Introductory Applications of Monte Carlo Methods in Social Physics

Joe Roberts

May 2024

Abstract

The Ising model and its conserved-order-parameter version are introduced and simulated using Metropolis, Glauber, and Kawasaki algorithms. We have then motivated their use within social physics, of which a brief history and a note on empirical data has been given. Through our simulations we find that low social temperature results in a polarised society regarding opinion, and a once unpopular opinion can be spread to the masses under an increase in social temperature. In addition, we investigated the relationship between pubic opinion and an external influence, the latter being modelled by an external magnetic field obeying the mean-field approximation. We found that the external influence has a categorical effect on public opinion; as well as identifying reason to believe (hysteresis) that humans have an inherent collective reluctance to new and opposing ideas. Next, self-organising segregation and integration have been simulated - segregation (coexisting phase) at low social temperature and integration (homogeneous phase) at high social temperature, deemed self-organising due to a lack of external magnetic field. Finally, we analysed the relationship between concentration and social temperature, concluding that the greater the concentration of one group within a population, the greater the social temperature needed for integration.

Non-technical Summary

Believe it or not, evidence suggests that the collective behaviour of humans is analogous to certain concepts within theoretical physics. This evidence comes from a field of research known as social physics. Social physics is wholly focused on explaining and predicting the behaviour of the masses and not the individual. This paper attempts to outline the simplest and most commonly used ideas within the field, identifying the equivalences between physical quantities and sociological quantities. From these equivalences, and by employing what are known as Monte Carlo methods, we simulated the following aspects of social systems: opinion dynamics and segregation/integration.

We introduced a theoretical model which proposes a grid of upward and downward pointing arrows, which interact in some way. These grids can be used to explain certain physical phenomena associated with magnets, by running the grids through Monte Carlo methods, as mentioned. These have very little to do with the place, other than the idea of *random numbers*.

Monte Carlo methods allow us to simulate the interactions between all the squares, and to compute various quantities of the grid (or "system"). Monte Carlo methods will simulate systems at certain temperatures, and at different temperatures we see drastically different behaviour from the system.

Before we go any further, let us clarify that the term "tolerance" is used as a numerical quantity in this paper; it should be assumed to have its normal definition, as well as a general initiative for integration within a society by its people.

Through our simulations we found that low tolerance results in a polarised society in respect to opinion, and a once unpopular opinion can be spread to the masses under an increase in tolerance. In addition, we investigated the relationship between pubic opinion and an external influence (the news, social media, and so on). We found that the external influence has a categorical effect on public opinion; as well as identifying reason to believe that humans have an inherent collective reluctance to new and opposing ideas.

Next, segregation and integration have been simulated - segregation resulting from low tolerance and integration resulting from high tolerance; deemed "self-organising" due to a lack of external influence. Finally, we analysed the relationship between concentration and tolerance. Here, we are defining concentration as the proportion of one demographic within a population. We conclude that the greater the concentration, the greater the tolerance needed for integration.

It should be noted that the specific numerical values used within the simulations and results are not to be taken too literally. One cannot really put a number on a whole society's "tolerance", not to mention the many seemingly ignorant oversights regarding the model's approach to the individual. We can see that these are completely irrelevant however: let us reiterate what was put forward at the start of this summary. Social physics is not attempting in any way to model the behaviour or psychology of the individual - it is the collective behaviour of a people that is of interest.

Contents

Introduction

Can the activity and psychology of humankind be understood through the lens of physics? Can we quantitatively explain the dynamics of opinions within a society? Is a one hundred year old model, intended to explain magnetisation, the key to explaining self-organising segregation? It is the interdisciplinary work of statistical physics and the social sciences, resulting in social physics, that attempts to answer these questions.

For general context, Monte Carlo methods use random sampling to simulate complex systems. (It is this randomness which causes them to bear the name of the Monaco district, and this is the only association the two have.) They are used when an analytical approach is unavailable or impractical; they are employed within an array of disciplines, ranging from theoretical physics, to finance, to the social sciences. Their main goal is to find the estimator of a quantity within the given system, without which one would not be able to attain much meaning from the simulations executed.

In the present work, they are applied to the 2D zero-field Ising model and its conserved-order-parameter counterpart, in order to demonstrate the critical phenomena, spontaneous magnetisation, and the loss of magnetisation of magnets. The Ising model was first seen in 1920; in its simplest form, it proposes a lattice of any positive dimension where each site can take a binary value of either "spin up" or "spin down" (or $+1$ and -1, red and blue, and so on). Crucially, these sites are said to be interacting and influencing each other in such a way that would cause them to be aligned. As will be shown, different quantities can be calculated for these lattices in different configurations, and are needed for a Monte Carlo simulation to operate.

We will explain in detail how three specific Monte Carlo algorithms (Metropolis, Glauber, and Kawasaki) are to be setup and executed. We will then outline the ways in which one can draw meaningful conclusions from the simulations.

Eventually, we will begin our discussion of social physics, beginning with a brief history of the field and a note on the empirical data validating it. We will then be able to come on to covering the main goal of this work: outlining the initial and commonly used ideas within social physics, as well as how one would go about simulating certain social phenomena. The two areas we will be focusing on are opinion dynamics and self-organising segregation.

Social physics does not claim to be able to predict anything about the individual - it is wholly focused on explaining and predicting the behaviour of the masses. After all, it is based within the framework of statistical physics; an area of theoretical physics using the interactions of many microscopic particles to explain the macroscopic behaviour of the larger, collective system.

Statistical Mechanics

It is through the lens of statistical mechanics that we can explain the macroscopic behaviour of large groups of microscopic particles.[1] Monte Carlo methods can then be used to simulate this behaviour[2,3] through random sampling. This chapter will attempt to establish the necessary principles in analysing large complex systems.

2.1 Introduction to Statistical Mechanics

Consider a system in thermal equilibrium with a thermal reservoir. This fixes the temperature T and conserves the number of degrees of freedom N (or "particle number"). This is known as a canonical ensemble, which itself is a type of statistical ensemble (a set containing all the states $\Omega = \{\omega_1, \omega_2, \omega_3, ...\}$ a system can adopt at any given time).

Let us define the *partition function* as

$$Z = \sum_i e^{-\beta H_i} \tag{2.1}$$

and the *occupation probability* (the probability that the system adopts a specific state) as

$$p_i = \frac{1}{Z} e^{-\beta H_i}, \tag{2.2}$$

where β and H denote the *inverse temperature* $(1/T)$ and the *Hamiltonian* $H = H(\sigma)$ respectively.

2.2 Monte Carlo Simulations in Thermal Equilibrium

Consider the above system with some observable quantity Q. The main goal of a Monte Carlo simulation is to find the expectation value $\langle Q \rangle$ of this quantity. This is formally achieved by averaging over all states, and has to be weighted by its Boltzmann probability.[3] This is only practical, however, to small systems, therefore with larger systems we calculate a best estimate of $\langle Q \rangle$, or the *estimator*[3] of $\langle Q \rangle$, which we define as

$$Q_M = \frac{\sum_{i=1}^{M} Q_{\mu_i} p_{\mu_i}^{-1} e^{-\beta H_{\mu_i}}}{\sum_{j=1}^{M} p_{\mu_j}^{-1} e^{-\beta H_{\mu_j}}} \tag{2.3}$$

for a system of M different states $\{\mu_i\}$, $i \in [1, M]$. Here p_μ is the probability distribution of the states; this p_μ being chosen by *importance sampling.*

Both sums in (2.3) will have the majority of their contributions from a limited number of different states (μ values) especially at low temperatures when the system is not excited into the higher states. Also, the number of possible states a system can be in is usually far too large for a computer to sum over.[3] It is because of this that we assume that the values p_{μ_i} follow the *Boltzmann Probability Distribution*,[3] which has the same form as the *occupation probability* (2.2) defined above, therefore

$$p_{\mu_i} = \frac{1}{Z} e^{-\beta H_{\mu_i}}. \tag{2.4}$$

It then follows that our *estimator* Q_M takes the form[3]

$$Q_M = \frac{1}{M} \sum_{i=1}^{M} Q_{\mu_i}. \tag{2.5}$$

Markov processes

The set of states $\{\mu_i\}$ is created mainly by Markov processes. This is why the set of states can be referred to as a *Markov chain.* I will be denoting the current state of the system as μ and the new state as ν. A Markov process will take μ and randomly generate ν, with the probability that the new state will be ν given its current state is μ being defined as the *transition probability*,[3] $P(\mu \to \nu)$, such that

$$\sum_{\nu} P(\mu \to \nu) = 1. \tag{2.6}$$

This requirement comes from the fact that the process will always create a ν when given a system in μ.

This *transition probability* is independent of time and independent of any state prior to μ. There are two final requirements for these Markov processes, namely *ergodicity* and *balance*, motivated by the fact that we need the states $\{\mu_i\}$ to exist with probabilities aligned with the Boltzmann distribution. This is because a system, in reality, at thermal equilibrium, exhibits these same qualities.

The ergodicity condition demands that the system, at some point in time, will have reached all $\{\nu_i\}$ from each and every $\{\mu_i\}$. If this were not a requirement, then it would be possible for the system to start in a μ_i but then not eventually reach a Boltzmann distribution.

The balance condition ensures thermal equilibrium. It essentially demands that, while a system can freely move from μ_i to ν_i as much as it pleases, this needs to be reciprocated (or "balanced" - hence the name) by transitions from ν_i to μ_i - resulting in the following expression:[3]

$$\sum_\nu p_\mu P(\mu \to \nu) = \sum_\nu p_\nu P(\nu \to \mu), \tag{2.7}$$

which, due to (2.6), can also be written as

$$p_\mu = \sum_\nu p_\nu P(\nu \to \mu). \tag{2.8}$$

Finally, as previously mentioned, we need the states $\{\mu_i\}$ to exist with probabilities aligned with the Boltzmann distribution; this can be enforced by letting the transition probabilities satisfy the following equation:[3]

$$\frac{P(\mu \to \nu)}{P(\nu \to \mu)} = \frac{p_\nu}{p_\mu} = e^{-\beta \Delta E}. \tag{2.9}$$

Here we have let $H_\nu - H_\mu = \Delta E$.

Acceptance and selection probabilities

The relations (2.8) and (2.9) can be satisfied in many ways by different choices of transition probabilities. The main distinction between different Monte Carlo algorithms lies in this choice.

Therefore, what we are tasked with is to define a certain set of Markov processes that will correlate to our choice of transition probabilities. This may be difficult and somewhat inexact, motivating the use of *acceptance probabilities* (denoted by $A(\mu \to \nu)$).[3] These allow us to define the set of Markov processes in any way, while still obtaining a desired set of transition probabilities. Simply put, they represent the probability that a system (defined as in section 2.1) will "accept" this new state ν created by the Markov process, given an original state μ, and adopt its form.

Let us also define *selection probabilities* (denoted by $g(\mu \to \nu)$)[3] as the probability that, when given a system in μ, the Markov process will create a new "target"[3] state ν. (For absolute clarity, one should not see this as the probability that a new state will be created, as opposed to a new state *not* being created; it is the probability that a new state, specifically ν_i, will be created, as opposed to any other new state from the set of all possible new states $\{\nu_i\}$.)

One should also note at this stage that the transition probability for a system to stay in the same state during a Markov process (that is $P(\mu \to \mu)$) can be non-zero. This may sound contradictory to the previous statement that, when given a system in μ, the process will always create a ν. However, this logic is perfectly sound as the ν generated is simply one such that $\nu = \mu$.

One should be able to see that, in a purely hypothetical situation, if

$$A(\mu \to \nu) = 0 \quad \text{for all } \nu \neq \mu, \tag{2.10}$$

then

$$P(\mu \to \mu) = 1. \tag{2.11}$$

This is because if the system never accepts/adopts a new state, then it will be *guaranteed* to stay in the same state indefinitely.

Therefore, it can be seen that by adjusting the values taken by the "stay-at-home" transition probability,[3] $P(\mu \to \mu)$, we directly influence the value of $A(\mu \to \nu)$.

Circling back to our original goal, we can express these transition probabilities as a combination of acceptance and selection probabilities,[3] such that

$$P(\mu \to \nu) = g(\mu \to \nu)A(\mu \to \nu). \tag{2.12}$$

Substituting this into the constraint that is equation (2.9) gives

$$\frac{P(\mu \to \nu)}{P(\nu \to \mu)} = \frac{g(\mu \to \nu)A(\mu \to \nu)}{g(\nu \to \mu)A(\nu \to \mu)}. \tag{2.13}$$

In summary then, when carrying out a Monte Carlo algorithm, one designs a Markov process with chosen $g(\mu \to \nu)$ and $A(\mu \to \nu)$, such that (2.9) is satisfied. These make up the desired transition probabilities, because by satisfying (2.9) (and therefore (2.13)), we end up with a set of states $\{\mu_i\}$ that exist with probabilities aligned with the Boltzmann distribution. We employ this general procedure in a later section to define each algorithm.

Finally, one should note that we want the acceptance probabilities to be as large as possible - of course while staying within $[0, 1]$. We can see from equations (2.10) and (2.11), that there is inverse proportionality between $A(\mu \to \nu)$ and $P(\mu \to \mu)$, and therefore the reason for wanting $A(\mu \to \nu)$ to be large is that we do not want a system that can easily stay in the same state after each iteration of our Monte Carlo algorithm, that is $P(\mu \to \mu) = 0$ or near zero.

Essentially, we can set one of the acceptance probabilities in (2.13) to be one - indicating that state changes will always be accepted in that particular direction (but not that the system itself will always go from μ to ν, because of the involvement of selection probabilities) - and then let the other acceptance probability be whatever then satisfies (2.13) and (2.9).

2.3 The Ising Model

The Ising model is one of the simplest models in statistical mechanics. It proposes a lattice (or grid) of arrows each representing an atom inside a ferromagnetic material (a material that can become magnetised without an external magnetic field).

The original aim of the Ising model was to understand why magnets behave the way they do when heated; it explains the loss of magnetisation as the system surpasses a *critical temperature*, and the reason for spontaneous magnetisation at low temperatures.[4]

Models can be one dimensional (a chain), two dimensional (a lattice as mentioned), or in fact any $d \in \mathbb{Z}^+$, where d specifies the dimensionality of the lattice. Each arrow can take one of two orientations, (hence one can refer to each site as a "dipole") namely "spin up" or "spin down". Therefore we can represent the set of spins as:

$$\sigma = \{\sigma_k\}, \quad \text{where } \sigma_k \in \{+1, -1\}. \tag{2.14}$$

Here we denote spins parallel to the magnetic field as $\sigma_k = 1$ and spins anti-parallel to the magnetic field as $\sigma_k = -1$.

We define any one *micro-state* (or "state") to be particular combination or pattern of spins that a system has adopted. The set of micro-states is denoted as

$$\Omega = \{\omega_i\}, \quad \text{where } i = 0, 1, 2, ..., M. \tag{2.15}$$

Here M is the total number of micro-states. Finally, we let N be the total number of spins (or "sites") on the lattice, resulting in the following equation:[5]

$$M = 2^N \tag{2.16}$$

Project specifics

In this project we will be considering a $d = 2$ dimensional L by L square lattice Ising model with, unless specified otherwise, a $B = 0$ magnetic field (also known as a "zero-field").

Note that it is an implicit assumption within Ising models that the all the spins on the lattice *interact*. We take any spin on the lattice to "prefer"[6] that its neighbours be of the same spin alignment, due to the sign convention of our coupling coefficient J.

As we are working with a finite lattice, it is also important to give special consideration to the boundary sites. We assume that the system adopts *periodic boundary conditions*: that is, a site in the far right column of the lattice will interact with, and influence, a site on the same row in the far left column, and vice versa. This also applies to sites on the very top row and the very bottom row in exactly the same way.

When discussing and implementing Monte Carlo algorithms, it is necessary to evaluate the *change* in a quantity, resulting from a single-spin-flip; in addition to calculating the quantity itself. We do exactly this in the next section, and the reasons for this will be discussed later.

