Lisa Larrimore
Physics 114 - Seminar 14

Monte Carlo Simulation of the 2D Ising Model

The Metropolis Algorithm

We know that the expectation value of an observable A can be written as

$$\langle A \rangle = \frac{\sum_r A_r e^{-\beta E_r}}{\sum_r e^{-\beta E_r}}, \tag{1}$$

where A_r is the value of A for the state r. So given a system that has a discrete number of states, we could, using a computer, calculate A for each state and weight these values by their Boltzman factors to find the average A. This might be feasible for a system with a small number of states, but if we have a 20×20 spin lattice interacting via the Ising model, there are 2^{400} states, so we cannot possibly examine all of them.

What if we decide to just sample some of the states? How would we decide which ones? This is where the "Monte Carlo" part comes in. Named for the Mediterranean casino town, a Monte Carlo method is any algorithm that involves a pseudorandom number generator.

One (bad) way of using random numbers would be to randomly pick a lot of states, measure A for each of them, and weight these values of A by their Boltzman factors. We might get close to the right answer if we sampled a lot of states, but we would spend a lot of time calculating A for states that contribute very little to the final result (an Ising lattice at very high temperature is unlike to be in the state with all spins pointing in one direction).

Instead of sampling (measuring parameters like A for) a lot of states and then weighting them by their Boltzman factors, it makes more sense to *choose states based on their Boltzman factors* and to then weight them equally. This is known as the Metropolis algorithm, which is an *importance sampling* technique. One pass through the algorithm is described here:

1. A trial configuration is made by randomly choosing one spin.
2. The energy difference of the trial state relative to the present state, δE, is calculated.
3. If $\delta E \leq 0$, the trial state is energetically favorable and thus accepted. Otherwise, a random number $0 \leq \eta \leq 1$ is generated, and the new state is only accepted if $\exp(-\beta \delta E) > \eta$. This condition can be rewritten as $-\beta \delta E > \log \eta$, which is what I used in the code.

Calculating Observables

We can obtain some qualitative information about our simulation by watching the spin array during a simulation. I have written an IDL program, `see_spins.pro`, that allows us to do this. For high temperatures, the spins remain randomly aligned after long periods of equilibration, whereas for low temperatures, the spins end up pointing in mostly the same direction.

To get more quantitative results, we can measure the energy and the magnetization at each step of the routine. Before we start taking statistics, we should allow the system to equilibrate for a long time (my code equilibrates for `nequil` passes). We can then measure the magnetization by taking the sum of all the spins in the lattice, and we can calculate the energy by determining the energy for each spin and dividing by two for double counting.

What about the specific heat or susceptibility? There isn't a good way to claculate a derivative of the partition function in our code, but it turns out that the specific heat can also be written in terms of the variance of the energy:

$$\begin{aligned} C_V =& \frac{\partial \langle E \rangle}{\partial T} \\ =& -\frac{\beta}{T}\frac{\partial \langle E \rangle}{\partial \beta} \\ =& \frac{\beta}{T}\frac{\partial^2 \ln Z}{\partial \beta^2} \\ =& \frac{\beta}{T}\frac{\partial}{\partial \beta}\left(\frac{1}{Z}\frac{\partial Z}{\partial \beta}\right) \\ =& \frac{\beta}{T}\left[\frac{1}{Z}\frac{\partial^2 Z}{\partial \beta^2} - \frac{1}{Z^2}\left(\frac{\partial Z}{\partial \beta}\right)^2\right] \\ =& \frac{\beta}{T}\left[\left\langle E^2 \right\rangle - \langle E \rangle^2\right]. \end{aligned} \tag{2}$$

Incidentally, this is known as the Fluctuation Dissipation Theorem.

