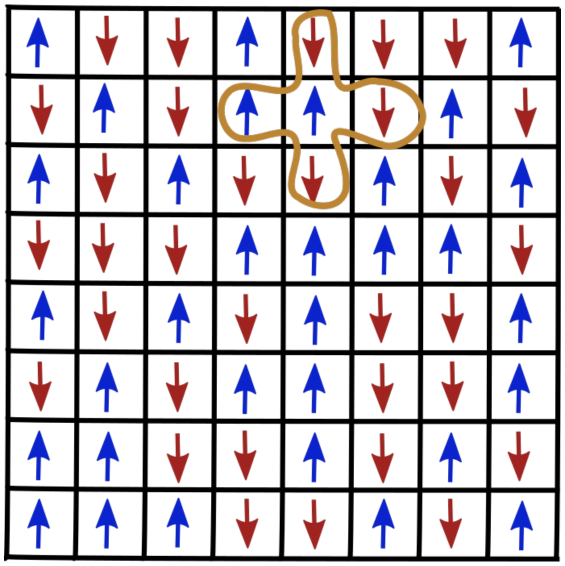
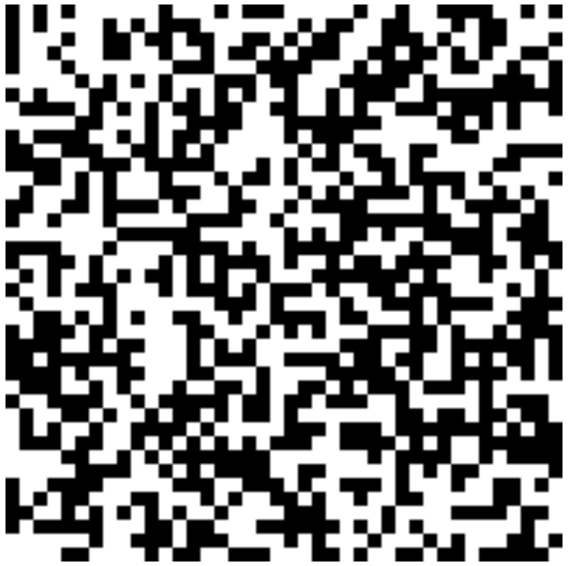
Project 10 – Phase Transitions of the Ising Model

The Ising model is a simplified discrete classical system for representing phase transitions. It's typically employed with isometric (crystalline) geometry, with periodic boundary conditions. This is the technique we use here, in 2 dimensions, to model magnetic phase transitions. The phase transition in physical terms is between a ferromagnetic (ordered) state and a paramagnetic (disordered) one. In non-physical terms, this is akin to a binary grid (made of 1's and 0's), which undergoes a transition between a low entropy (high symmetry) state, and a highly entropic one. Our bits represent electron spins s_i, whose spin vectors are taken to be parallel to their magnetic moments (as opposed to antiparallel) by convention.





Here we show the arbitrary initial state of our "spin" lattice (left), compared with how the program represents our system (right) – with s^+ as 1 and s^- as 0. The colored outline (left) emphasizes how each spin element is only affect by its four adjacent spin elements – this both simplifies the calculation and is also physically realistic, as we'll see.

The phase transition we seek to model is consequence of two competing "forces": these are the organizing and disorganizing forces. Physically, the organizing force is the tendency for each electron's spin induced magnetic moment to align parallel with its neighbors'. And the disorganizing force is none other than temperature.

Since we're not dealing with any externally applied field in this project – and certainly not a variable one, and since our system is holonomic and monogenic, the Hamiltonian of the system is equal to the total energy E, where

$$E = -J \sum_{<ij>} s_i \cdot s_j$$

Where J is the organizing force, acting on all nearest neighbor $\langle ij \rangle$ spin pair's $s_i \cdot s_j$.