2.3.1 Energy

For the situation described above, the internal energy of the system is fully described by the *Hamiltonian*, $E(\sigma)$.[1]

$$E(\sigma) = -\sum_{\langle jk \rangle} J_{jk}\sigma_j\sigma_k - \sum_n B_n\sigma_n \tag{2.17}$$

where $J \in \mathbb{R}$ is a proportionality factor (representing the strength of the interaction between nearest neighbour spins on the lattice) and $B \in \mathbb{R}$ is the external magnetic field.

We will be making the assumption that the *coupling coefficient* $J = 1$ throughout the lattice, and therefore

$$E(\sigma) = -\sum_{\langle jk \rangle} \sigma_j \sigma_k - \sum_n B_n \sigma_n. \tag{2.18}$$

It is important to note here that the subscript $\langle jk \rangle$ indicates that we are only summing over the "nearest neighbours" of site j; in other words the four individual sites directly above, below, to the left, and to the right of site j.

As mentioned, we will be computing the change in each quantity after a single-spin-flip. Splitting up each of the sums in (2.18), we can write

$$E_\mu = -\sigma_j \sum_{k \in \langle jk \rangle} \sigma_k - \sum_{\langle mk \rangle \neq j} \sigma_m \sigma_k - \sum_n B_n (\sigma_j + \sum_{k \neq j} \sigma_k). \tag{2.19}$$

Now consider the spin flip $\sigma_j \to -\sigma_j$ which characterises a change of state $\mu \to \nu$. Our energy is now

$$E_\nu = \sigma_j \sum_{k \in \langle jk \rangle} \sigma_k - \sum_{\langle mk \rangle \neq j} \sigma_m \sigma_k - \sum_n B_n (-\sigma_j + \sum_{k \neq j} \sigma_k). \tag{2.20}$$

The change in energy then takes the form

$$\begin{aligned}
\Delta E &= E_\nu - E_\mu \\
&= \sigma_j \sum_{k \in \langle jk \rangle} \sigma_k - \sum_{\langle mk \rangle \neq j} \sigma_m \sigma_k - \sum_n B_n (-\sigma_j + \sum_{k \neq j} \sigma_k) \\
&+ \sigma_j \sum_{k \in \langle jk \rangle} \sigma_k + \sum_{\langle mk \rangle \neq j} \sigma_m \sigma_k + \sum_n B_n (\sigma_j + \sum_{k \neq j} \sigma_k) \\
&= 2\sigma_j \sum_{k \in \langle jk \rangle} \sigma_k + 2\sum_n B_n \sigma_j - \sum_n B_n \sum_{k \neq j} \sigma_k + \sum_n B_n \sum_{k \neq j} \sigma_k \\
&= 2\sigma_j (\sum_n B_n + \sum_{k \in \langle jk \rangle} \sigma_k). \qquad (2.21)
\end{aligned}$$

As mentioned above, we will be focusing (mostly) on the *zero-field Ising model*. In such a case, we obtain

$$\Delta E = 2\sigma_j \sum_{k \in \langle jk \rangle} \sigma_k. \tag{2.22}$$

However, there may be a time when we need to account for an external magnetic field applied to the system - we can do this via the *mean-field approximation*.

2.3.2 Magnetisation

The magnetisation of a whole system in state μ is simply the sum of all the sites' spins:[3]

$$M_\mu = \sum_i \sigma_i^\mu. \tag{2.23}$$

Then the change in magnetisation caused by a single-spin-flip will take the form:

$$\Delta M = M_\nu - M_\mu = \sum_i \sigma_i^\nu - \sum_i \sigma_i^\mu \tag{2.24}$$

We can see that, due to single-spin-flip-dynamics, the only terms that would not cancel are the terms in reference to the site being flipped; that is, site j. Therefore, it follows that

$$\begin{aligned}\Delta M &= \sum_i \sigma_i^\nu - \sum_i \sigma_i^\mu \\ &= \sigma_j^\nu - \sigma_j^\mu \quad \text{where } \sigma_j^\nu = -\sigma_j^\mu \implies \sigma_j^\mu = -\sigma_j^\nu \\ \Delta M &= 2\sigma_j^\nu. \end{aligned} \tag{2.25}$$

It will become clear later (when evaluating estimators and other derived quantities) as to why it is pertinent to establish an expression for the magnetisation per site, not of the whole system, as is M_μ. For now though, we simply posit that m is

$$m = \frac{M_\mu}{N}, \quad \text{N being the number of sites on the lattice.} \tag{2.26}$$

Mean field approximation

Recall (2.26)

$$\begin{aligned} m &= \frac{M_\mu}{N} \\ \implies m &= \frac{1}{N}\sum_n \sigma_n \end{aligned} \tag{2.27}$$

and our general expression for the energy of the Ising model (2.18)

$$E(\sigma) = -\sum_{\langle jk\rangle} \sigma_j\sigma_k - \sum_n B_n\sigma_n \tag{2.28}$$

$$= -\sum_{\langle jk\rangle} \sigma_j\sigma_k - B\sum_n \sigma_n, \quad \text{for constant magnetic field.} \tag{2.29}$$

We then impose that

$$\langle m\rangle = \frac{1}{N}\sum_n \langle\sigma_n\rangle. \tag{2.30}$$

Using this, we expand the interacting spins term $\sigma_j\sigma_k$, such that

$$\sigma_j\sigma_k = [(\sigma_j - \langle m\rangle) + \langle m\rangle][(\sigma_k - \langle m\rangle) + \langle m\rangle] \tag{2.31}$$

$$\implies \sigma_j\sigma_k = (\sigma_j - \langle m\rangle)(\sigma_k - \langle m\rangle) + (\sigma_j + \sigma_k)\langle m\rangle - \langle m\rangle^2. \tag{2.32}$$

Due to assertion that the system contains, on average, *small* fluctuations around the value $\langle m\rangle$, we can ignore (2.32)'s first term. Combining with (2.18) brings us the following.[5]

$$\sigma_j\sigma_k = (\sigma_j + \sigma_k)\langle m\rangle - \langle m\rangle^2 \tag{2.33}$$

$$\implies E(\sigma) = -\sum_{\langle jk\rangle}((\sigma_j + \sigma_k)\langle m\rangle - \langle m\rangle^2) - B\sum_n \sigma_n \tag{2.34}$$

$$= -\sum_{\langle jk\rangle}(\sigma_j + \sigma_k)\langle m\rangle + \sum_{\langle jk\rangle}\langle m\rangle^2 - B\sum_n \sigma_n \tag{2.35}$$

$$\implies E_{MF} = 2N\langle m\rangle^2 - (4\langle m\rangle + B)\sum_n \sigma_n \tag{2.36}$$

We have been able to get from the penultimate line of work to the last by the following logic: in a 2D lattice the sum over nearest neighbours $\langle jk\rangle$ counts $\langle m\rangle^2$, $4N$ times; and in the sum of $(\sigma_j + \sigma_k)$ over nearest neighbours, each spin will appear 4 times.

Let us define the *effective magnetic field* B_{eff} as

$$B_{\text{eff}} = (4\langle m\rangle + B). \tag{2.37}$$

As always, we will need an expression for the change in energy caused by the *flipping* of σ_j; we declare that this satisfies[5]

$$\Delta E = 2B_{\text{eff}}\sigma_j. \tag{2.38}$$

2.3.3 Phase Transitions

Phase transitions are a key aspect of many areas of scientific research today.[2] This section attempts to provide a brief, qualitative explanation of phase transitions and their consequences.

Phase transitions describe the transition between a *disordered phase* and an *ordered phase*. Let us define the disordered phase of a system to be when there is an approximately equal number of sites with $\sigma = +1$ as there are sites with $\sigma = -1$. If one were to assume an infinite lattice where $N \to \infty$ (in other words, the *thermodynamic limit*) then the overall magnetisation of the system would approach zero.

Let us define the ordered phase of a system to be when most, if not all of the sites on the lattice, are aligned with the same spin. In the thermodynamic limit, this manifests itself as the magnitude of the estimator of each site's magnetisation approaching 1. That is,

$$\textbf{Disordered phase:} \quad M_\mu \to 0 \quad \text{as } N \to \infty \implies \langle m\rangle \to 0. \tag{2.39}$$

$$\textbf{Ordered phase:} \quad |\langle m\rangle| \to 1 \quad \text{as } N \to \infty. \tag{2.40}$$

The disordered phase occurs at temperatures above the *critical temperature*, T_c, and the ordered phase occurs below it. We define the critical temperature to be the temperature at which the phase transition occurs.

Let us now define *control parameters* and *order parameters*. Control parameters are the parameters of the system that are deliberately changed, whereas order parameters are derived from the derivative of the *free energy*[5]. In our particular system, the control parameter is temperature and the order parameter is magnetisation. Energy is simply a useful observable quantity.

As one can see from (2.39) and (2.40), the order parameters of a system (in the thermodynamic limit) are zero in the disordered phase and non-zero in the ordered phase.

One should also note here that it is possible for a system (with more than one order parameter) to undergo more than one phase transition at different critical temperatures. This is because each phase transition will be characterised by the evolution (over temperature) of one order parameter.

At temperatures near the critical temperature, we find large areas of aligned spins; we will refer to these as *domains*[6,7] or *clusters*.[3] These clusters greatly influence the values of the order parameters, and hence when they are flipped there are noticeable fluctuations in the quantities. These are known as *critical fluctuations*.[3]

More formally, we can define[5] a *phase transition* to be a divergence or discontinuity in the order parameter (or a derivative of the order parameter) for a system in the thermodynamic limit. A *first-order* phase transition is one in which the actual order parameter diverges, and a *second-order* phase transition is one in which its derivative (with respect to the control parameter) diverges.

We can show that our specific system, a 2D zero-field Ising model, has a second-order phase transition through analysis of the *magnetic susceptibility*. We define the magnetic susceptibility to be[3]

$$\chi = \beta N(\langle m^2 \rangle - \langle m \rangle^2). \tag{2.41}$$

This is the equation that our algorithms used to compute the magnetic susceptibility.

Later, we plot graphs of χ vs T for various algorithms. If these graphs involve an abrupt peak in χ (which is discontinuity in the derivative of an order parameter) we can conclude that the 2D zero-field Ising model has a second-order phase transition.

Moreover, we can again show there is a second-order phase transition in our particular system through analysis of the *specific heat*. We define the specific heat c of the system to be

$$c = \frac{dE}{dT}. \tag{2.42}$$

The specific heat *per site* per site can also be computed in the following form:[3]

$$c = \beta^2 N(\langle \epsilon^2 \rangle - \langle \epsilon \rangle^2) \tag{2.43}$$

where ϵ denotes the energy per lattice site, such that

$$\epsilon = \frac{E}{N}. \tag{2.44}$$

(2.43) is the equation that our algorithms actually used to evaluate the specific heat. It can be shown that (2.43) is equivalent to (2.42)[3,5].

Later, we plot graphs of c vs T for various algorithms. If these graphs involve an abrupt peak in c we can conclude that the 2D zero-field Ising model has a phase transition, as phase transitions can also - more generally - be characterised by an abrupt change in observable quantities.

Exact results

Finally, it should be noted[1] that Ising models of one or two dimensions are known to have exact, analytic solutions. That is, the critical temperature of the two-dimensional case is a known value (the one-dimensional case does not undergo a phase transition and therefore does not have a "critical temperature" so to speak).

The one-dimensional case was solved by Ernst Ising in 1925 (and was published in his doctoral thesis) hence the model bears his name. It was not Ising however, who initially came up with the general model: it was in fact his doctoral advisor Wilhelm Lenz who first proposed an arrangement of spins on a lattice of dimension $d \in \mathbb{Z}^+$. It should be noted here that, by "solved", we mean finding exact solutions to the partition function and various other derived thermodynamic quantities.

It was not until 1944 when Lars Onsager exactly solved the two-dimensional case. Therefore, the critical temperature of the 2D zero-field Ising model is known.

$$\textbf{Exact Critical Temperature:} \qquad T_c = \frac{2}{\ln(1+\sqrt{2})} \approx 2.269 \tag{2.45}$$

Critical exponents and finite-size scaling

There are more quantitative ways of describing the behaviour of a system near its critical point which we should mention. Quantities like specific heat and magnetic susceptibility will follow power-law behaviour with *critical exponents*. Let us first define the *reduced temperature* t of a system to be[3]

$$t = \frac{T - T_c}{T_c}. \tag{2.46}$$

The divergence of specific heat and magnetic susceptibility in the thermodynamic limit can then be described with two different critical exponents of this reduced temperature, like so.[3]

$$c \sim |t|^{-\alpha} \qquad\qquad \chi \sim |t|^{-\gamma} \tag{2.47}$$

There are many techniques to measure these critical exponents but we will only mention *finite-size scaling.*[3] This method can be used for the 2D Ising model because the critical temperature is known exactly.

To explain this method let us first vaguely define the *correlation time* τ to be the time taken (during a Monte Carlo simulation) for the system to get from one state to a state somewhat unrecognisable from the first. In other words, a state which is no more similar to the first state than some other random state. Often it is assumed to be equal to the *equilibration time* - the time taken for a system to reach equilibrium (measured in Monte Carlo "sweeps" as will be seen later).

Let us also define the *correlation length*[3] to be the average size of the domains formed by a 2D Ising model at equilibrium. We can write correlation time and length in power-law form, and then join the two equations to obtain an expression for correlation time as a function of correlation length.

$$\tau \sim |t|^{-z\nu}, \quad \xi \sim |t|^{-\nu} \tag{2.48}$$

$$\implies \tau \sim \xi^{z} \tag{2.49}$$

Consider a graph of temperature vs correlation length. In the thermodynamic limit, the correlation length will diverge at the critical temperature. We would see a similar shape to this divergence for a finite lattice until we got to the size of one side of the square lattice. This is fairly simple when considered, as the correlation length (the size of the domains) cannot possibly exceed the size of the finite lattice. (2.49) then becomes

$$\tau \sim L^{z}. \tag{2.50}$$

Finally, as the critical temperature of this system is known, we can carry out multiple simulations at different lattice sizes all at this temperature T_c, while measuring correlation time. This should then give us all the information we need to calculate the value of z, the *dynamic exponent.* This is the method of finite-size scaling.

We have now seen how finite-size scaling and critical exponents can give numerical explanations and analysis to critical phenomena.

2.4 The COP Ising Model

The COP Ising model (*conserved order parameter*) is very similar to the Ising model described above, apart from one key difference: the order parameter, in our case magnetisation, is conserved during any thermodynamic simulation. This is achieved through the *exchanging* of spins as opposed to the flipping, hence why the Kawasaki algorithm (which we will come on to) is sometimes called "spin-exchange sampling"[2].

The reason we want to exchange instead of flip is linked to our definition of a system's magnetisation,

$$M_{\mu} = \sum_{i} \sigma_{i}^{\mu}.$$

If we never change any one site's spin, the total sum of all spins should never change.

Another reason to exchange spins is that of real world applications. If you were to model the motion of gas particles in a lattice gas, then it makes much more sense to say that they have swapped places, as opposed to saying they have had their "spin" flipped; as in this situation and a lot of other situations, the term *spin* does not really apply, and is simply used as a differentiator between two types of particles.

So to recap, we still have a set of spins on a 2D lattice which can be represented by:

$$\sigma = \{\sigma_i\}, \quad \text{where } \sigma_i \in \{+1, -1\}.$$

The set of micro-states is denoted as

$$\Omega = \{\omega_i\}, \quad \text{where } i = 0, 1, 2, ..., M$$

where M is the total number of micro-states; and we let N be the total number of spins on the lattice. Due to the conserved-order-parameter nature of our system, we now have a different form of the relationship between M and N:[5]

$$M = \binom{N}{n_+} \tag{2.51}$$

We will still be working with a finite zero-field model, which employs periodic boundary conditions.

2.4.1 Energy

One important consequence of the conserved order parameter is that, our ΔE will take a different form. We will now derive this.

For the Hamiltonian, we will straight away be asserting a zero-field, and obtain the following form:

$$E(\sigma) = -\sum_{\langle kj \rangle} \sigma_j \sigma_k. \tag{2.52}$$

This will step through all "j" sites, each time multiplying by the sum over its four nearest neighbours, the σ_k's. Therefore the expression above can be thought of as

$$E_\mu = -(\sigma_1 \sum_{\langle p1 \rangle} \sigma_p + \sigma_2 \sum_{\langle q2 \rangle} \sigma_q + \sigma_3 \sum_{\langle r3 \rangle} \sigma_r + \sigma_4 \sum_{\langle t4 \rangle} \sigma_t + \cdots + \sigma_j \sum_{\langle kj \rangle} \sigma_k). \tag{2.53}$$

where $p \neq q \neq r \neq t$.