Similarly, the magnetic susceptibility, χ, can be written in terms of the variance in the magnetization:

$$\begin{aligned} \chi =& \frac{\partial \langle M \rangle}{\partial H} \\ =& \beta\left[\left\langle M^2 \right\rangle - \langle M \rangle^2\right]. \end{aligned} \tag{3}$$

So by keeping statistics on E, E^2, M, and M^2, we can plot the energy, the magnetization, the specific heat, and the magnetic susceptibility. On each of these graphs, each circle represents an independent run of $100,000$ steps of equilibration and $100,000$ more steps of data taking.

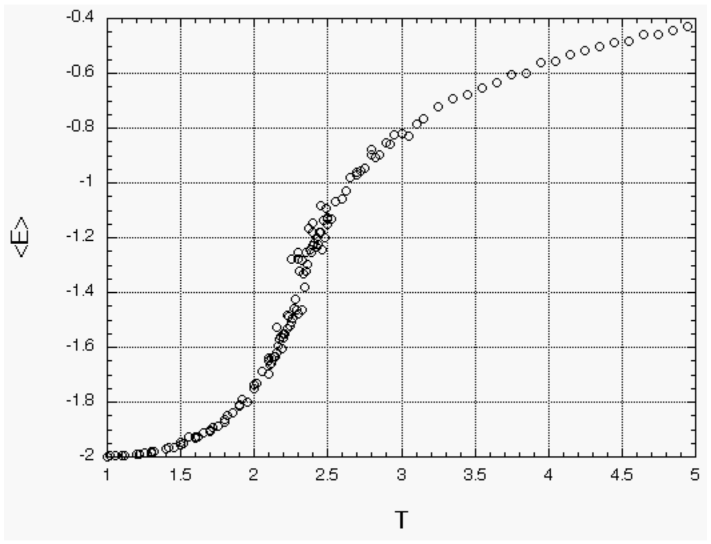

Figure 1: The energy is a continuous function of temperature, which, as we expect, increases as a function of T.

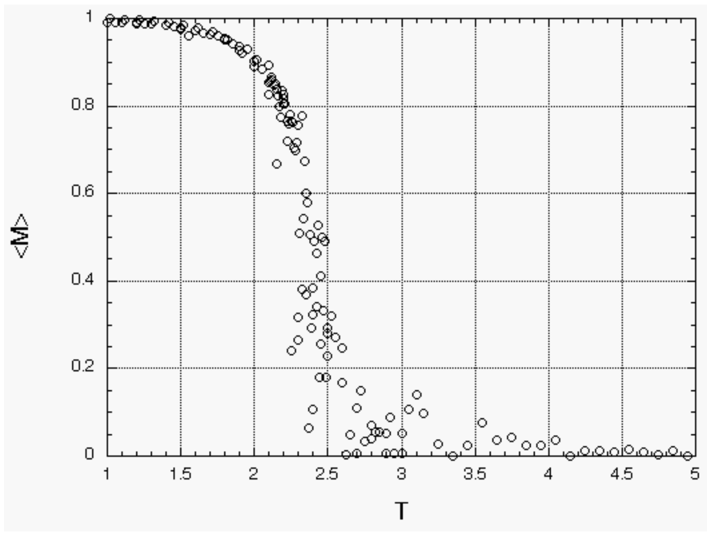

Figure 2: The magnetization drops off sharply near the critical temperature, which, in our units where $k = J = 1$, is approximately 2.3.

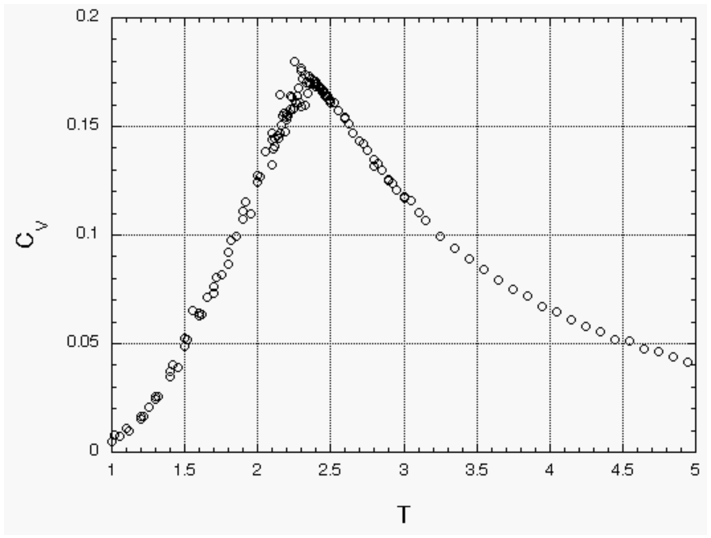

Figure 3: The specific heat has a peak at the critical temperature.

