Markov Chains and Tensor Networks: Numerical Techniques in Phase Transitions

Yinhei Chow
supervised by Prof. Zi Yang Meng

22 Semptember 2023

Abstract

I write these notes in an attempt to orient myself as I first tread into this vast landscape of many-body physics. I go over some techniques like Markov chain Monte Carlo, finite-size scaling and renormalization group. This writing serves also as the report of a self-initiated project I did under Prof. Meng's guidance in the summer of my second year studying at The University of Hong Kong

KEYWORDS: Ising model. Second order phase transition. Markov chain Monte Carlo. Metropolis algorithm. Scaling forms. Wolff cluster algorithm. Transverse field Ising model.

Contents

1 Ising **1**
1.1 Convolution 1

2 Coarse graining probability vectors **2**
2.1 Theory 2
2.2 Implementation 3

3 Wolff **4**
3.1 Variant 4
3.2 Generalization 4
3.3 Transformation 5

4 Scaling **5**
4.1 Generalized homogeneity hypothesis . 5
4.2 Theory 5
4.3 Implementation 5

5 Performance evaluation **6**
5.1 Time-series analysis 6

6 Tensor network renormlization **8**
6.1 Theory 8
6.1.1 Rank fluidity 8
6.2 Implementation 8
6.3 World line 9

A Appendix **10**

We exploit the convergence property of Markov chains to study collective phenomena. We limit ourselves to the simplest nontrivial example - Ising model. It is simple enough to admit an analytic solution in one and two dimensions, making it a nice candidate to test new numerical methods before they are applied to actual intractable problems.

1 Ising

An Ising model is an interacting system of localized spin degrees of freedom. It is a special case of O_n model, defined by the Hamiltonian

$$H = -J \sum_{<ij>} \sigma_i \cdot \sigma_j - Jh \cdot \sum_i \sigma_i,$$

where the spin degrees of freedom lives on the $(n-1)$-sphere $\sigma : \Lambda \to S_{n-1}$, with Λ being a lattice with periodic boundary condition, and the sum runs over all bonds of the lattice. In our discussion below, we will set external magnetic field h, i.e. the force conjugating to our scalar field σ, to zero. This is because the system otherwise exhibits no disordered phase, no to mention a phase transition.

The model is so-called because the symmetry group of its Hamiltonian, or precisely the action within, is the n^{th} dimensional orthogonal group O_n. In other words, the Hamiltonian is invariant under the simultaneously transformation of all its spins $(\sigma_i)_i$ by any element from O_n. Ising model then corresponds to choosing $n = 1$, so that the spins simply take values ± 1. We further choose to define it on a two-dimensional square lattice.

1.1 Convolution

Inspired by how much the Ising model looks like a convolutional[1] neural network (they are similar in that both are obtained by assuming locality on their respective more general problems: the fully interacting problem for localized spin models - i.e. every

[1] More precisely, a convolutional layer actually uses cross-correlation, but that chirality doesn't matter here.

spins interacts with every other spins - ; and a dense layer in neural networks), we write

$$H = -J\sigma : \tilde{\Delta} * \sigma$$

where $*$ is a convolution respecting the periodic boundary of matrix σ, : is the Frobenius product. Note that the kernel here is not the Laplacian mask, but

$$\tilde{\Delta} := \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix},$$
$$\tilde{\Delta} * \sigma = \Delta * \sigma + 4\sigma.$$

The reason I do this is because it makes the later code implementation effortless - the messy picture of summing over nearest neighbors simplifies into two global multiplications: one pulls the neighbors together, the second compares them with the spin they encircle. Also, when our lattice gets large, well-established algorithms like the fast Fourier transform (FFT) is going to help us with speed. Recall that convolution is a multiplication but in Fourier space.

The second reason is that, by simply swapping the Frobenius product for Hadamard product $\odot$, we get to visualize the energy distribution automatically (see figure 1)

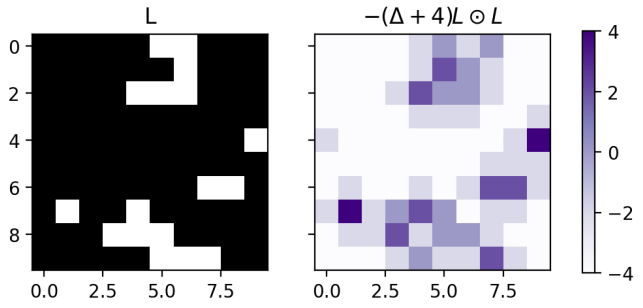

Figure 1: A thermalizing state, with the left panel showing the configuration of up and down spins, and the right panel showing the corresponding energy distribution.

2 Coarse graining probability vectors

To determine the time averages of thermodynamic quantities, which is assumed to equals the ensemble average, we estimate this ensemble average with one that runs over only a subset of the state space. The question is then how should we choose this subset to serve as a good representative (this is often referred to as importance sampling). A good choice is naturally one that allows the probability distribution on the state space be invariant upon truncation of much of the state space in obtaining that subset. In other words, we need a way to generate samples that together follows a coarse-grained version of the original probability vector.

2.1 Theory

Markov chain then lends itself as a convenient way to cruise this overwhelmingly vast state space. This is because of its nice convergent property: as long as we design the dynamics (or equivalently its stochastic matrix) such that it satisfies the detailed balance condition corresponding to our distribution of interest, convergence into it and it only is guaranteed [2].

With the statics set, we proceed to endow it with a dynamics.

Despite the process being random, if $\vec{p}(t)$ is the probability vector, its evolution can be described deterministically as

$$\vec{p}(t+1) = P\,\vec{p}(t) \quad \forall t,$$
$$P := (P_{ij})_{i,j\in C},\ P_{ij} := P(j \to i),$$

where P is the stochastic matrix - a matrix made up of stochastic vectors as its columns. In other words, elements are partition of unity along every columns. The transition probabilities housed in P is up to us to design and is equivalent to choosing a dynamics.

The expression can be unpacked into the master equation

$$\partial_t\, p_\mu(t) = \sum_{\nu\neq\mu} p_\nu(t)P(\nu \to \mu) - p_\mu(t)P(\mu \to \nu)$$

by noting that the diagonals cancel (- obviously, the probability leaking out of a state heading towards itself will be entirely compensated by the probability entering that state originating from itself). All that's left to contribute to a change in local probability density is then the net amount of probability flowing into the state from its environment. That's why homogeneity leads immediately to a continuity equation.

