\left(\sum_{j=0}^{M-1}X_{t+k}^{(j)}\right)^2}{M}\right]^{1/2}}.$$

Remark. Since the process is stationary, one may be tempted to reuse the X_t variance estimator for X_{t+k} — after all, they estimate the same quantity σ_X^2. In practice, however, doing so tends to produce autocorrelation estimates which fall (quite far) outside the range $[-1,1]$, violating inequality B.7.3. That is, the second equality in equation (B.7.2) holds theoretically but not at the estimator level. This same remark holds for the sliding-window estimator, to be defined next.

The difficulty with the multi-realization estimator is that realizations X_t can be expensive to compute. Rather than running M processes from $t=0$ up to some N, which takes $O(MN)$ process-generation time, perhaps we can (carefully) use the stationarity of the process, estimating the autocorrelation using only a single realization. This will take only $O(N)$ process-generation time.

Definition B.7.7. Given a single realization $X_0,\ldots,X_{N-1}$, take k from $0,1,2,\ldots,N-2$. The *sliding-window estimator* of the autocorrelation $\mathrm{Corr}(X_0,X_k)$, is

$$\hat{c}(k) = \frac{\frac{1}{N-k}\sum_{i=0}^{N-k-1}(X_iX_{i+k}) - \frac{1}{(N-k)^2}\left(\sum_{i=0}^{N-k-1}X_i\right)\left(\sum_{j=0}^{N-k-1}X_{j+k}\right)}{\left(\frac{1}{N-k-1}\right)\left[\sum_{i=0}^{N-k-1}X_i^2 - \frac{\left(\sum_{i=0}^{N-k-1}X_i\right)^2}{N-k}\right]^{1/2}\left[\sum_{j=0}^{N-k-1}X_{j+k}^2 - \frac{\left(\sum_{j=0}^{N-k-1}X_{j+k}\right)^2}{N-k}\right]^{1/2}}. \tag{B.7.8}$$

This formula is perhaps intimidating, but is made quite simple with the aid of the example below, wherein $N=10$ and $k-2$. Namely:

- We consider all pairs separated by k time steps: X_0X_k, X_1X_{k+1}, $\ldots$, $X_{N-k-1}X_{N-1}$. There are $N-k$ such pairs.

- The first elements in each pair form a window from X_0 to X_{N-k-1}.

- The second elements in each pair form a window from X_k to X_{N-1}.
- We estimate the mean and variance of X_0 by the sample mean and sample variance over the first window.
- We estimate the mean and variance of X_k by the sample mean and sample variance over the second window.
- We estimate the cross-moment $\mathbb{E}[X_t X_{t+k}]$ by the sample mean over pair products.

Example B.7.9. $\triangleright$ There are $N = 10$ samples, X_0 through X_9:

X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9

Picking $k = 2$, there are two windows of length $N - k = 8$:

X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7		
		X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9

Equation (B.7.8) has five distinct sums: the sum of X_0 through X_7, the sum of squares of X_0 through X_7, the sum of X_2 through X_9, the sum of squares of X_2 through X_9, and the cross sum $X_0X_2 + \ldots + X_7X_9$. $\triangleleft$

Remark B.7.10. One would hope that $\hat{c}(t)$ is an unbiased estimator of $c(t)$. Finding its expectation using the definition is intimidating: we have a ratio of products of sums of correlated random variables. Taking an experimental approach instead, making multiple plots of the form of figure B.4, one finds that $\hat{c}(t)$ does in fact fractionally underestimate $c(t)$. This affects the estimated integrated autocorrelation time, as discussed in remark B.9.2.

This estimator has the benefit of making use of all the data in a single realization. Its drawback is that, for larger k, the sample size $N - k$ is small. Thus, the error in the estimator increases for larger k.

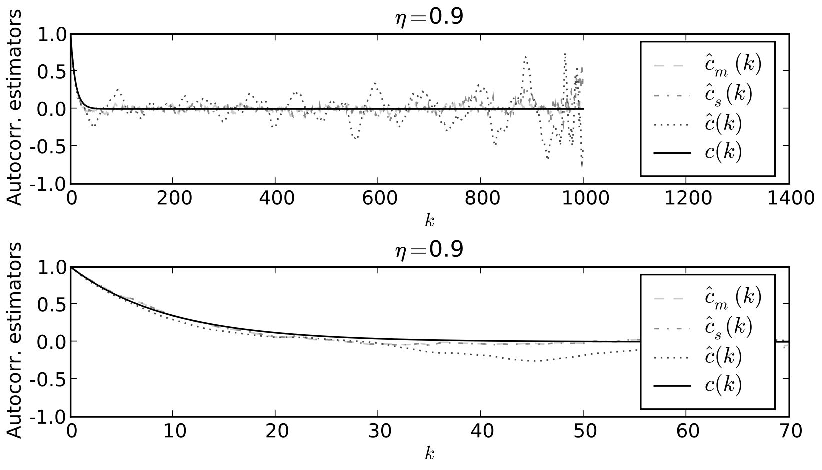


Figure B.4. Autocorrelation and estimators thereof for Y_t with $\eta = 0.9$. Burn-in iterates have been discarded. The second plot zooms in on the first 50 samples of the first plot.

Figure B.4 compares estimators against the true value $c(k) = \mathrm{Corr}(X_0, X_k) = \eta^k$ for $\eta = 0.9$. Here, $N = 1000$ time steps have been used; $M = 1000$ realizations for the multi-realization estimator $\hat{c}_m(k)$. Note that the decreasing sample size, $N - k$, of the sliding-window estimator $\hat{c}(k)$ increases the error of this estimator. For this reason, $\hat{c}_s(k)$ is also plotted. This is the same as $\hat{c}_m(k)$, but with $M = N - k$. The first plot shows the autocorrelation estimators for $k = 0$ to 998; the second zooms in on the first 50 values of k.

Remark B.7.11. We observe the following:

- Comparing the full-length and short-length multi-realization estimators $\hat{c}_m(k)$ vs. $\hat{c}_s(k)$ shows that decreasing sample size does have an effect for larger k. Nonetheless, the sliding-window estimator $\hat{c}(k)$ shows markedly wilder behavior for larger k, which cannot be accounted for by small-sample-size effects alone.
- For all three estimators, errors are small when k is small, which is when the true autocorrelation $c(k) = \eta^k$ is non-negligible.

- Thus, one should examine estimators of the autocorrelation only for values of k until the estimators approach zero. Values past that point are neither accurate nor needed.