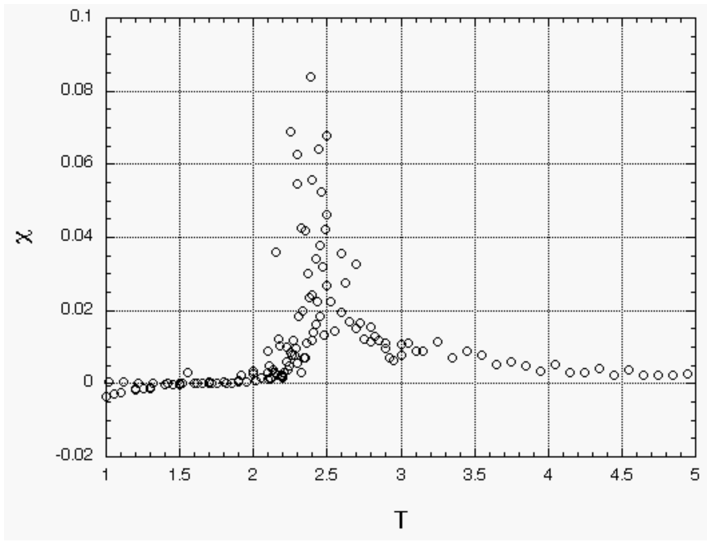

Figure 4: The magnetic susceptibility has a sharp jump at the critical temperature.

