二维三角格子 Ising 模型的 Monte Carlo 方法研究

姓名：马磊　学号：10210190005

一、程序设计

本次作业采用 Metropolis 算法实现二维三角格子 Ising 模型的物理量计算。
二维三角格子如图 1 所示。

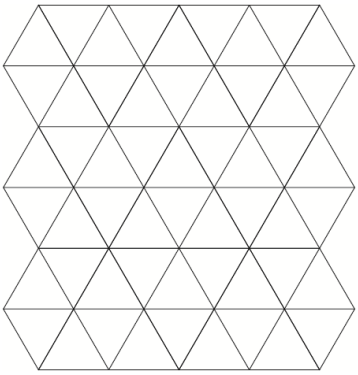


图 1 三角格子图示

其中交点为格点，连线为键。哈密顿量为

$H=\sum_{i,j}Js_is_j$ ，其中 i, j 为最近邻格点。

本次作业中为研究不同二维晶格物理性质的差异，将哈密顿量改写为

$H=\sum_{i,j}J_1s_is_j+\sum_{i',j'}J_2s_{i'}s_{j'}$ ，其中 i, j 为三个最近邻中两个，i', j'为另一个最近邻。

格点示意为图 2。

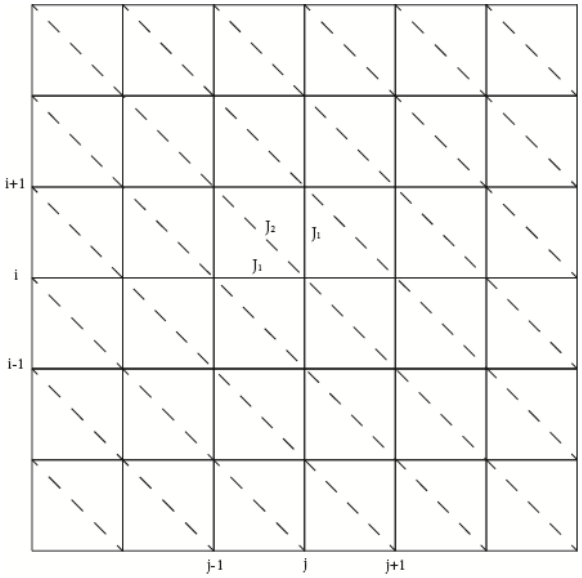


图 2 扩展的三角格子

将某格点与附近格点相互作用按照行列写出来，

$$H_{ij}=s_{i,j}(J_1(s_{(i-1)\bmod L,j}+s_{(i+1)\bmod L,j}+s_{i,(j-1)\bmod L}+s_{i,(j+1)\bmod L})+J_2(s_{(i+1)\bmod L,(j-1)\bmod L}+s_{(i-1)\bmod L,(j+1)\bmod L}))$$

其中，L 每一维度上的格点数目。

在程序中，温度取作T/J_1，并取$R=J_2/J_1$，E，C，M，X 分别表示体系的能量，热容，总磁矩，磁导率。相应参数取值如下：

格点大小	32*32
T	1.5-10.5/0.5-10.5
R	-6.0-6.0
温度步长	0.05
R 步长	0.1
总步数	5100000
忽略步数	100000

二、程序实现

本作业中使用 C 语言实现。

三、计算结果

当 R>0 时，计算结果如图 3 所示。

由图可见，对于格点大小为 32 X 32 的晶格，在J_2/J_1取值为 0.1 到 6 时，一直有非常明显的相变。相变温度从 2.45（R=0.1）变化到 8.55（R=6.0）。

R 非常小时（如图 4，R=0.1），相变温度（R=0.1 为 2.45）接近正方格子的相变温度。这是因为$J_2 << J_1$，导致格子的非常类似于正方格子。随着 R 增大，由于格点之间相互作用越来越大，相变温度越来越高，即体系从铁磁相转变到顺磁相的温度越来越高。只是因为，当J_1, J_2都大于零时，格点之间的相互作用越强烈，打乱铁磁相所需的温度越高，即相变温度升高是体系内部格点之间相互作用与热运动相互竞争的结果。

当 R<0 时，计算结果如图 7 所示。

由图可见，对于格点大小为 32 X 32 的晶格，在J_2/J_1取值为-6.0 到 0.0 时，相变温度从 2.3（R=0.0）随着 R 的减小而减小。

从 M-T 图和 X-M 图可见，当 R=-1.0 时，就已经没有相变了。这是个假象。注意到我的温度范围是 0.5 到 10.5，仔细观察会发现，其实在 R=1.0 的时候，三维图的峰值移动到了 T<0.5 的地方，所以本次计算并没能够给出。之所以能够从 E-T 图和 C-T 图猜测极有可能有相变，是因为图中出现了很强烈的涨落现象。从上面的计算可以发现 C-T 图的峰宽要比 X-T 的峰宽

