

# Long range Ising model

Santiago Sanz Wuhl

May 6, 2025

## 1 Introduction

A (classical) spin system with  $N$  particles is governed by the Edwards-Anderson Hamiltonian

$$H = - \sum_{1 \leq j < i}^N J_{ij} s_i s_j, \quad (1)$$

with  $J_{ij}$  the elements of a 2-dimensional symmetric matrix, dictating the interactions between the particle at the site  $i$  and the particle at the site  $j$ , and  $s_i$  is the spin of the particle at the site  $i$ . Any two particles interact via a power law

$$J_{ij} \sim \frac{1}{r_{ij}^{d+\sigma}}, \quad (2)$$

with  $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$ ,  $d$  the spacial dimensions and  $\sigma$  a free parameter characterizing the range of the interaction.

**This should appear below, not here** The sites are set up in a 1-dimensional chain with periodic boundary conditions, and thus  $r_{ij} = \min(|i - j|, |i - j| - N)$ .

### 1.1 The Ising model

#### Notation for states and spins are ambiguous

A fundamental quantity in statistical mechanics is the partition function

$$Z = \sum_s \exp(-\beta E_s),$$

where the sum is performed over states,  $\beta = \frac{1}{T}$  is the inverse temperature and  $E_s$  is the energy of the state  $s$ . By use of the partition function, one is able to calculate observables e.g. the average energy of the system

$$\begin{aligned} \langle E \rangle &= \sum_s E_s P_s = \frac{1}{Z} \sum_s E_s \exp(-\beta E_s) \\ &= -\frac{1}{Z} \frac{\partial}{\partial \beta} Z \\ &= -\frac{\partial \ln Z}{\partial \beta}, \end{aligned} \quad (3)$$

with  $P_s$  the probability of finding the system in the state  $s$ , with energy  $E_s$ .

The averaging over states is however very computationally expensive, as it is to be summed over the  $2dN$  dimensional phase space, with  $d$  the dimensions of the configuration space. One instead calculates thermal averages by use of an algorithm by Metropolis et al. [1953], where the thermal average of an observable  $A$  is calculated by generating a sequence of  $M$  configurations  $\{s_1, \dots, s_M\}$  and calculating

$$\langle A \rangle = \frac{1}{M} \sum_{m=0}^M A_m. \quad (4)$$

A configuration  $s_j$  is generated with a probability  $\pi_{ij}$  from a configuration  $s_i$ . The algorithm proposed by Metropolis et al. [1953] relies on the sampling of the state  $s_j$  with an *a priori* probability  $\alpha_{ij}$ , and it is accepted with a probability  $P_{ij}$ . While  $\alpha_{ij}$  is set by the system, the acceptance probability is

$$P_{ij} = \begin{cases} \exp[-\beta(E_j - E_i)] & E_j > E_i \\ 1 & E_j \leq E_i. \end{cases} \quad (5)$$

This way, a new state  $s_j$  is always accepted if it lowers the total energy of the system, but also accepts states of total higher energy with a probability that decreases with the temperature.

The simplest case that allows one to study the formation of spontaneous magnetization of ferromagnetic systems is the Ising model, where only interaction between nearest neighbors is considered. This is modeled by the Hamiltonian

$$H = -J \sum_{\langle ij \rangle}^N s_i s_j. \quad (6)$$

Here  $\langle ij \rangle$  represents the nearest neighbors of the site  $i$ , and  $J$  is a real constant. In terms of the Hamiltonian (1),

$$J_{ij} = \begin{cases} J & i \& j \text{ neighbors} \\ 0 & \text{else.} \end{cases} \quad (7)$$

For the Ising model, at each iteration  $m$ , a new state  $s_{m+1}$  is proposed by flipping the spin at a random site with a probability  $\alpha_{ij} = N^{-1}$ . The change in energy of the total system depends only on the spins at the nearest neighbors of the site  $i$ . By calculating the change in energy, the new configuration  $s_{m+1}$  is accepted with a probability  $P_{ij}$  given by (5).