Codes

This FORTRAN 90 code generates statistics on energy, heat capacity, magnetization, and magnetic susceptibility for a range of temperatures:

```
program ising  ! 2D Monte Carlo Simulation of Ising Model

! Lisa Larrimore, lisal@sccs.swarthmore.edu
! 3 May 2002
! Physics 114 Final Project

! This program is adapted from the Ising Model program written in
! BASIC by Elaine Chandler that appears on p. 184 of David Chandler's
! Introduction to Modern Statistical Mechanics.

! The input parameters for this program are in "ising.in", and they
! allow the size, length, and initial configuration of the simulation
! to be changed. See comments in file.

! This program has three output files:
!
!   "spin-array"        Contains snapshots of the spin lattice at the end of
!                       each temperature run (or throughout the middle of the
!                       run, if only looking at one temperature). Can be
!                       visualized with the IDL program see_spins.pro
!
!   "magnetization"     Contains four columns: each temperature, the
!                       average magnetization at that temp, the ave magnetizaion
!                       squared at that temp, and the susceptibility.
!
!   "energy"            Contains four columns: each temperature, the
!                       average energy at that temp, the ave energy squared
!                       at that temp, and the heat capacity.

implicit none

! Variable declarations:
integer :: i, j, m, n, m2, n2   ! dummy integers
integer, allocatable :: A(:,:)  ! matrix containing spins
integer :: nrows, ncols         ! number of rows and cols of A
real :: temp, beta              ! temperature, inverse temperature
integer :: ConfigType           ! starting configuration type
integer :: npass                ! number of passes for MC algorithm
integer :: ipass                ! the current pass number
integer :: nequil               ! number of equilibration steps
integer :: trial_spin           ! values of changed spin
real :: high_temp               ! starting temp for scan
real :: low_temp                ! final temp for scan
real :: temp_interval           ! interval between scan points
integer :: nscans               ! number of scans (each at diff T)
integer :: iscan                ! current scan number
logical :: MovieOn              ! set to .true. to make movie of 1 temp
real :: deltaU                  ! change in energy between 2 configs
real :: deltaU1, deltaU         ! energy changes for lattice gas
real :: log_eta                 ! log of random number to compare to
real :: magnetization           ! magnetization of all spins in lattice
real :: magnetization_ave       ! cumulative average magnetization
real :: magnetization2_ave      ! cumulative average of mag. squared
real :: energy                  ! energy of all spins in lattice
real :: energy_ave              ! cumulative average of energy
real :: energy2_ave             ! cumulative average of energy squared
integer :: output_count         ! # times things have been added to averages

print*, "_______________MONTE CARLO 2D ISING MODEL_______________"
print*, "Monte Carlo Statistics for 2D Ising Model with"
print*, "  periodic boundary conditions."
print*, "The critical temperature is approximately 2.3, as seen on"
print*, "  Chandler p. 123."

! Read in input parameters from file "ising.in"
open(unit=11,file='ising.in',status='old',action='read')
read(11,*);read(11,*) nrows
read(11,*);read(11,*) ncols
read(11,*);read(11,*) npass
read(11,*);read(11,*) nequil
read(11,*);read(11,*) high_temp
read(11,*);read(11,*) low_temp
read(11,*);read(11,*) temp_interval
read(11,*);read(11,*) ConfigType
read(11,*);read(11,*) MovieOn
close(11)

! Set the dimensions of the matrix of spin arrays. This program uses
! periodic boundary conditions, so the first two rows and columns are
! the same as the last two.
allocate(A(nrows+2,ncols+2))

! Open output files:
open(unit=32,file='spin-array',status='replace',action='write')
write(32,*) nrows
write(32,*) ncols
nscans = int((high_temp - low_temp)/temp_interval) + 1
if (MovieOn) then
  write(32,*) 51
  write(32,*) 1
else
  write(32,*) nscans
  write(32,*) 2
endif

open(unit=33,file='magnetization',status='replace',action='write')
write(33,*) "temp    ave_magnetization    ave_magnetization^2  susceptibility"
open(unit=34,file='energy',status='replace',action='write')
write(34,*) "temp    ave_energy    ave_energy^2    C_v"

scan_loop: do iscan = 1, nscans
```

```
temp = high_temp - temp_interval*(iscan-1)
print*, "Running program for T =", temp

! Initialize variables
beta = 1.0/temp