The negative here guarantees the fact that antiparallel pairs $s^+ \cdot s^-$ have higher energy than parallel pairs, which is to say the system *prefers* to align its spins at low temperatures.

To implement temperature T, we have to consider the probability P of any given total energy state for the system E_α, which is given by the Boltzmann factor

$$P(E_\alpha) = e^{-\frac{E_\alpha}{k_B T}}$$

We will not go into the rationale for this specific canonical form for probability distributions, other than to say it is necessary to reconcile informational (Shannon) entropy $S = k_B \sum p_i \ln(p_i)$, with thermodynamic entropy $dS = \delta(Q/T)$ for reversible systems.

Our first approximation is that of the *Mean Field Approximation,* which essentially replaces a tedious, and often intractable, n-body problem, with a 1-body problem, by summing over average *effective fields* (the details of which are left aside).

We start by considering a single spin state, and if our energy is to be unity (it will actually be proportional to J), we see probabilities p for s^+ and s^- are

$$p^+ = C * e^{\frac{1}{k_B T}}$$
$$p^- = C * e^{-\frac{1}{k_B T}}$$

Where the coefficient C is to ensure our probabilities add up to 1 (conserving information, each element is definitely either 1 or zero $\Rightarrow p^+ + p^- = 1$).

$$\therefore C = \frac{1}{e^{\frac{1}{k_B T}} + e^{-\frac{1}{k_B T}}}$$

And so to get our approximation for the effective field on a single spin – which can either be 1 or 0 (representationally speaking), we go from discrete to continuous (which is okay because we will re-sum these individual effects) and look for a function whose *slope* is either 1 or 0.

We therefore take the difference between a s^+ and s^- (a numerical derivative),

$$\langle s \rangle = p^+ - p^- = C * e^{\frac{1}{k_B T}} - C * e^{-\frac{1}{k_B T}} = \tanh\left(\frac{1}{k_B T}\right) \star$$

Those familiar with the $\tanh(x)$ function will find this to be exactly what we need.
⋆ The constants are later absorbed, or compensated, by the proportionality of J/T.

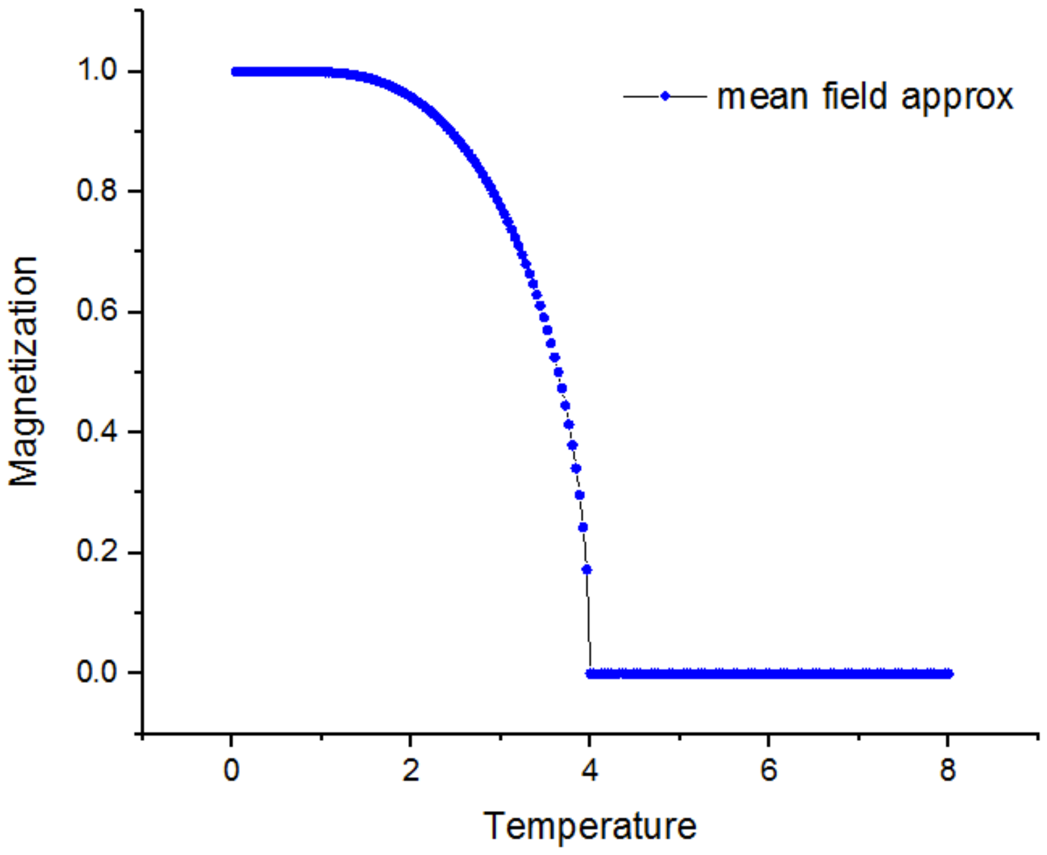
Re-summing (numerical integration), we find

