AP157 Output 2 - February 6, 2026

Benedict A. Pangilinan (bapangilinan5@up.edu.ph)

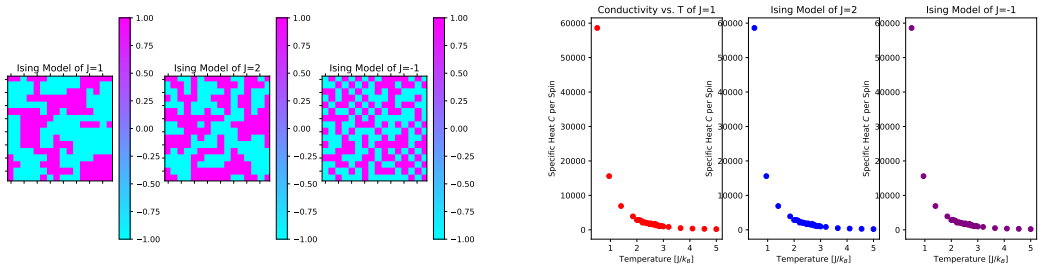

Figure 1: **Picture of a cat.** Figure on the left describes the spin model in different values of J. On the right, meanwhile, is the conductivity across different J's, normalized with respect to Boltzmann's constant k_B

Spin Model

This is a sample code for Spin Model, which plots the spin states of each site in Ising Model, via Metropolis-Hastings Algorithm.

```
def ising_model(J, h, beta, N, init_p, mc_iters):
    # How does this work?
    ising_lattice = np.sign(rng.random(size=(N,N))
         - init_p)
    energy_list = []

    # To be used on total energy calculation
    left, up, right, down = roll(ising_lattice) #
        use our roll function to shorten the code
    total_energy = -J*np.sum(ising_lattice*(left+
        up)) - h*np.sum(ising_lattice)
    energy_list.append(total_energy)

    for i in range(mc_iters):
        ## Pick a random spin in the lattice
        pos_x, pos_y = rng.integers(N, size=2)
        site_spin = ising_lattice[pos_x, pos_y]

        ## Compute the change in energy
        # Define total_energy of the flip

        # Here, we define the E_new - E_old
        delta_argument = np.roll(ising_lattice, 1,
            axis=1)[pos_x, pos_y] + np.roll(
            ising_lattice, 1, axis=0)[pos_x, pos_y]
        + np.roll(ising_lattice, -1, axis=1)[pos_x,
             pos_y]+ np.roll(ising_lattice, -1,
            axis=0)[pos_x, pos_y]
        delta_energy = 2*J*site_spin*(
            delta_argument)

        # config_p:
        config_p = np.exp(-beta*delta_energy) # For
             configuration spin
        number = rng.random()
        if rng.random() < config_p:
          ising_lattice[pos_x, pos_y] = -site_spin
        else:
          delta_energy = 0

        ## Record the relevant stuff
        total_energy = total_energy - delta_energy
        energy_list.append(total_energy)

        ## Record the relevant stuff
            # energy_list.append(total_energy)

    return ising_lattice, energy_list
```

`ising_lattice` is where the left plot came from, while the right plot came from `energy_list`.

For Loop

The right plot now came from:

```
    ## We compose the heat conducitivities to
        prepare for plotting
    temp_space = sorted(set().union(np.linspace
        (0.5, 5, 11), np.linspace(2, 3, 20)))
    c_init = []
    c_pos = []
    c_neg = []
    for T in temp_space:
        beta = 1.0/T
        sample_out, sample_energy = ising_model(J
            =1, h=0, N=16, beta=beta, init_p=0.5,
            mc_iters=sweeps*N*N)
        c_init.append(np.var(sample_energy[(-(
            sweeps//4)*N**2)::(N**2)])/((N**2)*(T
            **2)))
        sample_out_pos, sample_energy_pos =
            ising_model(J=2, h=0, N=16, beta=beta,
            init_p=0.5, mc_iters=sweeps*N*N)
        c_pos.append(np.var(sample_energy[(-(sweeps
            //4)*N**2)::(N**2)])/((N**2)*(T**2)))
        sample_out_neg, sample_energy_neg =
            ising_model(J=-1, h=0, N=16, beta=beta,
```

```
        init_p=0.5, mc_iters=sweeps*N*N)
    c_neg.append(np.var(sample_energy[(-(sweeps
        //4)*N**2)::(N**2)])/((N**2)*(T**2)))
```

Due to the constraints in computational constraints, limitation in the number of `sweeps`, that is, the number of times I am iterating the lattice, as if the lattice is propagating across time for energy equilibration, is too short to achieve the desired stabilization. This explains the monotonic decrease in conductivity across temperatures instesd of peaking around $T = 2.5 \frac{J}{k_B}$