[2] For convergence, the Markov chain needs also to satisfies ergodicity, normalization and homogeneity, which can be easily verified for our case here: ergodicity comes from the ergodicity hypothesis - via detailed balance asserting that transition probabilities be nonzero if stationary distribution has no zero component, normlization is inherent in the definition of a stochastic matrix, and homogeneity is equivalent to the fact that we can even write the above matrix notation for updating the probabiltilies.

Solving for stationary distributions

$$\partial_t \vec{p}(t) = 0$$

$$\iff \quad \sum_{\nu \neq \mu} p_\nu(t) P(\nu \to \mu) = p_\mu(t) \sum_{\nu \neq \mu} P(\mu \to \nu) \quad \forall \mu$$

(global balance)

$$\impliedby \qquad p_\nu(t) P(\nu \to \mu) = p_\mu(t) P(\mu \to \nu) \quad \forall \nu, \mu$$

(detailed balance)

then gives detailed balance as a sufficient condition.

To let the system converges into the thermodynamic equilibrium, fixing the temperature, we can then assume the Boltzmann distribution as one stationary distribution

$$p_\nu = \frac{e^{-\beta E(\nu)}}{\sum_\mu e^{-\beta E(\mu)}}.$$

This is precisely the softmax activation, which maximizes entropy[3] among the classes ν, with β controlling the base of the exponential rescaling on logits.

As will be made clear in a later section on Wolff cluster, it is convenient to factor explicitly the transition probability into two parts: trail (T) and acceptance (A) probabilities. For our single-spin update algorithm, T is $\frac{1}{L^2}$ or zero, and it is nonzero only when the two states it connects differ by one spin. Hence, it is symmetric for any two states μ and ν, and must therefore cancels to reveal only acceptance probabilities into direct contact with Boltzmann distribution in the condition of detailed balance:

$$e^{-\beta \Delta E_{\mu \to \nu}} = \frac{e^{-\beta E(\nu)}}{e^{-\beta E(\mu)}} = \frac{p_\nu(t)}{p_\mu(t)} = \frac{A(\mu \to \nu)}{A(\nu \to \mu)}$$

where $\Delta E_{\mu \to \nu} := E_\nu - E_\mu$, which also gives A of reverse transitions as reciprocals. It is now apparent that one choice for acceptance probability is $A(\mu \to \nu) := min\{1, e^{-\beta \Delta E_{\mu \to \nu}}\}$, by forcefully capping the left-most value at one. This is however not the only possible choice that bring about detailed balance. For example, here's a differentiable alternative making using of logistic function:

$$A(\mu \to \nu) := \frac{1}{e^{\beta \Delta E_{\mu \to \nu}} + 1}.$$

We will adopt the former choice - referred to as the Metropolis algorithm (MA) - because accepting as many proposed transitions as possible is the best use of computational resource.

[3] That is, to suppress prior information.

2.2 Implementation

Looping over a lattice of lattice sizes L and temperatures T, we draw the order parameter m, internal energy u, and their respective response functions x, c_v(see figure 2).

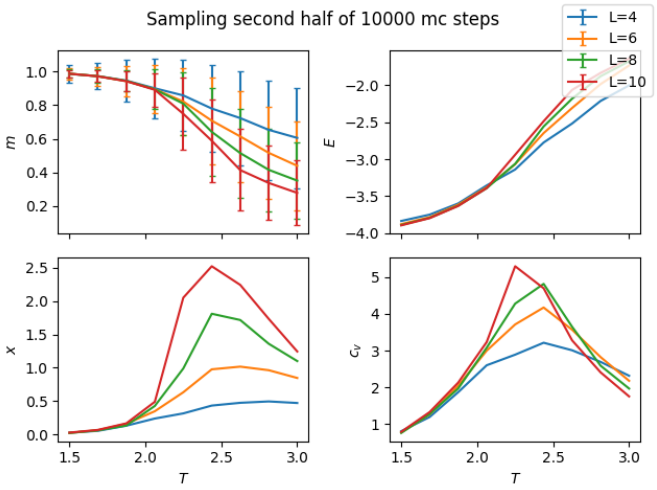

Figure 2: Figure showing order parameter, internal energy, and their respective response functions as estimated by Metropolis algorithm. The error bars in the plot of the order parameter m represents the standard deviation rather than the standard error, since the latter is too small. In other words, the error bars are standard errors scaled up by $\sqrt{N}$ where N is the number of samples at each temperature. Metaphorically speaking, the above x-axis is a transect laid along $h = 0$ crossing the phase transition point, sampling the ecology on the phase diagram.

Energy is continuous over T, as expected of a continuous phase transition. We see from the standard deviation of the sample generated that the algorithm gets less confident as it moves into the phase transition from low temperatures, but then the error return to a lower value as it treads deeper into the disordered phase (high temperatures). Both specific magnetic susceptibility x and heat capacity c_v go through a spike in the phase transition. As shown, the spike gets sharper for a larger system size, since the estimate is getting closer to the thermodynamic limit[4]. We know that x diverges at precisely the phase transition, so we can use

$$T_c^i := \underset{T}{\operatorname{argmax}}\, x_i(T)$$

as an estimator for T_c. This way, the value of T_c can

[4] A phase transition point is a nonanalyticity in the system's partition function. So, a finite-system never exhibits an actual phase transition, with its partition function being a finite sum of analytic terms. The transition is of order n if it is what impedes the partition function Z from being welcomed into the differentiability class C_n. Loosely speaking, the order states how convertly the discontinuity of phases lies in Z.

be obtained to arbitrary precision by doing the simulation for ever larger linear dimensions L, switching to a finer temperature scale whenever the estimate T_c^i becomes stationary in i.

There is a much more effective finite-size scaling method for determining properties (e.g. T_c) of the criticality, by using a crossing point induced upon rescaling the axes in a specific way. Yet, before we move onto that, we will upgrade our current Markov chain Monte Carlo algorithm to one better suited for sampling physics near criticality - cluster update.

3 Wolff

As mentioned, changing the dynamics to that of Wolff algorithm (WA) is the same as changing the Markov matrix P, made up entirely by individual T, A probabilities.