B.8 Integrated autocorrelation time

Following [Berg], we develop the notion of integrated autocorrelation time as follows. We reconsider the variance of the sample mean (see section B.6) from a different point of view. Again, X_t is a stationary Markov process with common mean μ_X and common variance σ_X^2. We have

$$\begin{aligned}
\operatorname{Var}(\overline{X}_N) = \mathbb{E}[(\overline{X}_N - \mu_X)^2] &= \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\mathbb{E}[(X_i-\mu_X)(X_j-\mu_X)] \\
&= \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\mathbb{E}[X_iX_j - \mu_X X_i - \mu_X X_j + \mu_X^2] \\
&= \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left(\mathbb{E}[X_iX_j] - \mu_X^2\right) = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\operatorname{Cov}(X_i,X_j) \\
&= \frac{1}{N^2}\left[\sum_{i=0}^{N-1}\operatorname{Var}(X_i) + 2\sum_{t=1}^{N-1}(N-t)\operatorname{Cov}(X_0,X_t)\right] \\
&= \frac{\sigma_X^2}{N} + 2\sigma_X^2\sum_{t=1}^{N-1}(N-t)\operatorname{Corr}(X_0,X_t) \\
&= \frac{\sigma_X^2}{N}\left[1 + 2\sum_{t=1}^{N-1}\left(1-\frac{t}{N}\right)\operatorname{Corr}(X_0,X_t)\right] \approx \frac{\sigma_X^2}{N}\left[1+2\sum_{t=1}^{\infty}\operatorname{Corr}(X_0,X_t)\right].
\end{aligned}$$

If X_t is IID then we recover the familiar $\operatorname{Var}(\overline{X}_N) = \sigma_X^2/N$; otherwise we have

$$\operatorname{Var}(\overline{X}_N) = \frac{\sigma_X^2}{N}\,\tau_{\text{int}} \tag{B.8.1}$$

where τ_{int} is the last bracketed expression above. Note as well that if $\operatorname{Corr}(X_0, X_k) = \eta^k$, then

$$\tau_{\text{int}} = 1 + 2\sum_{t=1}^{\infty}\eta^t = 1 + \frac{2\eta}{1-\eta} = \frac{1+\eta}{1-\eta} \tag{B.8.2}$$

which is what we would have expected by comparing equations (B.6.2) and (B.8.1). As a consequence, when $c(t) = \eta^t$ we have

$$\tau_{\text{int}} = \frac{1+\eta}{1-\eta} \qquad \text{and} \qquad \eta = \frac{\tau_{\text{int}} - 1}{\tau_{\text{int}} + 1}. \tag{B.8.3}$$

Some values are shown for reference in table B.1.

η	0	0.1	0.2	0.5	0.6	0.9	0.990	0.999
$(1+\eta)/(1-\eta)$	1	1.222	1.500	3.000	4.000	19	199	1999

Table B.1. η vs. $(1+\eta)/(1-\eta)$.

Remark B.8.4. If the process is IID, i.e. $\eta = 0$, then $c(0) = 1$, $c(t) = 0$ for all $t \geq 1$, and $\tau_{\text{int}} = 1$.

Definition B.8.5. Recall that $s_N^2(X_t)$ (equation (B.7.5)) estimates σ_X^2. Using equation (B.8.1), the *naive estimator* and *corrected estimator* of $\text{Var}(\overline{X}_N)$ are

$$t_N^2(X_t) = \frac{s_N^2(X_t)}{N} \qquad \text{and} \qquad u_N^2(X_t) = \frac{s_N^2(X_t)}{N}\,\hat{\tau}_{\text{int}}, \tag{B.8.6}$$

as long as we have an estimator $\hat{\tau}_{\text{int}}$ of τ_{int}.

B.9 Estimation of the integrated autocorrelation time

Recall from remark B.7.11 that $\hat{c}(t)$ is a rather wild estimator of $c(t)$ at high t. Since

$$\hat{\tau}_{\text{int}} = 1 + 2\sum_{t=1}^{\infty} \hat{c}(t)$$

is nothing more than a sum of $c(t)$, we can expect it to be ill-behaved as well.

Definition B.9.1. The *running-sum estimator* of τ_{int} is

$$\hat{\tau}_{\text{int}}(t) = 1 + 2\sum_{k=1}^{t} \hat{c}(k).$$

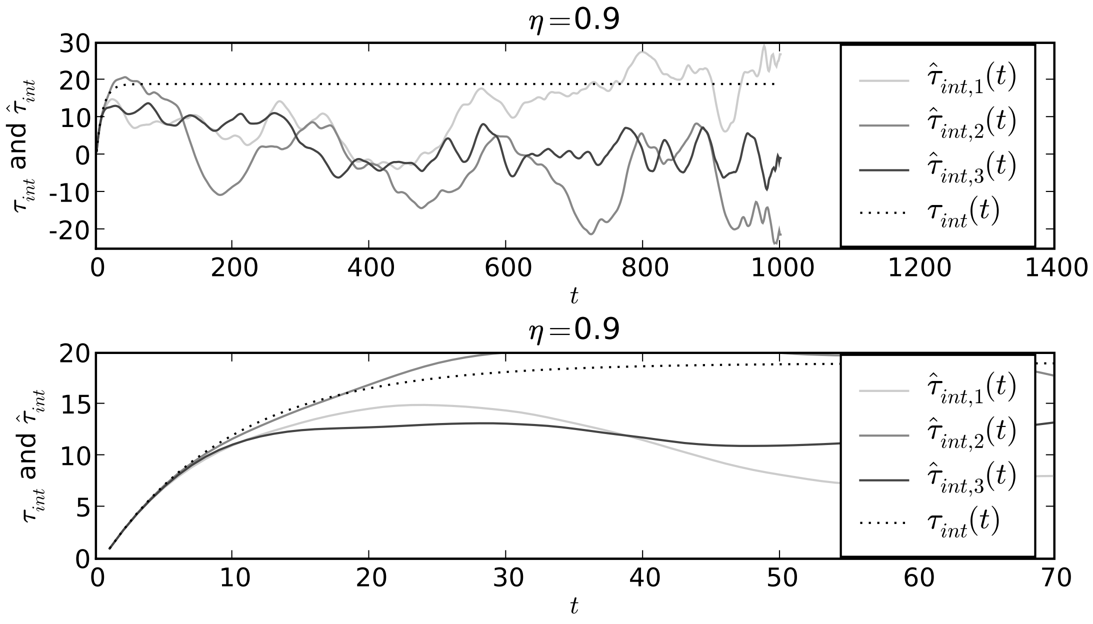


FIGURE B.5. Estimated and exact integrated autocorrelation times for Y_t with $\eta = 0.9$, using three realizations similar to the one in figure B.4. Burn-in iterates have been discarded. The second plot zooms in on the first 50 samples of the first plot. The flat-spot estimator $\hat{\tau}_{\text{int}}$ of τ_{int} is found by reading off the vertical coordinate of the first turning point of each solid-line plot; the true τ_{int} is the horizontal asymptote of the dotted-line plot. Two of the three turning points yield a $\hat{\tau}_{\text{int}}$ which is less than the true τ_{int}. This is the general case: we find that $\hat{\tau}_{\text{int}}$ underestimates more often than it overestimates. See also figure B.8 on page 186.