大的多，在 R>1.0 的时候的 C-T 图中的强烈的涨落应该就是后面会出现的相变对于的峰。

另外，发现在 R=-3.0 时，X-T 曲线在 T 较大是出现了类似开口向下的抛物线的情况，这个可能有两个原因：一个是因为计算方法导致的，即方法的原因使得高温的时候 M 的涨落太大，偏离正常值；另一个就是事实确实如此，因为温度升高时，热涨落变得更加剧烈，所以磁导率 X 变大，但是当温度升高到一定的值以后，这种因素达到饱和，X-T 曲线开始下降。

R=1.0 的情况如图 5，从图中可得临界温度 T_c=3.65。临界指数的获取使用线性拟合的方法。定义

$$t = |T - T_c| / T_c$$

由临界指数定义可以得到

$$\ln C = -\alpha \ln |t| + C_C$$

$$\ln M = -\beta \ln |t| + C_M$$

$$\ln X = -\alpha \ln |t| + C_X$$

所以只需要做出 lnC~ln|t|, lnM~ln|t|, lnX~ln|t|曲线，然后计算斜率，响应的斜率的绝对值就是临界指数。

最后所得部分临界指数为下表。

α	2.07
β	0.13
γ	1.07

四、<u>误差分析</u>

当 R=1.0 时，临界温度以及部分临界指数的相对误差为

α	100%
β	4%
γ	39%

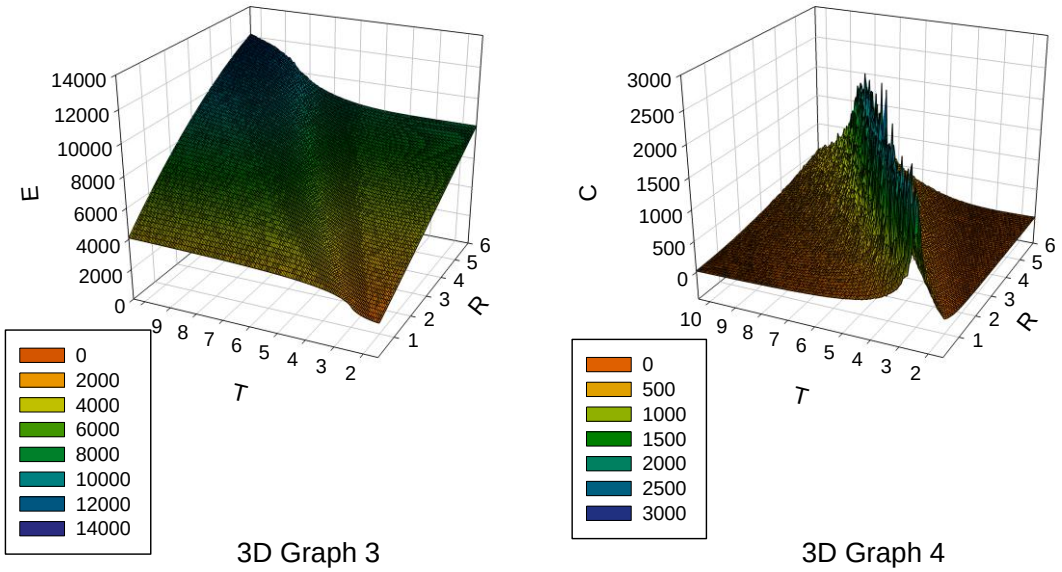
五、<u>结果讨论</u>

本作业中，临界指数并未通过 MC 直接给出，磁滞现象也未计算。而且并未进行多次重复计算来求的计算 Error，这在计算中是不允许的。

计算中临界指数的计算可以简单的使用热浴的方法，通过关联长度、关联函数和 Fisher 标度律来计算，这样可以使得计算结果足够好。或者使用 Wolf 选择 Cluster 的方法。