$$E_{eff} = -J\sum\langle s\rangle = -\sum \tanh\left(\frac{J\langle s\rangle}{k_B T}\right)$$

Here we haven't forgotten that J is a function of s, or, more aptly, that s is a function of itself – i.e., we've shown the *implicit relation* between spins, which needs to be solved numerically.

We use the Newton Raphson Method, and add a factor of 4 for nearest neighbors, generating the following plot of magnetization $M(T)$, where M is simply

$$M = \sum_i s_i$$



We could make it so our plot goes from $[1, -1]$, so that it better represents spin, but that is mere intuition (the very first figure is a misrepresentation); for real spin states, despite intuition, $s^+ \perp s^-$. Moreover, this representation better suits our digital needs. Additionally, our temperature units are contrived to fit units of J/k_B.

As we see, while temperature rises, the net magnetization drops drastically to zero.

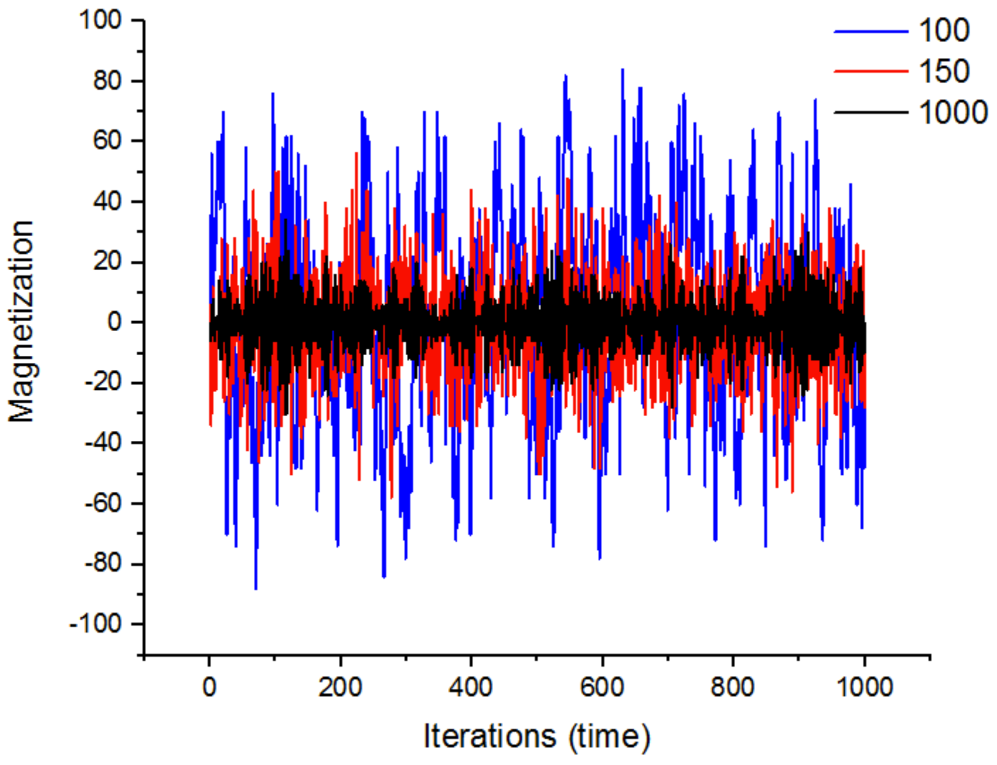
This is because the spins are initially all aligned, yet as temperature increases, they reluctantly start cancelling out, until a critical temperature T_c (here at 4) – where temperature dominates the interactions and no net magnetization remains.