One way to graduate to a nonlocal update procedure is to consider bond percolation. Letting adjacent spin degrees of freedom connect with probability p, one treats the clusters arising from percolation as the smallest unit for an update, and proceed to fine-tune such p to satisfy detailed balance (recall this condition ensures that simulation converges). We build from the previous unit of single spins larger units - called Wolff clusters.

The original paper [1] deduces the form for p by separating Z into terms satisfying specific restrictions. The later paper [2] merely states p, followed by an explicit demonstration of how the transition probability satisfy detailed balance.

I personally found the derivation of p to be most natural by making use of the similarity between Wolff algorithm and Metropolis algorithm. Unfortunately, I am unable found any proofs applying this fact, nor even any mention of this nice similarity. So, I would have to elaborate on such idea below. Succinctly, this is the connection that I wish to depict:

$$T_{wol}(\mu \to \nu) \coloneqq \prod_{\langle ij\rangle \in \partial\Lambda'} A_{met}\Big|_{\langle ij\rangle}(\mu \to \nu),$$

which gives $p(\nu_i, \mu_j) = 1 - A_{met}\big|_{\langle ij\rangle}(\mu \to \nu)$ (eq. 1).

Purely syntactically, WA can be thought of as the double-negative of MA. MA flips a spin if doing so lowers energy. In contrast, WA refuses to flip a spin if doing so raises energy (see eq. 1). In additional to this, each spin gets at most four independent chances of getting flipped from the four directions that a cluster could extend into it. From this viewpoint, the latter is much laxer in extending clusters into spins than the former is to flip spins. This added slackness eventually allows WA to perform collective Monte Carlo updates without violating detailed balance.

To draw our analogy rigorously, we construct

$$\Delta E^{\langle ij\rangle}_{\mu\to\nu} \coloneqq -J(\nu_i\nu_j - \mu_i\mu_j)$$

as the change in energy brought about by the activation of bond $\langle ij\rangle$. Recall in the previous section, we have derived that $\forall \mu, \nu \; s.t. \; |\mu - \nu| = 1$,

$$\begin{aligned} A_{met}(\mu \to \nu) &\coloneqq min\{1, e^{-\beta\Delta E_{\mu\to\nu}}\} \\ \Rightarrow \frac{A_{met}(\mu \to \nu)}{A_{met}(\nu \to \mu)} &= e^{-\beta(E_\nu - E_\mu)}. \end{aligned}$$

Now, in direct analogy to this, for every $\mu, \nu \in \{\pm 1\}^\Lambda$ such that they differ by a connected region $\Lambda' \subset \Lambda$,

$$\begin{aligned} T(\mu \to \nu) &\coloneqq \prod_{\langle ij\rangle \in \partial\Lambda'} min\{1, e^{-\beta\Delta E^{\langle ij\rangle}_{\mu\to\nu}}\} \\ \Rightarrow \frac{T(\mu \to \nu)}{T(\nu \to \mu)} &= \prod_{\langle ij\rangle \in \partial\Lambda'} e^{-\beta\Delta E^{\langle ij\rangle}_{\mu\to\nu}} = e^{-\beta(E_\nu - E_\mu)}. \end{aligned}$$

All that remains is the realization that A can be arbitrary symmetric functions, so we may as well as flip all the Wolff clusters we have so painstakingly generated (i.e. to set $A = 1$).

Let $p(\nu_i, \mu_j)$ denotes the probability of activating bond $\langle ij\rangle$. The selection probability of transition $\mu \to \nu$ is the probability of the Wolff cluster Λ' forming. Yet, a cluster is entirely defined by its boundary, so it suffices to cumulate probabilities of each boundary segment arising independently, resulting in the equality $T(\mu \to \nu) = \prod_{\langle ij\rangle \in \partial\Lambda'} (1 - p)$.

This holds $\forall \Lambda' \subset \Lambda$, so

$$p = 1 - min\{1, e^{K(\nu_i\nu_j - \mu_i\mu_j)}\} = \begin{cases} 1 - e^{-2K}, & \mu_i\mu_j = 1 \\ 0, & \mu_i\mu_j \neq 1 \end{cases}.$$

We will need p as the operational parameter for realizing the defined T.

3.1 Variant

A variant is the Swendsen-Wang algorithm, which after partitioning spin clusters into Wolff clusters, flips the latter with 0.5 probability. This is less inefficient since half of the clusters grown end up not being realized, though some prefer this algorithm if they wish to e.g. estimate susceptibility with the mean cluster size, rather than as the fluctuation of order parameter.

3.2 Generalization

Individual energy changes $\Delta E^{\langle ij\rangle}_{\mu\to\nu} = \mp 2J$ is but a way to check whether or not the two spins μ_i, μ_j starts out in the same spin cluster. But in the $O(n)$ model, $\Delta E^{\langle ij\rangle}_{\mu\to\nu} \coloneqq -J(\nu_i \cdot \nu_j - \mu_i \cdot \mu_j) \in [-2J, 2J]$

takes a continuum of value, and measures how much the pair fails to align.

Then $p = 1 - e^{-\beta \Delta E^{\langle ij \rangle}_{\mu \to \nu}}$ whenever $\mu_i \cdot \mu_j > 0$, and $p = 0$ otherwise.

3.3 Transformation

To make explicit the analogy, there's nothing stopping us from defining a transformation $(T_{met}, A_{met}) \mapsto (T_{wol}, A_{wol})$ by

$$T_{wol}(\mu \to \nu) \coloneqq \prod_{\langle ij \rangle \in \partial \Lambda'} A_{met} \Big|_{\langle ij \rangle} (\mu \to \nu),$$
$$A_{wol}(\mu \to \nu) \coloneqq |\Lambda| \cdot T_{met}(\mu \to \nu).$$

All that is insightful about the transformation resides in the first equation. It encodes the observation that Wolff selection probability of forming a cluster is obtained by carrying out individual Metropolis acceptance probabilities across the boundary.