Let us now choose two spins at random, say σ_1 and σ_3. One should note that flipping both of these spins individually is equivalent to exchanging them, as we will be attempting to model in later sections. Our system's Hamiltonian now takes the form

$$E_\nu = -(-\sigma_1 \sum_{\langle p1 \rangle} \sigma_p + \sigma_2 \sum_{\langle q2 \rangle} \sigma_q - \sigma_3 \sum_{\langle r3 \rangle} \sigma_r + \sigma_4 \sum_{\langle t4 \rangle} \sigma_t + \cdots + \sigma_j \sum_{\langle kj \rangle} \sigma_k). \tag{2.54}$$

Now let us compute the change in energy going from state μ to ν:

$$\begin{aligned}\Delta E &= E_\nu - E_\mu \\ &= -[-\sigma_1 \sum_{\langle p1\rangle} \sigma_p + \sigma_2 \sum_{\langle q2\rangle} \sigma_q - \sigma_3 \sum_{\langle r3\rangle} \sigma_r + \cdots + \sigma_j \sum_{\langle kj\rangle} \sigma_k] \\ &+ [\sigma_1 \sum_{\langle p1\rangle} \sigma_p + \sigma_2 \sum_{\langle q2\rangle} \sigma_q + \sigma_3 \sum_{\langle r3\rangle} \sigma_r + \cdots + \sigma_j \sum_{\langle kj\rangle} \sigma_k] \\ &= -[-\sigma_1 \sum_{\langle p1\rangle} \sigma_p - \sigma_3 \sum_{\langle r3\rangle} \sigma_r] + [\sigma_1 \sum_{\langle p1\rangle} \sigma_p + \sigma_3 \sum_{\langle r3\rangle} \sigma_r] \\ &= 2[\sigma_1 \sum_{\langle p1\rangle} \sigma_p + \sigma_3 \sum_{\langle r3\rangle} \sigma_r]. \end{aligned} \tag{2.55}$$

Given that σ_1 and σ_3 were chosen at random, we can rewrite them as σ_i and σ_j, where sites i and j are two sites chosen at random. Finally, our equation for the change in energy of a 2D COP zero-field Ising model, after a spin-exchange, takes the form

$$\Delta E = 2[\sigma_i \sum_{\langle pi\rangle} \sigma_p + \sigma_j \sum_{\langle qj\rangle} \sigma_q]. \tag{2.56}$$

2.4.2 Phase Transitions

Another key result of the conserved magnetisation is the different behaviour we see in phase transitions. When taking a system from a low temperature to a high temperature, passing through the critical temperature, we do not see an ordered phase transition into a disordered phase. Instead, we go from what is called a *coexisting phase* to a *homogeneous phase.* The coexisting phase involves multiple differently aligned ordered phases, in other words, domains. The homogeneous phase can be thought of in a similar vein to the disordered phase. To explain the reasons for this and to simulate this in an algorithm, we must first take a step back and consider again our model - then acquire some new mathematical tools.

Consider for a moment that only a fraction of the sites on the lattice above actually contain particles. We will let this fraction be denoted as ρ, the *particle density.*[3]

Next, let us define *occupation number* as the number of particles occupying a single site on the lattice:

$$n = \{n_i\}, \quad \text{where } n_i \in \{0, 1\}. \tag{2.57}$$

Here, $n_i = 1$ denotes a site occupied by a particle and $n = 0$ denotes a site that is not occupied by a particle. The requirement of $n_i \in \{0, 1\}$ is applied so that the system cannot have more than one particle occupying a site. This is to replicate and enforce the *Pauli exclusion principle.*

By requiring that the particle density be *constant*, the following equation is satisfied.

$$\sum_i^N n_i = \rho N \tag{2.58}$$

Next, let us write our spins σ_i as a function of the occupation numbers, such that

$$\sigma_i = 2n_i - 1. \tag{2.59}$$

One can see that, through the two allowed values of n_i, we arrive at the same values of σ_i as before:

$$n_i = 0 \implies \sigma_i = 2(0) - 1 = -1 \tag{2.60}$$

$$n_i = 1 \implies \sigma_i = 2(1) - 1 = 1. \tag{2.61}$$

Therefore we have now arrived at a set of variables that indicate a site to have a spin value of $+1$ when it is occupied by a particle, and -1 when it is not.[3] At first glance, this may seem strange. How can a site without a particle have a non-zero value of spin? How can *nothing* have spin?

One must recall what was established at the start of this section; in other words, the term "spin" should not be taken as literally as it is usually meant in the world of physics - it is simply used to distinguish between two sites on a lattice differing in some specific way.

Rearranging (2.59) for the occupation number, we arrive at

$$n_i = \frac{1}{2}(\sigma_i + 1). \tag{2.62}$$

Substituting back into (2.58), and recalling that the total magnetisation of the lattice is given by $M_\mu = \sum_i^N \sigma_i$, we find

$$\begin{aligned} \rho N = \sum_i^N \frac{1}{2}(\sigma_i + 1) &\implies 2\rho N = \sum_i^N \sigma_i + N \\ &\implies N(2\rho - 1) = M_\mu \\ &\implies 2\rho - 1 = m \\ &\implies \rho = \frac{1}{2}(m + 1). \end{aligned} \tag{2.63}$$

Here we have also used the fact that magnetisation per site is given by $m = \frac{M_\mu}{N}$.

We have now arrived at an expression for particle density as a function of magnetisation (per site). This therefore gives us the ability to fix the magnetisation by fixing the particle density.

Let us now return to the discussion of phase transitions in COP Ising models. We have already seen in (2.39) that, for an Ising model in the thermodynamic limit, the disordered phase has an average magnetisation per site approaching zero. We can say the same for the homogeneous phase:

$$\textbf{Homogeneous phase:} \qquad M_\mu \to 0 \quad \text{as } N \to \infty \implies \langle m \rangle \to 0. \tag{2.64}$$

Substituting this back into our expression for particle density in a COP Ising model, (2.63), we find

$$\rho \to \frac{1}{2}(0+1) \implies \rho = \frac{1}{2} \quad \text{for } N \to \infty. \tag{2.65}$$

This simply means that half of our spins will have spin $\sigma = +1$ and the other half will have spin $\sigma = -1$ for a model with a zero-field. This is fairly trivial and self-explanatory. However, this logic becomes more useful when looking at the coexisting phase.

Again, consider again a normal Ising model in the thermodynamic limit. This time in the ordered phase, (2.40). We see that the absolute value of average magnetisation approaches one. However, due to the conserved magnetisation within the COP Ising model, we find a different result than (2.40) for the coexisting phase:

$$\textbf{Coexisting phase:} \qquad |\langle m \rangle| \to 0 \quad \text{as } N \to \infty. \tag{2.66}$$

Through (2.63), this leads to particles densities of

$$\begin{aligned} |m| = 1 &\implies m_+ = 0, m_- = 0 \\ &\implies \rho_+ = \frac{1}{2}(m_+ + 1) = \frac{1}{2} &(2.67) \\ &\implies \rho_- = \frac{1}{2}(m_- + 1) = \frac{1}{2}. &(2.68) \end{aligned}$$

It would seem we have come to the same conclusion as for the homogeneous case. So, have we gone wrong somewhere? No. All we have done is discover that, for a COP Ising model, the thermodynamic limit is meaningless to us. Given that no real world system is in the thermodynamic limit (as there are not infinite particles in the observable universe,[5]) this shouldn't be an issue.

What the logic above *does* tell us, is that the particle densities of the domains formed below T_c, fall somewhere between 0 and 1. That is,

$$\rho_+, \rho_- \in [0,1] \quad \text{where } \rho_+ > \rho_-. \tag{2.69}$$

We call this phenomenon *phase separation*,[2] because we end up with the lattice *separating* into multiple regions of a certain density ρ_+ or ρ_-. These domains do however *coexist*, hence we refer to the phase involving phase separation as the coexisting phase.

2.5 Metropolis, Glauber, and Kawasaki

The Metropolis, Glauber, and Kawasaki algorithms are specific types of Monte Carlo algorithms. Recall from section 2.1 that it is the specific choice of transition probabilities - which is determined by the choice of acceptance probabilities and selection probabilities - that distinguishes one Monte Carlo algorithm from another. Another key differentiator, which will be discussed in this section, is the selection and state-change mechanism that the algorithm carries out on each iteration.

This section will include an initial implementation of each algorithm on the Ising model described above.

Note on single-spin-flip dynamics

For each algorithm we will be using *single-spin-flip dynamics*, which means, for each iteration of the algorithm, only one dipole can have its orientation flipped. (That is, one and no more than one - as opposed to meaning that there is only one spin on the lattice that can ever be flipped and the others are fixed after the initial creation of the lattice - this would be rather counterproductive.) Therefore, we will end up with any state ν being different from the previous state μ, by only one site's orientation.

The reason for enforcing this is to limit energy change for each individual state change. This is simply to replicate what we see in real physical systems. As previously mentioned, for a system in thermal equilibrium, contribution to sums in (2.3) comes from a limited number of different states, especially at low temperatures. The transitions between these states would not be of large energy change, as they are not states that differ massively in energy. This is why we want to limit energy change within our algorithms.

A general comparison of Glauber, Metropolis, and Kawasaki algorithms

It is useful at this point to establish the main differences and features of each algorithm. The Glauber algorithm always involves single-spin-flip dynamics, whereas the Metropolis algorithm does not always have to. (In this project however, the Metropolis algorithm will always involve single-spin-flip dynamics, as previously mentioned.) In order to change state, they both change the orientation of a single dipole.

The key difference between these two algorithms and Kawasaki algorithms, is that the latter will *exchange* two dipoles on the lattice (or the orientations of the two dipoles) during a state change - as opposed to flipping one spin on its own.

The main difference between Glauber and Metropolis algorithms lies within the exact criterion of acceptance for each algorithm, and will be defined later.

The different dynamics within each algorithm means that each one lends itself better to representing certain physical systems or real-life scenarios. The Glauber and Metropolis algorithms, applied to the Ising model, have been used as follows: in physics, to represent the mechanics of a ferromagnet;[8] and in sociology, to represent opinion dynamics within voter models.[7,6] On the other hand, the Kawasaki algorithm, applied to the *conserved-order-parameter* Ising model, has been used like so: in physics: to represent the properties of lattice gases;[3] and in sociology, to represent ethnic segregation.[9,6]

The next few subsections will illustrate how one would go about setting up each algorithm and taking measurements. Then, in the next section, we will discuss how they can be modified or interpreted to represent the sociological situations mentioned above.

Measurements

We must now discuss how our chosen algorithms can be used to make measurements of physical quantities of a system, namely: internal energy and magnetisation.

To do this one must wait for the system to come to *equilibrium*, which is a term we defined at the start of section 2.1. That is, for a system to be in thermal equilibrium, it has to obey (2.2).

The way to make sure our system has come to equilibrium is to observe that a physical quantity has become reasonably constant after passing a certain point in time. That is, we plot a Q vs t graph where Q is some observable quantity.

However, there is a problem with this method - systems can reach what is called a "local energy minimum".[3] This is where a system finds itself in a configuration of spins where flipping any single spin would result in an increase in the internal energy of the system, and thus systems often stay in this state for a period of time. (Note we are using the terms "configuration" and "state" interchangeably here.) This is not the state where equilibrium is most likely to occur, which is in fact the "global energy minimum"[3] - the state with the lowest possible energy a given system can have.

In order to not be deceived by local energy minima, we actually plot two Q vs t graphs. One can then conclude that the system is at thermal equilibrium when the observable quantity in *both* systems has become reasonably constant (at the same value of course). One should still run the system for some time after this however, because the local energy minima of two systems can align.

The two systems should be started in different initial states with different "seeds"[3] (the initial value that a random number generator - in our case, Python's inbuilt function - uses to generate a string of pseudo-random numbers). However, one should use the same T (temperature) value, so that the equilibrium quantities reached are the same.

We have now established the use and plotting of physical quantities. Having already explained that the energy of a system can be given by (2.18), and the magnetisation of a system can be given by (2.23), let us explore exactly how and when the equations for E, M, ΔE, and ΔM are used within Monte Carlo algorithms.

First of all, note the concept of auto-correlation.[3] We define a single "sweep" to be one full run through of N Monte Carlo iterations, where N is the total number of lattice sites.

The four quantities mentioned above will therefore be measured once at the end of every sweep, as opposed to every time the system is updated. This ensures there are not hundreds of somewhat useless data being stored, and also plotted. This in turn increases the overall efficiency of the algorithm. It is important to note here that by *measured* we mean that the quantity be saved into some long list of values (or, an "array" in programming terms); not the actual calculation of the quantity.

Consider again (2.18). As this has to step through every site on the lattice, it would be extremely inefficient to compute this on every sweep, let alone on every iteration. A more efficient method, and the one we have employed, is one that takes advantage of single-spin-flip-dynamics. That is, on each iteration, we only calculate the change in energy (2.22) of the system caused by the single-spin-flip. We then add this to the running total of our energy to update the quantity accurately and efficiently.

It should be noted here that the quantities E and M are calculated on every iteration, simply so that the running total is correct. This is also an inherent feature of the way Monte Carlo algorithms work, as we will see later. They are only "measured" on every sweep however, for reasons we have already discussed.

It is of paramount importance however, that to utilise this method correctly, there must be a calculation of the internal energy of the whole system in its initial state (immediately after creation) using the Hamiltonian.

Magnetisation can be approached in a very similar fashion. In other words, one should first calculate the magnetisation of the whole system in its initial state (by (2.23)), and then subsequently update this value on each iteration by calculating the change in magnetisation (by (2.25)) caused by a single-spin-flip. The running total of this is then measured every sweep.

Estimators

If we just step back for a moment and look to section 2.2, we can see that a key goal we set out to achieve was to find the *estimator* of a quantity - given by

$$Q_M = \frac{1}{M}\sum_{i=1}^{M} Q_{\mu_i}. \tag{2.70}$$

Therefore, by choosing a set of values of our quantity during the global energy minimum (these will be in a number of different states M) and then averaging them, we can arrive at quantities like $\langle E \rangle$, the estimator of the energy, and $\langle m \rangle$, the estimator of the magentisation (per site).

2.5.1 The Metropolis Algorithm

We will now follow the general procedure outlined towards the end of section 2.2 to explicitly define the Metropolis algorithm.

Stochastic framework

We choose that the selection probability be set to zero for any new state that is not in $\{\nu_i\}$. We also require that all $g(\mu \to \nu)$ be equal. This implies, for a system of N degrees of freedom,[3]

$$\sum_{i=1}^{N} g(\mu \to \nu) = 1 \implies g(\mu \to \nu) = \frac{1}{N}. \tag{2.71}$$

It is helpful to note that

$$|\{\nu_i\}| = N \tag{2.72}$$

where N here is the same N we defined in 2.1 (that is, the number of degrees of freedom). The reason for (2.72) is that we are using single-spin-flip dynamics. This means that the amount of possible new states ν_i is one per every degree of freedom. Hence if we have N degrees of freedom, the size of the set $\{\nu_i\}$ will be N. And finally, this is why there are N different selection probabilities $g(\mu \to \nu)$, and therefore why (2.71) holds true.

Enforcing the constraint of (2.9) leads us to[3]

$$\frac{P(\mu \to \nu)}{P(\nu \to \mu)} = \frac{g(\mu \to \nu)A(\mu \to \nu)}{g(\nu \to \mu)A(\nu \to \mu)} = \frac{A(\mu \to \nu)}{A(\nu \to \mu)} = e^{-\beta \Delta E}. \tag{2.73}$$

Now we will choose our acceptance probabilities. We have already established and justified in section 2.2 that these will be chosen to be as large as possible, while still satisfying (2.9) and (2.73). They take the form[3]

$$A(\mu \to \nu) = \begin{cases} e^{-\beta(\Delta E)} & \text{if } \Delta E > 0 \\ 1 & \text{otherwise.} \end{cases} \tag{2.74}$$

This is the specific acceptance criterion (and indeed the unique aspect) of the Metropolis algorithm.