As one might guess, this approximation is only so good. It's an excellent tool for modeling our phase transition to 1st order. But importantly, this particular phase transition (ferromagnetic $\leftrightarrow$ paramagnetic) is categorically 2nd order. Here we cannot well evaluate the critical point T_c, as the slope there is not analytical, which is to say it is discontinuous at T_c.

Therefore we turn to a more advanced technique, the *Monte Carlo Method*, which uses stochastic RNGs to simulate more realistic temperature fluctuations, as well as implementing randomized initial conditions.

The details can be seen in the Fortran code provided at the end, but the essential concept is rather straightforward:

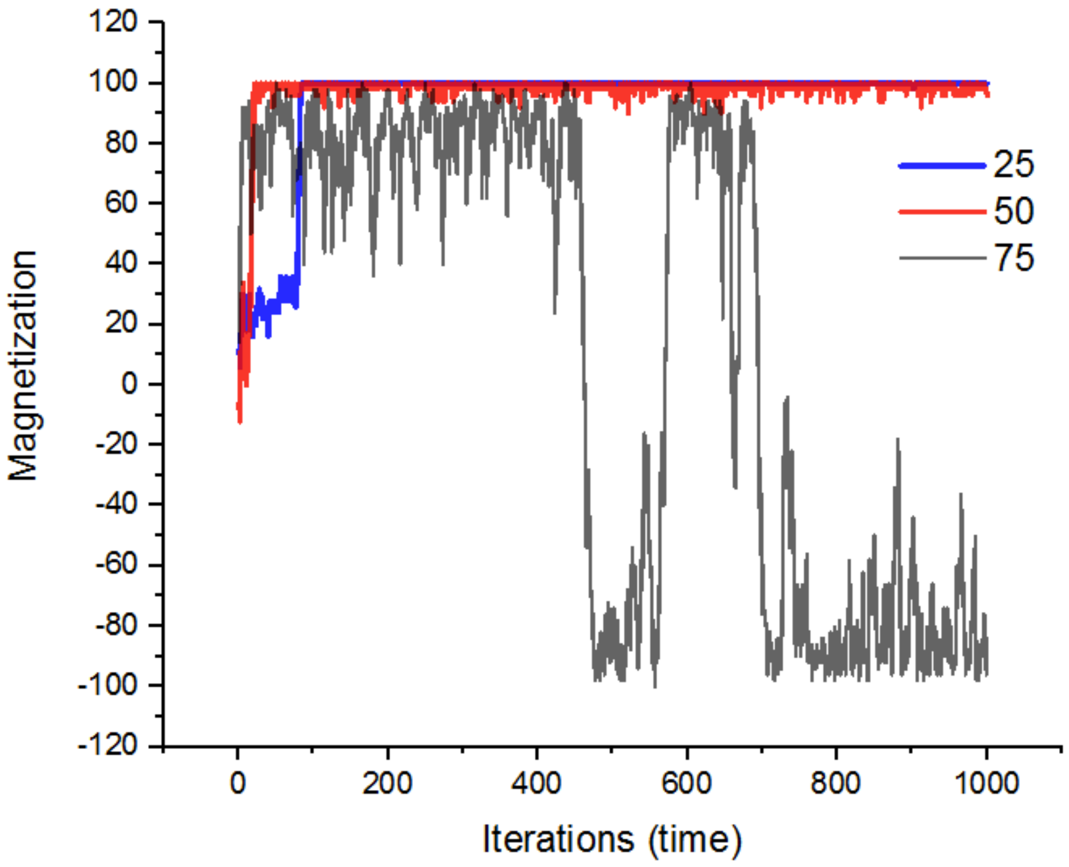
- Start with an arbitrary configuration, at a given temperature
- Calculate the energy of each spin element E_i
    - Compare that energy with the energy required to flip a spin ⋆
        - If $E_i < E_{flip}$ then flip the spin
        - Otherwise, flip with a random chance proportional to the temperature
- Iterate this process for a constant duration



