Monte Carlo methods for 2D Heisenberg Model of magnetic two-dimensional material

Yehui Zhang 171767
(Southeast University, Nanjin, Jiangsu, 211189, China)

ABSTRACT: This study aim at the Monte Carlo method of 2D Heisenberg Model. Before I perform the Monte Carlo simulation, distribution of points on the spherical surface and Monte Carlo change of each step were the main issues we investigated. Here we systematically study a 2D ferromagnetic monolayer CrI3 which was first one be measured the average Tc for the monolayer samples to be 45K. Summary of previous work, Ising Model may overestimate the Curie temperature. Therefore, this article would preform the Monte Carlo method by using 2D Heisenberg Model to simulate monolayer CrI3.

Keywords: 2D Heisenberg Model(XYZ), Monte Carlo method, Curie temperature, distribution.

I. Introduction

The statistical mechanics of complex physical systems poses many hard problems which are difficult to solve by analytical approaches. Numerical simulation techniques will therefore be indispensable tools on our way to a better understanding of. The numerical tools can roughly be divided into molecular dynamics (MD) and Monte Carlo (MC) simulations.

In physics-related problems, Monte Carlo methods are useful for simulating systems with many coupled degrees of freedom, such as fluids, disordered materials, strongly coupled solids, and cellular structures. In principle, Monte Carlo methods can be used to solve any problem having a probabilistic interpretation. By the law of large numbers, integrals described by the expected value of some random variable can be approximated by taking the empirical mean of independent samples of the variable. When the probability distribution of the variable is parametrized, mathematicians often use a Markov chain Monte Carlo (MCMC) sampler. The central idea is to design a judicious Markov chain model with a prescribed stationary probability distribution.

II. Heisenberg Model

The classical Ising model and the classical Heisenberg model are both simplified models of magnetism in materials. In the Ising model, there are spins S_i that can take on the values +1 or −1 living on each of N sites of a lattice. We'll leave the shape and dimensionality of the lattice unspecified for now. Typically, the spins interact with their nearest neighbors, with the energy of a configuration of spins given by:

$$H_{\text{Ising}}=-J\cdot\sum_{<ij>}S_iS_j$$

Here, the sum over <ij> runs over all nearest-neighbor pairs of sites on the lattice. If J<0, it's energetically preferable for neighboring spins to point in opposite directions, while if J>0, it's energetically preferable for neighboring spins to point in the same direction. In two dimensions or greater, the Ising model exhibits a continuous phase transition between a low-temperature ordered (ferromagnetic) phase and a high-temperature disordered (paramagnetic) phase with zero average magnetization.

The Heisenberg model looks very similar, except that the spins Si are three-dimensional unit vectors that can point anywhere on the unit sphere. The energy function for a configuration of spins is instead:

$$H_{\text{Heisenberg}}=-J\cdot\sum<ij>\hat{S}_i\cdot\hat{S}_j$$

III. Distribution

First, initializing the spin state should be discuss before performing the monte carlo method. In order to get a uniform distribution, five different kinds of distribution methods would be study.

① Phi-Z (aka Phi-cos[Theta])

Phi = rand(-pi,pi)

Thtea = arccos(rand(-1,1))

X = cos(Phi)*sin(Thtea)

Y = sin(Phi)*sin(Theta)

Z = cos(Theta)

Code 1. Phi-Z method Generator

```
void phi_z(Point& tmp, mt19937& rng)
{
    const uniform_real_distribution<double> rand(-1, 1);
    tmp.phi = M_PI * rand(rng);
    tmp.theta = acos(rand(rng));
    tmp.x = cos(tmp.phi) * sin(tmp.theta);
    tmp.y = sin(tmp.phi) * sin(tmp.theta);
    tmp.z = cos(tmp.theta);
}
```

② Theta-Phi

Phi = rand(0,2*pi)

Thtea = rand(0,pi)

X = cos(Phi)*sin(Thtea)

Y = sin(Phi)*sin(Theta)

Z = cos(Theta)