However, the Ising model presents no phase transition Ising [1925] for 1-dimensional spin chains, unless long-range interactions are allowed. In order to study the fully connected spin chain with long range interactions (1), one needs to compute  $\mathcal{O}(n^2)$  at each iteration  $m$  of the Metropolis algorithm. In the interest of reducing this computation time, a mode dilution approximation is proposed.

## 1.2 Mode dilution

The mode dilution approximation studies the Hamiltonian given by (1), by setting all the interaction strengths to a constant  $J$ , and only allowing  $N_l$  pairs

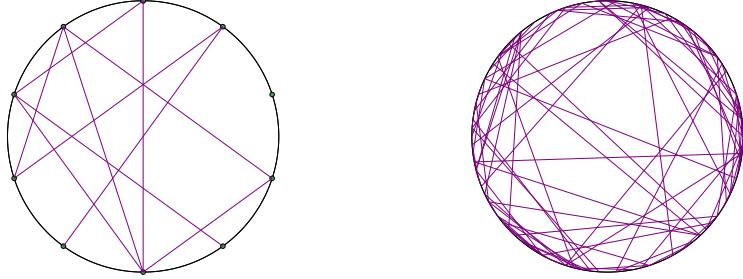


Figure 1: Partially connected graphs of different  $N$  and  $N_l$ , but same  $\sigma = 5$ . In the left panel,  $N = N_l = 10$ , and in the right panel  $N = N_l = 100$ .

of sites to interact<sup>1</sup>, any two pair of sites  $i, j$  is connected with a probability (without normalising) of  $r_{ij}^{-(d+\sigma)}$ . Figure 1 displays two examples of partially connected graphs with the same  $\sigma = 5$ , but  $N = N_l = 10$  for the left panel, and  $N = N_l = 100$  for the right panel. Both of these graphs present the same coordination number

$$z = 2 \frac{N_l}{N},$$

i.e. the average number of bonds (both in- and outgoing) for each node.

### 1.3 Set up

We consider a 1-dimensional spin-chain with periodic boundary conditions, so that the distance between any two nodes  $i, j$ , is the minimum distance along the circle that joins them,  $r_{ij} = \min(|i - j|, |i - j| - N)$

## 2 Algorithms

### 2.1 Generating the $J_{ij}$

To simulate the dilute bond approximation, we need an algorithm with which to generate the matrix  $J_{ij}$  with non-uniformly distributed non-zero entries. An alias algorithm proposed by Walker [1974] is used. This method allows one to sample integers from 1 to  $\alpha$  with a non-uniform probability distribution in  $\mathcal{O}(1)$  computation complexity with  $\mathcal{O}(N)$  memory complexity, by splitting the sampling into sampling a uniformly generated integer from 1 to  $\alpha$ , and then *tossing an unfair coin* with a certain probability, defined by Walker's alias algorithm.

**The connection set algorithm** We will only be interested in the indices  $i, j$  for which  $J_{ij}$  is non-zero. Furthermore, since  $J_{ij}$  is symmetric, we characterize connections by the (unordered) set data structure  $\{i, j\}$ . The drawn connections  $\{i, j\}$  are stored in another set `connectionSet`. Since sets are unordered, repeated connections do not change `connectionSet`, therefore the algorithm

---

<sup>1</sup>Thus,  $J_{ij}$  will have  $2N_l$  non-zero elements.

proceeds while `length(connectionSet) < N_1`. The algorithm to draw random connections is straight-forward: Draw an *uniformly* distributed integer from 1 to N to choose the first particle. The second site  $j$  is then chosen by generating a random integer  $m \in [1, N - 1]$ , where the probability  $P_m$  of sampling  $m$  is  $P_m = m^{-(1+\sigma)}$ .

The sampled connection is thus  $\{n, \text{ mod } (n + m, N)\}$ , where  $\{\}$  indicates the data structure `set`. The use of sets is not only motivated by convenience of the code, but they also present  $\mathcal{O}(1)$  lookup time in Julia, as it was tested by a user Kamiński [2022].