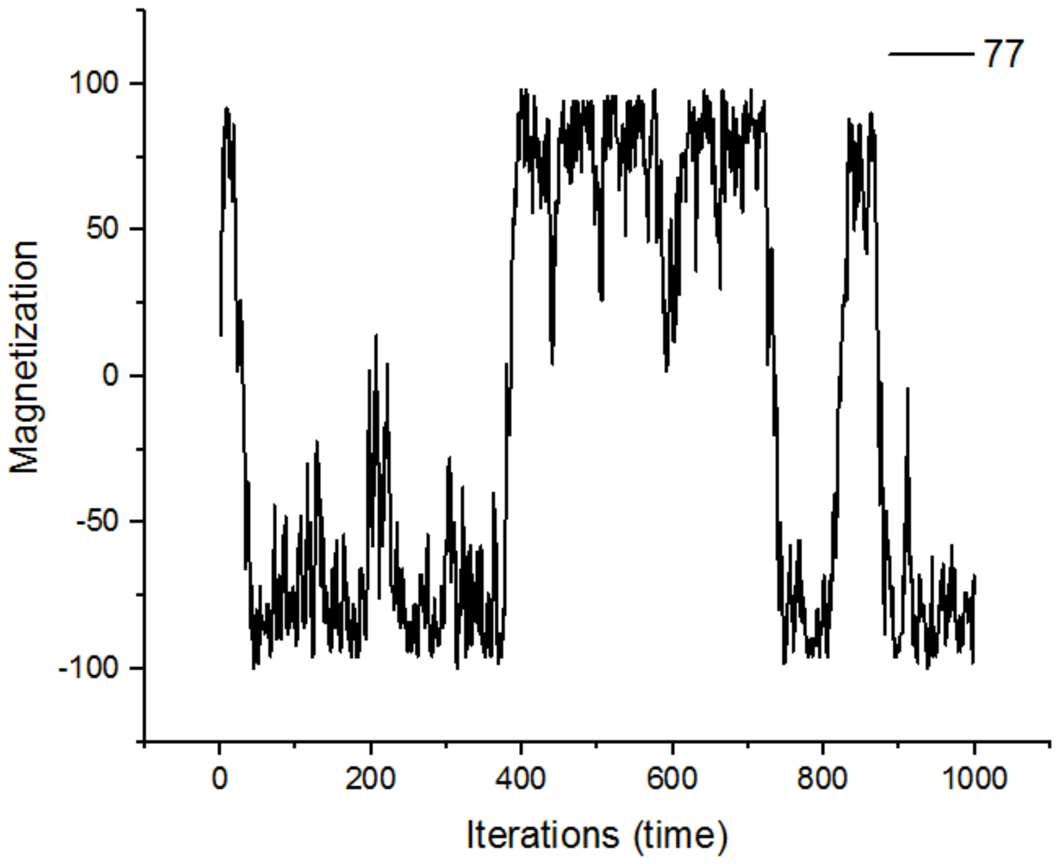
Now we are able to plot the dynamics of the system – specifically the net magnetization M, for a given number of iterations (here 1000) at a specified temperature of $T = [100, 150, 1000]$.