output_count = 0
energy_ave = 0.0
energy2_ave = 0.0
magnetization_ave = 0.0
magnetization2_ave = 0.0

! Set up the initial spin configuration.
select case(ConfigType)
  case(1)  ! checkerboard setup
    A(1,1) = 1
    do i = 1, nrows+1
      A(i+1,1) = -A(i,1)
    enddo
    do j = 1, ncols+1
      A(:,j+1) = -A(:,j)
    enddo
    ! (note: the requirement that nrows and ncols are even is to
    ! ensure that the first two rows/cols start out the same as the
    ! last two)
  case(2)  ! interface
    do i = 1, nrows+2
      do j = 1, (ncols+2)/2
        A(i,j) = 1
      enddo
      do j = (ncols+2)/2 + 1, ncols+2
        A(i,j) = -1
      enddo
    enddo
  case(3)  ! unequal interface
    do i = 1, nrows+2
      do j = 1, (ncols+2)/4
        A(i,j) = 1
      enddo
      do j = (ncols+2)/4 + 1, ncols+2
        A(i,j) = -1
      enddo
    enddo
  case default
    print*, "Error! Check ConfigType parameter in ising.in"
    stop
end select

! Main loop containing Monte Carlo algorithm:
MC_passes: do ipass = 0, npass

  ! If MovieOn is .true., write the spin array to an output every
  ! npass/50 steps.
  if ((MovieOn) .and. (mod(ipass,npass/50) == 0)) then
```

```
        do i = 2, nrows+1
          do j = 2, ncols+1
            write(32,*) A(i,j)
          enddo
        enddo
      endif

      ! If ipass is greater than nequil (the number of equilibration steps),
      ! calculate the magnetization and energy:
      if (ipass > nequil) then
        output_count = output_count + 1
        magnetization = sum(A(2:nrows+1,2:nrows+1))/(ncols*nrows*1.0)
        magnetization_ave = magnetization_ave + magnetization
        magnetization2_ave = magnetization2_ave + magnetization**2
        energy = 0.0
        do i = 2, nrows + 1
          do j = 2, ncols + 1
            energy = energy - A(m,n)*(A(m-1,n)+A(m+1,n)+A(m,n-1)+A(m,n+1))
          enddo
        enddo
        ! Divide the energy by the total number of spins to get the ave
        ! energy per spin, and divide by 2 to account for double counting.
        energy = energy/(ncols*nrows*2.0)
        energy_ave = energy_ave + energy
        energy2_ave = energy2_ave + energy**2
      endif

      ! Randomly choose a spin to change:
      m = nint((nrows-1)*ran1(5) + 2)  ! choose a random row
      n = nint((ncols-1)*ran1(5) + 2)  ! choose a random column
      trial_spin = -A(m,n)                     ! trial spin value

      ! Find change in energy (deltaU) due to trial move.
      ! If exp(-beta*deltaU) > eta, where eta is random, accept move:
      deltaU = -trial_spin*(A(m-1,n)+A(m+1,n)+A(m,n-1)+A(m,n+1))*2
      log_eta = dlog(ran1(5) + 1.0d-10)  ! random number 0-1 (+ tiny offset)
      if (-beta*deltaU > log_eta) then
        A(m,n) = trial_spin
        if (m == 2) A(nrows+2,n) = trial_spin
        if (m == nrows+1) A(1,n) = trial_spin
        if (n == 2) A(m,ncols+2) = trial_spin
        if (n == ncols+1) A(m,1) = trial_spin
      endif

    enddo MC_passes

    ! Write final spin array to output file
    if (.not. MovieOn) then
      do i = 2, nrows + 1
        do j = 2, ncols + 1
          write(32,*) A(i,j)
        enddo
      enddo
```

```
  endif
  write(33,*) temp, abs(magnetization_ave/output_count), &
    magnetization2_ave/output_count, &
    beta*(magnetization2_ave/output_count - (magnetization_ave/output_count)**2)
  write(34,*) temp, energy_ave/output_count, energy2_ave/output_count, &
    (beta**2)*(energy2_ave/output_count - (energy_ave/output_count)**2)

enddo scan_loop

close(32)
close(33)
close(34)

print*, "Program ising.f90 complete!"
print*, "Look at 'spin-array' with IDL program see_spins.pro"

contains


!_______RANDOM NUMBER GENERATING FUNCTION______!

double precision function ran1(idum)
implicit none
double precision ::  r(97)
integer, intent(IN) :: idum
save
integer, parameter :: M1=259200,IA1=7141,IC1=54773
real, parameter :: RM1=1.0d0/M1
integer, parameter :: M2=134456,IA2=8121,IC2=28411
real, parameter :: RM2=1.0d0/M2
integer, parameter :: M3=243000,IA3=4561,IC3=51349
integer :: IX1, IX2, IX3, jjj
integer ::  iff=0
if (idum < 0 .or. iff == 0) then
  iff = 1
  IX1 = mod(IC1-idum,M1)
  IX1 = mod(IA1*IX1+IC1,M1)
  IX2 = mod(IX1,M2)
  IX1 = mod(IA1*IX1+IC1,M1)
  IX3 = mod(IX1,M3)
  do jjj = 1,97
    IX1 = mod(IA1*IX1+IC1,M1)
    IX2 = mod(IA2*IX2+IC2,M2)
    r(jjj) = (dfloat(IX1)+dfloat(IX2)*RM2)*RM1
  end do
end if
IX1 = mod(IA1*IX1+IC1,M1)
IX2 = mod(IA2*IX2+IC2,M2)
IX3 = mod(IA3*IX3+IC3,M3)
jjj = 1+(97*IX3)/M3
if (jjj > 97 .or. jjj < 1) PAUSE
ran1 = r(jjj)
r(jjj) = (dfloat(IX1)+dfloat(IX2)*RM2)*RM1
```

```
end function ran1

end program ising
```