但是由于时间关系，这些工作都为做，实为遗憾。

附录 1：相关图表



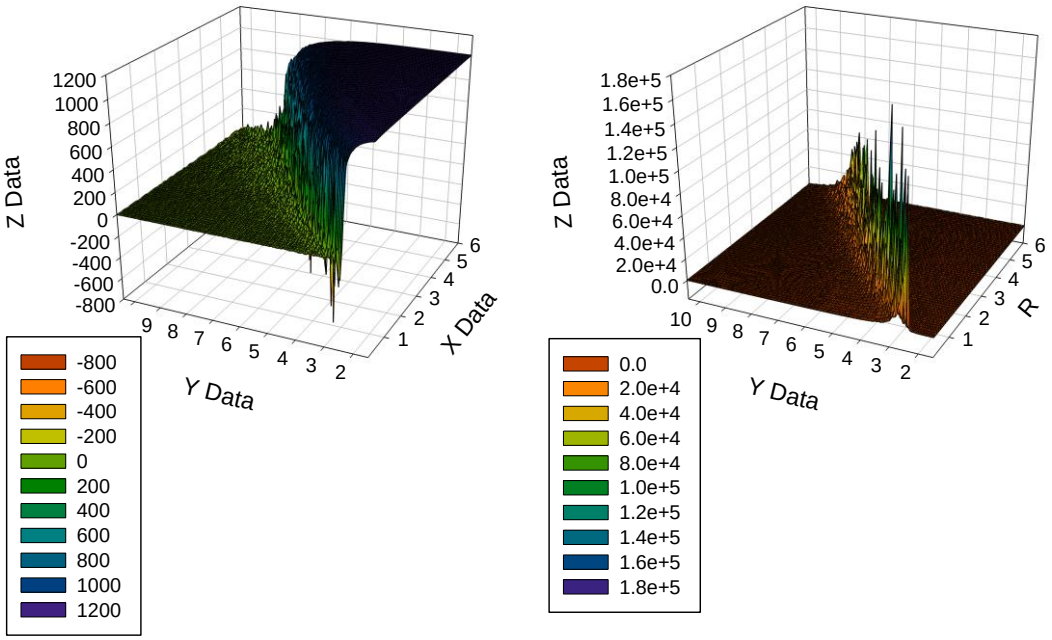


图 3 2D 三角格子 Ising 模型计算结果（R=0.1~6.0）

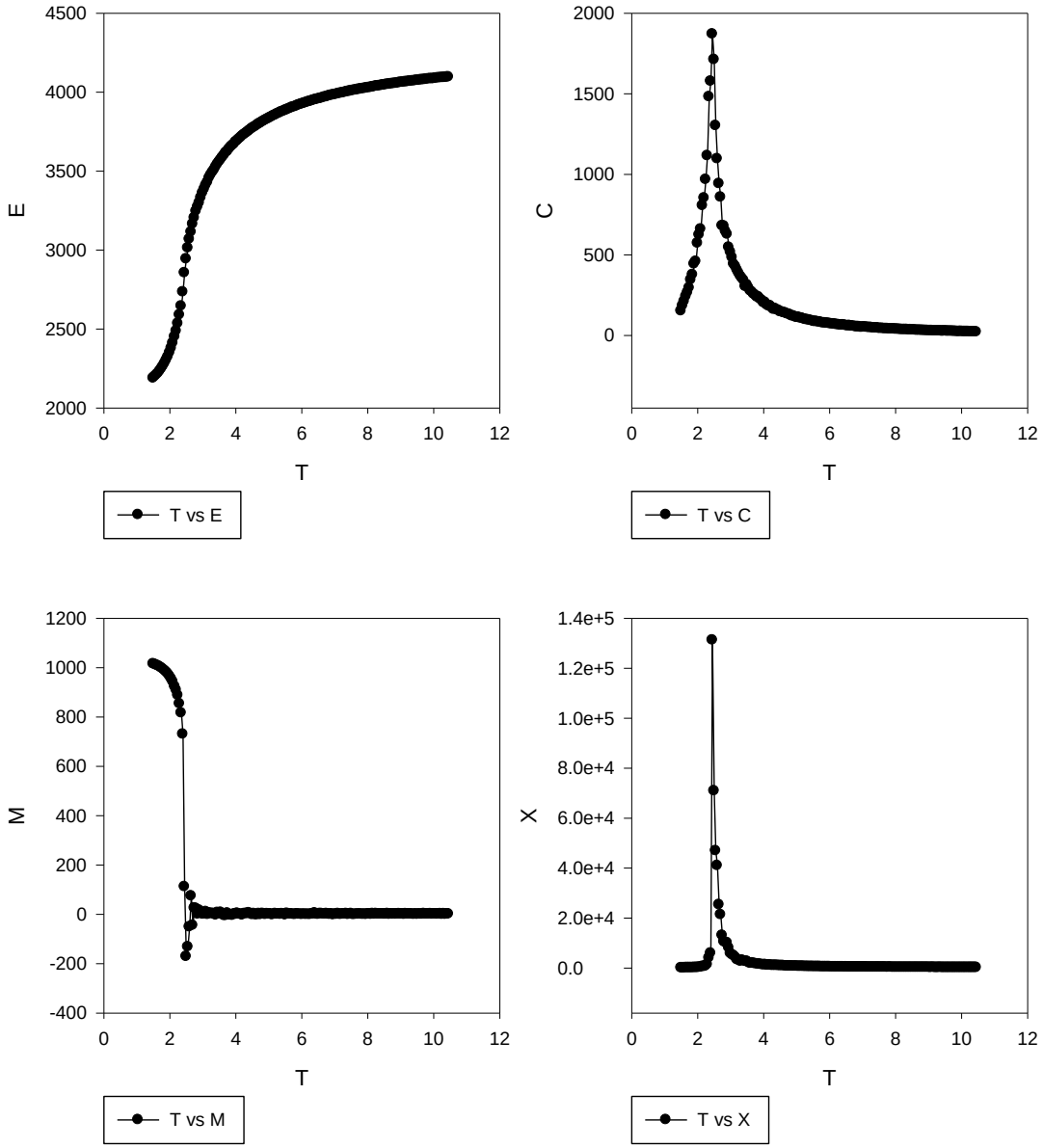


图 4 2D 三角格子 Ising 模型 R=0.1

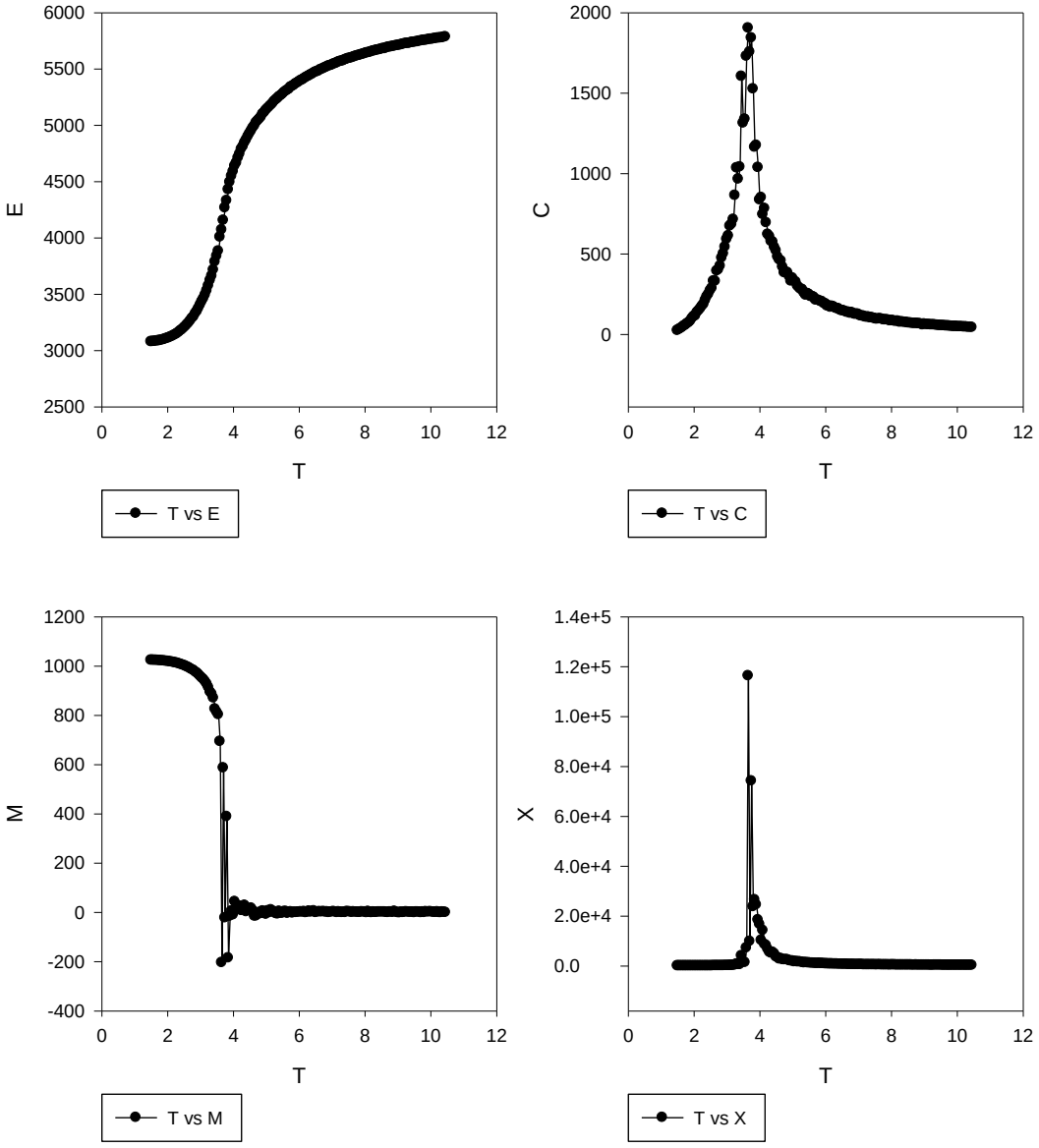


图 5 2D 三角格子 Ising 模型 R=1.0

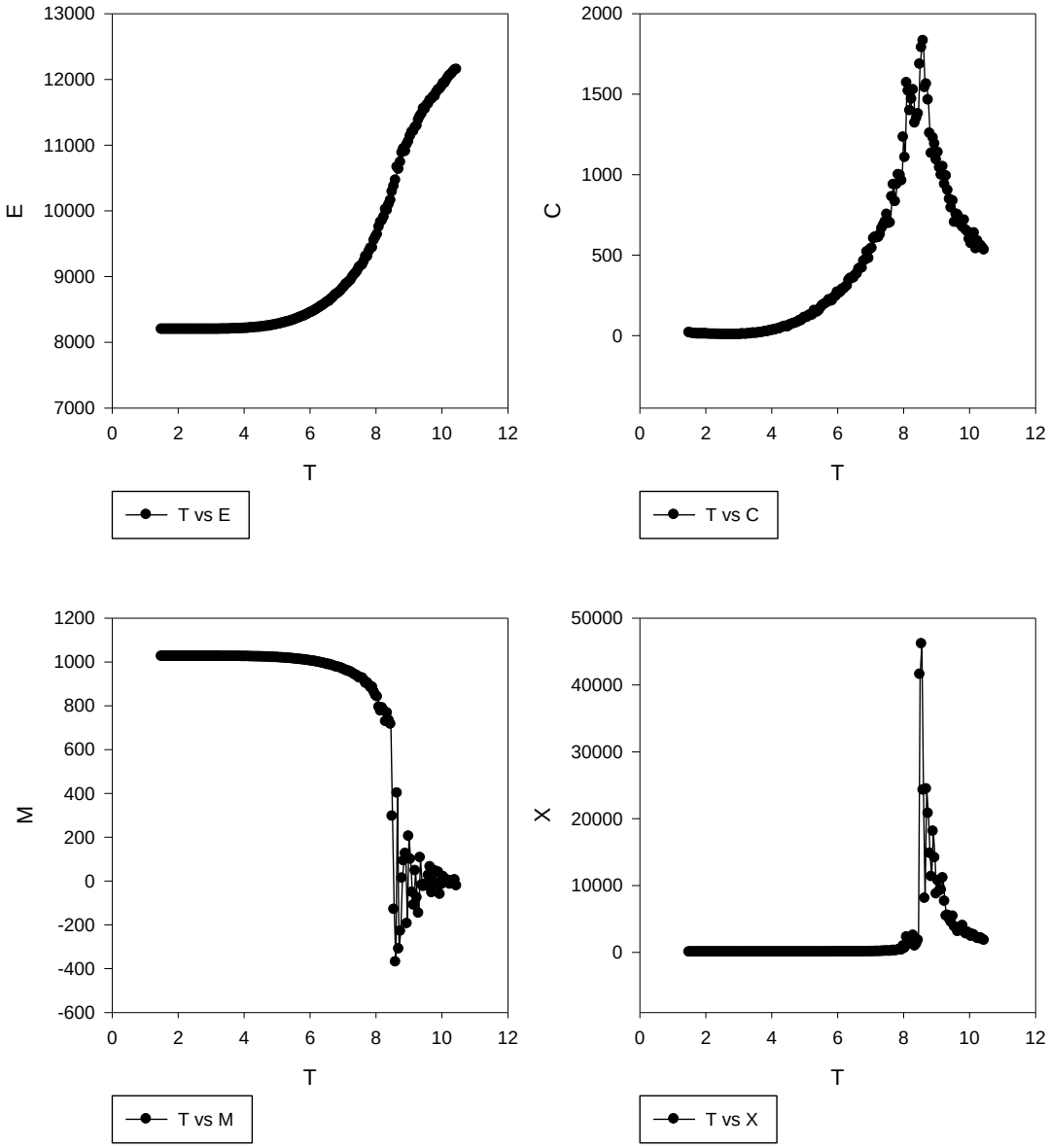


图 6 2D 三角格子 Ising 模型 R=6.0

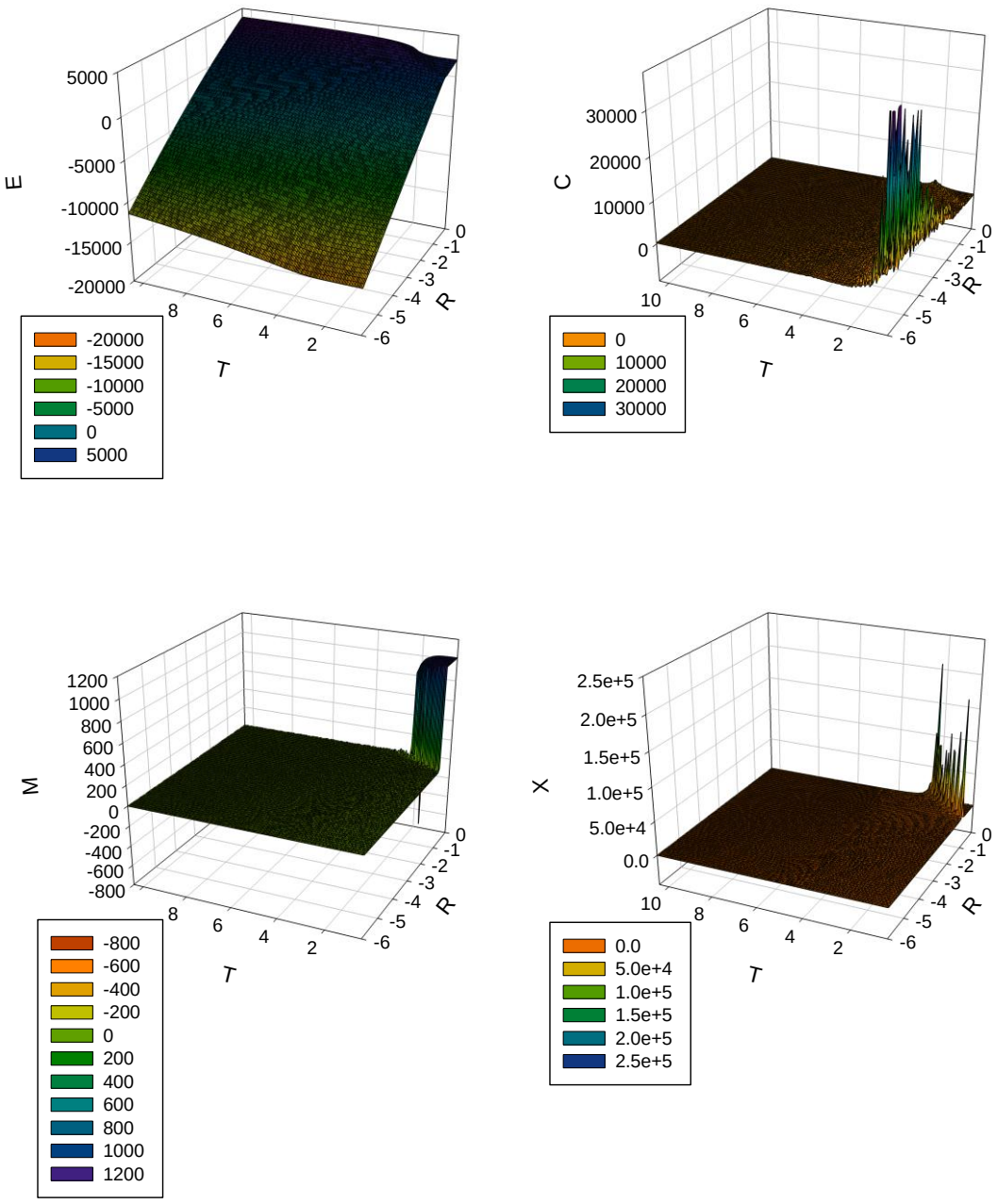


图 7 2D 三角格子 Ising 模型计算结果（R=-6.0~0.0）

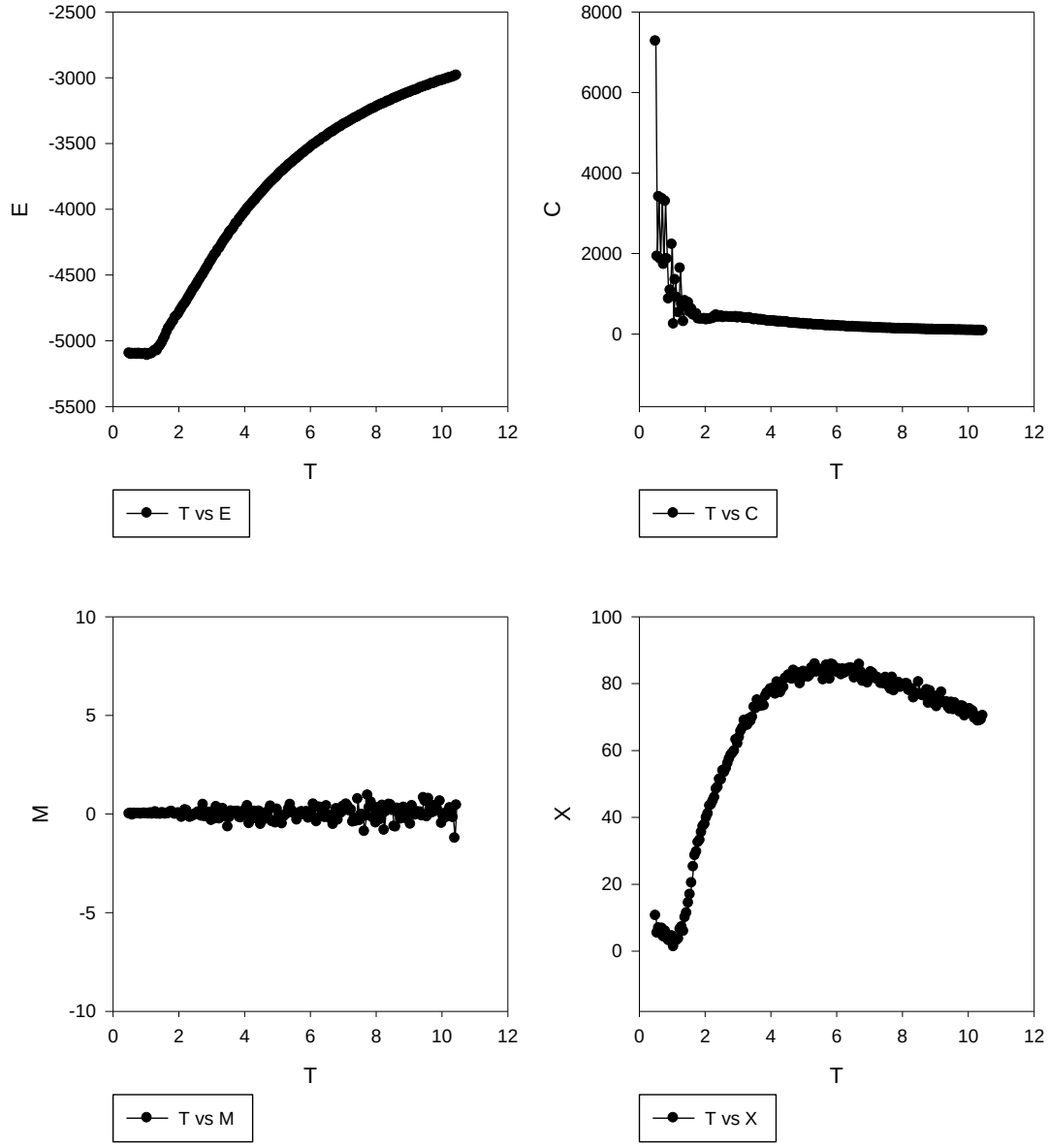


图 8 三角格子 Ising 模型 R=-3.0

附录 2： 程序源代码

```
/*This program calculates the energy, heat capacity, magnetization, magnetic permitivity.*/
/*Boundary condition: periodic boundary condition.*/
/*In this program, J&k are put into T. To fill it to the expressions, just consider the dimensions.*/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

