Monte Carlo Simulation of Spin-fermion Model

Based on

Phase Separation in Electronic Models for Manganites
S. Yunoki, J. Hu, A. L. Malvezzi, A. Moreo, N. Furukawa, and E. Dagotto.

Phys. Rev. Lett., 80:845–848, Jan 1998.

Amit Bikram Sanyal

May 4, 2020

1 Hubbard Model

In condensed matter, there are generally two ways to model a crystal. The first approach involves treating each atom as a potential and solving the Schrödinger equation to find bound states, treating the electrons entirely as waves. A second approach is to consider that each atom is a "site" on the lattice, and the electrons are particles which hop from site to site, with it being localized entirely on a given site at a given moment of time. This is known as the tight-binding model, where we add the further simplification that electrons can only jump from a site to their nearest neighbors.

At this point, it becomes relatively simple to add the effect of electron-electron interaction to the system. The interaction can be assumened to be short range, as in, two particles interact only when they sit on the same site. This is called the Hubbard model.

$$\hat{H} = -t\left(\sum_{\sigma=\uparrow,\downarrow}\sum_{\langle i,j\rangle}\hat{c}^{\dagger}_{i,\sigma}\hat{c}_{j,\sigma} + \text{h.c.}\right) + U\sum_{i=1}^{N}\hat{n}_{i,\uparrow}\hat{n}_{i,\downarrow} \tag{1}$$

The Hubbard model, despite its simplifications, is a highly effective model that predict magnetic effects in a crystal very accurately.

However, the Hubbard inetraction term makes this an intrinsically many-body model and as a result, the Hilbert space of this system grows as $\mathcal{O}(N!)$ if the lattice has N sites. This makes diagonalizing this problem computationally challenging. A solution is to decouple the interaction into an effective model where we can use a one-body Hilbert space of $\mathcal{O}(N)$ instead. One of these methods is the Mean-field solution, where we assume that a single electron is moving in a a effective field created by the rest of the electrons. We can show that interaction can be approximately decomposed into *auxiliary fields*, which act as an effective classical spin sitting at each location that interacts with the itinerant electron.

2 The Spin-fermion model

The Spin-fermion (SF) model was initially a phenomenological model inspired by the mean field solution. We pretend that each site is actually occupied by a classical spin that can take any orientation, and at each site, a single fermionic interaction term appears that can simulate the effects of the electron-electron interaction. For an $L_x \times L_y$ lattice, this only creates a much smaller Hilbert space of dimension $2L_xL_y$, as opposed to $\mathcal{O}((L_xL_y)!)$ in the many-body case. The factor of 2 appears because of the two spin sectors of the problem. Although phenomenological in nature, it was shown to be highly effective in capturing the effects of interaction in the system.

Later on, it was discovered that there are actual materials in nature with multi-orbital crystals, such as cuprates and manganites, which realistically have a structure similar to the SF model![1]. This further cemented the importance of this model as an investigative tool.

The single-orbital SF Hamiltonian is given as

$$\begin{aligned}\hat{H} &= -t \sum_{\langle i,j\rangle,\sigma} \left(\hat{a}^{\dagger}_{i,\sigma}\hat{a}_{j,\sigma} + h.c.\right) - J_H \sum_i \vec{s}_i \cdot \vec{S}_i + J_{AF} \sum_{\langle i,j\rangle} \vec{S}_i \cdot \vec{S}_j \\ &= -t \sum_{\langle i,j\rangle,\sigma} \left(\hat{a}^{\dagger}_{i,\sigma}\hat{a}_{j,\sigma} + h.c.\right) - \frac{J_H}{2} \sum_{i,\alpha,\beta} \hat{a}^{\dagger}_{i,\alpha}\vec{\sigma}_{\alpha,\beta}\hat{a}_{i,\beta} \cdot \vec{S}_i \\ &\quad + J_{AF} \sum_{\langle i,j\rangle} \vec{S}_i \cdot \vec{S}_j \end{aligned} \tag{2}$$

For our problem, we will set $J_{AF} = 0$ and $\left|\vec{S}_i\right| = 1 \implies \vec{S}_i = [\sin(\theta_i)\cos(\phi_i), \sin(\theta_i)\sin(\phi_i), \cos(\theta_i)]$. For sake of simplicity, we fix the classical spins on plane, implying all $\phi_i = 0$, which is known as the XY-model for the spins.

