实践和修改本讲附件相关程序

- 调整相关参数，给出一个尺寸为32x16的二维Ising相变模拟程序和有意义结果

解答: 代码如下（matlab）：

In [ ]:
```
% Ising 模型模拟参数
Nrows = 32; % 格点的行数
Ncols = 16; % 格点的列数
NMaxSim = 100000;  % 最大模拟步数
N0 = 30000;        % 达到稳定状态之前的步数
n_verbose = 10000; % 每隔多少步输出一次信息
J = 1;             % 交换参数
H = 0.2;           % 外部磁场
Ncase = 9;         % 不同温度的模拟次数
x = [400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000]; % 模拟的温度值

% 初始化能量和图像存储变量
Energy = zeros(1, Ncase);
Image = zeros(Nrows, Ncols, Ncase);

% 主循环：遍历不同的温度
for k = 1:Ncase
    T = x(k); % 当前模拟的温度
    beta = 1 / T; % 逆温度参数
    E = 0; % 系统能量初始化
    S = 2 * (randi([0, 1], Nrows, Ncols) - 0.5); % 初始化自旋矩阵
    fprintf("Start simulation for case %d, where T = %d ...\n", k, T);

    % 内循环：Metropolis 算法
    for t = 1:NMaxSim
        if (mod(t, n_verbose) == 0)
            fprintf("    Try #%d ...\n", t);
        end

        % 随机选择一个自旋
        i = randi([1, Nrows]);
        j = randi([1, Ncols]);

        % 计算能量变化
        deltaE = 2 * J * S(i, j) * (...
            S(mod(i - 2, Nrows) + 1, j) + S(mod(i, Nrows) + 1, j) + ...
            S(i, mod(j - 2, Ncols) + 1) + S(i, mod(j, Ncols) + 1)) + ...
            2 * H * S(i, j);

        % Metropolis 判据
        if (deltaE <= 0 || rand() < exp(-deltaE * beta))
            S(i, j) = -S(i, j); % 翻转自旋
            E = E + deltaE;
        end
    end

    % 计算并存储模拟结束时的平均能量
    Energy(k) = E / (NMaxSim - N0);

    % 存储模拟结束时的自旋配置
    Image(:, :, k) = S;

    fprintf("==== Simulation Finished for case %d with E = %f. ====\n\n", ...
    k, Energy(k));
end
```

```
% 绘制能量分布
figure;
scatter(x, Energy, 'filled');
xlabel('Temperature (k)');
ylabel('Energy');
title('Energy Distribution');

% 绘制自旋配置图像
figure;
for icase = 1:Ncase
    subplot(3, 3, icase);
    imagesc(Image(:, :, icase));
    axis equal tight;
    colormap(gray);
    title(['T = ', num2str(x(icase)), 'k']);
end
```

得到结果： Start simulation for case 1, where T = 400 ...

```
    Try #10000 ...
    Try #20000 ...
    Try #30000 ...
    Try #40000 ...
    Try #50000 ...
    Try #60000 ...
    Try #70000 ...
    Try #80000 ...
    Try #90000 ...
    Try #100000 ...
```