Equation two of above transformation would probably come about as meaningless, or downright pretentious. I mean, defining a constant by a scalar multiple of another constant sounds like I am overdoing it, and it is not well-justified because unlike the first equation, there's no real structure to be passed down from T_{met}. But all I mean to convey is their ratio being the lattice size $|\Lambda|$. It speaks to the contrast that one of them is local, the other is global. We will adopt this ratio in setting Monte Carlo (MC) steps in the later section Performance evaluation. We define one MC step for WA as one collective update, and that for MA as $|\Lambda|$ single-spin updates. That is, they agree at low temperatures. Of course, the performances measured depend not on this scale we choose for respective virtual time axes.

My hypothesis is that this idea can be developed into a general local-nonlocal transformation procedure for generating collective Monte Carlo algorithms satisfying certain conditions.

Ergodicity is automatic, since Wolff algorithm can be seen as a generalization of Metropolis algorithm, that is, given any path in the state space induced by Metropolis algorithm, there is a nontrivial probability for Wolff algorithm to generate exactly that path. More briefly, a single spin is a possible Wolff cluster.

With the Wolff algorithm defined, we proceed to implement the promised crossing-point method.

4 Scaling

4.1 Generalized homogeneity hypothesis

The generalized homogeneity hypothesis, or briefly, scaling hypothesis, asserts all thermodynamic functions and their correlation functions be generalized homogeneous about the system's critical point. From it we get the natural concept of critical exponents. A collection of them belonging to a theory is known as a universality class. The punchline is that a universality class depends only on the dimension of the system, as well as the dimension and range of the microscopic interactions.

4.2 Theory

To write the power law scaling form of a thermodynamic function f, we defined its associated critical exponent as

$$\beta_f \coloneqq \lim_{t \to 0} \frac{ln|f(t)|}{ln|t|},$$

and write the scaling form $f \sim |t|^{\beta_f}$. Note the double-logarithm in the definition - this is characteristic to power laws and also the reason we will use a log-log visualization in figure () to expose dynamic exponents z as slopes. When β_f is negative, for example in how x, c_v diverges in figure (), it is customary to replace it with $-\beta_f$.

Simply put, a critical exponent is the degree of homogeneity for a thermodynamic function, in a reduced coordinate system $(h, t) = (h - h_c, \frac{T - T_c}{T_c})$ ($h_c = 0$ as mentioned previously).

$$m_L(T) = L^{-\beta/\nu} \tilde{m}(L^{1/\nu}(T - T_c))$$

4.3 Implementation

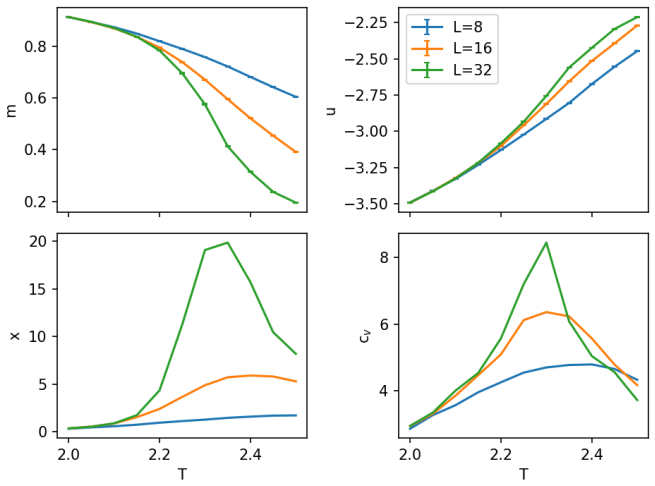

Figure 3: Same as figure 2, but with Wolff algorithm replacing Metropolis algorithm, and the error bars are the standard errors. Note the finer temperature scale and larger system sizes.

Notice that at $T = T_c$, the L-dependence of $L^{\beta/\nu} m_L(T)$ vanishes. In other words, scaling the y-axis accordingly creates a crossing-point of all simulations with different L, whose x-coordinate gives an estimate for T_c. This is demonstrated in figure 4.

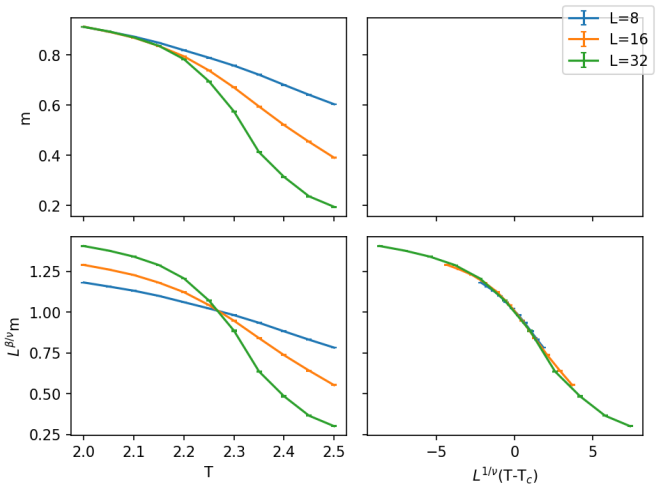

Figure 4: Figure showing the crossing-point, formed upon rescaling the axes of the data collapse lying in the top-left panel of figure 3

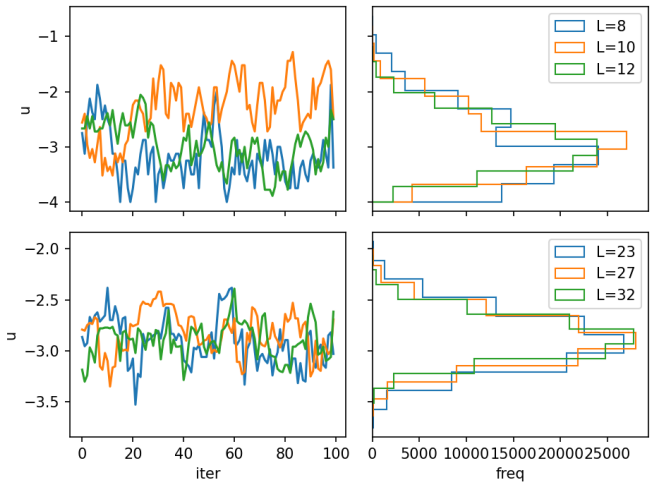

Figure 6: The left panels display a 100-length interval random sampled from the time series. The right panels plot histograms obtained by collapsing the entire time series. The first and second rows correspond to the surveyed energy densities of Metropolis algorithm and Wolff algorithm respectively.

$$(\beta, \nu) \mapsto T_c$$

This phenomenon of universality is best explained by the theory of renormalization group. Unfortunately, at this stage I lack the expertise to give full justice to this graduate topic, so I could only point readers to [3]. That said, I would like to point out one key construct in the derivation, the block-spin transformation, since it connects naturally to the later section on tensor renormalization group.