Code 2. Theta-Phi method Generator

```
void theta_phi(Point& tmp, mt19937& rng)
{
    const uniform_real_distribution<double> rand(0, 2);
    tmp.phi = M_PI * rand(rng);
    tmp.theta = M_PI_2 * rand(rng);
    tmp.x = cos(tmp.phi) * sin(tmp.theta);
    tmp.y = sin(tmp.phi) * sin(tmp.theta);
    tmp.z = cos(tmp.theta);
}
```

③ X-Y-Z

X = rand(-1,1)

Y = rand(-1,1)

Z = rand(-1,1)

Norm(X, Y, Z)

Code 3. X-Y-Z method Generator

```
void x_y_z(Point& tmp, mt19937& rng)
{
    const uniform_real_distribution<double> rand(-1, 1);
    const auto x = rand(rng), y = rand(rng), z = rand(rng);
    const auto norm = sqrt(x * x + y * y + z * z);
    tmp.x = x / norm; tmp.y = y / norm; tmp.z = z / norm;
}
```

④ X-Y-Z & abs()<=1

X = rand(-1,1)

Y = rand(-1,1)

Z = rand(-1,1)

Norm(X, Y, Z) & abs()<=1

Code 4. X-Y-Z<1 method Generator

```
void x_y_z_lessthan1(Point& tmp, mt19937& rng)
{
    const uniform_real_distribution<double> rand(-1, 1);
    auto x = 0.0, y = 0.0, z = 0.0, norm = 2.0;
    for (; norm > 1;)
    {
        x = rand(rng); y = rand(rng); z = rand(rng);
        norm = sqrt(x * x + y * y + z * z);
    }
    tmp.x = x / norm; tmp.y = y / norm; tmp.z = z / norm;
}
```

⑤X-Y-Z-Normal

X = normal_rand(-1,1)

Y = normal_rand(-1,1)

Z = normal_rand(-1,1)

Norm(X, Y, Z)

Code 5. X-Y-Z-Normal method Generator

```
void ormal_x_y_z(Point& tmp, mt19937& rng)
{
    normal_distribution<double> rand(0, 1);
    const auto x = rand(rng), y = rand(rng), z = rand(rng);
    const auto norm = sqrt(x * x + y * y + z * z);
    tmp.x = x / norm; tmp.y = y / norm; tmp.z = z / norm;
}
```

Spherical coordinate system use theta phi and r three parameters to determine one point, naturely spherical surface distribution parameters choose theta and phi. However some studies point out that it was not a correctly distributed points generator. Technical application choose x y and z three parameters to determine one point, without theta and phi, thus application does not need transform.

Therefore, five different kinds of methods would be perform below.

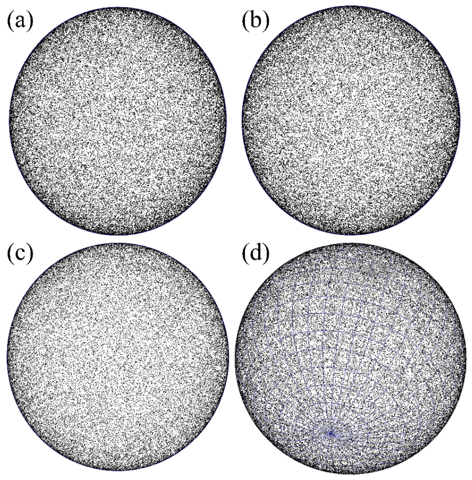

Figure 1.Phi-Z method distribution. (a).X-axis view (b).Y-axis view (c). Z-axis view (d). normal view

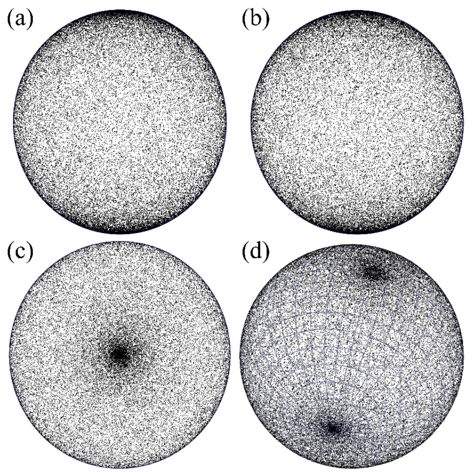

Figure 2.Theta-Phi method distribution. (a).X-axis view (b).Y-axis view (c). Z-axis view (d). normal view