We can then zero-in on and analyze T_c.

First we recognize the wild fluctuations in M, centered about 0. This behavior is indicative of $T \gg T_c$. In this temperature regime, we are *far from equilibrium,* and here temperature is too high for any magnetic domains to materialize for long – if this was a sheet of metal, it may be paramagnetic, but it would not exhibit any stable magnetic field. Another interesting dynamic to notice here is that with higher temperature, M is squeezed closer to zero, as magnetic moments are canceled out by stochastic fluctuations quicker and quicker as they form.



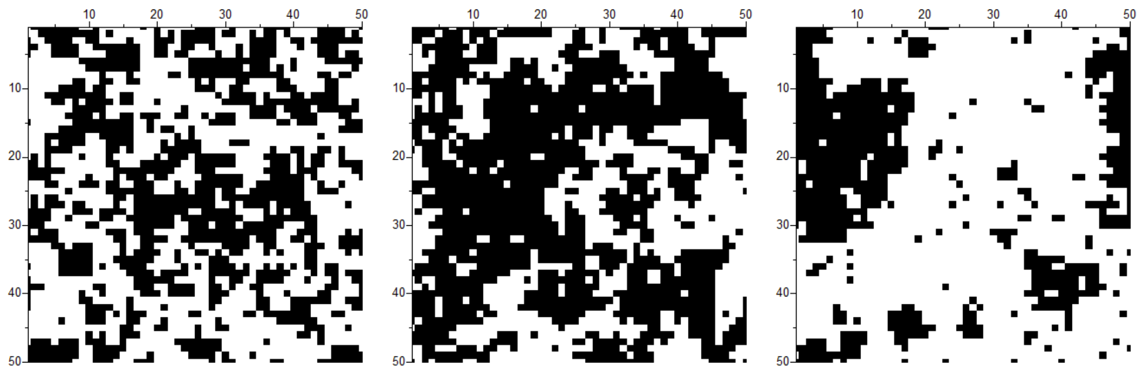
Here above we see a fascinating spread: At $T = 25$, the system takes awhile to reach 100 M (total alignment for a 10x10 grid). At $T = 50$, the system quickly transitions, but then fluctuates slightly about $M \sim 100$. At $T = 75$, we see it tries to align completely, but temperature keeps kicking it out of order, and as a result it is able to access full reversal $M \sim -100$, before getting bumped back out. With these results we can estimate that, given this specific system, $T_c \approx 75$.