The **Julia** code used to generate the non-zero  $J_{ij}$  is found in Appendix A, and examples of diluted spin models with  $N = N_l$  and  $\sigma = 5$  are displayed in Figure 1, with  $N = 10$  on the left panel and  $N = 100$  on the right panel.

## 2.2 Cluster counting

Another useful quantity will be the identification of clusters, this is, all simply connected subsets of nodes, either directly or indirectly. This is done by a Depth First Search (DFS) algorithm, which is explained below. In order to apply this algorithm, the connections set must be given a new format so that the DFS algorithm can be easier applied. An array `connectionArray` of  $N$  entries is created, where each of its entries is an empty array. The array `connectionArray[i]` contains the indices of the nodes to which the node  $i$  is connected. An example of a `connectionArray` corresponding to a fully connected graph with  $N = 4$  is

```
connectionArray = [[2,3,4], [1,3,4], [1,2,4], [1,2,3]].
```

The DFS algorithm is in the worst case  $\mathcal{O}(V + E)$ , with V the number of vertices of the graph and E the number of edges (of each cluster). Two sets are defined

1. `clusters`: The identified clusters
2. `visited`: The visited nodes

Starting (arbitrarily) at the first node `n=1`. Initialize the array `stack` containing only said node, and an empty set `cluster`. Then,

1. While the stack is non-empty: pop the stack to the variable `curr = pop(stack)`,
2. If `curr` is not in `visited`, add it to `cluster`, and to the `visited` set.
3. Retrieve the „neighbors” of `curr` from `connectionArray[curr]`, and add them to the stack.
4. Repeat step 1.

Reaching an empty stack means there are not any more non-visited nodes in the current cluster, and therefore `cluster` is returned and added to the set `clusters`. This procedure is repeated for every node, if it is not in `visited`.

The corresponding **Julia** code can be found in A.2.

### 2.3 Ising model

Application of the Metropolis algorithm with the current set up is simple. Generate a random `spinArray` of length  $N$  containing only 1 or  $-1$ . Then,

1. Choose a site  $i$ ,
2. Calculate the energy change  $E$  corresponding to flipping the spin `spinArray[i]`,
3. Flip the spin at site  $i$  with the probability  $P_{ij}$  given by Equation (??). Now the neighboring spins are `connectionArray[i]`,
4. Repeat step 1.

However, care should be taken in the generation of the  $J_{ij}$ , as having a too low coordination number avoids the system from being almost fully connected, and prevents phase transitions.

## 3 Results

Figure 2 displays the change in number of clusters for different configurations, for a constant  $N$ , as a function of the number of bonds  $N_l$ . Different curves correspond to different  $\sigma$ , as indicated by the legend. The standard deviation of the sample mean is reported.

Figure 3 displays the size distribution of the clusters for different  $N_l$ , for fixed  $N$ . The three panels display the distributions for different values of  $\sigma$ . For visualization purposes, the curves are normalised, since there are less available clusters of length  $N$ , than there are clusters of length 1 (isolated sites), which leading to underrepresentation.

The order parameter of choice is the magnetization  $\mu = \sum s_i$  of the spin chain. Figure 4 shows the magnetization of the spin chain, as the Metropolis algorithm advances, with  $N = N_l = 4096, \sigma = 0.5$ . The top panel shows the magnetization for 50 different starting spin seeds, and the bottom panel presents the averaging of  $\mu^2$  across said spin seeds in blue. The curve  $y = 10^{-5.5}x^{1.7}$  is given in red, for reference.