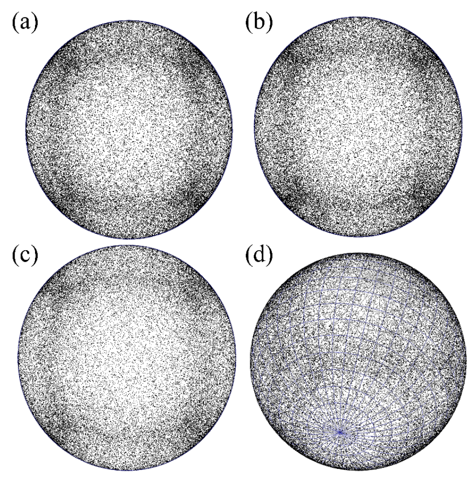

Figure 3.X-Y-Z method distribution. (a).X-axis view (b).Y-axis view (c). Z-axis view (d). normal view

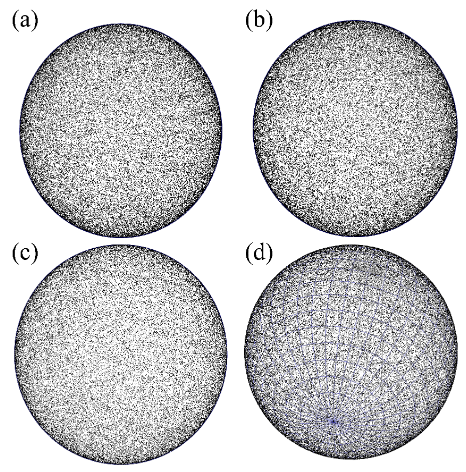

Figure 4.X-Y-Z<=1 method distribution. (a).X-axis view (b).Y-axis view (c). Z-axis view (d). normal view

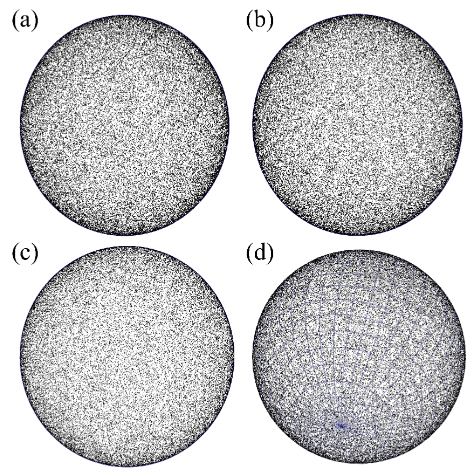

Figure 5.X-Y-Z-Normal method distribution. (a).X-axis view (b).Y-axis view (c). Z-axis view (d). normal view

Apparently, Phi-Z method, X-Y-Z<1 method and X-Y-Z-Normal method can generate correctly distributed points. For better performance, Compare the generator efficiency.

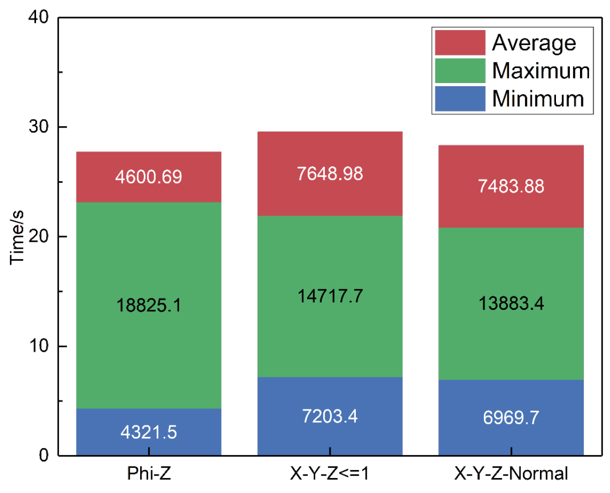

Figure 6. Compare the generator efficiency

Different methods generated 50,000 points and repeated 5,000 times. Phi-Z generator less stable which may caused by anti-trigonometric function. But average preformance much faster than X-Y-Z<=1 and X-Y-Z-Normal method.