Example

Putting together everything we've covered so far - Markov processes, the zero-field Ising model and its quantity calculations, and the general process of the Metropolis algorithm - we can produce the following graph:

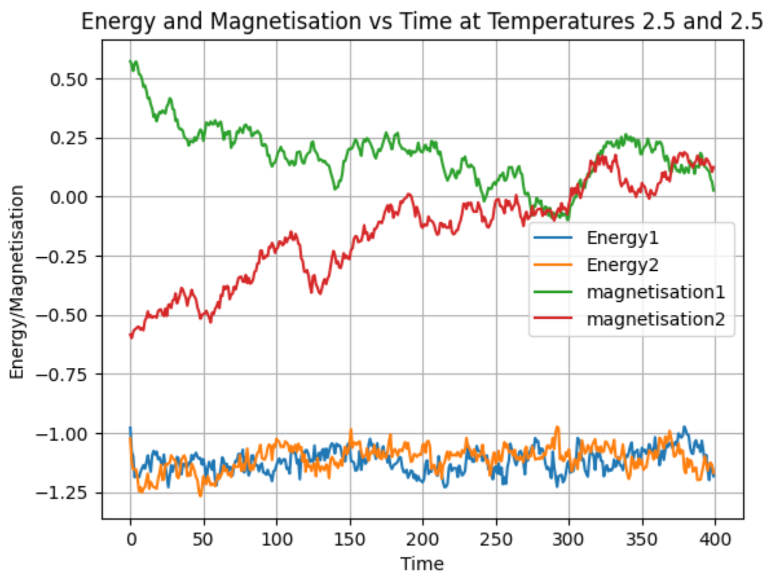


Figure 2.1: Two 2D Ising models simulated using the Metropolis algorithm, started in two different initial states, with T = 2.5, lattice size 50x50, and 1e6 iterations (split into 400 sweeps). We have then plotted the energy per site vs time and the magnetisation per site vs time for both models.

We can see here that, even when started in different initial states, two models at the same temperature will come to equilibrium at the same values for energy and magnetisation (that is, equilibrium within their own individual systems, they are in no way interacting). We can identify a possibly local energy minimum from around t = 275 to t = 300; and we deduce that the global energy minimum, hence the system coming to equilibrium, occurs at around t = 370.

In order to demonstrate a phase transition, we ran 25 simulations of our system at temperatures $0.2 \leq T \leq 5.0$ (waiting for each to come to equilibrium) and then plotted χ vs T, to obtain the following graph:

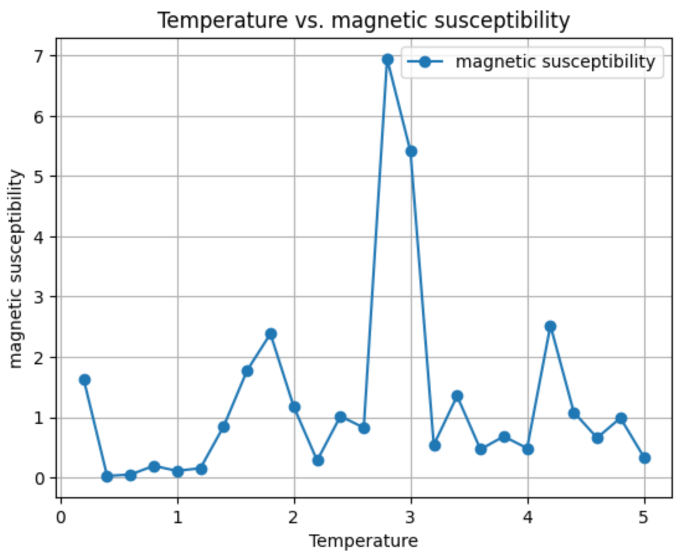


Figure 2.2: Graph of Magnetic Susceptibility vs Temperature for a 2D Ising model simulated using the Metropolis algorithm; each time started in an initial random state, with lattice size 50x50 and let run for $1e6$ iterations (split into 400 sweeps).

We can clearly see a peak in the magnetic susceptibility around $T = 2.8$ and we therefore take this to be our critical temperature, T_c:

$$\textbf{Measured Critical Temperature:} \qquad 2.6 \leq T_c \leq 3.0 \tag{2.75}$$

In addition, because this graph involves discontinuity in the derivative of an order parameter, we can conclude that the 2D zero-field Ising model has a second-order phase transition.

For this same 25 simulations of our system at temperatures $0.2 \leq T \leq 5.0$, we also plotted c vs T.

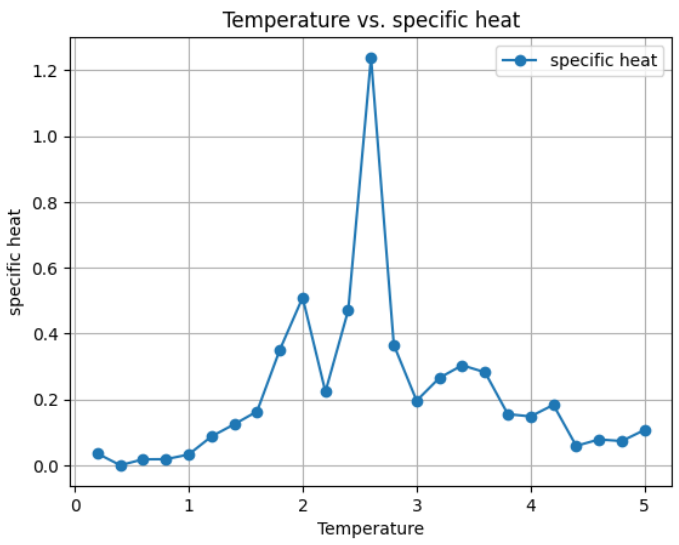


Figure 2.3: Graph of Specific Heat vs Temperature for a 2D Ising model simulated using the Metropolis algorithm; each time started in an initial random state, with lattice size 50x50 and let run for $1e6$ iterations (split into 400 sweeps).

We can clearly see a peak in the specific heat around $T = 2.6$ and we therefore take this to be our critical temperature, T_c:

$$\textbf{Measured Critical Temperature:} \qquad 2.4 \leq T_c \leq 2.8 \tag{2.76}$$

Because this graph involves an abrupt change in an observable quantity, we can interpret this as more evidence for a phase transition.

As shown in (2.43) and (2.41), we had to use ϵ and m to obtain the previous graphs. The corresponding graphs of energy and magnetisation vs temperature for the 25 simulations are given below.

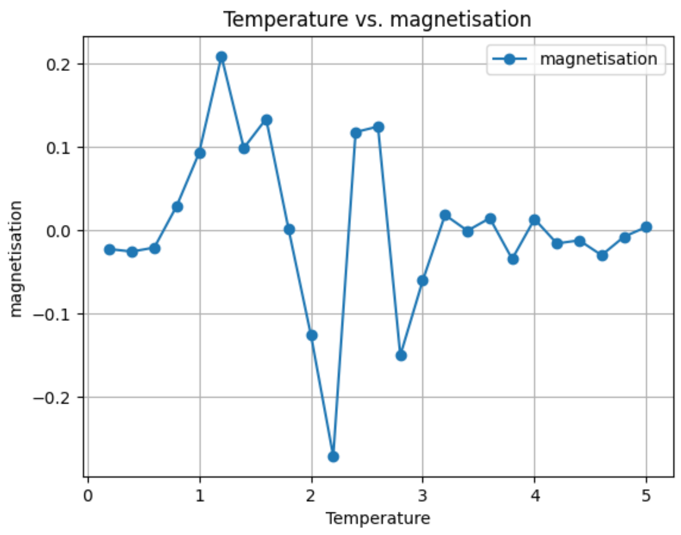


Figure 2.4: Graph of Magnetisation vs Temperature for a 2D Ising model simulated using the Metropolis algorithm; each time started in an initial random state, with lattice size 50x50 and let run for $1e6$ iterations (split into 400 sweeps).

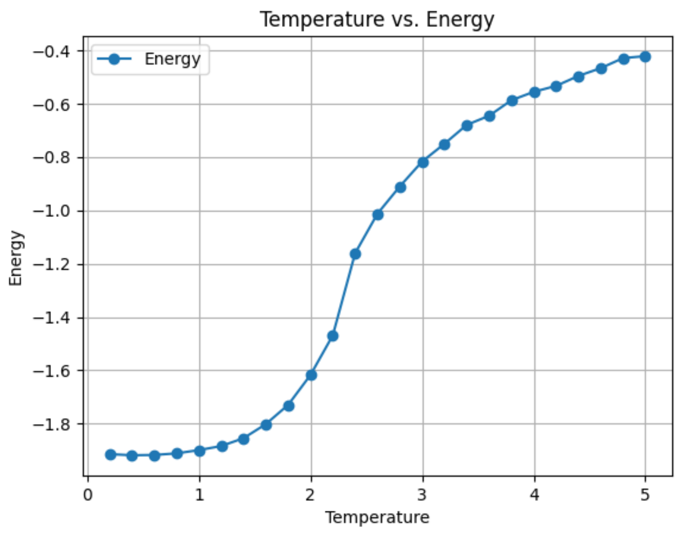


Figure 2.5: Graph of Energy vs Temperature for a 2D Ising model simulated using the Metropolis algorithm; each time started in an initial random state, with lattice size 50x50 and let run for $1e6$ iterations (split into 400 sweeps).

In this graph we can see spontaneous, non-zero magnetisation below the critical temperature T_c, and little to no magnetisation above the critical temperature. This is exactly what we would expect to see.

2.5.2 The Glauber Algorithm

Let us now turn to the Glauber algorithm, and discuss its setup in more detail. As previously mentioned in section 2.5, it is the exact criterion of acceptance that separates the Glauber algorithm from the Metropolis algorithm. All other knowledge assumed for the Metropolis algorithm also applies to the Glauber algorithm.

Stochastic framework

Equations (2.71) and (2.72) still hold true in the exact same way, therefore

$$\sum_{i=1}^{N} g(\mu \to \nu) = 1 \implies g(\mu \to \nu) = \frac{1}{N},$$

where

$$|\{\nu_i\}| = N.$$

In this case, the acceptance ratio (or "acceptance probability") is chosen to be:

$$A(\mu \to \nu) = \begin{cases} \dfrac{e^{-\beta \Delta E}}{1 + e^{-\beta \Delta E}}, & \text{if } \Delta E > 0, \\ 1, & \text{otherwise.} \end{cases} \tag{2.77}$$

High temperature limits

The behavioural differences caused by each algorithm's unique $A(\mu \to \nu)$ are usually quite subtle. However, at high temperatures in particular, there is quite a distinction.[2]

In the Metropolis algorithm, $e^{-\beta \Delta E} \to 1$ as $T \to \infty$, $\beta \to 0$. We then end up with

$$A(\mu \to \nu) \to \begin{cases} 1 & \text{if } \Delta E > 0 \\ 1 & \text{otherwise.} \end{cases} \tag{2.78}$$

This means that on every iteration, a spin is flipped; on every sweep, all spins are flipped. The model will then switch between two states for as long as the program is run, not coming to or approaching one single state.

Furthermore, in order to satisfy ergodicity, there needs to be a non-zero probability of going from μ_i to all ν_i. When a system is continuously oscillating between two states, this is clearly not satisfied.

However, in the Glauber algorithm, $\frac{e^{-\beta \Delta E}}{1+e^{-\beta \Delta E}} \to \frac{1}{2}$ as $T \to \infty$, $\beta \to 0$. We then end up with

$$A(\mu \to \nu) \to \begin{cases} 1/2 & \text{if } \Delta E > 0 \\ 1 & \text{otherwise.} \end{cases} \tag{2.79}$$

This will satisfy ergodicity for the system at all high temperatures.

With this all being said however, there is actually a method to avoid the Metropolis algorithm violating ergodicity; it is one that we have implemented. Instead of visiting every site on every sweep, we probabilistically select which sites to visit. As is enforced by translation invariance,[5,2] sites need to be visited with equal probabilities. Note, by “visited” we mean that the algorithm runs a Metropolis iteration (a flip check and possibly a spin-flip) on that particular site.

Employing this method will mean that, when large domains have formed or even taken over the whole lattice, they are not completely flipped on every subsequent sweep, and there are not simply two states that are gone between. Therefore we have ensured that both algorithms do not violate the principle of ergodicity.

Example

Just as for the Metropolis algorithm, we will now carry out two simulations of the zero-field Ising model, while measuring energy and magnetisation for both. This leads us to the following graph:

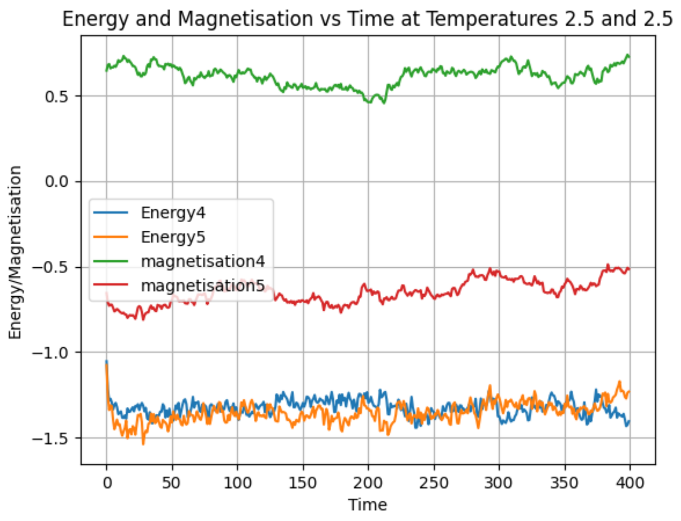


Figure 2.6: Two 2D Ising models simulated using the Glauber algorithm, started in two different initial states, with T = 2.5, lattice size 50x50, and 1e6 iterations (split into 400 sweeps). We have then plotted the energy per site vs time and the magnetisation per site vs time for both models.

Note that if we were to have plotted the absolute value of magnetisation, the results would in fact overlap.

Recall from section 2.3.3 the equations for specific heat and magnetic susceptibility, (2.43) and (2.41). We have used these in conjunction with the Glauber algorithm to once again show that the 2D zero-field Ising model experiences a second-order phase transition; as is seen on the following graphs:

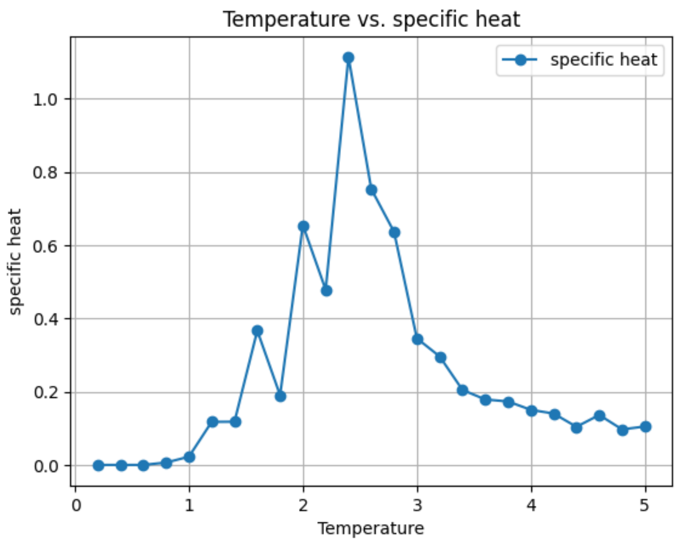


Figure 2.7: Graph of Specific Heat vs Temperature for a 2D Ising model simulated using the Glauber algorithm; each time started in an initial random state, with lattice size 50x50 and let run for $1e6$ iterations (split into 400 sweeps).