The idea is to accumulate the reliable low-t values of $\hat{c}(t)$ until the sum becomes approximately constant at some s, then stop and declare $\hat{\tau}_{\text{int}}$ to be $\hat{\tau}_{\text{int}}(s)$. This is the *flat-spot estimator* or *turning-point estimator* for τ_{int}. See figure B.5 for illustration, where s is approximately 24 for the blue realization and 29 for the red. From the plots, we estimate $\tau_{\text{int}} \approx 15$; using equation (B.8.3), we estimate $\eta = (15-1)/(15+1) = 0.875$. This is reasonable since the data were obtained with $\eta = 0.9$, for which the true τ_{int} is 19 by equation (B.8.2).

It is clear from the figure that estimators $\hat{\tau}_{\text{int}}$ can vary noticeably from one realization to the next. Our estimator for the variance of the sample mean, i.e. the

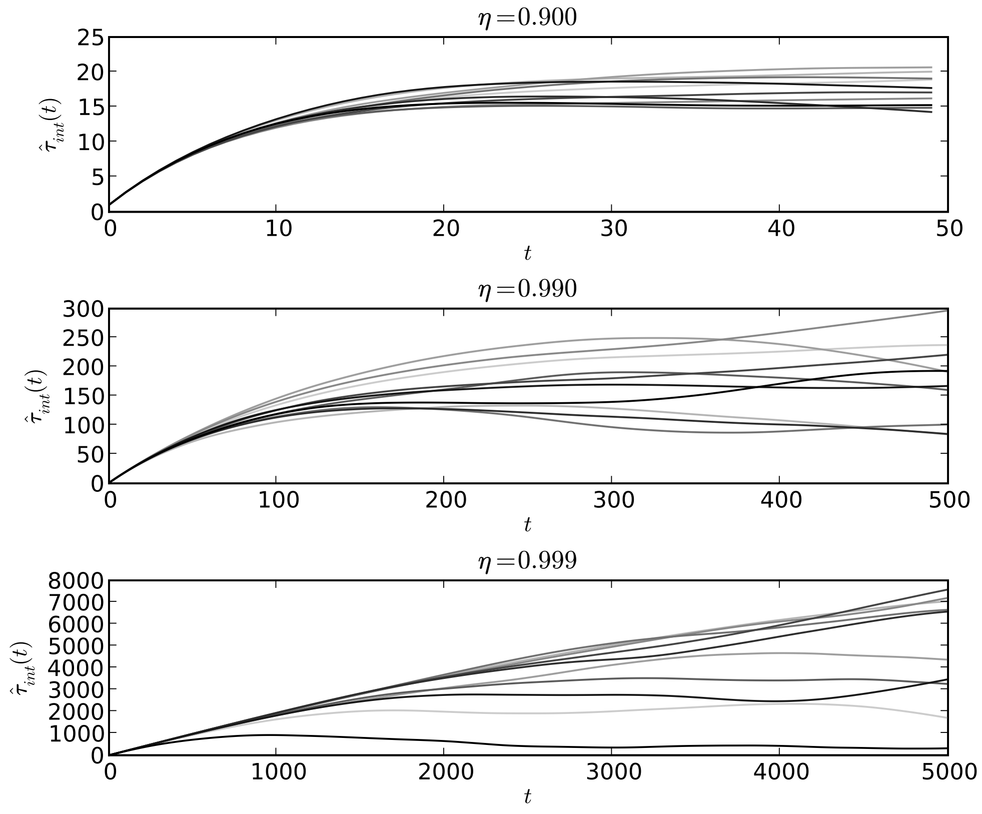


FIGURE B.6. Estimated integrated autocorrelation times for Y_t with $\eta = 0.9, 0.99, 0.999$, using ten realizations each. N is 100,000; burn-in iterates have been discarded. Recall that true τ_{int} values are 19, 199, and 1999, respectively. The variation in the vertical coordinate of the first flat spot in each plot, which increases with η, gives rise to the error of the error bar on the sample mean.

error bar on the sample mean, is $u_N^2(X_t)$ (equation (B.8.6)). Since $\hat{\tau}_{\text{int}}$ is a factor in $u_N^2(X_t)$, variations in $\hat{\tau}_{\text{int}}$ will result in error of the error bar. Figure B.6 shows that variations in $\hat{\tau}_{\text{int}}$ increase with η.

At present I know of no solution to this problem other than the running of multiple experiments — larger M, using the notation of section B.1. As long as τ_{int} is estimated based on a single experimental result $X_0, \ldots, X_{N-1}$, one must be aware that the error bars on the sample mean are crude.

Remark B.9.2. As was noted in remark B.7.10, $\hat{c}(t)$ underestimates $c(t)$. Since $\hat{\tau}_{\text{int}}$

is formed from a sum of $\hat{c}(t)$'s, $\hat{\tau}_{\text{int}}$ is also a fractional underestimator of τ_{int}, as will be seen in section B.10.

B.10 Estimation of the variance of the sample mean

Given the flat-spot estimator $\hat{\tau}_{\text{int}}$ of τ_{int} from section B.9 and the naive estimator of the variance of the sample mean from equation (B.8.6), we may now compute the corrected estimator of the the variance of the sample mean:

$$u_N^2(X_t) = \frac{s_N^2(X_t)}{N}\,\hat{\tau}_{\text{int}}.$$

We use the correlated-uniform Markov process to illustrate, since for this process all quantities have known theoretical values. As in section B.1, we display standard deviations in our plots and tables, rather than variances: the units of measurement of the former match those of the mean, and they correspond visually to variations in the data.

- The mean and variance of Y_t are $\mu_Y = 1/2$ and $\sigma_Y^2 = 1/12$; $\sigma_Y \approx 0.289$. Using $\eta = 0.0, 0.9, 0.999$, the true τ_{int} is $1, 19, 1999$, respectively. We conduct $M = 100$ experiments of collecting and analyzing $N = 10000$ time-series samples $Y_0, \ldots, Y_{N-1}$.

- The true mean is shown in row 1 of table B.2. Estimators $\overline{Y}_N$ are shown in figure B.7. The average of these over all M experiments is shown in row 2 of table B.2.

- The true naive variance of the sample mean is σ_Y^2/N, with true naive standard deviation of the sample mean $\sigma_Y/\sqrt{N} \approx 0.00289$. The true corrected variance of the sample mean is $\sigma^2_{\overline{Y}_N} = \tau_{\text{int}}\,\sigma_Y^2/N = 1/120000, 19/120000, 1999/120000$. The true standard deviations of the sample means are then $\sigma_{\overline{Y}_N} \approx 0.0028868$, 0.0125831, 0.1290672. These are shown in row 3 of table B.2.

- The multi-experiment estimator $s_M(\overline{Y}_N)$ of $\sigma_{\overline{Y}_N}$ is the sample standard deviation of the M values $\overline{X}_N^{(0)}, \ldots, \overline{X}_N^{(M-1)}$. These estimators are shown in row 4 of table B.2. As expected, the multi-experiment estimator is a good one.