IV. Monte Carlo changes

Second, Monte Carlo changes are much more important than distribution. Because for low temperture, initializing the spin state coule point at the same place. Simply, extending the Phi-Z generator method, Monte carlo changes could be minor change for phi and z. However, from where I stand, it was an Incorrectly change method.

Therefore three different kinds of method would perform below.

① Phi-Z (aka Phi-cos[Theta])

Code 6. Phi-Z method change(move)

```
void theta_z(Point& tmp, Point& start, mt19937& rng, const double delta)
{
    const uniform_real_distribution<double> rand(-1, 1);
    tmp.phi = start.phi + delta * M_PI * rand(rng);
    tmp.z = start.z + delta * rand(rng);
    if (tmp.z > 1)
    { tmp.z = 2 - tmp.z; tmp.phi = 2 * M_PI - tmp.phi; }
    if (tmp.z < -1)
    { tmp.z = -2 - tmp.z; tmp.phi = 2 * M_PI - tmp.phi; }
    tmp.theta = acos(tmp.z);
    tmp.x = cos(tmp.phi) * sin(tmp.theta);
    tmp.y = sin(tmp.phi) * sin(tmp.theta);
    tmp.z = cos(tmp.theta);
}
```

② Inclined Angle

Code 7. Inclined Angle method change(move)

```
void inclined_angle(Point& tmp, Point& start, mt19937& rng,
 const double delta)
{
    auto bak = start;
    for (;;)
    {
        uniform_real_distribution<double> rand(-1, 1);
        tmp.phi = M_PI * rand(rng);
        tmp.theta = acos(rand(rng));
        tmp.x = cos(tmp.phi) * sin(tmp.theta);
        tmp.y = sin(tmp.phi) * sin(tmp.theta);
        tmp.z = cos(tmp.theta);
        if (bak.x * tmp.x + bak.y * tmp.y + bak.z * tmp.z
            - (1 - 2 * delta) > 0) break;
    }
}
```

③ Euler Angle

$$\begin{bmatrix} cos(theta) * cos(phi) & cos(theta) * sin(phi) & -sin(theta) \\ -sin(phi) & cos(phi) & 0 \\ sin(theta) * cos(phi) & sin(theta) * sin(phi) & cos(theta) \end{bmatrix}$$

Code 8. Euler Angle method change(move)

```
void euler_angle(Point& tmp, Point& start, mt19937& rng, co
nst double delta)
{
    const uniform_real_distribution<double> rand(0, 2);
    const auto costhetai = cos(start.theta);
    const auto sinthetai = sin(start.theta);
    const auto cosphii = cos(start.phi);
    const auto sinphii = sin(start.phi);

    const auto phi = M_PI * rand(rng);
    const auto theta = acos(1 - delta * rand(rng));
    const auto x = sin(theta) * cos(phi);
    const auto y = sin(theta) * sin(phi);
    const auto z = cos(theta);
    tmp.x = costhetai * cosphii * x - sinphii * y
        + sinthetai * cosphii * z;
    tmp.y = costhetai * sinphii * x + cosphii * y
        + sinthetai * sinphii * z;
    tmp.z = -sinthetai * x + costhetai * z;
}
```

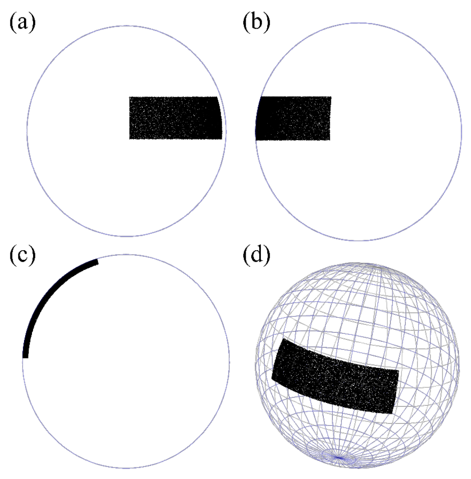

Figure 7.Phi-Z method change. (a).X-axis view (b).Y-axis view (c). Z-axis view (d). normal view