We can clearly see a peak in the specific heat around $T = 2.4$ and we therefore take this to be our critical temperature, T_c:

$$\textbf{Measured Critical Temperature:} \qquad 2.2 \leq T_c \leq 2.6 \tag{2.80}$$

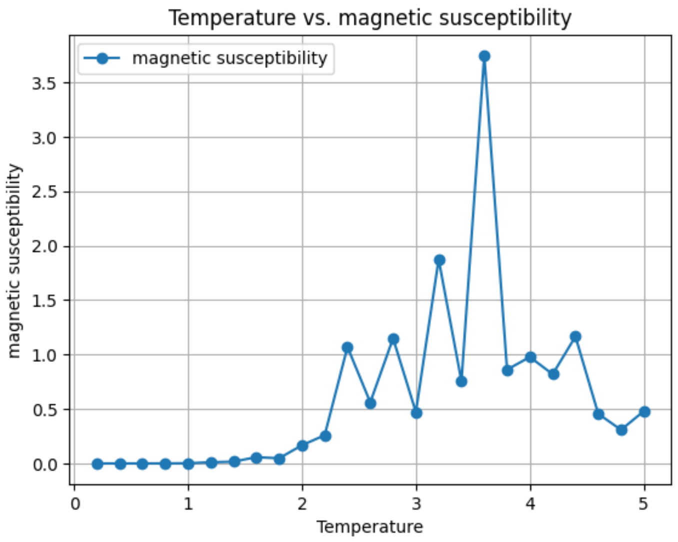


Figure 2.8: Graph of Magnetic Susceptibility vs Temperature for a 2D Ising model simulated using the Glauber algorithm; each time started in an initial random state, with lattice size 50x50 and let run for $1e6$ iterations (split into 400 sweeps).

We can clearly see a peak in the magnetic susceptibility around $T = 3.6$ and we therefore take this to be our critical temperature, T_c:

$$\textbf{Measured Critical Temperature:} \qquad 3.4 \leq T_c \leq 3.8 \tag{2.81}$$

Just like we did for the Metropolis Algorithm, we also plotted graphs of ϵ and m vs T. These are given below.

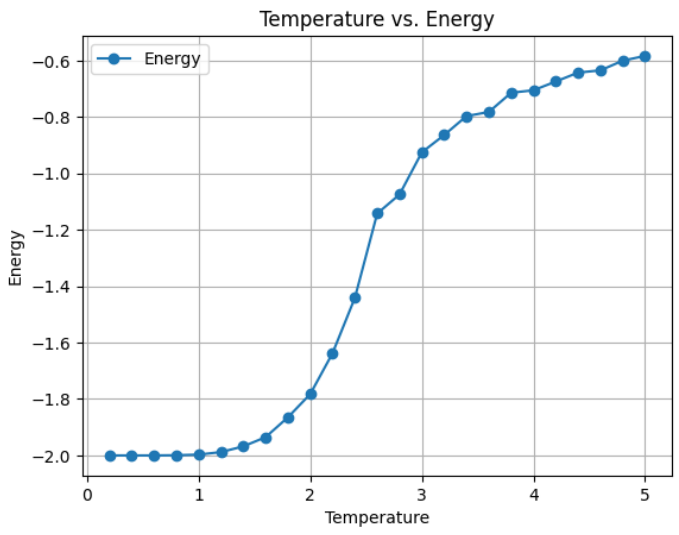


Figure 2.9: Graph of Energy vs Temperature for a 2D Ising model simulated using the Glauber algorithm; each time started in an initial random state, with lattice size 50x50 and let run for $1e6$ iterations (split into 400 sweeps).

We can see here that the low temperature results reach the equilibrium energy much faster than in the Metropolis algorithm.

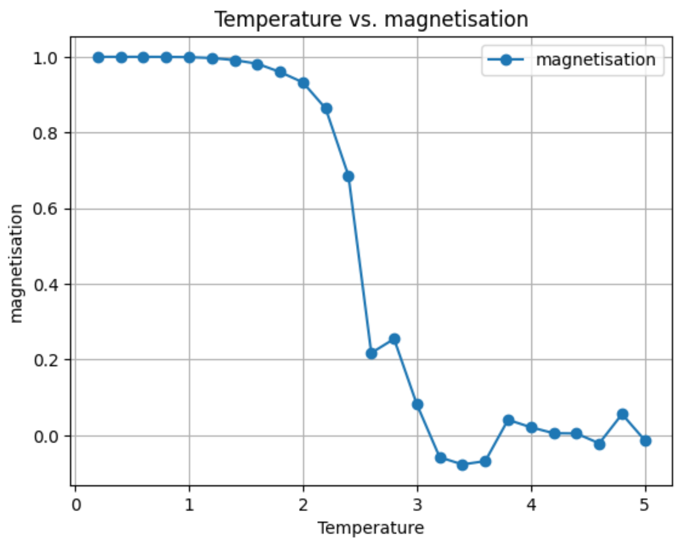


Figure 2.10: Graph of Magnetisation vs Temperature for a 2D Ising model simulated using the Glauber algorithm; each time started in an initial random state, with lattice size 50x50 and let run for $1e6$ iterations (split into 400 sweeps).

In this graph we can see spontaneous magnetisation below the critical temperature T_c, and little to no magnetisation above the critical temperature. This is exactly what we would expect to see.

2.5.3 The Kawasaki Algorithm

The Kawasaki algorithm, or "spin-exchange sampling",[2] is the algorithm applied to the COP Ising model. We will discuss how this differs from single-spin-flip-dynamics and establish other essential ingredients for this Monte Carlo method.

Spin-exchange and its initial similarity check

The main difference between Kawasaki and the other two algorithms already discussed, is the mechanics of each state-change. In a single iteration of the Kawasaki algorithm, two sites are chosen at random, which can be either *local* or *non-local* (nearest neighbours or not nearest-neighbours). If the two spins selected have the same orientation or are the exact same site, the algorithm does not do anything and moves onto the next iteration.

The purpose of this initial check is to maximise efficiency within the algorithm - both exchanging two aligned particles and exchanging one particle with itself do not change the system (and its observable quantities) in any way. Therefore we avoid useless computation by checking this first and, if need be, moving straight onto the next iteration.

On the other hand, if the two sites are opposed in orientation, they are then passed through the acceptance probability as normal. These two spins then have a chance of being *exchanged*. In essence, this means that one of the randomly selected spins is moved to the location (with its orientation kept intact) of the other randomly selected spin, while this second spin is moved to the old position of the first spin, now vacant.

However the algorithm will actually implement the exchange in a slightly different but ultimately equivalent way: our code will simply flip both spins at the same time. This is still essentially an exchange as each spin's value has effectively been transferred to the other. And after all, there is nothing more than the value to exchange, there is nothing "physical", so to speak, about these sites.

Crucially, all of these different ways of thinking about an exchange still keep magnetisation constant throughout a simulation, because any change in one site's spin is matched by an equal and opposite change in another site's spin.

Finally, it is important to note that the expression used for ΔE in the Kawasaki method is the one derived for the COP Ising model, (2.56).

Note on initial random states

As we have seen in the examples of the Metropolis and Glauber algorithms, the systems are started in "initial random states". To clarify, this means that, at initial creation of the array, the arrangement of spins on the lattice is random *and* the proportion of $\sigma = +1$ spins to $\sigma = -1$ spins is random (even if only to a certain extent, as we do specify a certain probability when setting up the lattice - this is then compared with a random number to decide which spin value to give that particular site). In other words, the proportion of $\sigma = +1$ spins to $\sigma = -1$ spins is *approximate*.

However, when working with the COP Ising model, and here the Kawasaki algorithm, we set this proportion to be a certain value. This proportion, also known as the *particle density* (2.63), is inherently related to the magnetisation of the system. Therefore, what we are able to do is set the constant magnetisation of our system by setting the proportion of $\sigma = +1$ spins to $\sigma = -1$ spins in our initial state.

The arrangement, or configuration, of spins on the lattice is still random however, and therefore we still use the phrase "initial random state".

Stochastic framework

Equations (2.71) and (2.72) still apply, therefore

$$\sum_{i=1}^{N} g(\mu \to \nu) = 1 \implies g(\mu \to \nu) = \frac{1}{N},$$

where

$$|\{\nu_i\}| = N.$$

The Kawasaki algorithm's acceptance probability is the same as that of the Glauber algorithm.[9]

$$A(\mu \to \nu) = \begin{cases} \dfrac{e^{-\beta \Delta E}}{1 + e^{-\beta \Delta E}}, & \text{if } \Delta E > 0, \\ 1, & \text{otherwise.} \end{cases} \tag{2.82}$$

Example

As the magnetisation of our system will be held constant throughout, there is no need to plot the magnetisation's evolution through time or Temperature. We also note that the magnetic susceptibility is irrelevant.

We can therefore skip straight to the graphs of energy and specific heat vs temperature.

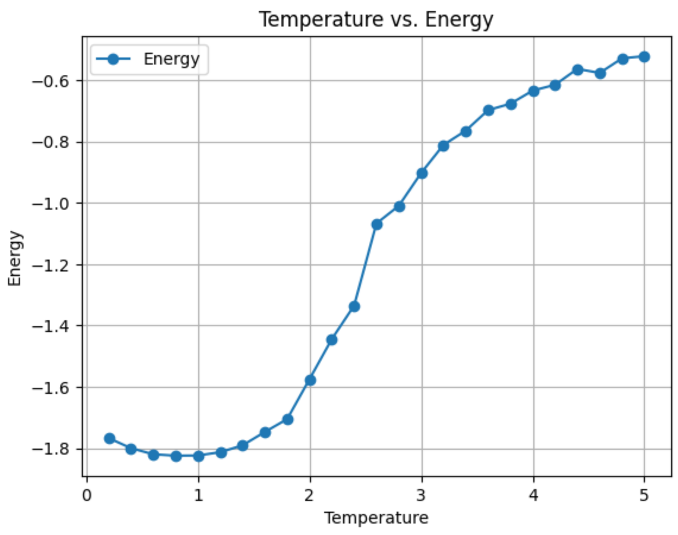


Figure 2.11: Graph of Energy vs Temperature for a 2D Ising model simulated using the Kawasaki algorithm; each time started in an initial random state, with lattice size 50x50 and let run for $1e6$ iterations (split into 400 sweeps).

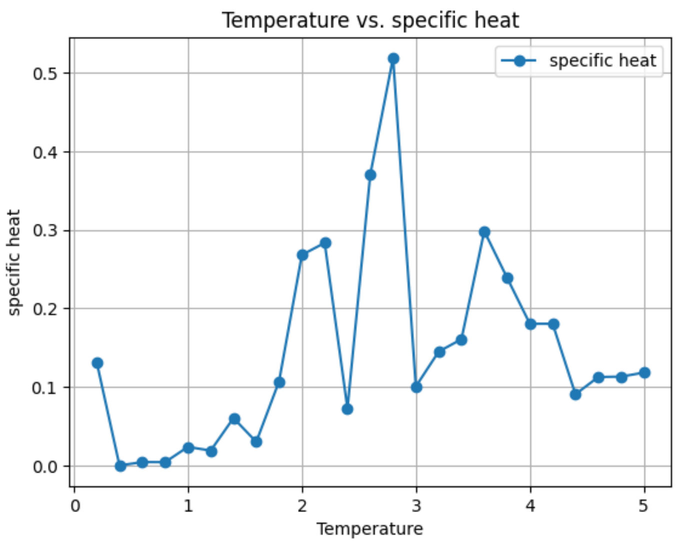


Figure 2.12: Graph of Specific Heat vs Temperature for a 2D Ising model simulated using the Kawasaki algorithm; each time started in an initial random state, with lattice size 50x50 and let run for $1e6$ iterations (split into 400 sweeps).

Both of these graphs exhibit abrupt changes around the 2D Ising model's critical temperature, (2.45), showing evidence for a phase transition.

More specifically, if we were to look at figures of the lattices themselves at different temperatures, we can see phase separation (into coexisting domains of opposing orientation) at lower temperatures, transitioning to the homogeneous phase at higher temperatures.

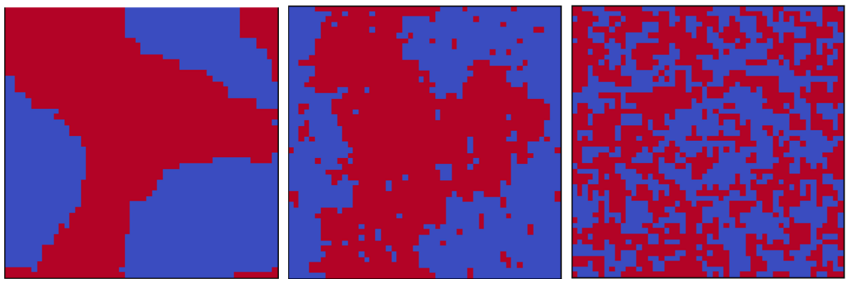


Figure 2.13: Visual representations of a 2D Ising model simulated using the Kawasaki algorithm; lattice size 50x50 and let run for $1e6$ iterations (split into 400 sweeps). The temperatures, from left to right, are $T = 1.0, T = 2.0, T = 4.0$.

Social Physics

Social physics (or "sociophysics" as it is also known) is the result of interdisciplinary research between the social sciences and theoretical (statistical) physics. This chapter will outline a brief history of the field and a general motivation for the application. We will then show and analyse simulations created for opinion dynamics and self-organising segregation.

Brief history

No less than twenty-five centuries ago it was proposed that humans behave like liquids, with some mixing well and others not (Empedokles[7]). Hence it has been believed for millennia that there are similarities and parallels between the natural sciences and the social sciences. Thomas Hobbes often related his theories of society to those of the motion of a particle.[10] Adam Smith's "invisible hand"[10] (18th century) and Hobbes' view of inherent human selfishness both suggest that humans obey deterministic physical laws, leading to self-organising phenomena.[10]

These intrinsic properties of humans proposed by Empedokles, Hobbes and Smith can be seen in the work of Schelling,[6,11] where it is suggested that people prefer to neighbour their own group or neighbour someone possessing the same *opinion* as their own. He would go on to model this with spins on a lattice wanting to align, resulting in the Schelling Model.[11] What has ensued is five decades of increasing interdisciplinary research.[12]

Note on empirical data

Evidence shows that mass psychology of groups of humans is analogous to the macroscopic behaviour of large groups of microscopic particles.[13,14,15] [14] shows that the October 1998 Brazil elections followed power-law behaviour (as seen in section 2.3.3) and scale-invariance. [13] reviews these findings and many more from other physicists, such as: birth rates,[16] number of mobile phones in a country,[16] applause dynamics,[17] and email exchanges.[18]

It is these empirical data which uphold the validity of this field of research. Without the above, the ideas and methods to be discussed may have been disregarded by now.

We can therefore say that, by utilising statistical physics and everything it involves, we can predict the behaviour of the masses[10,7] - but not the individual. The rest of this chapter shows how this can be approached at a basic level with the methods already established in previous chapters.

3.1 Opinion Dynamics

Quantity analogies

One main equivalence used time and time again within sociophyics is that of "social temperature".[9] Social temperature can be thought of as a general *tolerance* or initiative for integration within a society.

Next, one should recall the Ising model of previous sections. Each site could take one of two *spin* values: $\sigma = +1$ or $\sigma = -1$. In this section we will be applying this idea to model *opinions*: let us assume that the views of a people can be discretised into two opposing opinions, regardless of the matter at hand.

Within the Ising model, there is an implicit assumption of *interaction*. This, of course, we also apply to people, and say that the people surrounding a person influence the person's opinion, and vice versa. This though begs questions and criticisms such as: "is one person influenced by the same number of people as everyone else?" or "people are not just influenced by the people geographically closest to them". The answer to these doubts has already been covered: we are not attempting to model the activity and psychology of the individual, we are attempting to model the activity and psychology of a people.

This section will show the effect that a change in the tolerance of a society can have on people's opinions.

3.1.1 Opinion Spread

Let us now assume that the opinion of a person (=spin of one site) can change (=flip). We have therefore fully justified the use of the Glauber algorithm to simulate the opinions of a people. Implementing this algorithm at various tolerances will produce the following figures.