- Next we turn to single-experiment estimators of the variance of the sample mean. The estimated naive standard deviation of the sample mean is $t_N(Y_t) = s_N(Y_t)/\sqrt{N}$. These are not plotted for each experiment; their average over all M experiments is shown in row 5 of table B.2. Note that they match the true variance of the sample mean only in the IID ($\eta = 0$) case.

- True values of τ_{int} for each η are shown in row 8 of the table. The flat-spot estimators $\hat{\tau}_{\text{int}}$ for all $M = 100$ experiments are shown in figure B.8. Their average and sample standard deviation over all M experiments are shown in rows 9 and 10. As discussed in remark B.9.2, we see that $\hat{\tau}_{\text{int}}$ fractionally underestimates τ_{int}.

- Using the $\hat{\tau}_{\text{int}}$ values, the corrected estimators $u_N(Y_t) = t_N(Y_t)\sqrt{\hat{\tau}_{\text{int}}}$ are shown, for all $M = 100$ experiments, in figure B.9. Their averages over all M experiments are shown in row 6 of table B.2. (Again, the corresponding true values are in row 2 of the table.) The fractional underestimation of $\hat{\tau}_{\text{int}}$ carries over to $u_N(Y_t)$. One trades the quality of the estimator for the feasibility of its computation.

- Standard deviations over M experiments of $u_N(Y_t)$ are shown in row 7 of the table. Figure B.10 shows, for $\eta = 0.999$, the $M = 100$ values of $\overline{Y}_N$ along with their respective $u_N(Y_t)$'s. These show the error of the error bar.

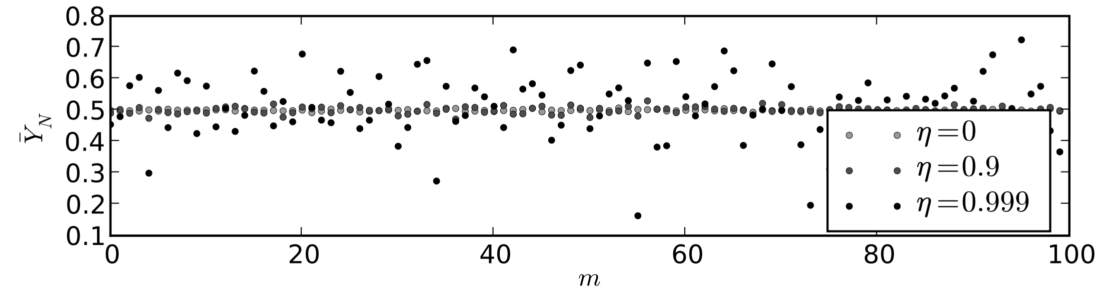


FIGURE B.7. $\overline{Y}_N$ over $M = 100$ experiments, where the true value is $\mu_X = 0.5$. Variance of $\overline{Y}_N$ increases with autocorrelation factor η.

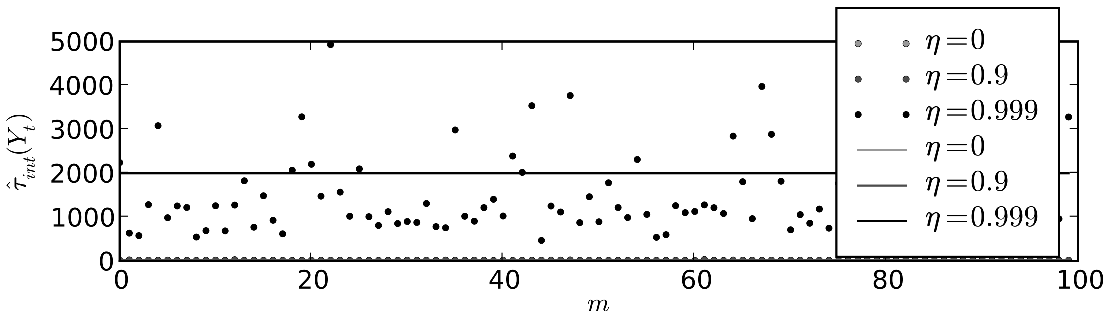


FIGURE B.8. $\hat{\tau}_{\text{int}}(Y_t)$ over $M = 100$ experiments, along with true values. Note that $\hat{\tau}_{\text{int}}(Y_t)$ fractionally underestimates the true $\tau_{\text{int}}(Y_t)$.

B.11 Integrated and exponential autocorrelation times

In remark B.7.1 of section B.7, we noted that if $\text{Corr}(X_0, X_t) = \eta^t$ for $\eta \in [0, 1)$, then we may define an exponential autocorrelation time via

$$\tau_{\text{exp}} = -1/\log \eta$$

such that $\text{Corr}(X_0, X_t) = \exp(-t/\tau_{\text{exp}})$. Yet section B.8 gave us something similar: the integrated autocorrelation time τ_{int}. In particular, if $\text{Corr}(X_0, X_t) = \eta^t$, then we had

$$\tau_{\text{int}} = \frac{1+\eta}{1-\eta}.$$

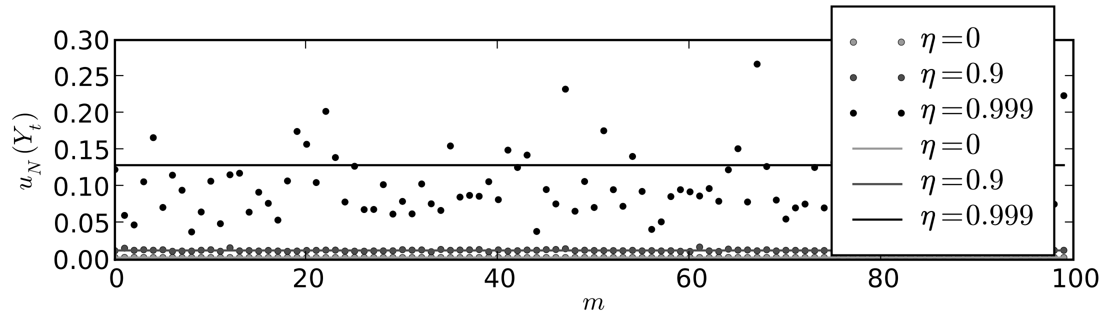


FIGURE B.9. $u_N(Y_t)$ over $M = 100$ experiments, along with true values. Note that $u_N(Y_t)$ fractionally underestimates the true standard deviation of the sample mean, $\sigma_{\overline{Y}_N} = \sigma_Y/\sqrt{N}$.

Figure B.11 compares these two.

B.12 Batched means

Introductory statistics tends to deal with the analysis of IID samples. Yet, realization sequences from an MCMC experiment tend to be highly correlated. The sample mean estimates the true mean, since expectation is linear. But when one wishes to place an accurate error bar on the sample mean, correlations must be taken into account.