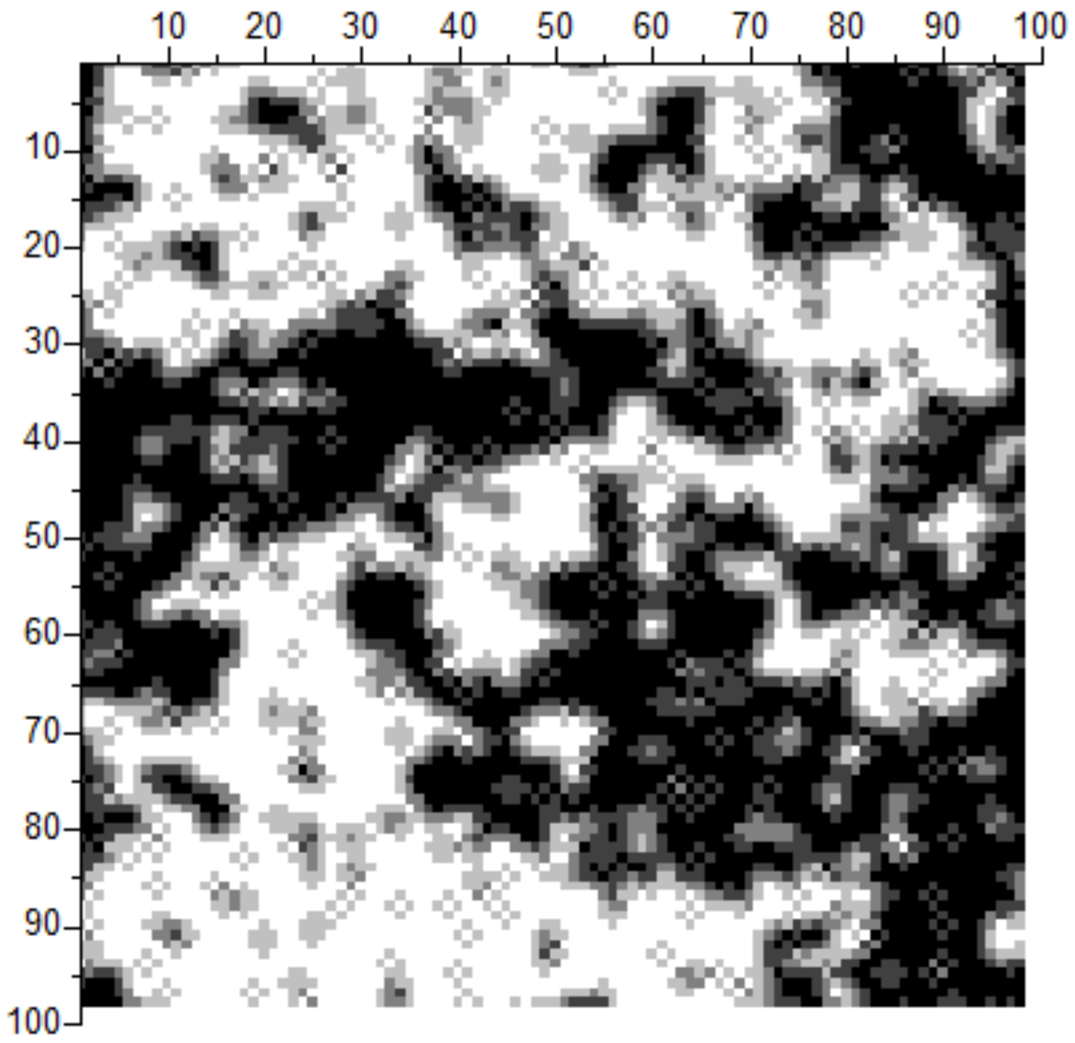
At $T = 77$, we are about as close as we can get to T_c, where the system oscillates back and forth between spin up and spin down, without ever reaching total polarity.

This regime, $\sim T_c$, is of great interest to a variety of physics – think of the value of the point at which water boils / freezes, or perhaps even the physics of life!

To investigate further, we take one last look at the dynamics around T_c. To do this, we use our previous model, and modify it for direct observation of the grid – now expanded to 50x50.



Here we see, from left to right, $T = 100, 85, 75$. Finally, it is revealed that somewhere between total order and total disorder is where the interesting patterns arise. Here below is a re-run $\sim T_c$ at higher iteration and grid count, with one run of a smoothing calculation to improve clarity. Admittedly there isn't much to see, but what's important is that both large and small scale structures are allowed to form (for a moving example, see https://upload.wikimedia.org/wikipedia/commons/e/e6/Ising_quench_b10.gif).



These patterns show up in nature as Li symmetry, relevant to what's called in the West *Turing patterns*. They are mere manifestations of a balancing act between various forces. In physics this is often of primary concern. Consider how liquid is the manifestation of a sort balance between kinetic and potential forces – between solid (where potential forces keep things organized), and gas (where kinetics decorrelate the elements). It is here one gets all the interesting properties of fluid dynamics, as well as, in more special cases, superfluid and superconductive states. These dynamics can be modeled and investigated with more advanced versions of the same methods used here – e.g., spin glass systems.