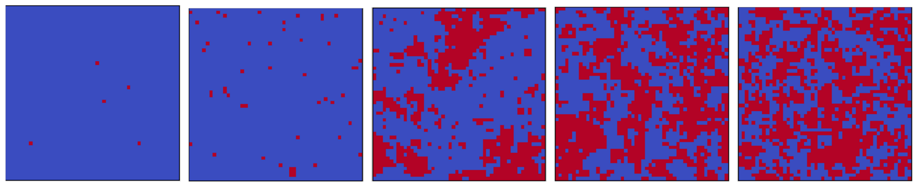


Figure 3.1: Visual representations of two opinions (here shown by blue and red dots) spreading through a society due to a change in social temperature. They have been simulated using the Glauber algorithm with a lattice size 50x50 and let run for $1e6$ iterations (split into 400 sweeps). The temperatures/tolerances from left to right are: $\frac{T}{T_c} = 0.5$, $\frac{T}{T_c} = 0.8$, $\frac{T}{T_c} = 1.0$, $\frac{T}{T_c} = 1.2$, $\frac{T}{T_c} = 1.5$.

What we can see in these figures is that, at low tolerance, we expect a completely polarised society in regard to opinion. Then, as the tolerance increases, we see the much less popular opinion of before spread from person to person. This results in a greater chance of one being surrounded (=nearest neighbours) by people of opposing opinions.

The results can be visualised in a different way: if we were to plot magnetisation vs temperature, exactly as in Figure 2.10, we would essentially get a *degree of polarisation* vs *tolerance* graph.

3.1.2 Opinion Dynamics and External Influence

Ising models can be subject to an external magnetic field. These can be modelled as uniform, site dependant, or in a plethora of other ways. One can incorporate a magnetic field into the calculation of energy and energy change through the mean-field approximation, as discussed in section 2.3.1.

Quantity analogies

We will establish here another quantity analogy: one can view the external magnetic field applied to a lattice of interacting spins as an overarching influence to a people, through some medium (news, social media, propaganda, and so on). Just as the magnetic field of a lattice of spins may vary from site to site, different people can be affected differently by an overarching influence given to all.

This section will attempt to model exactly this, based on [6].

Modelling external influence

Note that, in general, (2.38) shows a magnetic field increases the probability of spins to be aligned in the direction parallel to that field. Next, we see that [6] proposes asymmetric populations: this involves a first group of people ($\sigma = +1$) that affect the second group of people ($\sigma = -1$) differently than the first are affected by the second; all as a result of this external influence. This can be modelled through a site dependant effective magnetic field in the following form:[6]

$$B_j^{\text{eff}} = \begin{cases} \dfrac{n_A - n_B}{T} + B, & \text{if } n_A - n_B > 0, \\ B, & \text{otherwise.} \end{cases} \tag{3.1}$$

Here n_A represents the number of nearest neighbours to site j that are $\sigma = +1$, and n_B represents the number of nearest neighbours that are $\sigma = -1$.

What (3.1) is essentially saying is that an “A” person can affect a “B” person more than a “B” person can affect an “A” person. This can be deduced because having more A neighbours increases B_j^{eff} much more than when one has more B neighbours. It can therefore be said that A people are more influential and the two groups of people are *asymmetric*, as was proposed before.

In sum, we have established an effective magnetic field that will be incorporated into the ΔE caused by one spin flip:

$$\Delta E = 2B_j^{\text{eff}}\sigma_j. \tag{3.2}$$

Assume again the definition for opinions proposed at the start of this section - again, this justifies the use of the Glauber algorithm within these simulations.

We will be investigating how a change in the external influence (=external magnetic field) of a people affects opinion spread (=spins of each site). The following graph from [6] plots magnetisation vs magnetic field strength for three different temperatures, whereby the magnetic field strength at each site obeys (3.1). Recall that a high reading of magnetisation can be thought of as a high degree of polarisation, and vice versa.

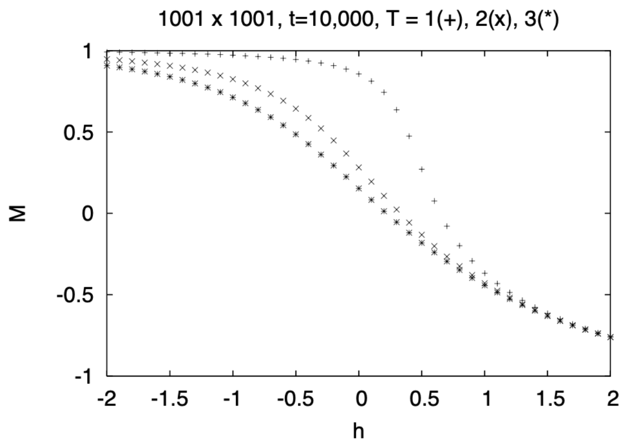


Figure 3.2: Composition of the population versus h at fixed T = 1,2,3, averaged over 10000 sweeps through a lattice of one million people. This simulation took 4 1/2 h.

Firstly, we should note that, when compared with Figure 2.10, there is an opposite sign convention being used for the magnetisation. Also, note that the points on Figure 2.10 that correspond to $T = 1, T = 2, T = 3$ are essentially an another version of the readings from Figure 3.2 that coincide with $h = 0$. From this, we can identify some discrepancies and give reasons for them.

At $T = 1$, $h = 0$ and $T = 3$, $h = 0$ the magnetisation is reasonably similar. (Note here that "h" and "B" are equivalent in standing for the magnetic field strength.) However, at $T = 2$ and $h = 0$, Figure 2.10 still shows *near* complete polarisation whereas Figure 3.2 shows a magnetisation of $M < 0.5$. The reason for this fairly substantial difference in measurement is most likely down to two things - lattice size and run time.

The lattice used in [6] is four hundred times bigger than that used here, allowing for much more subtlety and accuracy within the measurements. Also, their simulation was carried out for twenty five times more sweeps, not to mention that their one "sweep" consists of four hundred times more iterations. It is no wonder then that it took four and a half hours for this graph to be produced.

Lastly, let us define *hysteresis*.[19] Hysteresis (albeit a general concept reaching far away from magnets) describes how magnetic systems can depend on their past state. It is often characterised by a system, once put under a magnetic field of one direction, still possessing some magnetisation in this given direction even when this magnetic field is taken away. It is because of this that, if a new magnetic field - opposing the first in direction - were to be put in place, then the system may seem to not respond straight away.

We see exactly this within Figure 3.2. It could be argued that hysteresis represents how public opinion can, for a short while, remain unchanged even after an opposite external influence - possibly even show an inherent human reluctance to new and different ideas. However, to reiterate once more, statements like these can never really be measured, and hence never really be proven quantitatively (at least not in the scope of this project).

3.2 Self-organising Segregation

Quantity analogies

In this section we will be using the same definition of social temperature and will still be proposing interactions between people. However, instead of opinions, now consider that our spins correspond to one of two different types of people, differentiated by race, age, sex, religion; or whatever criteria of segregation that may exist. (We could even model a person of a certain opinion, as opposed to, in the last section, the opinion of a certain person.)

This section will show how the Ising model can be used to simulate the movement of people characterised by one of these *descriptors*.

3.2.1 Modelling Self-organising Segregation

Let us now assume that the descriptor of a person (=spin of one site) cannot change (=flip). Instead we model people to be able to *relocate* (=exchange). We have therefore fully justified the use of the Kawasaki algorithm, and we can refer back to the figures in 2.13, displayed again here for convenience.

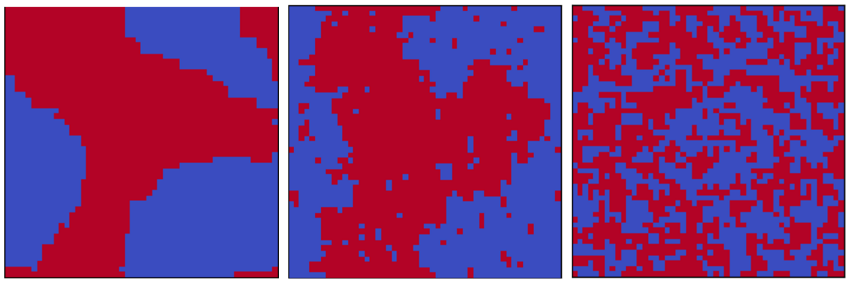


Figure 3.3: Visual representations of a 2D Ising model simulated using the Kawasaki algorithm; lattice size 50x50 and let run for $1e6$ iterations (split into 400 sweeps). The temperatures from left to right are: $T = 1.0, T = 2.0, T = 4.0$.

When viewing these figures from the current perspective, we can deduce that it is low tolerance that causes segregation (=coexisting phase), and high tolerance that causes integration (=homogeneous phase). The literature[6] refers to this as "self-organising" due to there being no external magnetic field taken into account in the simulation.

We will now move on to show a successful recreation and adaptation of tests run in [9].

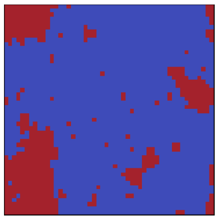
The relationship between concentration and tolerance

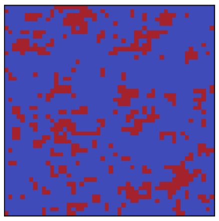
[9] investigates and executes something very similar to what we have shown above, but with more emphasis and analysis on the relationship between *concentration* and temperature. Concentration here specifies the proportion of one people to the other. Therefore, in this context, concentration is analogous to particle density. Also, note that we are still using the same idea of social temperature.

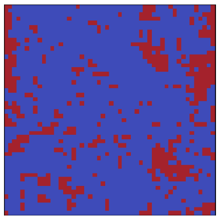
We have already shown in (2.63) how the particle density can be used to fix the magnetisation of a system. We will use this technique to test the balance between concentration and temperature - and whether it results in segregation or not - by changing the two fixed quantities (that is, multiple simulations with different concentration and temperature; but within each, they are fixed).

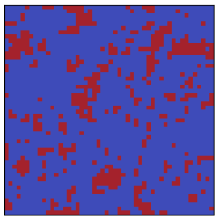
There will be three tests, each containing four simulations, to show "ghetto dissolution", "ghetto formation", and "no ghetto formation in equilibrium".[9] Note there is a fourth test in the literature which we omit. All of the following figures were obtained on a 50x50 lattice as opposed to a 79x100 lattice - a small difference between our adaptation and [9] (along with the exact lattice configurations produced of course). We denote the concentration as c, temperature as T, and the number of Monte Carlo iterations as N. (Hence N here should not be confused with its other use as the number of sites on the lattice.)

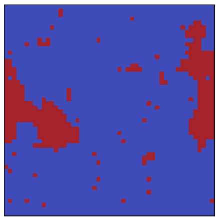
Note we prefaced this work in the last subsection as not only a recreation but an *adaptation* as well. This is because there is one key difference between the figures produced below and [9] - [9] only allows for the exchange of nearest neighbour spins (that is, “local”) whereas we allow for any non-local exchange; the latter being of course more realistic when trying to model humans.

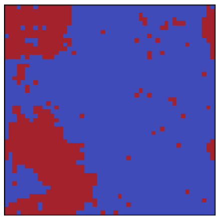

(a) $c = 0.2, \frac{T}{T_c} = 0.8, N = 10^7$

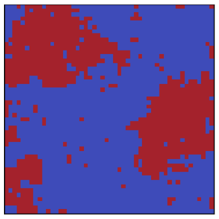

(b) $c = 0.2, \frac{T}{T_c} = 1.2, N = 10^2$

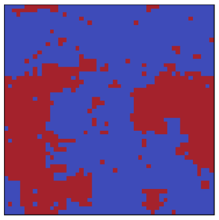

(c) $c = 0.2, \frac{T}{T_c} = 1.2, N = 10^4$

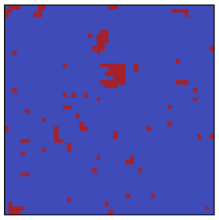

(d) $c = 0.2, \frac{T}{T_c} = 1.2, N = 10^6$

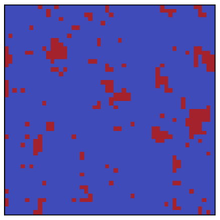

(e) $c = 0.15, \frac{T}{T_c} = 0.8, N = 600$

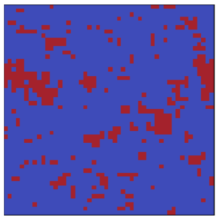

(f) $c = 0.25, \frac{T}{T_c} = 0.85, N = 600$


(g) $c = 0.35, \frac{T}{T_c} = 0.9, N = 10^3$


(h) $c = 0.35, \frac{T}{T_c} = 0.9, N = 10^7$


(i) $c = 0.05, \frac{T}{T_c} = 0.8, N = 10^7$


(j) $c = 0.10, \frac{T}{T_c} = 0.95, N = 10^7$


(k) $c = 0.15, \frac{T}{T_c} = 1.1, N = 10^7$

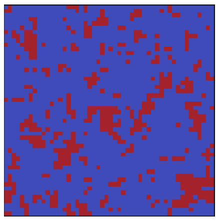

(l) $c = 0.2, \frac{T}{T_c} = 1.25, N = 10^7$

Figure 3.4: Figures demonstrating “ghetto dissolution” (a - d), “ghetto formation” (e - h), and “no ghetto formation in equilibrium” (i - l); simulated using the Kawasaki algorithm at size 50x50 with differing values of $c, T,$ and N.

These tests, especially the third, demonstrate exceptionally well that there is not one value of concentration or social temperature which decides whether segregation (=phase separation) occurs - it is the relationship between. One can conclude that the greater the concentration of one people, the greater a social tolerance is needed for integration (=homogeneous phase).

A comment on the model

There is one fairly simple but important criticism that the above model does not take into account, and it is a classic example of where people are not always like spins. When one person relocates, it is very rare that they exchange. In other words, it is very rare that when one moves house, they swap residences with another. Therefore to improve on the model we could have incorporated empty spaces[9] to arrive at the Schelling Model.[11]

Conclusion

This paper introduced Monte Carlo methods as algorithms that can simulate large complex systems through random sampling. We established how three specific Monte Carlo methods differ; whether it be by acceptance probability or their selection and state-change mechanism. Two of these algorithms have been applied to *social physics*, in which we attempt to simulate the activity and psychology of the masses but not the individual: the Glauber algorithm has been applied to the Ising model in order to simulate various opinion dynamics; the Kawasaki algorithm has been applied to the conserved-order-parameter Ising model to demonstrate self-organising segregation. Recall that we defined tolerance to assume its usual meaning, in addition to an initiative for integration within a society by its members.

When we simulated opinion spread at low tolerance, we ended up with a completely polarised society. As we increased the tolerance we saw a much greater spread of the initially unpopular opinion. These findings suggest, rather intuitively, that if the tolerance of a society increases, the chance of a once unpopular opinion to spread to the masses is greater. Is this to say that tolerance can cause some specific public opinion? Or could it be said that public opinion affects tolerance? However interesting these questions may be, they will not be explored here but left for the reader to ponder.

We employed the Glauber algorithm once more to investigate opinion dynamics under an external influence analogous to the news or social media. We modelled this external influence as an external magnetic field, and employed the mean field approximation. Whether parallel or anti-parallel, we see that the direction and magnitude of the external influence has a definitive effect on public opinion.

One can also identify the phenomenon of hysteresis, and propose that this shows how a people, once polarised in one view, can possess an initial reluctance towards a new and opposing view. To clarify, this is a *collective* initial reluctance - we have already established that this paper and area of research is concerned not with the individual, but the masses. It should be noted that our own simulations were not able to recreate Figure 3.2.

We then implemented the Kawasaki algorithm to analyse a system of people, each possessing some criteria of segregation, with no external influence imposed. It is this lack of external influence, as well as the phase separation itself, which leads us to the conclusion of a self-organising nature of segregation. This however, is only the low tolerance case. It is evident in Figure 2.13 that when the tolerance of a people is higher, integration is much more likely.

Expanding on these simulations we were able to investigate the relationship between concentration and tolerance, and whether or not each were the main factor of segregation. Concentration and tolerance were fixed quantities within our tests, but after twelve simulations (of varying c and T) demonstrating three different social phenomena, we were able to find our answer: it is not one of either concentration or tolerance that is the deciding factor for integration, but a combination. We can conclude that the greater the concentration of one group within a population, the greater the tolerance needed for integration.

Overall, we have demonstrated how fairly simple algorithms involving random numbers can simulate complex physical phenomena; and how one can draw parallels between a theoretical lattice of spins and the collective behaviour of humans.