One approach (see for example [Berg], who calls this process *binning*) is to subdivide $X_0, \ldots, X_{N-1}$ into $K = N/B$ batches of size B. The K sample means over batches may be treated as IID samples. The independence of the K samples means that the variance of their sample mean will be reduced, but reducing the sample size from N to K will increase the variance. We will show that these two effects cancel: binning N samples down to K samples does not change the variance of the sample mean. (As shown in [Berg], batched means have their uses: they may be used to construct a method to estimate τ_{int}, as an alternative to the method of section B.9.)

Definition B.12.1. Given $X_0, \ldots, X_{N-1}$ with common mean μ_X and variance σ_X^2, let B divide N and $K = N/B$. Then B is the *batch size* and K is the *number of*

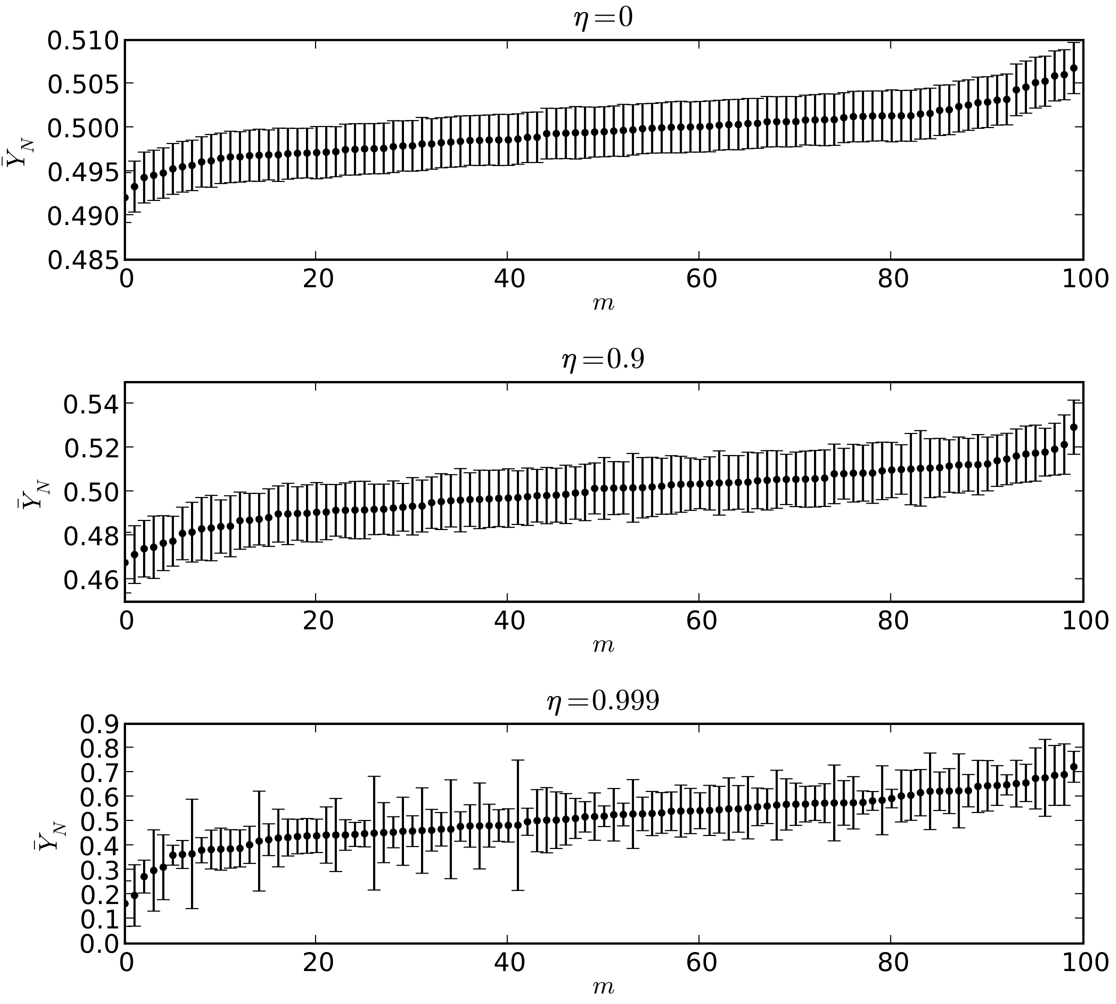


FIGURE B.10. $\overline{Y}_N$ with single-sigma error bars, $\eta = 0, 0.9, 0.999$, $M = 100$ experiments, sorted by increasing $\overline{Y}_N$. The magnitude and the variation of the error bars both increase with η.

batches. For $k = 0, \ldots, K-1$, the kth batch consists of $X_{kB}, \ldots, X_{(k+1)B-1}$. The sample mean of the kth batch is

$$A_k = \frac{1}{B} \sum_{i=0}^{B-1} X_{kB+i}.$$

Description	$\boldsymbol{\eta}$	**0**	**0.9**	**0.999**
1. True mean	$\mu_{\overline{Y}_N}$	0.50000	0.50000	0.50000
2. Sample mean	$m_M(\overline{Y}_N)$	0.49948	0.49952	0.51100
	$m_M(\overline{Y}_N)$	0.49987	0.50231	0.47341
	$m_M(\overline{Y}_N)$	0.49991	0.49895	0.47958
3. True standard deviation of sample mean	$\sigma_{\overline{Y}_N}$	0.00288	0.01258	0.12906
4. Multi-experiment estimator of $\sigma_{\overline{Y}_N}$	$s_M(\overline{Y}_N)$	0.00274	0.01166	0.10342
	$s_M(\overline{Y}_N)$	0.00274	0.01167	0.12303
	$s_M(\overline{Y}_N)$	0.00298	0.00986	0.11929
5. Averaged single-experiment naive estimators of $\sigma_{\overline{Y}_N}$	$m_M(t_N(Y_t))$	0.00288	0.00287	0.00263
	$m_M(t_N(Y_t))$	0.00288	0.00288	0.00260
	$m_M(t_N(Y_t))$	0.00288	0.00287	0.00250
6. Averaged single-experiment corrected estimators of $\sigma_{\overline{Y}_N}$	$m_M(u_N(Y_t))$	0.00288	0.01279	0.09957
	$m_M(u_N(Y_t))$	0.00289	0.01280	0.10037
	$m_M(u_N(Y_t))$	0.00289	0.01276	0.08947
7. Sample standard deviation of corrected estimators of $\sigma_{\overline{Y}_N}$	$s_M(u_N(Y_t))$	0.00004	0.00105	0.04402
	$s_M(u_N(Y_t))$	0.00005	0.00108	0.04665
	$s_M(u_N(Y_t))$	0.00006	0.00118	0.03851
8. True integrated autocorrelation time	τ_{int}	1	19	1999
9. Averages of estimated integrated autocorrelation time	$m_M(\hat{\tau}_{\text{int}})$	0.999	19.854	1442.627
	$m_M(\hat{\tau}_{\text{int}})$	1.002	19.763	1500.137
	$m_M(\hat{\tau}_{\text{int}})$	1.008	19.857	1279.173
10. Standard deviation across M experiments of $\hat{\tau}_{\text{int}}$	$s_M(\hat{\tau}_{\text{int}})$	0.028	3.162	865.992
	$s_M(\hat{\tau}_{\text{int}})$	0.031	3.092	1045.842
	$s_M(\hat{\tau}_{\text{int}})$	0.039	3.628	758.903

Table B.2. Statistics for three trials of $M = 100$ experiments on $N = 10000$ samples of Y_t: $\eta = 0.0, 0.9, 0.999$.