==== Simulation Finished for case 1 with E = 0.000011. ====

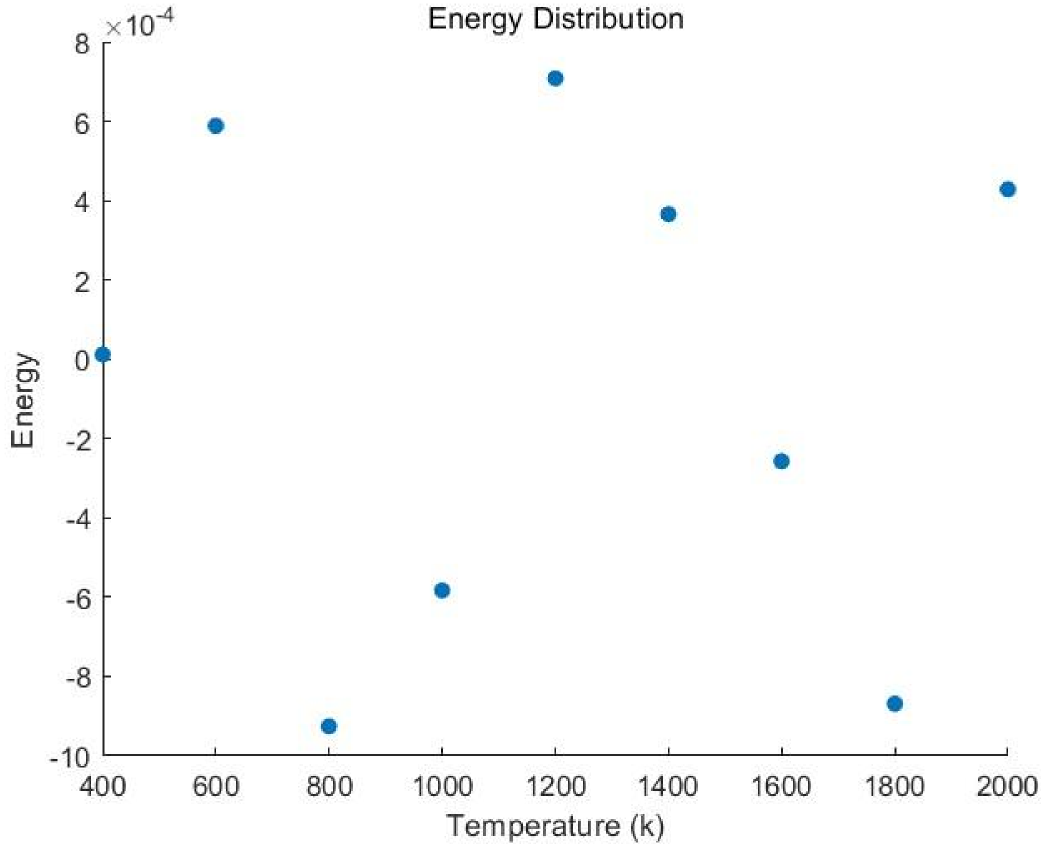
Start simulation for case 2, where T = 600 ...

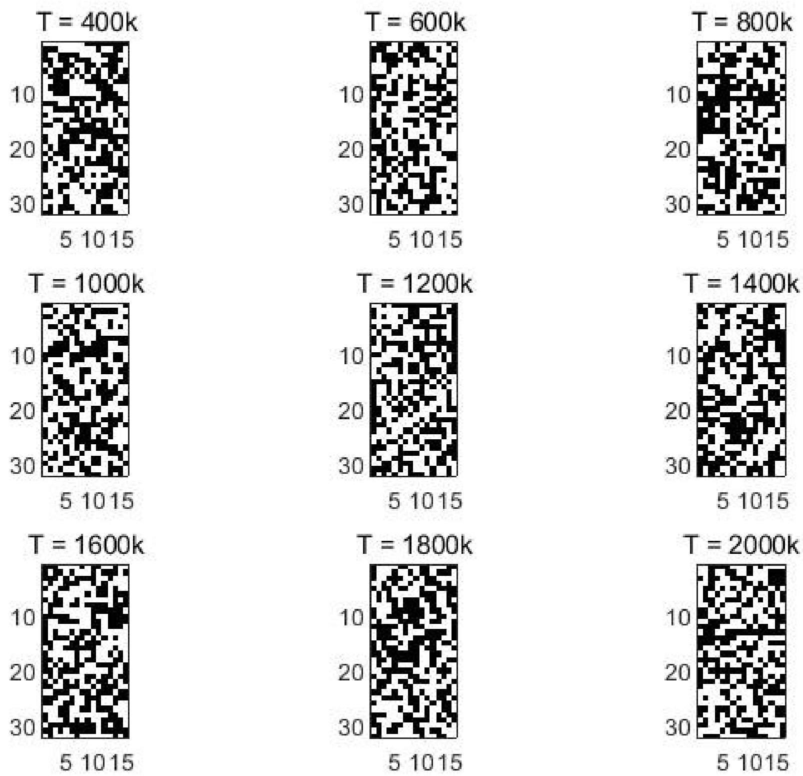
```
    Try #10000 ...
    Try #20000 ...
    Try #30000 ...
    Try #40000 ...
    Try #50000 ...
    Try #60000 ...
    Try #70000 ...
    Try #80000 ...
    Try #90000 ...
    Try #100000 ...
```