5 Performance evaluation

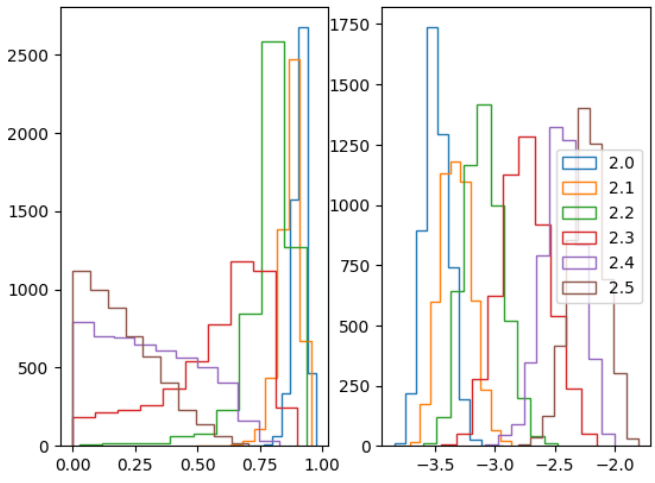

Figure 5: Figure showing the distributions of m and u, obtained from collapsing the time series along temporal dimension.

Try as you might, staring intently at the error bars don't seem to provide a clear picture of how the performance of the two algorithms compare. We are also interest to know at which temperature does an algorithm perform the best.

Judging from figure 5, fluctuations in the data generated using Wolff algorithm seem to be slightly tighter, or are we imagining things? As we can see, a subjective measure only goes so far. One can plot the histogram as I have done on the right panels, and for Wolff algorithm's data looking more like a Bell curve is a good indicator that the samples it generated are less correlated. But let's introduce a better measure - dynamic exponent.

5.1 Time-series analysis

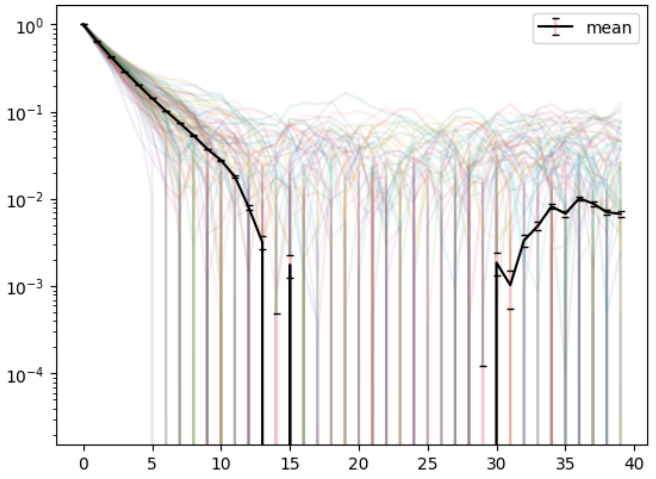

Figure 7: A graph showing the result of block bootstrapping the entire time series of energy density, to estimate the time-displaced autocorrelation function of energy.

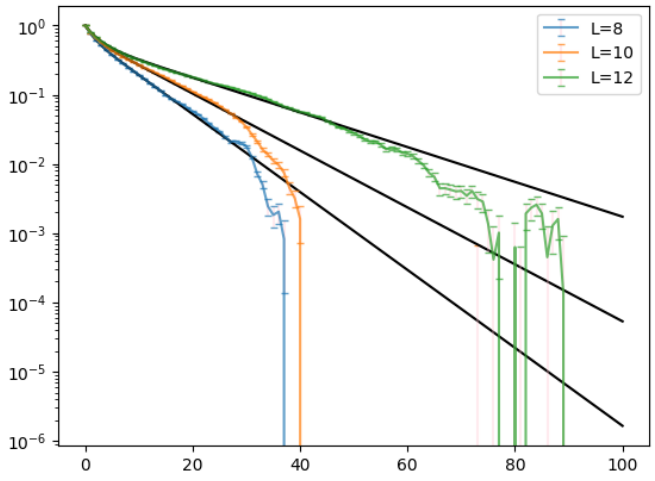

Figure 8: A graph showing the time-displaced autocorrelation of energy density u against virtual time lag. The black lines show the curves fitting to the experimental data.

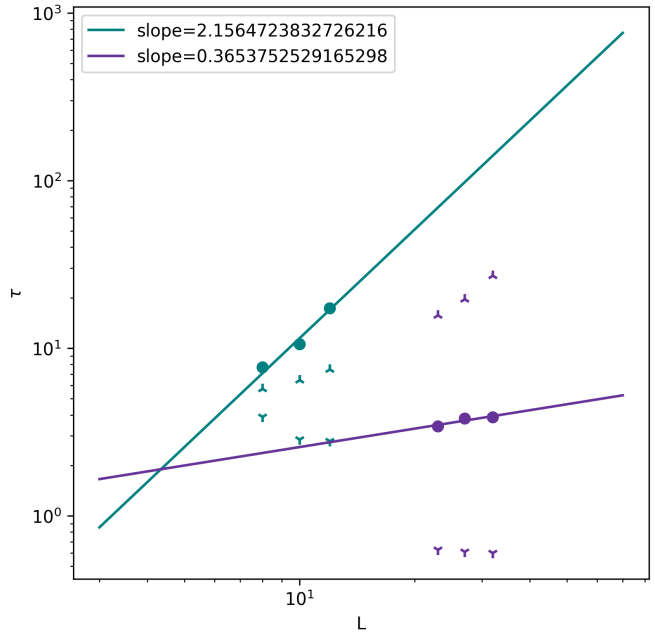

Figure 9: Figure comparing power-law finite-size scaling of correlation times for Metropolis (green) and Wolff algorithms (purple). Simulations are done at three temperatures: T_c for circle, $T_c + 0.5$ for upward tripod, $T_c - 0.5$ for downward tripod.