We now consider the sequence $A_0, \ldots, A_{K-1}$. We define the *batched mean* to be

$$\overline{X}_{N,B} = \frac{1}{K}\sum_{k=0}^{K-1} A_k.$$

By linearity of expectation, we immediately have $\mathbb{E}[\overline{X}_{N,B}] = \mu_X$. We next inquire about the variance of the batched mean, then compare that to the variance of the (non-batched) sample mean.

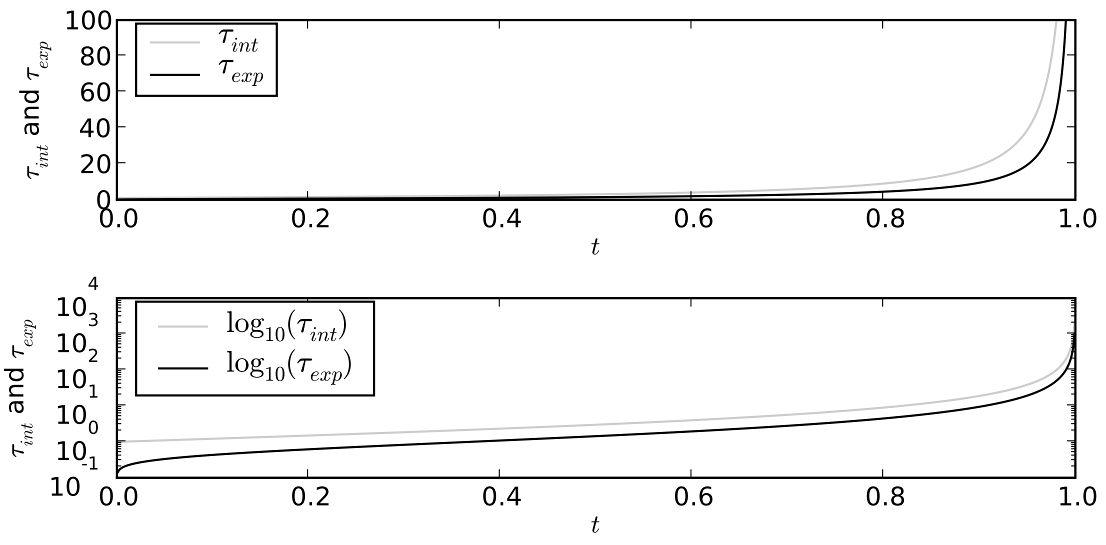


FIGURE B.11. Integrated and exponential autocorrelation times as a function of η.

B.13 Variance and covariance of batches

To compute $\mathrm{Var}(A_k)$ and $\mathrm{Corr}(A_0, A_k)$, we first need $\mathbb{E}[A_k A_\ell]$ for $k = \ell$ and $k \neq \ell$. In the $k = \ell$ case, the computation is the same as in section B.6, with B playing the role of N. We have

$$\mathbb{E}[A_k^2] \approx \mu_X^2 + \frac{\sigma_X^2}{B}\left(\frac{1+\eta}{1-\eta}\right) \qquad \text{and} \qquad \mathrm{Var}(A_k) \approx \frac{\sigma_X^2}{B}\left(\frac{1+\eta}{1-\eta}\right).$$

For $k \neq \ell$, without loss of generality assume $k < \ell$. Using equation (B.6.1), we have

$$\begin{aligned}\mathbb{E}[A_k A_\ell] &= \frac{1}{B^2}\sum_{i=0}^{B-1}\sum_{j=0}^{B-1}\mathbb{E}[X_{kB+i}X_{\ell B+j}] &&= \frac{1}{B^2}\sum_{i=0}^{B-1}\sum_{j=0}^{B-1}(\mu_X^2 + \sigma_X^2\eta^{\ell B+j-kB-i}) \\ &= \mu_X^2 + \frac{\sigma_X^2\eta^{(\ell-k)B}}{B^2}\sum_{i=0}^{B-1}\eta^{-i}\sum_{j=0}^{B-1}\eta^j &&= \mu_X^2 + \frac{\sigma_X^2\eta^{(\ell-k)B}}{B^2}\left(\frac{1-\eta^B}{1-\eta}\right)^2.\end{aligned}$$

If the batch size is chosen so that η^B is negligible, then

$$\mathbb{E}[A_k A_\ell] = \mu_X^2.$$

Now we have (for $\eta^B \approx 0$)

$$\operatorname{Var}(A_k) \approx \frac{\sigma_X^2}{B}\left(\frac{1+\eta}{1-\eta}\right) \quad \text{and} \quad \operatorname{Corr}(A_k, A_\ell) = \frac{\mathbb{E}[A_k A_\ell] - \mu_X^2}{\sigma_{A_0}^2} = \delta_{k,\ell}. \tag{B.13.1}$$

This justifies the hope that batches can be constructed to form an IID sequence.

B.14 Variance of the batched mean

We now find out what effect batching has on the variance of the sample mean: batching produces an IID sequence, which will reduce the variance (equation (B.8.1)), yet it reduces the sample size from N down to $K = N/B$, which by central-limit reasoning should increase the variance.

For the non-batched mean, we have the random variables $X_0, \ldots, X_{N-1}$; parameters are mean μ_X, variance σ_X^2, autocorrelation η^k, and (from equation (B.8.3)) integrated autocorrelation time $\tau_{\text{int}} = (1+\eta)/(1-\eta)$. Equation (B.6.2) gives

$$\operatorname{Var}(\overline{X}_N) = \frac{\sigma_X^2}{N}\left(\frac{1+\eta}{1-\eta}\right). \tag{B.14.1}$$

For the batched mean, we batch $X_0, \ldots, X_{N-1}$ into K IID batches of size B. We have the random variables $A_0, \ldots, A_{K-1}$, with mean μ_X, variance $(\sigma_X^2/B)(1+\eta)/(1-\eta)$ (equation (B.13.1)), autocorrelation $c(k) = \delta_{0,k}$ (since A_k is IID) and integrated autocorrelation time $\tau_{\text{int}} = 1$ (remark B.8.4). Then

$$\operatorname{Var}(\overline{X}_{N,B}) = \frac{\sigma_X^2}{KB}\left(\frac{1+\eta}{1-\eta}\right) = \frac{\sigma_X^2}{N}\left(\frac{1+\eta}{1-\eta}\right).$$

Thus, to first order in η and N, as long as B is large enough that η^B is negligibly small, we do not expect batching to change the variance of the sample mean.