#define L 32
#define NGL  100000          /*Samples Neglected*/
#define N_t 5100000        /*Total samples*/
#define N (N_t-NGL)
#define T_i 0.5
#define T_f 10.5
#define Tstep 0.05
#define SIZE_T 500         /*The size of emcx in T axis. Check this before compiling!*/
#define SIZE_R 60          /*The size of emcx in R axis. Check this before compiling!*/
#define R_i -6                   /*Ration of J2/J1. Initial value.*/
#define R_step 0.1            /*step*/
#define R_f 0                /*End of R*/

int sum_M;
int s[L][L];                        /*Size of the lattice*/
double sum_E;
double T;

struct Obsv
{
   double C;
   double X;
   double M;
   double E;
};

void Inist();
void Mag();
void Energy(double R);
void MCP(double R);
struct Obsv Cac_obsv(double R);

int main()
```

```
{
    int i,j;
    double R;
    struct Obsv emcx[SIZE_T][SIZE_R];
    FILE *ft,*fm,*fx,*fc,*fe,*fr;
    fr=fopen("R.txt","w");
    ft=fopen("T.txt","w");
    fx=fopen("X.txt","w");
    fc=fopen("C.txt","w");
    fm=fopen("M.txt","w");
    fe=fopen("E.txt","w");

    R=R_i;
    srand(time(NULL));
    for(j=0;R<=R_f;j++)
    {
        T=T_i;
        for(i=0;T<=T_f;i++)
        {
            Inist();
            emcx[i][j]=Cac_obsv(R);
            fprintf(fr,"%f\n",R);
            fprintf(ft,"%f\n",T);
            fprintf(fx,"%f\n",emcx[i][j].X);
            fprintf(fc,"%f\n",emcx[i][j].C);