The eigenvalues of the Hamiltonian can now be found by diagonalizing the Hamiltonian, which will depend on the actual orientation of the spins at a given temperature. In order to find the ground state, we need to determine this optimnal arrangement of spins, which will be done via the Monte Carlo Metropolis algorithm.

3 Computational modelling of the system

First, we need to model the tight-binding part of the Hamiltonian, which is done via the following function.

```
void makeTB()
{
  H.resize(2 * prm.Lx * prm.Ly, 2 * prm.Lx * prm.Ly);
  for (int i = 0; i < prm.Lx; i++)
  {
    for (int j = 0; j < prm.Ly; j++)
    {
      int pxy = k(i, j);

      int pxp1y = k((i + 1) % prm.Lx, j);
      int pxm1y = k((i - 1 + prm.Lx) % prm.Lx, j);

      int pxyp1 = k(i, (j + 1) % prm.Ly);
      int pxym1 = k(i, (j - 1 + prm.Ly) % prm.Ly);

      int pxp1ym1 = k((i + 1) % prm.Lx, (j - 1 + prm.Ly) % prm.Ly);
      int pxm1yp1 = k((i - 1 + prm.Lx) % prm.Lx, (j + 1) % prm.Ly);

      H(pxp1y, pxy) = cd(prm.tx, 0.0);
      H(pxm1y, pxy) = cd(prm.tx, 0.0);

      H(pxyp1, pxy) = cd(prm.ty, 0.0);
      H(pxym1, pxy) = cd(prm.ty, 0.0);

      H(pxp1ym1, pxy) = cd(prm.td, 0.0);
      H(pxm1yp1, pxy) = cd(prm.td, 0.0);

      H(pxp1y + ns, pxy + ns) = cd(prm.tx, 0.0);
      H(pxm1y + ns, pxy + ns) = cd(prm.tx, 0.0);

      H(pxyp1 + ns, pxy + ns) = cd(prm.ty, 0.0);
      H(pxym1 + ns, pxy + ns) = cd(prm.ty, 0.0);

      H(pxp1ym1 + ns, pxy + ns) = cd(prm.td, 0.0);
      H(pxm1yp1 + ns, pxy + ns) = cd(prm.td, 0.0);
    }
  }
}
```

This part of the Hamiltonian governs the non-interacting part of the problem and will remain constant throughout the computation.

The Hunds interaction needs to be recalculated each time we change the configuration of the spins, and is done by calling the following routine:

```
void makeHund()
{
  // Hund's terms
  for (int i = 0; i < ns; i++)
  {
    H(i, i) = 0.5 * prm.JH * cos(theta(0, i));
    H(i + ns, i + ns) = -0.5 * prm.JH * cos(theta(0, i));
    H(i, i + ns) = 0.5 * prm.JH * sin(theta(0, i)) * exp(-imagi * phi(0, i));
  }
}
```

The algorithm is now as follows:

1. Start with a random set of θ_i for each site.
2. Start at site one, and calculate the energy of the system via diagonalization.
3. Change the value of θ_i at that site, and diagonalize the Hamiltonian again to recalculate the energy of that configuration.
4. Calculate the probability of transition from the old state to the new one.
5. Roll a random number to probabilistically accept or reject this transition.
6. Move on to subsequent sites till last site.
7. Repeat entire process several times till energy values have settled.

To calculate the probabilty weight, we first define a function

$$\begin{aligned} P &= \mathrm{tr}(Z) \\ &= \prod_{\lambda} \left[1 + \exp(-\beta\,(\epsilon_\lambda - \mu))\right] \end{aligned} \tag{3}$$

We calculate P' after perturbing the angle at a given site. This changes P as changing the angle changes the Hamiltonian and hence, the spectrum. Probability of this change being accepted is (P/P').