Table B.3 shows some sample results of these calculations for the correlated-uniform Markov process Y_t. There are $M = 100$ experiments of $N = 10000$ samples. Each experiment was analyzed as-is ($B = 1$), as well as with batch size $B = 64$, 512, and 4096. (Recall from table B.1 that $\eta = 0, 0.9, 0.999$ correspond to $\tau_{\text{int}} = 1, 19, 1999$, respectively.) Now the process being analyzed is $A_0, \ldots, A_{K-1}$ where $K = N/B$. We note the following:

- For $B = 1$ (the original time series), the estimator u^2 of the variance of the sample mean employed was the corrected estimator of equation (B.8.6), while for $B = 64, 512, 4096$, the A_k's were treated as if they were IID. That is, for $B > 1$ we set $u^2 = t^2$.

- Batch size does not, of course, affect the sample mean (column 3 of the table). Likewise, it does not affect the multi-experiment estimator of the variance of the sample mean (column 4).

- The true variance of the sample mean, $\sigma_{\overline{A}_N} = \sqrt{\tau_{\text{int}}\sigma^2_{A_k}/K}$, is shown in column 5.

- The last two columns show the first two autocorrelation estimates. These show that for $\eta = 0$ (the IID case), Y_t samples are indeed approximately IID. For $\eta = 0.9$, the $B = 64$ batches are nearly independent, and the largest batches are quite weakly correlated. For $\eta = 0.999$, where $\tau_{\text{int}} = 1999$, batch sizes of 64 and 512 are too small, but batch size 4096 is large enough to produce weakly correlated batches.

- For $\eta = 0$, the average of the single-experiment estimator u of the variance of the sample mean is approximately constant with respect to batch size. For $\eta = 0.9$ and $\eta = 0.999$, once the batch size is large enough to get weakly correlated samples, the estimator u^2 on batches agrees with the estimator u^2 on the original time-series data.

- The multi-realization estimator and the averaged single-realization estimator of the variance of the sample mean (columns 4 and 6) roughly agree, for batch sizes large enough that batches are weakly correlated.

- The error of the error bar (column 7 of the table) is not improved by use of batched means.

η	B	$m_M(\overline{A}_N)$	$s_M(\overline{A}_N)$	$\sigma_{\overline{A}_N}$	$m_M(u_N(A_k))$	$s_M(u_N(A_k))$	$\hat{c}_{A_k}(0)$	$\hat{c}_{A_k}(1)$
0.000	1	0.5001	0.00113	0.00113	0.00113	0.000002	1.0000	0.0001
0.000	64	0.5001	0.00113	0.00113	0.00113	0.000027	0.9990	0.0378
0.000	512	0.5001	0.00113	0.00113	0.00114	0.000073	0.9922	0.0031
0.000	4096	0.5001	0.00113	0.00113	0.00111	0.000206	0.9375	0.2053
0.900	1	0.4996	0.00481	0.00492	0.00490	0.000034	1.0000	0.8982
0.900	64	0.4996	0.00481	0.00492	0.00451	0.000103	0.9990	0.1591
0.900	512	0.4996	0.00481	0.00492	0.00480	0.000280	0.9922	-0.0552
0.900	4096	0.4996	0.00481	0.00492	0.00476	0.000894	0.9375	0.1445
0.999	1	0.5112	0.04801	0.05042	0.04937	0.004180	1.0000	0.9987
0.999	64	0.5112	0.04801	0.05042	0.00874	0.000756	0.9990	0.9493
0.999	512	0.5112	0.04801	0.05042	0.02301	0.002276	0.9922	0.6462
0.999	4096	0.5112	0.04801	0.05042	0.04280	0.007354	0.9375	-0.0060

TABLE B.3. Statistics for $M = 100$ batched experiments on $N = 65536$ samples of Y_t: $\eta = 0.0, 0.9, 0.999$.

B.15 Conclusions on error bars, autocorrelation, and batched means

Given an MCMC experimental result $X_0, \ldots, X_{N-1}$, we may compute the sample mean $\overline{X}_N$ and an estimator $s_N^2(X_t)$ of the sample variance.

Batched means improve neither the bias nor the variation of the error bar. The variance reduction obtained by (approximate) independence of batches cancels out the variance increase caused by reduced sample size.

Computing autocorrelations and summing them as described in section B.9, we may obtain an estimate $\hat{\tau}_{\text{int}}$ of the integrated autocorrelation time τ_{int}. This is used to update the naive estimated variance of the sample mean $t_N^2(X_t) = s^2/N$ to the corrected estimator $u_N^2(X_t) = \hat{\tau}_{\text{int}} s^2/N$. With the understanding that $\hat{\tau}_{\text{int}}$ has itself a noticeable variance and a fractional underbias, u_N estimates the standard deviation of the sample mean.

Index

References

The random-cycle model:

[BU07] Betz, V. and Ueltschi, D. *Spatial random permutations and infinite cycles.* `arXiv:0711.1188`. Commun. Math. Phys. 285, 469-501 (2009).

[BU08] Betz, V. and Ueltschi, D. *Spatial random permutations with small cycle weights.* `arXiv:0812.0569v1`. Probabl. Th. Rel. Fields (2010).

[BU10] Betz, V. and Ueltschi, D. *Critical temperature of dilute Bose gases.* Physical Review A **91**, 023611 (2010). `arXiv:0910.3558`.

[BUV09] Betz, V. Ueltschi, D., and Velenik, Y. *Random permutations with cycle weights.* `arXiv:0908.2217`.

[GRU] Gandolfo, D., Ruiz, J., and Ueltschi, D. *On a model of random cycles.* `arXiv:cond-mat/0703315`. Statist. Phys. 129, 663-676 (2007).

[Lugo] Lugo, M. *Profiles of permutations.* Electronic Journal of Combinatorics, **16** (2009) R99.

[Sütő1] Sütő, A. *Percolation transition in the Bose gas.* J. Phys. A: Math. Gen. **26** (1993) 4689-4710.

[Sütő2] Sütő, A. *Percolation transition in the Bose gas II.* J. Phys. A: Math. Gen. **35** (2002) 6995-7002.

[U06] Ueltschi, D. *Feynman cycles in the Bose gas.* `arXiv:math-ph/0605002v2`.

[U07] Ueltschi, D. *The model of interacting spatial permutations and its relation to the Bose gas.* `arXiv:0712.2443v3`. Mathematical Results in Quantum Mechanics, pp. 225-272, World Scientific (2008).

Non-spatial permutations:

[DF] D.S. Dummit and R.M. Foote. *Abstract Algebra* (2nd ed.). John Wiley and Sons, 1999.

[Golomb] Golomb, S.W. *Random permutations.* Bull. Amer. Math. Soc. 70 (1964), 747.

[SL] Shepp, L.A. and Lloyd, S.P. *Ordered Cycle Length in a Random Permutation.* Trans. Amer. Math. Soc. 121, (1966), 340-357.