            fprintf(fm,"%f\n",emcx[i][j].M);
            fprintf(fe,"%f\n",emcx[i][j].E);
            printf("-------------\n");
            printf("R=%f\n",R);
            printf("T=%f\n",T);
            printf("X=%f\n",emcx[i][j].X);
            printf("C=%f\n",emcx[i][j].C);
            printf("M=%f\n",emcx[i][j].M);
            printf("E=%f\n",emcx[i][j].E);
            T = T+Tstep;
        }
        R+=R_step;
    }
    fclose(ft);
    fclose(fx);
    fclose(fc);
    fclose(fm);
    fclose(fe);
    printf("The End!\n");
}
```

```
void Inist()
{
    int i,j;
    for(i=0;i<L;i++)
    {
        for(j=0;j<L;j++)
        {
            s[i][j]=1;
        }
    }
}



void MCP(double R)
{
    int i,j;
    double sum,dE;
    i=rand()%L;
    j=rand()%L;



sum=s[i][j]*(s[(L+i-1)%L][j]+s[(i+1)%L][j]+s[i][(j-1+L)%L]+s[i][  (j+1)%L]+R*(s[(i+1)%L][(j-1+L)%L]+s[(i-1+
L)%L][(j+1)%L]));


    dE=2*sum;
    if( dE>0 )
    {
        if(((double)rand()/RAND_MAX)<=exp((-1.0)*dE/T))
        {
            s[i][j]=-s[i][j];
            sum_M+=2*s[i][j];
            sum_E+=dE;
        }
    }
    else
    {
        s[i][j]=-s[i][j];
        sum_M+=2*s[i][j];
        sum_E+=dE;
    }
 }


struct Obsv Cac_obsv(double R)
{
```

```
    int i;
    double E_avg=0.0;
    double E_as=0.0;
    double M_avg=0.0;
    double M_as=0.0;
    struct Obsv emcx;
    Mag();
    Energy(R);

    for(i=0; i<N_t; i++)
    {
       MCP(R);
       if(i>=NGL)
       {
          E_avg+=(double)sum_E/N;
          E_as+=(double)sum_E*sum_E/N;
          M_avg+=(double)sum_M/N;
          M_as+=(double)sum_M*sum_M/N;
       }
    }
    emcx.E=E_avg;
    emcx.M=M_avg;
    emcx.C=(E_as - E_avg*E_avg)/T/T;
    emcx.X=(M_as - M_avg*M_avg)/T;

    return emcx;
}



void Mag()
{
    int i,j;
    int dM_tmp=0;
    for(i=0;i<L;i++)
    {
       for(j=0;j<L;j++)
          dM_tmp+=s[i][j];
    }
    sum_M=dM_tmp;
}

void Energy(double R)
{
    double dE_tmp=0.0;
    int i,j;
```

```
    for(i=0;i<L;i++)
    {
        for(j=0;j<L;j++)
        {

dE_tmp+=0.5*(s[i][j]*(s[(i-1+L)%L][j]+s[(i+1)%L][j]+s[i][(j-1+L)%L]+s[i][(j+1)%L])+R*s[i][j]*(s[(i+1)%L][(
j-1+L)%L]+s[(i-1+L)%L][(j+1)%L]));
        }
    }
    sum_E=dE_tmp;
}
```