Below is modified Fortran code, which produced all the Monte Carlo data.

```
program montecarlo
real sm(100,100)
real esm(100,100)
dt=0.03
print*, "Enter itmp"
read(5,*) itmp
open(7,file="mag-temp")
110   format(100(2x,e12.5))
smtot=0.0
do i=1,100
do j=1,100
r1=rand(0)
sm(i,j)=1.0
if (r1<0.5) then
sm(i,j)=-1.0
endif
smtot=smtot+sm(i,j)
enddo
enddo
temp=itmp*dt
do its=1,500000 ! timesteps
smtot=0.0
etot=0.0
do i=2,99
do j=2,99
sm4=sm(i+1,j)+sm(i-1,j)+sm(i,j+1)+sm(i,j-1)
esm(i,j)=-sm(i,j)*sm4
call flip(sm(i,j),esm(i,j),temp)
smtot=smtot+sm(i,j)
etot=etot+esm(i,j)
enddo
enddo
sm11=sm(1,2)+sm(2,1)+sm(1,100)+sm(100,1)
esm(1,1)=-sm(1,1)*sm11
call flip(sm(1,1),esm(1,1),temp)
smtot=smtot+sm(1,1)
etot=etot+esm(1,1)

sm1001=sm(1,1)+sm(99,1)+sm(100,100)+sm(100,2)
esm(100,1)=-sm(100,1)*sm1001
call flip(sm(100,1),esm(100,1),temp)
smtot=smtot+sm(100,1)
etot=etot+esm(100,1)

sm1100=sm(1,99)+sm(2,100)+sm(1,1)+sm(100,100)
esm(1,100)=-sm(1,100)*sm1100
call flip(sm(1,100),esm(1,100),temp)
smtot=smtot+sm(1,100)
etot=etot+esm(1,100)

sm100=sm(100,99)+sm(99,100)+sm(1,100)+sm(100,1)
esm(100,100)=-sm(100,100)*sm100
call flip(sm(100,100),esm(100,100),temp)
smtot=smtot+sm(100,100)
etot=etot+esm(100,100)

do j=2,99
sm1j=sm(1,j-1)+sm(2,j)+sm(1,j+1)+sm(100,j)
esm(1,j)=-sm(1,j)*sm1j
call flip(sm(1,j),esm(1,j),temp)
smtot=smtot+sm(1,j)
etot=etot+esm(1,j)

sm100j=sm(100,j-1)+sm(99,j)+sm(100,j+1)+sm(1,j)
esm(100,j)=-sm(100,j)*sm100j
call flip(sm(100,j),esm(100,j),temp)
smtot=smtot+sm(100,j)
etot=etot+esm(100,j)
enddo

do i=2,99
sm1i=sm(i-1,1)+sm(i,2)+sm(i+1,1)+sm(i,100)
esm(i,1)=-sm(i,1)*sm1i
call flip(sm(i,1),esm(i,1),temp)
smtot=smtot+sm(i,1)
etot=etot+esm(i,1)

sm100i=sm(i-1,100)+sm(i,99)+sm(i+1,100)+sm(i,1)
esm(i,100)=-sm(i,100)*sm100i
call flip(sm(i,100),esm(i,100),temp)
smtot=smtot+sm(i,100)
etot=etot+esm(i,100)
enddo
print*, its
enddo ! time steps
do i=2,99
write(7,110) (0.25*(sm(i+1,j)+sm(i-1,j)+sm(i,j+1)+sm(i,j-
1)), j=2,99)
enddo
end

subroutine flip(smf,esmf,temp)
esm1=-esmf
if(esm1<esmf) then
smf=-smf !flip
esmf=esm1
else
bf=exp(-2.0*esm1/temp)
r1=rand(0)
if(r1<bf) then
smf=-smf
emsf=esm1
endif
endif
return
end
```