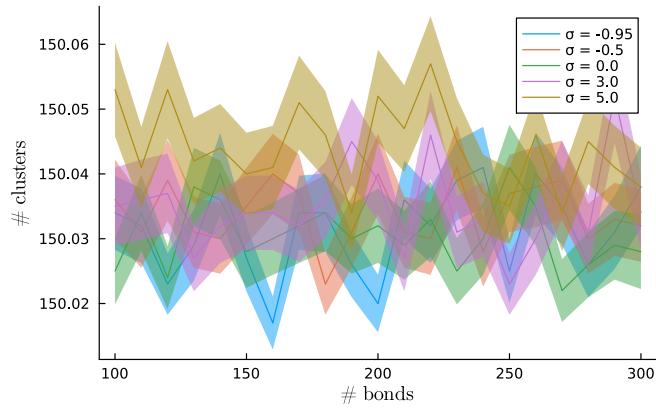


Figure 2: Average number of clusters and variation of the mean as a function of number of bonds, for  $N = 50$  and varying  $\sigma$ , as indicated by the legend.

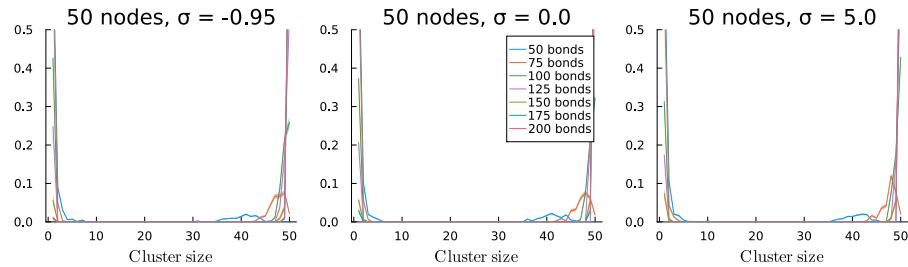


Figure 3: Normalized distributions of cluster sizes for fixed number of nodes  $N = 50$  and varying  $\sigma$  values, as indicated at the top of each panel, for different number of bonds among the nodes, as indicated in the legend of the middle panel.