Discussion Intuitively speaking, dynamic exponent z measures how poorly an algorithm scales with system size. Correlation times τ is a proxy for how slow an algorithm's dynamics washes away "memory" of a previous state. Recall that the essence of Markov chains is their being memoryless. It means that there's no momentum (nor higher time derivatives) in the convergence process, unlike convergence processes like ADAM. This is what makes them ideal for generating maximally independent data points.

Contrasting the two dynamic exponents zs (see figure 9), WA scales much better, and that its scaling is sub-exponential with respect to virtual time (MC step). It is still an exponential-time algorithm since we have to convert virtual time into actual computational time. It takes (on average) exponential time to complete one MC step - i.e. to grow one Wolff cluster - as lattice size creeps up to thermodynamic limit.

For MA, time required to complete a MC step is invariant across temperatures, since it consists of simply selecting with replacement L^2 spins. As a result, it suffices to consider τ as a function of T, and our graph (green data points) shows that its performance is the worst at criticality.

We now turn our attention to WA. States generated close enough along the virtual-time axis are less correlated as temperature decreases, since larger Wolff clusters are grown. This is in agreement with our data. However, similar to that mentioned above, it takes longer time to complete a MC step if the Wolff cluster is huge. While spin clusters put a hard limit on the shape and size of a Wolff cluster generated for that state, violent thermal fluctuations break spin clusters into smaller clusters. This consequently pushes down on the upper limit of Wolff cluster size for a given state. To get the optimal performance, we need to compromise between getting data as uncorrelated as possible and shortening the expected time it takes to generate a data point. Phrased differently: we seek large Wolff clusters, yet we need to control how frequently does spanning Wolff clusters arise. The sweet spot is a unique point where spin clusters of all sizes are equally probable. That is the critical point. As the polar opposite of MA, performance of WA peaks at criticality.

This definitely says something about the nature of criticality in physics. At criticality, the natural interacting unit is a cluster of arbitrary size, rather than individual spins. This presents a new way of understanding long-range correlations present in a many-body system, arising only under very specific environmental conditions. Let me unpack that. At criticality, the consequence of flipping a spin is not going to be contained locally, but that change would be immediately broadcasted to a connected collection of spins, arbitrarily distant from the location upon which the initial perturbation is inflicted.

Unsatisfied by the lack of error bars in the figure 9, as we should be, we come face to face with a major drawback of block bootstrap resampling, that while it gives a good estimate for any functions at a low computational cost, the estimate comes not with an estimate of the error. In other words, we get the values but we don't know their fluctuations across

independent samplings[5].

6 Tensor network renormlization

6.1 Theory

The tensor renormalization group method (TRG) [4] is a classical variant of density matrix renormalization group (DMRG). Compromising by shredding the quantum nature of the latter, it works in one higher dimension (2d). A tensor network approach working in three or higher dimensions remains however a hot research topic.

For a classical Ising model defined on a 2-dimensional square lattice,

$$Z = \sum_{\sigma} \prod_{i} e^{K\sigma_i(\sigma_{i+\hat{1}}+\sigma_{i+\hat{2}})}$$

where $\hat{1}, \hat{2}$ denote the smallest generators of the lattice. This is recast into a tensor network presentation

$$Z = \sum_{\sigma\sigma'} \prod_{i\in\Lambda} \underbrace{\delta_{\sigma_i\sigma'_i} e^{K(\sigma_i\sigma_{i+\hat{1}}+\sigma'_i\sigma'_{i+\hat{2}})}}_{T_{\sigma_i\sigma_{i+\hat{1}}\sigma'_i\sigma'_{i+\hat{2}}}}$$

by the artificial introduction of σ' to create the local tensors of rank-4 dimension-2. In other words, we are padding the values with zeros such that they fit nicely into a tensor framework. Notice that T is actually independent of i, thanks to translational invariance of the local interactions.

The tensor network can be alternatively defined on the dual lattice Λ^* comprising of square plaquettes. By the change of variable $\sigma_i\sigma_j \to \sigma_{\langle ij\rangle}$, which I like to think of as a shrinking followed by a rotation by 45°, we obtain the similar form

$$Z = \sum_{\sigma} \prod_{p\in\Lambda^*} \underbrace{\overbrace{\left(\frac{\sigma_{r_p}\sigma_{u_p}\sigma_{l_p}\sigma_{d_p}+1}{2}\right)}^{\bar{\delta}(\sigma,p)} e^{\frac{K}{2}(\sigma_{r_p}+\sigma_{u_p}+\sigma_{l_p}+\sigma_{d_p})}}_{T_{\sigma_{r_p}\sigma_{u_p}\sigma_{l_p}\sigma_{d_p}}}$$

keeping in mind that σ no longer represents spins but are now bond variables. Since the reparameterization breaks $\mathbb{Z}_2$ symmetry, and a symmetry is a redundancy in information, it is bound to introduce a complete copy of the state space housing but nonphysical states. This is the reason for the existence of a factor of $\bar{\delta}(\sigma, p)$ to ensure that if even one square plaquette has a nonphysical parameterization, that state has zero probability of occurring. In other words, it establishes that the only coset accounted for is the one adhering to the $\mathbb{Z}_2$-gauge constraint

$$\prod_{x\in\square_p} \sigma_x = 1,$$

where $\square_p := \{r_p, u_p, l_p, d_p\}$. This tensor T can be interpreted as the wavefunction of a quantum system defined on a loop in the classical lattice, as the loop size tends to infinity, with the basis being the bond variables [4].

6.1.1 Rank fluidity

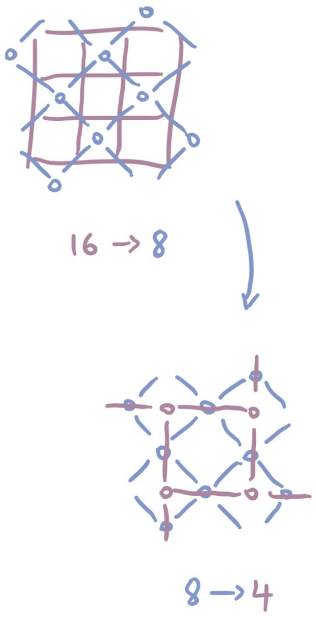

Figure 10: Diagram illustrating the last tensor contraction. Contrary to convention in prevailing literature, I prefer that repeated bonds and tensors not be drawn, since they only confuse the counting of tensors and obscure the boundary condition.