There is a computational challenge in calculating P. The product gets large very quickly and exceeds the floating point limit on a computer. Hence, I calculate $\log(P)$, which makes the probability weight $\exp(\log(P) - \log(P'))$ which is a smaller number. $\log(P)$ is calculated by the following routine that gets called in the Monte Carlo loops:

```
double lnP(int i)
{
  double sum = 0.0;
  double mu = getmu();
  for (int lambda = 0; lambda < eigs_.size(); lambda++)
  {
    sum += log((1.0 + exp(-(1 / (prm.T)) * (eigs_[lambda] - mu))));
    // cout << sum << endl;
  }
  return sum;
}
```

The final Monte Carlo loop is as follows:

```
for (int t = 0; t < prm.sweeps; t++)
{
  total_change = 0;
  accepted = 0;
  // cout << "Sweep number: " << t + 1 << endl;
  for (int i = 0; i < prm.Lx; i++)
  {
    for (int j = 0; j < prm.Ly; j++)
    {
      // cout << "Sweep: " << t + 1
      //      << " (" << i + 1 << ", " << j + 1 << ")\n";
      double theta_old, theta_new, P_old, P_new, P_ratio, r;
      int pos;
      pos = k(i, j);

      theta_old = real(theta(0, pos));
      makeHund();
      Diagonalize('V', H, eigs_);
      // cout << getmu() << " ";
      P_old = lnP(pos);

      theta_new = filter(theta_old + perturb());
      theta(0, pos) = theta_new;
      makeHund();
      Diagonalize('V', H, eigs_);
      // cout << getmu() << "\n";
      P_new = lnP(pos);

      r = rng.random();
      P_ratio = exp(P_old - P_new);
      // cout << r << " " << P_old << " " << P_new << " "
      // << exp(P_old) << " " << exp(P_new) << endl;
      if (r < P_ratio)
      {
        accepted += 1;
        theta(0, pos) = theta_new;
        // cout << "Accepted" << endl;
      }
      else
      {
        // cout << "Rejected" << endl;
        theta(0, pos) = theta_old;
      }
      total_change += 1;
    }
  }
  cout << "Sweep number: " << t + 1 << ", Acceptance ratio = "
  << (accepted * 1.0) / (total_change * 1.0) << endl;
}
```

4 Results

Although a large number of quantities can be calculated from this model, they chiefly rely on looking at the orientation of the spins, which tells the magnetic state of the system. By varying the number of particles and the strength of interaction, we can get a phase diagram.[2]

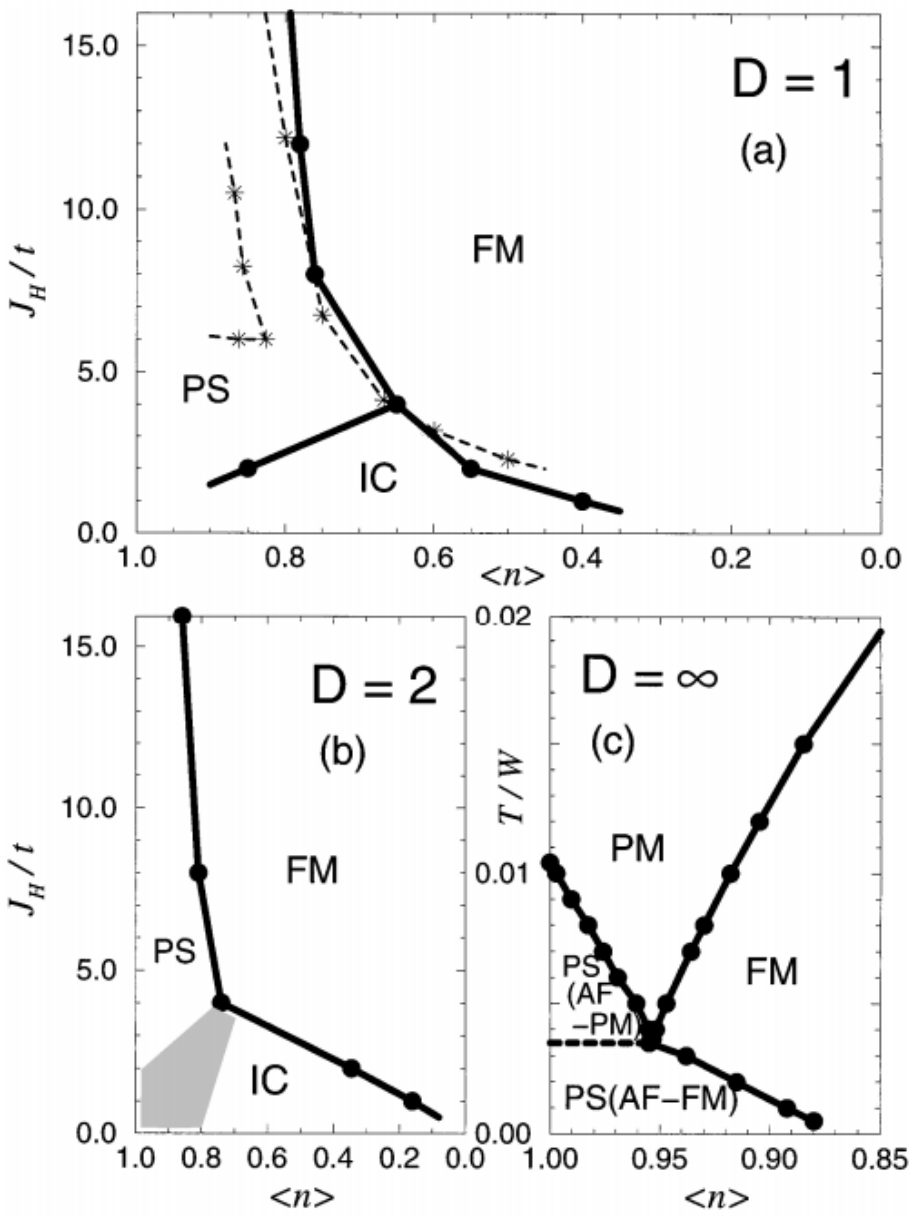
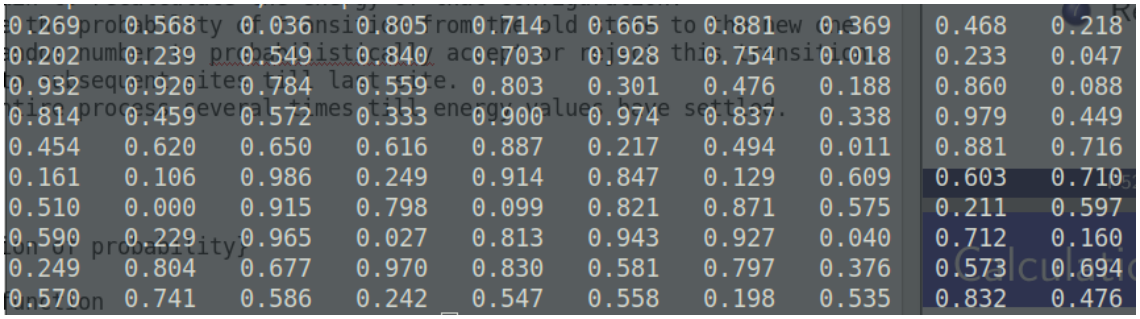