4.1 Further Discussion

There are many omissions from this work which would make quite the impact if implemented: calculations of error, correlation time, correlation length, equilibration time and auto-correlation functions, to name a few. We have already seen in section 3.1.2 that the lattice size and run time of our simulations pale in comparison to those found in the literature. Increasing both of these factors would increase the accuracy and reliability of our results.

Correlation functions, correlation length and correlation time would help to attain a much deeper understanding of the interactions at play, and hence give us more insight into the opinion dynamics discussed.[19]

Finally, it should be noted that the specific numerical values used for the simulations (lattice size, social temperature, time measured in sweeps) are not to be interpreted literally.[9]

Bibliography

[1] Sacha Friedli and Yvan Velenik. “Statistical Mechanics Of Lattice Systems: A Concrete Mathematical Introduction”. In: (2018). DOI: 10.1017/9781316882603.

[2] David P. Landau and Kurt Binder. “A Guide to Monte Carlo Simulations in Statistical Physics”. In: (2009). DOI: 10.1017/9781108780346.

[3] M. E. J. Newman and G. T. Barkema. “Monte Carlo Methods in Statistical Physics”. In: (1999).

[4] Charlie Wood. “The Cartoon Picture of Magnets That Has Transformed Science”. In: Quanta Magazine (2020). URL: https://www.quantamagazine.org/print.

[5] David Schaich. MATH327 Lecture Notes. Available at https://canvas.liverpool.ac.uk/courses/69036/files/10932838?module_item_id=1952653. 2024.

[6] D. Stauffer and S. Solomon. “Ising, Schelling and self-organising segregation”. In: Eur. Phys. J. B 57 (2007), pp. 473–479. DOI: 10.1140/epjb/e2007-00181-8.

[7] D. Stauffer. “Opinion Dynamics and Sociophysics”. In: (2009). DOI: 10.1007/978-0-387-30440-3_376.

[8] Daniel V. Schroeder. “An Introduction to Thermal Physics”. In: (2020). DOI: 10.1093/oso/9780192895547.001.0001.

[9] Hildegard Meyer-Ortmanns. “Immigration, Integration and Ghetto Formation”. In: International Journal of Modern Physics C 14 (2003). ISSN: 1793-6586. DOI: 10.1142/s0129183103004504.

[10] Matjaž Perc. “The social physics collective”. In: Sci Rep (2019). DOI: 10.1038/s41598-019-53300-4.

[11] Thomas C Schelling. Dynamic models of segregation. 1971. DOI: 10.1080/0022250X.1971.9989794.

[12] Wolfgang Weidlich. “Sociodynamics—a systematic approach to mathematical modelling in the social sciences”. In: Chaos, Solitons & Fractals 18.3 (2003), pp. 431–437. ISSN: 0960-0779. DOI: 10.1016/S0960-0779(02)00666-5. URL: https://doi.org/10.1016/S0960-0779(02)00666-5.

[13] Claudio Castellano, Santo Fortunato, and Vittorio Loreto. “Statistical physics of social dynamics”. In: Reviews of Modern Physics (2009).

[14] RN Costa Filho et al. “Scaling behavior in a proportional voting process”. In: Physical Review E 60.1 (1999), p. 1067.

[15] Gerardo Iñiguez Antonio F. Peralta János Kertész. “Opinion dynamics in social networks: From models to data”. In: Handbook of Computational Social Science (2022).

[16] Quentin Michard and J-P Bouchaud. “Theory of collective opinion shifts: from smooth trends to abrupt swings”. In: The European Physical Journal B-Condensed Matter and Con 47 (2005), pp. 151–159.

[17] Zoltán Néda et al. “The sound of many hands clapping”. In: Nature 403.6772 (2000), pp. 849–850.

[18] Jean-Pierre Eckmann, Elisha Moses, and Danilo Sergi. “Entropy of dialogues creates coherent structures in e-mail traffic”. In: Proceedings of the National Academy of Sciences 101.40 (2004), pp. 14333–14337.

[19] Han L. J. van der Maas, Jonas Dalege, and Lourens Waldorp. “The polarization within and across individuals: the hierarchical Ising opinion model”. In: Journal of Complex Networks 8 (2 Apr. 2020), cnaa010. DOI: 10.1093/comnet/cnaa010.

Metropolis Algorithm

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

# main parameters to be varied/created when running different tests
T1 = 2.5
T2 = 2.5
p1 = 0.2 # if either of these are set to 1: initial state completely spin
down
p2 = 0.8 # set to 0: initial state completely spin up
seed1 = 42
seed2 = 314159
sweeps = 400
```

```
# Ising model with metropolis kinetics.

# creating initial random arrays
maxDim = 50
vol = maxDim ** 2
one_over_vol = 1.0 / vol
one_over_sweeps = 1.0 / sweeps
N = sweeps * vol

rs1 = np.random.RandomState(seed1)
rs2 = np.random.RandomState(seed2)
spins1 = np.zeros((maxDim, maxDim))
spins2 = np.zeros((maxDim, maxDim))
for i in range(0, maxDim):
  for j in range(0, maxDim):
    if rs1.random() < p1:
      spins1[i,j] = -1
    else:
      spins1[i,j] = 1
    if rs2.random() < p2:
      spins2[i,j] = -1
    else:
      spins2[i,j] = 1


# function to compute change in energy of potential spin flip
def pot_dEfunction(spins, i, j):
  left = spins[i, (j - 1) % spins.shape[1]]
  right = spins[i, (j + 1) % spins.shape[1]]
  top = spins[(i - 1) % spins.shape[0], j]
  bottom = spins[(i + 1) % spins.shape[0], j]
  return 2 * spins[i, j] * (left + right + top + bottom)
```

```
# function to calculate Hamiltonian of whole system
def hamiltonian(spins):
  Ham = 0
  for i in range(spins.shape[0]):
    for j in range(spins.shape[1]):
      left = spins[i, (j - 1) % spins.shape[1]]
      right = spins[i, (j + 1) % spins.shape[1]]
      top = spins[(i - 1) % spins.shape[0], j]
      bottom = spins[(i + 1) % spins.shape[0], j]
      Ham += -spins[i, j] * (left + right + top + bottom)
  return Ham / 2

# metropolis algorithm
def metropolis(spins, T, N, randomState):
  E_o = hamiltonian(spins)
  E = E_o
  mag_o = np.sum(spins)
  mag = mag_o

  energy_axis = np.zeros(sweeps)
  energy_axis[0] = E_o
  mag_axis = np.zeros(sweeps)
  mag_axis[0] = mag_o
  time_axis = np.arange(sweeps)      # separate time axis for each
simulation

  for sweep in range(sweeps):
    # start of metropolis iterations
    for iteration in range(vol):
      i = randomState.choice(range(spins.shape[0]))   # randomly select
row
      j = randomState.choice(range(spins.shape[1]))   # randomly select
column
      pot_dE = pot_dEfunction(spins, i, j)
      if pot_dE < 0:                                # swap check
        spins[i, j] = -spins[i, j]
        mag += 2 * spins[i, j]
        E += pot_dE
      elif randomState.random() < np.exp(-pot_dE / T):  # "unique aspect"
        spins[i, j] = -spins[i, j]
        mag += 2 * spins[i, j]
        E += pot_dE
    # end of metropolis iterations
    # (therefore the following happens every 'sweep')

    energy_axis[sweep] = E * one_over_vol   # per site
    mag_axis[sweep] = mag * one_over_vol    # per site

  return time_axis, energy_axis, mag_axis # spec_heat_arr, mag_susc_arr
```

```
# applying metropolis algorithm to two systems differing by
# initial state & seed

time_axis1, energy_axis1, mag_axis1 = metropolis(spins1, T1, N, rs1)
time_axis2, energy_axis2, mag_axis2 = metropolis(spins2, T2, N, rs2)

# displaying first 2D Ising model
print()
plt.figure(figsize=(4, 3))
plt.imshow(spins1, cmap='coolwarm', interpolation='nearest')
plt.colorbar().remove()
plt.xticks([])
plt.yticks([])
plt.title("1st Monte Carlo simulation of a 2D Ising model \
using the Metropolis algorithm")
plt.text(-45, 2, f'Size: {maxDim} by {maxDim}', color='black',
fontsize=10)
plt.text(-45, 5, f'Temperature: {T1}', color='black', fontsize=10)
plt.text(-45, 8, f'Iterations: {N}', color='black', fontsize=10)
plt.text(-45, 11, f'Seed: {seed1}', color='black', fontsize=10)
plt.show()

# displaying second 2D Ising model
print()
plt.figure(figsize=(4, 3))
plt.imshow(spins2, cmap='coolwarm', interpolation='nearest')
plt.colorbar().remove()
plt.xticks([])
plt.yticks([])
plt.title("2nd Monte Carlo simulation of a 2D Ising model \
using the Metropolis algorithm")
plt.text(-45, 2, f'Size: {maxDim} by {maxDim}', color='black',
fontsize=10)
plt.text(-45, 5, f'Temperature: {T2}', color='black', fontsize=10)
plt.text(-45, 8, f'Iterations: {N}', color='black', fontsize=10)
plt.text(-45, 11, f'Seed: {seed2}', color='black', fontsize=10)
plt.show()

# plotting graphs of (time vs energy) & (time vs magnetisation)
# for both simulations
print()
plt.plot(time_axis1, energy_axis1, label='Energy1')
plt.plot(time_axis2, energy_axis2, label='Energy2')
plt.plot(time_axis1, mag_axis1, label='magnetisation1')
plt.plot(time_axis2, mag_axis2, label='magnetisation2')
plt.xlabel('Time')
plt.ylabel('Energy/Magnetisation')
plt.title(f'Energy and Magnetisation vs Time at Temperatures {T1} and
{T2}')
plt.legend()
plt.grid(True)
plt.show()
```

```
# note: graph in report, for results of above code, uses: p1 = 0.2, p2 =
0.8
```

```
# phase transition for an Ising model with metropolis kinetics.

T3 = np.arange(0.2, 5.2, 0.2) # ie T3 is an array of values from 0.5 to
5.0
p3 = 0.5                      # in steps of 0.2
seed3 = 100

rs3 = np.random.RandomState(seed3)
spins3 = np.zeros((maxDim, maxDim))
for i in range(0, maxDim):
  for j in range(0, maxDim):
    if rs3.random() < p3:
      #could be np.random.RandomState(seed3).random() < p3:
      spins3[i,j] = -1
    else:
      spins3[i,j] = 1

# a 3rd ising model has now been randomly generated.

def specific_heat_fn(avg_xsq, avg_x, T, vol):
  c = (1/(T**2)) * vol * (avg_xsq - avg_x**2)
  return c

def magnetic_susc_fn(avg_xsq, avg_x, T, vol):
  chi = (1/T) * vol * (avg_xsq - avg_x**2)
  return chi

# functions defined along with equations in report
```

```
vals_taken = 30
import csv
file_name = "simulation_results.csv"

with open(file_name, 'w', newline='') as csvfile:
  writer = csv.writer(csvfile)
  writer.writerow(['T', 'E', 'm', 'c', 'X'])
  for i in T3:
    # for every temperature, we calculate one value for energy,
    # magnetisation, specific heat, and magnetic susc

    #calculate arrays, then turn into single average value
    time_axis3, energy_axis3, mag_axis3 = metropolis(spins3, i, N, rs3)

    avg_E = np.mean(energy_axis3[-vals_taken:]) # single value obtained
    sq_sum = 0
    for j in energy_axis3[-vals_taken:]:
      sq_sum += j**2
    avg_Esq = sq_sum / vals_taken

    avg_mag = np.mean(mag_axis3[-vals_taken:])  # single value obtained
    sq_sum = 0
    for k in mag_axis3[-vals_taken:]:
      sq_sum += k**2
    avg_magsq = sq_sum / vals_taken

    c = specific_heat_fn(avg_Esq, avg_E, i, vol)      # single value
obtained
    chi = magnetic_susc_fn(avg_magsq, avg_mag, i, vol)   # single value
obtained

    # POTENTIAL ISSUE IDENTIFIED WITHIN CODE
    # equilibrium of any one quantity will not occur at the same point for
    # different temperatures, (as is incorrectly prdeicted by setting the
    # vals_taken to be constant)
    # therefore vals_taken should not be a constant
    # i predict that a solution to this would involve calculating
EQUILIBRATION
    # TIME for each simulation; this is beyond the scope of this project
and
    # hence will not be implemented

    # The following code displays our ising model at each integer value of
temp

    if i % 1.0 == 0:
      print()
      plt.figure(figsize=(4, 3))
      plt.imshow(spins3, cmap='coolwarm', interpolation='nearest')
      plt.colorbar().remove()
      plt.xticks([])
      plt.yticks([])
      plt.title(f"Monte Carlo simulation ({int(i)}) of a 2D COP Ising
model \
      using the Met algorithm")
      plt.text(-45, 2, f'Size: {maxDim} by {maxDim}', color='black',
fontsize=10)
      plt.text(-45, 5, f'Temperature: {i}', color='black', fontsize=10)
      plt.text(-45, 8, f'Iterations: {N}', color='black', fontsize=10)
      plt.text(-45, 11, f'Seed: {seed3}', color='black', fontsize=10)
      plt.show()

    # write single values into row, corresponding to its T
    # time axis not needed
    writer.writerow([i, avg_E, avg_mag, c, chi])

# at this point we have obtained results for average energy, average
# magnetisation, specific heat and magnetic susceptibility, for different
values
# of temperature
# and written these results into file_name = "simulation_results.csv"
```

```python



# the following code reads this csv file and then plots four graphs

import pandas as pd

df = pd.read_csv('simulation_results.csv')

# writer.writerow(['T', 'E', 'm', 'c', 'X'])


plt.plot(df['T'], df['E'], marker = 'o', label = 'Energy')
plt.xlabel('Temperature')
plt.ylabel('Energy')
plt.title('Temperature vs. Energy')
plt.grid(True)
plt.legend()
plt.show()

print()
plt.plot(df['T'], df['m'], marker = 'o', label = 'magnetisation')
plt.xlabel('Temperature')
plt.ylabel('magnetisation')
plt.title('Temperature vs. magnetisation')
plt.grid(True)
plt.legend()
plt.show()

print()
plt.plot(df['T'], df['c'], marker = 'o', label = 'specific heat')
plt.xlabel('Temperature')
plt.ylabel('specific heat')
plt.title('Temperature vs. specific heat')
plt.grid(True)
plt.legend()
plt.show()

print()
plt.plot(df['T'], df['X'], marker = 'o', label = 'magnetic
susceptibility')
plt.xlabel('Temperature')
plt.ylabel('magnetic susceptibility')
plt.title('Temperature vs. magnetic susceptibility')
plt.grid(True)
plt.legend()
plt.show()
```

Glauber Algorithm

```
# Ising model with glauber kinetics.


def glauber(spins, T, N, randomState):
  E_o = hamiltonian(spins)
  E = E_o
  mag_o = np.sum(spins)
  mag = mag_o

  energy_axis = np.zeros(sweeps)
  energy_axis[0] = E_o
  mag_axis = np.zeros(sweeps)
  mag_axis[0] = mag_o
  time_axis = np.arange(sweeps)      # separate time axis for each
simulation

  for sweep in range(sweeps):
    # start of glauber iterations
    for iteration in range(vol):
      i = randomState.choice(range(spins.shape[0]))   # randomly select
row
      j = randomState.choice(range(spins.shape[1]))   # randomly select
column
      pot_dE = pot_dEfunction(spins, i, j)
      if pot_dE < 0:                                # swap check
        spins[i, j] = -spins[i, j]
        mag += 2 * spins[i, j]
        E += pot_dE
      elif randomState.random() < (np.exp(-pot_dE/T)/(1 + np.exp(-
pot_dE/T))):
        spins[i, j] = -spins[i, j]              # "unique aspect" ^^^
        mag += 2 * spins[i, j]
        E += pot_dE
    # end of Glauber iterations
    # (therefore the following happens every 'sweep')

    energy_axis[sweep] = E * one_over_vol   # per site
    mag_axis[sweep] = mag * one_over_vol    # per site

  return time_axis, energy_axis, mag_axis
```

```
# we want to be able to compare the two algorithms and thus we use the
# exact same two intial states as before