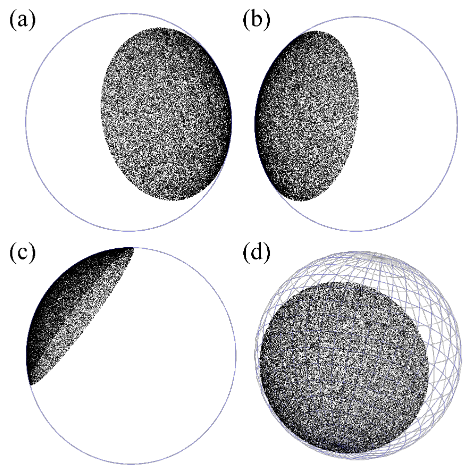

Figure 8. Inclined Angle method change. (a).X-axis view (b).Y-axis view (c). Z-axis view (d). normal view

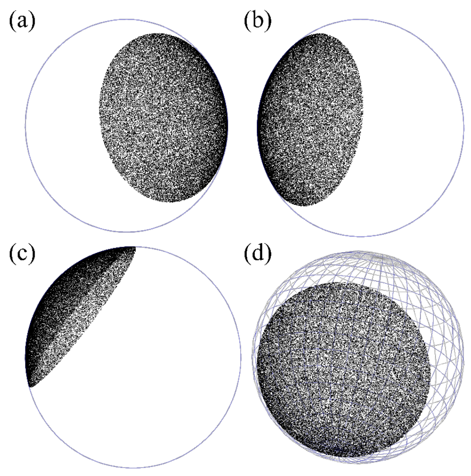

Figure 9. Euler Angle method change. (a).X-axis view (b).Y-axis view (c). Z-axis view (d). normal view

Obviously, Inclined Angle method and Euler Angle method could generate correctly changes. Whereas the former is not stable, which performance may degenerate seriously in low temperture.

V. Curie temperature

Up to now, CrI3 is the only one two-dimensional magnetic material which has been measured its Curie temperature to be 45K. Previously, some studies calculate its Curie temperture by using Ising Model. And the Curie temperature for CrI3 is estimated to be 107 K. However, Ising Model is widely believed overestimate the Curie temperture.

Definitions and symbols are defined below:

$$\langle M\rangle = (\sum M_i) / \mathbb{N} \quad \text{--} \quad \text{magnetization}$$

$$\langle H\rangle = (\sum H_i) / \mathbb{N} \quad \text{--} \quad \text{internal energy}$$

$$\chi(T)=[\langle|M|^2\rangle - \langle|M|\rangle^2] / (k_bT) \quad \text{--} \quad \text{susceptibility}$$

$$C_v(T)=[\langle H^2\rangle - \langle H\rangle^2] / (k_bT^2) \quad \text{--} \quad \text{heat capacity}$$

Now, using Ising Model's data to simulate Heisenberg Model.

$$H=-J\cdot\sum_{\langle ij\rangle}\hat{S}_i\cdot\hat{S}_j \text{ and } J = 2.519\ meV$$

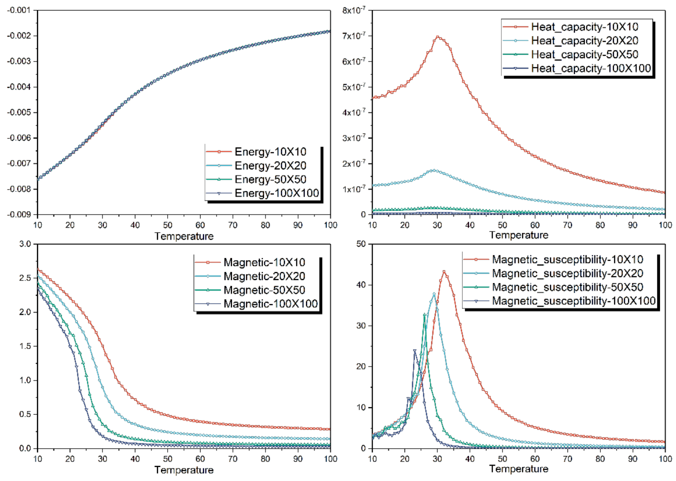

Figure 10. CrI3 different lattice simulation's energy, Heat capacity, magnetization and susceptibility