Figure 1: Phase diagram between J_H and μ

I ran my simulation at half-filling, very low temperature, at $J_H = 1.0$ on a 10×10 lattice and obtained the individual spins. Each spin is between 0 and 2π and is divided by 2π. Thus, a spin close to 1 or 0 points upwards and 0.5 means it points downwards.



0.269	0.568	0.036	0.805	0.714	0.665	0.881	0.369	0.468	0.218
0.202	0.239	0.549	0.840	0.703	0.928	0.754	0.018	0.233	0.047
0.932	0.920	0.784	0.559	0.803	0.301	0.476	0.188	0.860	0.088
0.814	0.459	0.572	0.333	0.900	0.974	0.837	0.338	0.979	0.449
0.454	0.620	0.650	0.616	0.887	0.217	0.494	0.011	0.881	0.716
0.161	0.106	0.986	0.249	0.914	0.847	0.129	0.609	0.603	0.710
0.510	0.000	0.915	0.798	0.099	0.821	0.871	0.575	0.211	0.597
0.590	0.229	0.965	0.027	0.813	0.943	0.927	0.040	0.712	0.160
0.249	0.804	0.677	0.970	0.830	0.581	0.797	0.376	0.573	0.694
0.570	0.741	0.586	0.242	0.547	0.558	0.198	0.535	0.832	0.476

Figure 2: Result from my simulation, at half-filling, very low temperature, at $J_H = 1.0$.

I also ran my simulation at very high values of J_H but failed to obtain a fully thermalized ferromagnetic state, as this requires the system to be cooled from a higher temperature, but I could not find out the exact calculations necessary to compute the chemical potential at high temperatures.

References

[1] Dagotto, E. *Nanoscale phase separation and colossal magnetoresistance: the physics of manganites and related compounds*. Springer, 2011.

[2] Yunoki, S., Hu, J., Malvezzi, A. L., Moreo, A., Furukawa, N., and Dagotto, E. Phase separation in electronic models for manganites. *Phys. Rev. Lett. 80* (Jan 1998), 845–848.