# therefore we do not need to redefine any new temperatures, seeds, random
# states, or choice probability
# all we have to redefine is two spin arrays, because the spins1 and
# spins2 have already been messed with and used in the metropolis demo

spins4 = np.zeros((maxDim, maxDim))
spins5 = np.zeros((maxDim, maxDim))
```

```
for i in range(0, maxDim):
  for j in range(0, maxDim):
    if rs1.random() < p1:
      spins4[i,j] = -1
    else:
      spins4[i,j] = 1
    if rs2.random() < p2:
      spins5[i,j] = -1
    else:
      spins5[i,j] = 1

time_axis4, energy_axis4, mag_axis4 = glauber(spins4, T1, N, rs1)
time_axis5, energy_axis5, mag_axis5 = glauber(spins5, T2, N, rs2)



# displaying first 2D Ising model
plt.figure(figsize=(4, 3))
plt.imshow(spins4, cmap='coolwarm', interpolation='nearest')
plt.colorbar().remove()
plt.xticks([])
plt.yticks([])
plt.title("1st Monte Carlo simulation of a 2D Ising model \
using the Glauber algorithm")
plt.text(-45, 2, f'Size: {maxDim} by {maxDim}', color='black',
fontsize=10)
plt.text(-45, 5, f'Temperature: {T1}', color='black', fontsize=10)
plt.text(-45, 8, f'Iterations: {N}', color='black', fontsize=10)
plt.text(-45, 11, f'Seed: {seed1}', color='black', fontsize=10)
plt.show()

# displaying second 2D Ising model
print()
plt.figure(figsize=(4, 3))
plt.imshow(spins5, cmap='coolwarm', interpolation='nearest')
plt.colorbar().remove()
plt.xticks([])
plt.yticks([])
plt.title("2nd Monte Carlo simulation of a 2D Ising model \
using the Glauber algorithm")
plt.text(-45, 2, f'Size: {maxDim} by {maxDim}', color='black',
fontsize=10)
plt.text(-45, 5, f'Temperature: {T2}', color='black', fontsize=10)
plt.text(-45, 8, f'Iterations: {N}', color='black', fontsize=10)
plt.text(-45, 11, f'Seed: {seed2}', color='black', fontsize=10)
plt.show()

# plotting graphs of time vs energy & magnetisation for both simulations
print()
plt.plot(time_axis4, energy_axis4, label='Energy4')
plt.plot(time_axis5, energy_axis5, label='Energy5')
plt.plot(time_axis4, mag_axis4, label='magnetisation4')
plt.plot(time_axis5, mag_axis5, label='magnetisation5')
plt.xlabel('Time')
plt.ylabel('Energy/Magnetisation')
plt.title(f'Energy and Magnetisation vs Time at Temperatures {T1} and
{T2}')
plt.legend()
plt.grid(True)
plt.show()
```

```
# phase transition for an Ising model with glauber kinetics.

# we will be using the same setup as the phase xsn demo with metrop
# therefore all we need redefine is the spins array

spins6 = np.zeros((maxDim, maxDim))
for i in range(0, maxDim):
  for j in range(0, maxDim):
    if rs3.random() < p3:
      spins6[i,j] = -1
    else:
      spins6[i,j] = 1

file_name2 = "simulation_results2.csv"

with open(file_name2, 'w', newline='') as csvfile:
  writer = csv.writer(csvfile)
  writer.writerow(['T', 'E', 'm', 'c', 'X'])
  for i in T3:
    # for every temperature, we calculate one value for energy,
magnetisation, specific heat, and magnetic susc

    #calculate arrays
    time_axis6, energy_axis6, mag_axis6 = glauber(spins6, i, N, rs3)

    # turn arrays into single values
    avg_E = np.mean(energy_axis6[-vals_taken:])
    sq_sum = 0
    for j in energy_axis6[-vals_taken:]:
      sq_sum += j**2
    avg_Esq = sq_sum / vals_taken

    avg_mag = np.mean(mag_axis6[-vals_taken:])
    sq_sum = 0
    for k in mag_axis6[-vals_taken:]:
      sq_sum += k**2
    avg_magsq = sq_sum / vals_taken


    c = specific_heat_fn(avg_Esq, avg_E, i, vol)
    chi = magnetic_susc_fn(avg_magsq, avg_mag, i, vol)

    # write single values into row, corresponding to its T
    # time axis not needed
    writer.writerow([i, avg_E, avg_mag, c, chi])

import pandas as pd

df2 = pd.read_csv('simulation_results2.csv')

# writer.writerow(['T', 'E', 'm', 'c', 'X'])


plt.plot(df2['T'], df2['E'], marker = 'o', label = 'Energy')
plt.xlabel('Temperature')
plt.ylabel('Energy')
plt.title('Temperature vs. Energy')
plt.grid(True)
plt.legend()
plt.show()

print()
plt.plot(df2['T'], df2['m'], marker = 'o', label = 'magnetisation')
plt.xlabel('Temperature')
plt.ylabel('magnetisation')
plt.title('Temperature vs. magnetisation')
plt.grid(True)
plt.legend()
plt.show()

print()
plt.plot(df2['T'], df2['c'], marker = 'o', label = 'specific heat')
plt.xlabel('Temperature')
plt.ylabel('specific heat')
plt.title('Temperature vs. specific heat')
plt.grid(True)
plt.legend()
plt.show()

print()
plt.plot(df2['T'], df2['X'], marker = 'o', label = 'magnetic
susceptibility')
plt.xlabel('Temperature')
plt.ylabel('magnetic susceptibility')
plt.title('Temperature vs. magnetic susceptibility')
plt.grid(True)
plt.legend()
plt.show()
```

Kawasaki Algorithm

```
# COP Ising model with kawasaki kinetics
# note to self
# if (i1, j1) == (i2, j2):    # if the indexes represent the same location
# if spins[i1,j1] == spins[i2,j2]:  # if the indexes are the same value



# function to calculate dE
def pot_dE_fn_Kaw(spins, i1, j1, i2, j2):
  left_p = spins[i1,(j1-1) % spins.shape[1]]
  right_p = spins[i1,(j1+1) % spins.shape[1]]
  top_p = spins[(i1-1) % spins.shape[0],j1]
  bottom_p = spins[(i1+1) % spins.shape[0],j1]

  left_q = spins[i2,(j2-1) % spins.shape[1]]
  right_q = spins[i2,(j2+1) % spins.shape[1]]
  top_q = spins[(i2-1) % spins.shape[0],j2]
  bottom_q = spins[(i2+1) % spins.shape[0],j2]

  # the following checks are to make sure of p =/ j and q =/ i.
  # we simply set to zero so that they are not counted within the sum
later.
  # note we will only be checking the former, as the latter can be
  # considered a result of the former.

  # another advantage of this method is that we are not changing the
actual
  # index's value to 0, we are just changing how it will affect the value
of dE

  if (i1,(j1-1) % spins.shape[1]) == (i2,j2):  # leftp = j
    left_p = 0
    right_q = 0
  if (i1,(j1+1) % spins.shape[1]) == (i2,j2):
    right_p = 0
    left_q = 0
  if ((i1-1) % spins.shape[0],j1) == (i2,j2):
    top_p = 0
    bottom_q = 0
  if ((i1+1) % spins.shape[0],j1) == (i2,j2):
    bottom_p = 0
    top_q = 0

  dE = 2 * ( spins[i1,j1] * (left_p + right_p + top_p + bottom_p) + \
             spins[i2,j2] * (left_q + right_q + top_q + bottom_q))

  return dE


# function to carry out the kawasaki algorithm
def kawasaki(spins, T, N, randomState):
  E = hamiltonian(spins)
  energy_axis = np.zeros(sweeps)
  energy_axis[0] = E
  time_axis = np.arange(sweeps)      # separate time axis for each
```

```
simulation
  for sweep in range(sweeps):
    for iteration in range(vol):
      i1 = randomState.choice(range(spins.shape[0]))
      j1 = randomState.choice(range(spins.shape[1])) # randomly selected
site1
      i2 = randomState.choice(range(spins.shape[0]))
      j2 = randomState.choice(range(spins.shape[1])) # randomly selected
site2

      if spins[i1,j1] == spins[i2,j2] or (i1,j1) == (i2,j2):
        dE = 0.0
        E += dE
      else:
        dE = pot_dE_fn_Kaw(spins, i1, j1, i2, j2)
        if dE < 0:
          spins[i1, j1] = -spins[i1, j1]
          spins[i2, j2] = -spins[i2, j2]
          E += dE
        elif randomState.random() < (1 / (1 + np.exp(dE / T))):
          spins[i1, j1] = -spins[i1, j1]
          spins[i2, j2] = -spins[i2, j2]
          E += dE
    energy_axis[sweep] = E * one_over_vol   # per site
  return time_axis, energy_axis

# at this point we have defined the kawasaki algorithm itself, and the
function
# for finding potential energy change of a spin-exchange (which is of
course
# then used within kawasaki)

# creating initial arrays
T7 = 2.5
T8 = 2.5
p7 = 0.5
p8 = 0.5
seed7 = 123
seed8 = 100
rs7 = np.random.RandomState(seed7)
rs8 = np.random.RandomState(seed8)
spins7 = np.zeros((maxDim, maxDim))
spins8 = np.zeros((maxDim, maxDim))
for i in range(0, maxDim):
  for j in range(0, maxDim):
    if rs7.random() < p7:
      spins7[i,j] = -1
    else:
      spins7[i,j] = 1
    if rs8.random() < p8:
      spins8[i,j] = -1
    else:
      spins8[i,j] = 1
```

```
# running the algorithm on the arrays defined
# and obtaining values for energy
time_axis7, energy_axis7 = kawasaki(spins7, T7, N, rs7)
time_axis8, energy_axis8 = kawasaki(spins8, T8, N, rs8)

# displaying first 2D COP Ising model
print()
plt.figure(figsize=(4, 3))
plt.imshow(spins7, cmap='coolwarm', interpolation='nearest')
plt.colorbar().remove()
plt.xticks([])
plt.yticks([])
plt.title("1st Monte Carlo simulation of a 2D COP Ising model \
using the Kawasaki algorithm")
plt.text(-45, 2, f'Size: {maxDim} by {maxDim}', color='black',
fontsize=10)
plt.text(-45, 5, f'Temperature: {T7}', color='black', fontsize=10)
plt.text(-45, 8, f'Iterations: {N}', color='black', fontsize=10)
plt.text(-45, 11, f'Seed: {seed7}', color='black', fontsize=10)
plt.show()

# displaying second 2D COP Ising model
print()
plt.figure(figsize=(4, 3))
plt.imshow(spins8, cmap='coolwarm', interpolation='nearest')
plt.colorbar().remove()
plt.xticks([])
plt.yticks([])
plt.title("2nd Monte Carlo simulation of a 2D COP Ising model \
using the Kawasaki algorithm")
plt.text(-45, 2, f'Size: {maxDim} by {maxDim}', color='black',
fontsize=10)
plt.text(-45, 5, f'Temperature: {T8}', color='black', fontsize=10)
plt.text(-45, 8, f'Iterations: {N}', color='black', fontsize=10)
plt.text(-45, 11, f'Seed: {seed8}', color='black', fontsize=10)
plt.show()

# plotting graphs of time vs energy
print()
plt.plot(time_axis7, energy_axis7, label='Energy7')
plt.plot(time_axis8, energy_axis8, label='Energy8')
plt.xlabel('Time')
plt.ylabel('Energy')
plt.title(f'Energy vs Time at Temperature {T8}')
plt.legend()
plt.grid(True)
plt.show()
```

```
# phase separation for an Ising model with kawasaki kinetics.

# we will be using the same setup as the phase xsn demo with metrop
# therefore all we need redefine is the spins array

spins9 = np.zeros((maxDim, maxDim))
for i in range(0, maxDim):
  for j in range(0, maxDim):
    if rs3.random() < p3:
      spins9[i,j] = -1
    else:
      spins9[i,j] = 1

file_name3 = "simulation_results3.csv"

with open(file_name3, 'w', newline='') as csvfile:
  writer = csv.writer(csvfile)
  writer.writerow(['T', 'E', 'c'])
  for i in T3:
    # for every temperature, we calculate one value for energy and
specific heat

    #calculate arrays
    time_axis9, energy_axis9, = kawasaki(spins9, i, N, rs3)

    # turn arrays into single values
    avg_E = np.mean(energy_axis9[-vals_taken:])
    sq_sum = 0
    for j in energy_axis9[-vals_taken:]:
      sq_sum += j**2
    avg_Esq = sq_sum / vals_taken

    c = specific_heat_fn(avg_Esq, avg_E, i, vol)

    # write single values into row, corresponding to its T
    # time axis not needed

    # graphic of model at T = 1,2,3,4,5
    if i % 1.0 == 0:
      print()
      plt.figure(figsize=(4, 3))
      plt.imshow(spins9, cmap='coolwarm', interpolation='nearest')
      plt.colorbar().remove()
      plt.xticks([])
      plt.yticks([])
      plt.title(f"Monte Carlo simulation ({int(i)}) of a 2D COP Ising
model \
      using the Kawasaki algorithm")
      plt.text(-45, 2, f'Size: {maxDim} by {maxDim}', color='black',
fontsize=10)
      plt.text(-45, 5, f'Temperature: {i}', color='black', fontsize=10)
      plt.text(-45, 8, f'Iterations: {N}', color='black', fontsize=10)
      plt.text(-45, 11, f'Seed: {seed3}', color='black', fontsize=10)
      plt.show()

    writer.writerow([i, avg_E, c])
```

```
# we have now obtained results for energy and specific heat for different
# temperatures

import pandas as pd

df3 = pd.read_csv('simulation_results3.csv')

# writer.writerow(['T', 'E', 'c'])

print()
plt.plot(df3['T'], df3['E'], marker = 'o', label = 'Energy')
plt.xlabel('Temperature')
plt.ylabel('Energy')
plt.title('Temperature vs. Energy')
plt.grid(True)
plt.legend()
plt.show()

print()
plt.plot(df3['T'], df3['c'], marker = 'o', label = 'specific heat')
plt.xlabel('Temperature')
plt.ylabel('specific heat')
plt.title('Temperature vs. specific heat')
plt.grid(True)
plt.legend()
plt.show()
```

Social Applications

```
# kawasaki APPLICATIONS code
# for the paper: Immigration, integration and ghetto formation by
# Hildegard Meyer-Ortmanns

# the following variables are to be changed for each test
concentration = 0.2
test_T = 1.25
N = 10**7

# the following is initial random array creation with fixed
# proportion/concentration of +1s to -1s
# note migrants denoted by +1, natives denoted by -1
T_c = 2.269
T = test_T * T_c
maxDim = 50
seed = 42
rs = np.random.RandomState(seed)
num_ones = int(concentration * maxDim**2)
num_neg_ones = int(maxDim **2 - num_ones)
spins = np.concatenate([np.ones(num_ones), -np.ones(num_neg_ones)])
np.random.shuffle(spins)
```

```
    spins = spins.reshape((maxDim, maxDim))

    # apply kawasaki algorithm to each test's spec
    time_axis, energy_axis = kawasaki(spins, T, N, rs)

    # print model for each test
    plt.figure(figsize=(4, 3))
    plt.imshow(spins, cmap='coolwarm', interpolation='nearest')
    plt.colorbar().remove()
    plt.xticks([])
    plt.yticks([])
    plt.title(f"Monte Carlo simulation of a 2D COP Ising model \
using the Kawasaki algorithm")
    plt.text(-45, 2, f'Size: {maxDim} by {maxDim}', color='black',
fontsize=10)
    plt.text(-45, 5, f'Temperature T/Tc: {test_T}', color='black',
fontsize=10)
    plt.text(-45, 8, f'Iterations: {N}', color='black', fontsize=10)
    plt.text(-45, 11, f'Seed: {seed}', color='black', fontsize=10)
    plt.text(-45, 14, f'Concentration: {concentration}', color='black',
fontsize=10)
    plt.show()

    # print graph of time vs energy
    print()
    plt.plot(time_axis, energy_axis, label='Energy')
    plt.xlabel('Time')
    plt.ylabel('Energy')
    plt.title(f'Energy vs Time at Temperature T/Tc: {test_T}, Concentration:
{concentration}, after {N} Iterations.')
    plt.legend()
    plt.grid(True)
    plt.show()
```