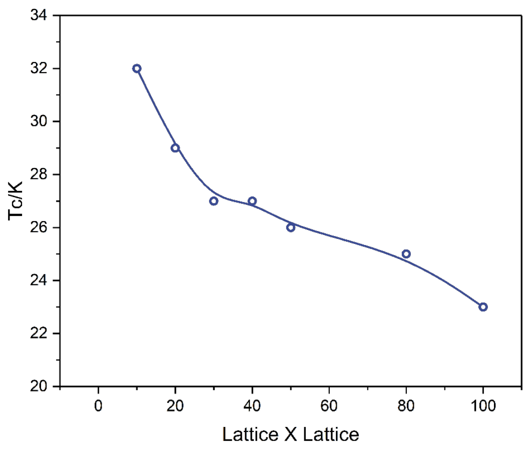

Figure 11. With the lattice increases, the Curie temperture continues to decrease

Meanwhile Mermin–Wagner theorem indicate that there is no phase with spontaneous breaking of a continuous symmetry for T>0, in 2D Heisenberg Model. However, adding magnetic anisotropy could fix it. Now, Hamiltinian to be:

$$H=-J\cdot\sum_{\langle ij\rangle}\hat{S}_i\cdot\hat{S}_j - A\cdot \hat{S}_z^2 \text{ and } J = 2.519\ meV,\ A = 4.45meV$$

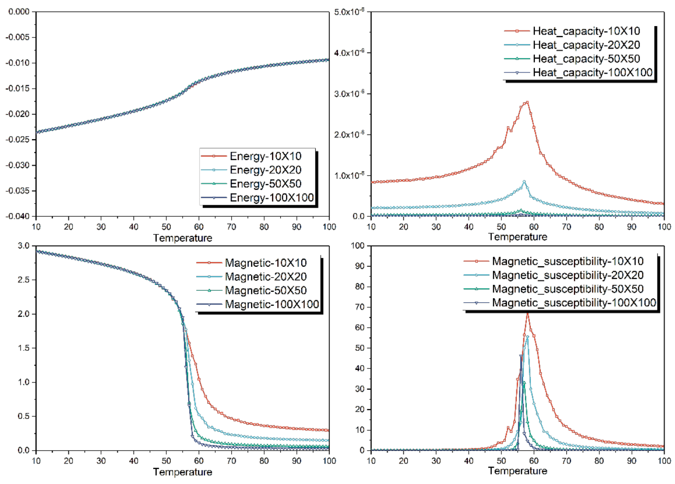

Figure 12. CrI3 different lattice simulation's energy, Heat capacity, magnetization and susceptibility

Supplement: for Phi-Z method change

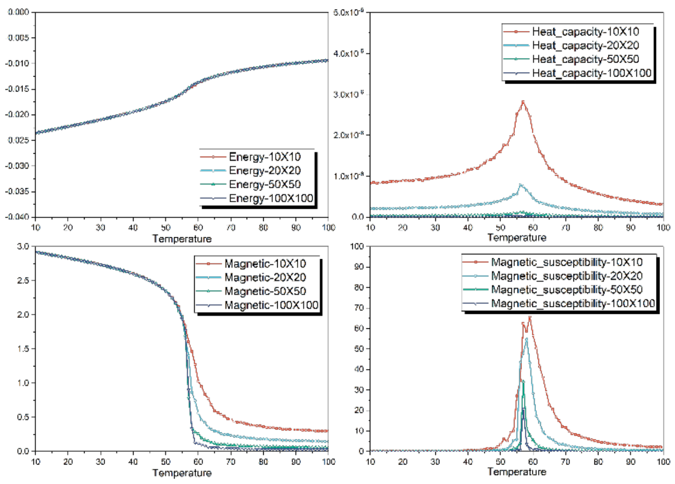

Figure 13. CrI3 different lattice simulation's energy, Heat capacity, magnetization and susceptibility

This seems to achieve the same result for CrI_3, but I still think this Monte Calro move is inappropriate.