The advantage of tracing over the state space by doing tensor contractions is that information can now be truncated at every length scales. Rank fluidity is the basis for rewiring and decimation, which is performed before each tensor contraction bringing us to the next (coarser) length scale. We have the isomorphism

$$\mathbb{R}^D\otimes\mathbb{R}^D\otimes\mathbb{R}^D\otimes\mathbb{R}^D \cong (\mathbb{R}^{D^2})^*\otimes\mathbb{R}^{D^2} \cong hom(\mathbb{R}^{D^2}, \mathbb{R}^{D^2})$$

where D is the bond dimension, which in the current simple case is two. This maps the local tensor into but a linear map, on which singular value decomposition is possible as a way of getting a list of orthogonal dimensions ordered in terms of their amount of information, measured as the standard deviations of the value they host.

6.2 Implementation

Above a certain threshold of bond dimension, the algorithm fails precisely at the critical point. This is testimony to the divergence of correlation length

[5] That said, just to roughly gauge the self-consistency, a second independent run gives 2.234, 0.341 respectively.

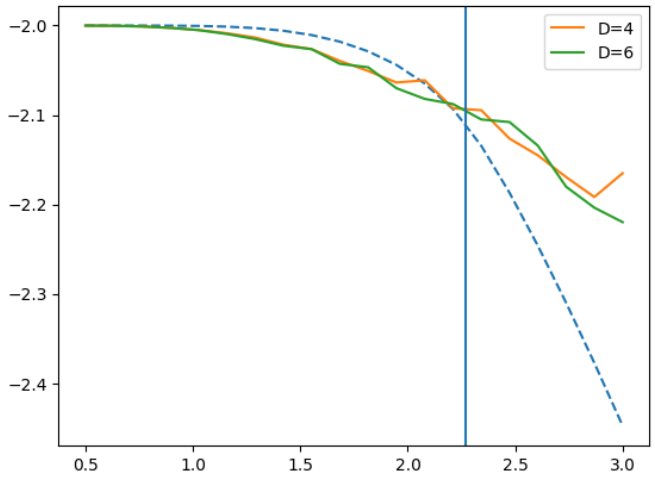

Figure 11: (Preliminary result) Free energy density against temperature at low bond dimensions. The dotted-line is Onsager's exact solution [5].

there, which renders truncation of left singular vectors (serving as basis states) unjustified. The bond dimensions of our two runs (shown in figure 11) are certainly too low to show any of this.

6.3 World line

It may seem strange that this local tensor that originates from the classical partition function could be intepreted as a quantum wavefunction. But this is related to a general procedure of quantum-classical mapping. Below I demonstrate one such mapping using the transverse field Ising model as an example, since this concurrently show how the technique in 3 is used to sample quantum systems, by growing clusters in spacetime.

Consider expansion of $e^{-\beta\hat{H}}$ into imaginary time slices, followed by Trotter decomposition,

$$\begin{aligned} Z_Q^{(d)} &= \sum_{\sigma^z}\prod_t \langle\sigma^z(t)|\otimes_i e^{\beta\hat{\sigma}_i^z\cdot(\hat{\sigma}_{i+\hat{1}}^z+\cdots+\hat{\sigma}_{i+\hat{d}}^z)+\beta g\hat{\sigma}_i^x}|\sigma^z(t+1)\rangle \\ &\approx \sum_{\sigma}\prod_{i,t} e^{\sigma_i(t)\cdot\left\{\tau\left[\sigma_{i+\hat{1}}(t)+\cdots+\sigma_{i+\hat{d}}(t)\right]+f(\tau g)\sigma_i(t+1)\right\}} \\ &= Z_C^{(d+1)}, \end{aligned}$$

where $\sigma^z = \otimes_i\sigma_i^z$, $\sigma = (\sigma_i : [1,m] \to S_0)_i$, $f(x) \coloneqq -\frac{1}{2}ln\,tanh(x)$, $\beta = m\tau$. Note that $f^{-1} = f$.

References

[1] Robert H. Swendsen and Jian-Sheng Wang. "Nonuniversal critical dynamics in Monte Carlo simulations". In: *Phys. Rev. Lett.* 58 (2 Jan. 1987), pp. 86–88. DOI: 10.1103/PhysRevLett.58.86. URL: https://link.aps.org/doi/10.1103/PhysRevLett.58.86.

[2] Ulli Wolff. "Collective Monte Carlo Updating for Spin Systems". In: *Phys. Rev. Lett.* 62 (4 Jan. 1989), pp. 361–364. DOI: 10.1103/PhysRevLett.62.361. URL: https://link.aps.org/doi/10.1103/PhysRevLett.62.361.

[3] John Cardy. "The renormalization group idea". In: *Scaling and Renormalization in Statistical Physics*. Cambridge Lecture Notes in Physics. Cambridge University Press, 1996, pp. 28–60. DOI: 10.1017/CBO9781316036440.004.

[4] Michael Levin and Cody P. Nave. "Tensor Renormalization Group Approach to Two-Dimensional Classical Lattice Models". In: *Phys. Rev. Lett.* 99 (12 Sept. 2007), p. 120601. DOI: 10.1103/PhysRevLett.99.120601. URL: https://link.aps.org/doi/10.1103/PhysRevLett.99.120601.

[5] Lars Onsager. "Crystal Statistics. I. A Two-Dimensional Model with an Order-Disorder Transition". In: *Phys. Rev.* 65 (3-4 Feb. 1944), pp. 117–149. DOI: 10.1103/PhysRev.65.117. URL: https://link.aps.org/doi/10.1103/PhysRev.65.117.