[Ewens] Ewens, W.J. *The sampling theory of selectively neutral alleles.* Theor. Popul. Biol. 3, 87-112 (1972).

Bose-Einstein condensation:

[AEMWC] Anderson, M.H., Ensher, J.R., Matthews, M.R., Wieman, C.E., and Cornell, E.A. *Observation of Bose-Einstein Condensation in a Dilute Atomic Vapor. Science*, vol. 269, issue 5221 (Jul. 14 1995), 198-201.

[BBHLV] Baym, G., Blaizot, J.-P., Holzmann, M., Laloë, F., and Vautherin, D. *Bose-Einstein transition in a dilute interacting gas.* `arXiv:cond-mat/0107129v2`.

[AM] Arnold, X. and Moore, X. *BEC transition temperature of a dilute homogeneous imperfect Bose gas.* Phys. Rev. Lett. vol. 87, no. 12, Sept. 2001.

[BP] Buffet, E. and Pulè, J.V. *Fluctuation properties of the imperfect Bose gas.* J. Math. Phys. vol. 24, no. 6, June 1983.

[Ceperley] Ceperley, D.M. *Path integrals in the theory of condensed helium.* Rev. Mod. Phys. vol. 67, no. 2, April 1995.

[Feynman] Feynman, R.P. *Atomic Theory of the λ Transition in Helium.* The Physical Review, vol. 91, no. 6 (1953).

[Grüter] Grüter, P. *Contribution à la Théorie des Gaz Dilués-Dégénérés.* Doctoral dissertation, Université de Paris Sud, Département de Physique de l'École Normale Supérieure, 1996.

[Holzmann] Holzmann, M. *La Transition de Bose-Einstein dans un gaz dilué.* Doctoral dissertation, Université de Paris VI, Département de Physique de l'École Normale Supérieure, 2000.

[LLS] Lebowitz, J., Lenci, M., and Spohn, H. *Large deviations for ideal quantum systems.* J. Math. Phys. 41, 1224-1243 (2000).

[LSSY] Lieb, E.H., Seiringer, R., Solovej, J.P., and Yngvason, J. *The Mathematics of the Bose Gas and its Condensation.* Oberwolfach Seminars, vol. 34. Birkhäuser, 2005.

[London] London, F. Phys. Rev. vol. 54, no. 947 (1938).

[PO] Penrose, O. and Onsager, L. *Bose-Einstein Condensation and Liquid Helium.* The Physical Review, vol. 104, no. 3 (1956).

[SU09] Seiringer, R. and Ueltschi, D. *Rigorous upper bound on the critical temperature of dilute Bose gases.* `arXiv.org:0904.0050`. Phys. Rev. B 80, 014502 (2009).

Worm algorithm for path-integral Monte Carlo:

[BPS06] Boninsegni, M., Prokof'ev, N.V., and Svistunov, B.V. *Worm algorithm and diagrammatic Monte Carlo: A new approach to continuous-space path integral Monte Carlo simulations.* Physical Review E **74**, 036701 (2006).

[PST98] Prokof'ev, N.V., Svistunov, B.V., and Tupitsyn, I.S. *Exact, complete, and universal continuous-time worldline Monte Carlo approach to the statistics of discrete quantum systems.* Journal of Experimental and Theoretical Physics, vol. 87, no. 2 (1998).

Finite-size scaling for path-integral Monte Carlo:

[Barber] Barber, M.N. *Finite-size scaling.* Phase Transitions and Critical Phenomena, Vol. 8., pp 146-266. Academic Press, London, 1983.

[GCL97] Grüter, P., Ceperley, D., and Lalöe, F. *Critical temperature of Bose-Einstein condensation of hard-sphere gases.* Physical Review Letters, vol. 79, no. 19 (1997).

[HK99] Holzmann,M. and Krauth, W. *Transition temperature of the homogeneous, weakly interacting Bose gas.* Physical Review Letters, vol. 83, no. 14 (1999).

[KPS] Kashurnikov, V.A., Prokof'ev, N.V., and Svistunov, B.V. *Critical temperature shift in weakly interacting Bose gas.* Physical Review Letters, vol. 87, no. 12 (2001).

[LB] Landau, D.P. and Binder, K. *A Guide to Monte Carlo Simulations in Statistical Physics* (2nd ed.). Cambridge University Press, 2005.

[NL04] Nho, K. and Landau, D.P. *Bose-Einstein condensation temperature of a homogeneous weakly interacting Bose gas: Path integral Monte Carlo study.* Physical Review A **70**, 053614 (2004)

[PC87] Pollock, E.L. and Ceperley, D.M. *Path-integral computation of superfluid densities.* Physical Review B, vol. 36, no. 16 (1987).

[PGP08] Pilati, S., Giorgini, S., and Prokof'ev, N.V. *Critical temperature of interacting Bose gases in two and three dimensions.* Physical Review Letters **100**, 140405 (2008).

[PR92] Pollock, E.L. and Runge, K.J. *Finite-size-scaling analysis of a simulation of the ^{4}He superfluid transition.* Physical Review B, vol. 46, no. 6 (1992).

[PV] Pelissetto, A. and Vicari, E. *Critical Phenomena and Renormalization-Group Theory.* Physics Reports, 368 (6), p.549-727, Oct 2002. `arxiv:cond-mat/0012164`.

Quantum mechanics and statistical mechanics:

[Griffiths] Griffiths, D.J. *Introduction to Quantum Mechanics* (2nd ed.). Pearson Prentice Hall, 2005.

[Huang] Huang, K. *Introduction to Statistical Physics.* CRC Press, 2001.

[Sakurai] Sakurai, J.J. *Modern Quantum Mechanics* (2nd ed.). Addison Wesley, 1993.

Probability, stochastic processes, and statistics:

[Berg] Berg, B. *Markov Chain Monte Carlo Simulations and Their Statistical Analysis.* World Scientific Publishing, 2004.

[CB] Casella, G. and Berger, R.L. *Statistical Inference* (2nd ed.). Duxbury Press, 2001.

[FG] Fristedt, B. and Gray, L. *A Modern Approach to Probability Theory.* Birkhäuser, 1997.

[GS] Grimmett, G. and Stirzaker, D. *Probability and Random Processes*, 3rd ed. Oxford, 2001.

[Lawler] Lawler, G. *Introduction to Stochastic Processes* (2nd ed.). Chapman & Hall/CRC, 2006.

[Øksendal] Øksendal, B. *Stochastic Differential Equations* (6th ed.). Springer, 2007.

[Young] Young, H.D. *Statistical Treatment of Experimental Data.* McGraw-Hill, 1962.

Algorithms:

[MN] Matsumoto, M. and Nishimura, T. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Transactions on Modeling and Computer Simulation, Volume 8, Issue 1 (Jan. 1998), pp 3-30.

[NR] Press, W. et al. *Numerical Recipes* (2nd ed.). Cambridge, 1992.