VI. Magnetic hysteresis loop

For previous code, it could easily add mpi for different temperture even without MPI_Reduce, MPI_Allreduce and MPI_Barrier. But for magnetic hysteresis loop simulation, different magnetic state correlated. In order to simulate magnetic domain in real materials, we perform 100 times different initialization states with mpi, then average all the data by MPI_Allreduce and MPI_Barrier.

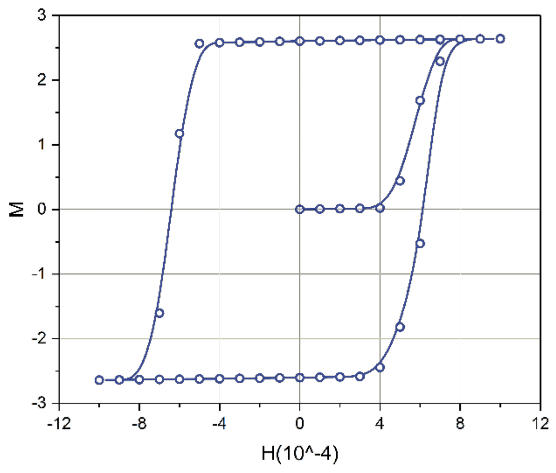

Figure 14. Magnetic hysteresis loop for CrI3 in 40K

VII. Conclusion

This article focus on how to correctly perform Monte Carlo method by using 2D Heisenberg Model. By using Phi-Z generator method and Eular Angle move method to simulate Monte Carlo method. But we ignore the deinition of the unit, in-depth study should consider this issue.

References

[1] A. Codello and G. D'Odorico, Phys Rev Lett **110**, 141601 (2013).

[2] C. Gong *et al.*, Nature **546**, 265-269 (2017).

[3] R. B. Griffiths, Physical Review **136**, A437-A439 (1964).

[4] B. Huang *et al.*, Nature **546**, 270-273 (2017).

[5] M. Kan, J. Zhou, Q. Sun, Y. Kawazoe, and P. Jena, J Phys Chem Lett **4**, 3382-3386 (2013).

[6] H. Kumar, N. C. Frey, L. Dong, B. Anasori, Y. Gogotsi, and V. B. Shenoy, ACS Nano **11**, 7648-7655 (2017).

[7] X. Li, X. Wu, and J. Yang, J Am Chem Soc **136**, 5664-5669 (2014).

[8] X. Li, X. Wu, and J. Yang, J Am Chem Soc **136**, 11065-11069 (2014).

[9] X. Li and J. Yang, Journal of Materials Chemistry C **2**, 7071 (2014).

[10] J. Liu and Q. Sun, Chemphyschem **16**, 614-620 (2015).

[11] J. Liu, Q. Sun, Y. Kawazoe, and P. Jena, Phys Chem Chem Phys **18**, 8777-8784 (2016).

[12] Y.-S. Liu, J.-H. Yong, H. Zhang, D.-M. Yan, and J.-G. Sun, Computer-Aided Design **38**, 55-68 (2006).

[13] M. A. McGuire, H. Dixit, V. R. Cooper, and B. C. Sales, Chemistry of Materials **27**, 612-620 (2015).

[14] M. E. Muller, Communications of the ACM **2**, 19-20 (1959).

[15] Z. Nehme, Y. Labaye, R. Sayed Hassan, N. Yaacoub, and J. M. Greneche, AIP Advances **5** (2015).

[16] P. S. Rakić, S. M. Radošević, P. M. Mali, L. M. Stričević, and T. D. Petrić, Physica A: Statistical Mechanics and its Applications **441**, 69-80 (2016).

[17] N. Samarth, Nature **546**, 216 (2017).

[18] Y. Sun, Z. Zhuo, X. Wu, and J. Yang, Nano Lett **17**, 2771-2777 (2017).

[19] J. Zhou and Q. Sun, J Am Chem Soc **133**, 15113-15119 (2011).

[20] H. L. Zhuang, Y. Xie, P. R. C. Kent, and P. Ganesh, Physical Review B **92** (2015).