A Appendix

I include here some selected code implementations of the algorithms covered, just to give a more down-to-earth perspective of these abstract contructions. These codes were written for readability not performance.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.ndimage import convolve,generate_binary_structure
from scipy.signal import correlate
import pickle
from scipy.optimize import curve_fit
from google.colab import drive
drive.mount('/content/drive', force_remount=False)


# basic functions
def init_lattice(L,ratio):
  l=np.random.random((L,L))
  l=(l<ratio).astype(int)
  l=2*l-1
  return l

def get_energy(l):
  # filter
  ker=np.array([[0,1,0],[1,0,1],[0,1,0]])

  # neighborhood magnetic field
  h=convolve(l,ker, mode='wrap')

  # energy
  E=-h*l
  return E
```

Listing 1: Common functions

```
# metropolis functions
djs=np.array([[1,0],[0,1],[-1,0],[0,-1]])
def metro_flip(l,b):
    L=l.shape[0]
    for i in range(L**2): # 1
        j=np.random.randint(0,L,2)
        dE=-2*get_energy(l)[j[0],j[1]]
        p=min(1,np.exp(-b*dE))
        l[j[0],j[1]]*=(-1)**(np.random.random()<=p)
    return l

def metropolis(l,iter,cut):
    L=l.shape[0]
    u_t=np.zeros((iter-cut))
    T=2/np.log(1+np.sqrt(2))-0.5
    b=1/T
    for i in range(iter):
            '''if i%1==0:
            plt.imshow(l,cmap='Greys')
            plt.title(f't={i}')
            plt.colorbar()
            plt.show()''' # for generating animations
        l=metro_flip(l,b)
        if i>=cut:
            u_t[i-cut-1]=get_energy(l).sum()/L**2
    return u_t

def loop_over_lattice_size(Ls,iter):
    cut=int(iter/2)#1000
    u_t_L=np.zeros((len(Ls),iter-cut))
    for i,L in enumerate(Ls):
    ratio=1
    l=init_lattice(L,ratio)
```

```
    u_t=metropolis(l,iter,cut)
    u_t_L[i]=u_t
    print('Completed: ',Ls[i])
    return u_t_L
```

Listing 2: Metropolis dynamics

```
# switching to some wolff functions
djs=np.array([[1,0],[0,1],[-1,0],[0,-1]])
def flip(l,b):
    p=1-np.exp(-2*b)
    L=l.shape[0]
    j=np.random.randint(0,L,2)
    surface_of_growth=[j]
    while len(surface_of_growth)!=0:
    #print(surface_of_growth)
    j=surface_of_growth[0]
    for i in (j+djs)%L:

        if l[i[0],i[1]]==l[j[0],j[1]]:
        dice=np.random.random()
        if dice<=p:
            n=0
            for k in surface_of_growth:
            n+=(i==k).prod()
            if n==0:
            surface_of_growth.append(i)

    l[j[0],j[1]]*=-1
    surface_of_growth.pop(0)
    return l

def wolff(l,iter,cut):
    L=l.shape[0]
    u_t=np.zeros((iter-cut))
    T=2/np.log(1+np.sqrt(2))-0.5
    b=1/T
    for i in range(iter):
            '''if i%1==0:
            plt.imshow(l,cmap='Greys')
            plt.title(f't={i}')
            plt.show()'''
        l=flip(l,b)
        if i>=cut:
            u_t[i-cut-1]=get_energy(l).sum()/L**2
    return u_t

def loop_over_lattice_size(Ls,iter):
    cut=int(iter/2)#1000
    u_t_L=np.zeros((len(Ls),iter-cut))
    for i,L in enumerate(Ls):
    ratio=1
    l=init_lattice(L,ratio)
    u_t=wolff(l,iter,cut)
    u_t_L[i]=u_t
    print('Completed: ',Ls[i])
    return u_t_L
```

Listing 3: Wolff cluster dynamics

```
# cut_ from 15 to 100, since now tail is suppressed by zero-padding
# no restriction to 1 for coef, no abs for taus

def fit(cut,cut_):
    taus=np.zeros((len(Ls)))
    fig,ax=plt.subplots()
    for i,autos in enumerate(autos_L):
        markers,caps,bars=ax.errorbar(range(cut),autos.mean(0)[:cut],yerr=autos.std(0)[:
    cut]/np.sqrt(autos.shape[0]) ,capsize=2,label=f'L={Ls[i]}',alpha=0.7)
```

```
        [bar.set_color('pink') for bar in bars]
        [bar.set_alpha(0.3) for bar in bars]

        # fit sum_i a_i e**(-t/tau_i) to auto
        popt, pcov = curve_fit(lambda t,tau0,tau1,tau2,a0,a1,a2 : abs(a0)*np.exp(-t/tau0)+abs(a1)*np.exp(-t/tau1)+abs(a2)*np.exp(-t/tau2), np.arange(cut_), autos.mean(0)[:cut_])
        tau0,tau1,tau2,a0,a1,a2=popt
        a0,a1,a2=abs(a0),abs(a1),abs(a2)
        taus_=np.array([tau0,tau1,tau2])
        aS=np.array([a0,a1,a2])
        print(taus_,aS)
        taus[i]=taus_.max()

        # plot fitted line
        t=np.linspace(0,cut,cut+1)
        fit=a0*np.exp(-t/tau0)+a1*np.exp(-t/tau1)+a2*np.exp(-t/tau2)
        ax.plot(fit,alpha=1,color='black')
    print(taus)
    ax.legend()
    plt.yscale('log')
    plt.show()

    # fit AL**z to tau(L)
    popt, pcov = curve_fit(lambda L,A,z :A*L**z, Ls, taus)
    A,z=popt
    plt.scatter(Ls,taus)
    Lss=np.linspace(1,50,50)
    plt.plot(Lss,A*Lss**z,label=f'slope={z}')
    plt.xscale('log')
    plt.yscale('log')
    plt.legend()
    return taus


# changing to zero-padding from real-data-padding to try to suppress the noise (to insignificant effect)
def bootstrap_auto(u_t_L,blocksize,no_of_bootstraps):
  autos_L=np.zeros((len(Ls),no_of_bootstraps,blocksize))
  for k,u_t in enumerate(u_t_L):
    for i in range(no_of_bootstraps):
      j=np.random.randint(0,len(u_t)-2*blocksize+1)
      block=u_t[j:j+blocksize]
      #auto=correlate(u_t[j:]-u_t[j:].mean(),block-block.mean(),mode='valid') # Alternative
      #auto=auto[:blocksize]
      auto=correlate(block-block.mean(),block-block.mean(),mode='full')
      auto=auto[blocksize-1:]
      autos_L[k][i]=auto/auto[0]
  return autos_L
```

Listing 4: Bootstrap resampling