LAB IX
Random magnet model: avalanches and hysteresis

Jakub Tworzydło

Institute of Theoretical Physics
Jakub.Tworzydlo@fuw.edu.pl

7/05/2024 ul. Pasteura 5, Warszawa

Plan

1 Algorithm

2 Tasks and hints

Plan

1. Algorithm

2. Tasks and hints

Process

- each spin flips when it can gain energy
- its local field $J \sum_{j(i)} s_j + h_i + H(t)$ at site i changes sign
- spin can be triggered by
  1) one of the neighbours flips
  2) increase of $H(t)$

Slowly changing the external field:

- search for the unflipped spin that is next to flip
- jump the field H just enough to flip
- propagate the avalanche.

Use $J = 1$ and periodic boundary conditions.

Block code: draft

Propagating an avalanche

(1) Find the triggering spin i for the next avalanche, which is the unflipped site with the largest internal field $J \sum_{j \text{ nbr to } i} s_j + h_i$ from its random field and neighbors.

(2) Increment the external field H to minus this internal field, and push the spin onto a first-in–first-out queue

(3) Pop the top spin off the queue.

(4) If the spin has not been flipped,* flip it and push all unflipped neighbors with positive local fields onto the queue.

(5) While there are spins on the queue, repeat from step (3).

(6) Repeat from step (1) until all the spins are flipped.

Task 1 (0.5pts)

Benchmark: calculate mean size of the first avalanche with 1000 realizations of disorder for 100×100 lattice and $R = 0.7, 0.9, 1.4$.

Optional: while developing your code plot the system/magnetic field configurations for a better insight.

Example results:

```
R = 1.4  mean_size = 1.042(1)
R = 0.9  mean_size = 1.39(1)
R = 0.7  mean_size = 660(30)
```

Task 2 (0.5pts)

A) Display the avalanches. While forming an avalanche assign the `count` number to the sites visited. Make a “pixel” plot of all the avalanches from the whole run, mark the avalanches with different colors. Use disorder strength $R = 0.9, 1.4, 2.1$.

B) Perform the simulation on a 300×300 system with $R = 0.9,\ 1.4,\ 2.1$.
Plot the accumulated result for $H(M)$ in the range $H \in (-3, 3)$ and $M \in (-1, 1)$. Magnetization: `M = np.sum(s)/(L**2)`.

Extra

There are many more things one can compute:

- histogram of avalanche sizes (on a log-log plot)
- colored shells (subsequent triggered neighbourhoods) of a growing avalanche
- time series of an avalanche (time is shell number)

Try at least one of those. For more details consult the literature (!)

One may also perform a 3D simulation (critical value is $R_c = 2.16$). (The speed would incease considerably when using `numba`, but it is not quite trivial; algorithmic tricks are also possible.)

Hints I

We need some simple data structure

```
# lattice of spins
s = np.ones( (L, L), dtype = int ) * (-1)
# recording of avalanches
aval = np.zeros( (L, L), dtype = int )

# random magnetic fields
h_rnd = np.random.randn(L, L) * R
# ... and the local fields
h_loc = np.ones( (L, L) ) * (-4.0) + h_rnd
```

Hints II

It is useful to prepeare a routine for calulating the neighbours

```
def neighbours(i):
    ix, iy = i
    return [  ( ix, (iy+1) % L ), ( ix, (iy-1) % L ),
              ( (ix+1) % L, iy ), ( (ix-1) % L, iy )  ]
```

Spin update: flip the spin and adjust local field of the neighbours

```
def update(i):
    s[i] = 1
    for j in neighbours(i):
        h_loc[j] += 2.0
    return
```

Hints III

FIFO queue is available as a Python list

```
d = []
d.append(i)
itmp = d.pop(0)
```

Important trick to calculate the triggering spin:

```
# find the triggering spin
i_trig = np.unravel_index(np.argmax(h_loc + (s+1)*(-100)),
                                                h_loc.shape)
# increment the external field
H = - h_loc[i_trig]
```

Quick way to plot the avalanches

```
plt.imshow(aval,interpolation='none',cmap=cm.gist_rainbow)
```