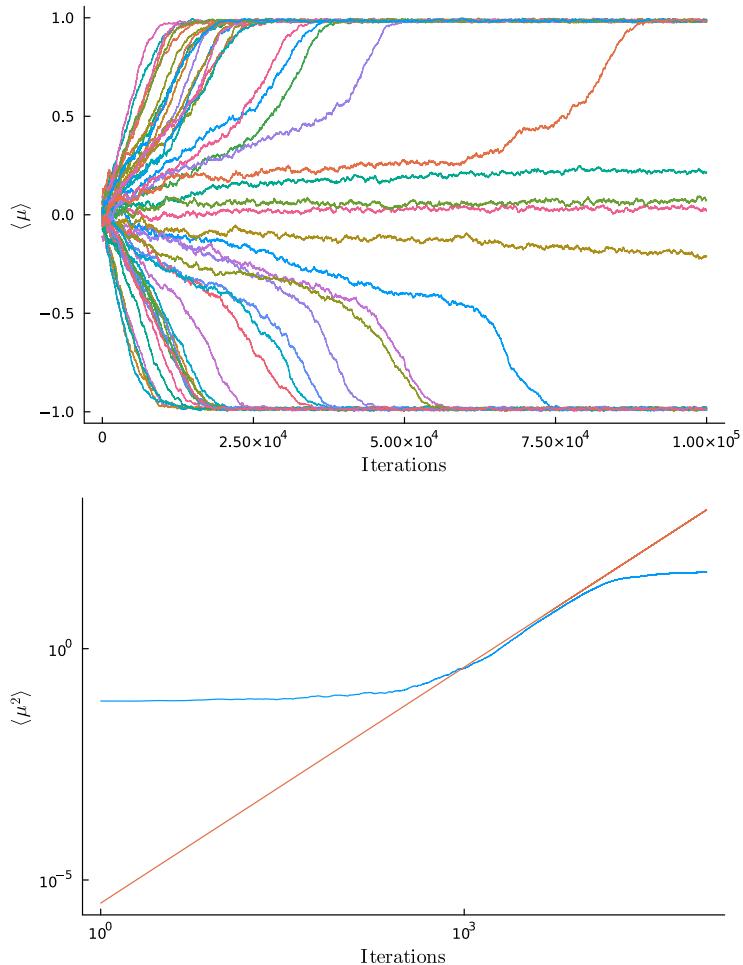


Figure 4: Magnetization  $\mu$  of the 1-dimensional spin chain with periodic boundary conditions, as iterations of the Metropolis advance, for different spin seeds and the same  $J_{ij}$ . In the top panel, the magnetization of the spin chain for different starting seeds. In the bottom panel, the magnetization squared, averaged across seeds. The curve  $y = 10^{-5.5}x^{1.7}$  is shown in red, for reference. Calculations here are done with  $N = N_l = 4096, \sigma = 0.5$ .

## A Code snippets

### A.1 Code for the generation of connection sets

```
using AliasTable

function generateConnectionsSet(N, N_l, sigma)

    distancesArray = 1:(N-1)
    chooseAT = AliasTable(ones(nPoints))
    distanceAliasTable = AliasTable(
        1 ./ distancesArray .^ (1+sigma)
    )

    # Stores connections
    connectionsSet = Set()

    # Stops if N_l is reached
    while len(connectionsSet) < N_l
        # Uniform distribution
        particle1Choice = rand(chooseAT, 1)[1]

        # Rolls integer m with probability P_m = m^(-(1+sigma))
        particle2Addition = rand(distanceAliasTable, 1)[1]
        particle2Choice = (particle2Addition + particle1Choice) % N

        # Stores the sampled connection. Since sets do not repeat elements
        # and are also unordered, if the connection already existed,
        # it will not be stored.
        push!(connectionsSet,
              Set([particle1Choice, particle2Choice]))
    end
    connectionsSet
end
```

### A.2 Code for the clustering algorithms

```
function clusterIdentification(connectivityArray)
    nPoints = length(connectivityArray)
    visited = Set{Int}()
    clusters = []

    function dfs(node, cluster)
        stack = [ node ]
        while !isempty(stack)
            curr = pop!(stack)
            if !(curr in visited)
```

```

        push!(visited, curr)
        push!(cluster, curr)
        for neighbor in connectivityArray[curr]
            if !(neighbor in visited)
                push!(stack, neighbor)
            end
        end
    end
end
cluster
end

for i in 1:nPoints
    if !(i in visited)
        cluster = Set{Int}()
        cluster = dfs(i, cluster)
        append!(clusters, [cluster])
    end
end
length(clusters)
end

```

## References

- E. Ising. Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik*, 31(1):253–258, 1925.
- B. Kamiński, 2022. URL "[https://www.juliabloggers.com/  
set-vs-vector-lookup-in-julia-a-closer-look/](https://www.juliabloggers.com/set-vs-vector-lookup-in-julia-a-closer-look/)".
- E. Luijten. *Introduction to Cluster Monte Carlo Algorithms*, pages 13–38. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-35273-0. doi: 10.1007/3-540-35273-2\_1. URL [https://doi.org/10.1007/3-540-35273-2\\_1](https://doi.org/10.1007/3-540-35273-2_1).
- N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 06 1953. ISSN 0021-9606. doi: 10.1063/1.1699114. URL <https://doi.org/10.1063/1.1699114>.
- R. H. Swendsen and J.-S. Wang. Nonuniversal critical dynamics in monte carlo simulations. *Phys. Rev. Lett.*, 58:86–88, Jan 1987. doi: 10.1103/PhysRevLett.58.86. URL <https://link.aps.org/doi/10.1103/PhysRevLett.58.86>.
- A. Walker. New fast method for generating discrete random numbers with arbitrary frequency distributions. *Electronics Letters*, 10:127–128, 1974. doi: 10.1049/el:19740097. URL [https://digital-library.theiet.org/  
doi/abs/10.1049/el%3A19740097](https://digital-library.theiet.org/doi/abs/10.1049/el%3A19740097).