This is the required input file for the above program:

```
nrows - number of rows of spins (even number)
20
ncols - number of columns of spins (even number)
20
npass - number of passes for each temperature
200000
nequil - number of equilibration steps for each temperature
100000
high_temp - temperature to start scan at
2.92
low_temp - temperature to finish scan at
0.92
temp_interval - scanning interval
.1
ConfigType - 1: checkerboard, 2: interface, 3: unequal interface
1
MovieOn - set to .true. when running for 1 temp to make movie
.false.
End of file.
```

This is the IDL helper program for visualizing the final spin arrays at each temperature:

```
pro see_spins

inputfile = 'spin-array'
openr, inlun, inputfile, /get_lun
readf, inlun, nrows
readf, inlun, ncols
readf, inlun, nframes
readf, inlun, MovieOn
print, "MovieOn is", MovieOn
A = intarr(ncols,nrows)
window, 5, xsize=ncols*20, ysize=nrows*20, $
  title='2D Ising Model: light = +, dark = -'
for n = 0, nframes-1 do begin
  for i = 0, nrows-1 do begin
    for j = 0, ncols-1 do begin
      readf, inlun, s
      A(j,nrows-1-i) = s
    endfor
  endfor
  if (MovieOn eq 2) then begin
    if (total(A) < 0) then A = -A
    for i = 0, nrows-1 do begin
      for j = 0, ncols-1 do begin
        if (A(j,nrows-1-i) eq -1) then A(j,nrows-1-i) = 1 $
          else A(j,nrows-1-i) = -1
      endfor
    endfor
  endif
  A = A*1000
  A = congrid(A, ncols*20, nrows*20)
  tv, A
  A = intarr(ncols, nrows)
  print, "Frame", n
  wait, 0.1
endfor
free_lun, inlun

end
```

Onsager's Exact Solution

I happened to find this while I was looking for information for my presentation, and I thought it was somewhat amusing.

> In 1942, Onsager developed an exact solution to the problem of Ising spins in a plane, the "two-dimensional Ising model." This work stands, to this day, as a pinnacle of the achievements of theoretical physics of our time. Onsager's solution yielded the thermodynamic properties of the interacting system, and demonstrated the phase transition at T_c but in a form quite unlike that of Curie-Weiss. In particular, the infinite specific-heat anomaly at T_c is a challenge for approximate, simpler theories to reproduce. Onsager's discovery was not without an amusing sequel. The original solution was given by Onsager as a discussion remark, following a paper presented to the New York Academy of Science in 1942 by Gregory Wannier, but the paper, based on an application of Lie algebras, only appeared two years later. However, his formula for the spontaneous magnetization below T_c which requires substantial additional anaylsis, $M = (1 - x^{-2})^{1/8}$, $x = \sinh(2J_1/kT)\sinh(2J_2/kT)$, was never published by him, but merely "disclosed." It required four years for its decipherment. It was first exposed to the public on 23 August 1948 on a blackboard at Cornell University on the occasion of a conference on phase transitions. Laslo Tisza had just presented a paper on The General Theory of Phase Transitions. Gregory Wannier opened the discussion with a question concerning the compatibility of the theory with some properties of the Ising moel. Onsager continued this discussion and then remarked that – incidentally, the formula for the spontaneous magnetization of the two-dimensional model is just that (given above.) To tease a wider audience, the formula was again exhibited during the discussion which followed a paper by Rushbrooke at the first postwar IUPAP statistical mechanics meeting in Florence in 1948; it finally appeared in print as a discussion remark. However, Onsager never published his derivation. The puzzle was finally solved by C.N. Yang and its solution published in 1952. Yang's analysis is very complicated . . .
>
> — D.C. Mattis, in *The Theory of Magnetism I*