==== Simulation Finished for case 2 with E = 0.000589. ====

Start simulation for case 3, where T = 800 ...

```
    Try #10000 ...
    Try #20000 ...
    Try #30000 ...
    Try #40000 ...
    Try #50000 ...
    Try #60000 ...
```





- 运用计算加速技术，你能获得更大尺度的模拟结果吗？请给出相关解释和说明。

解答： 因为我的matlab配置问题，没有配置gpu,故采用了并行运算方法。代码如下（matlab）：

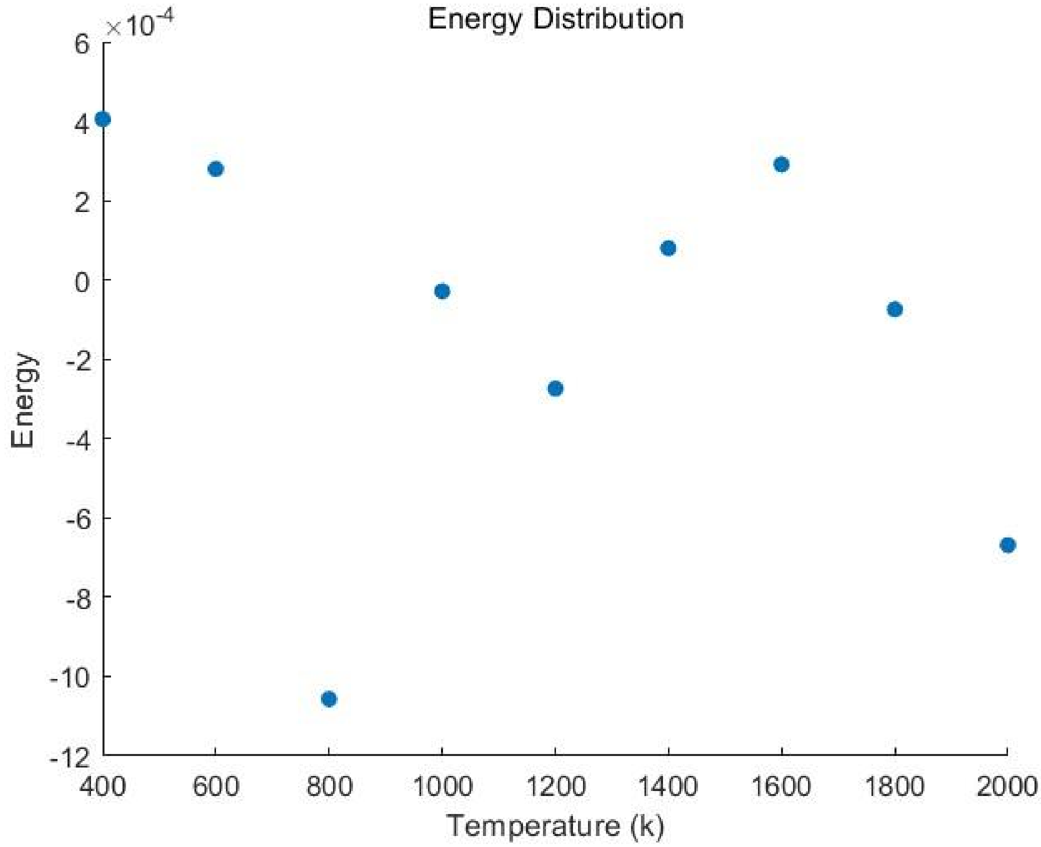
In [ ]:

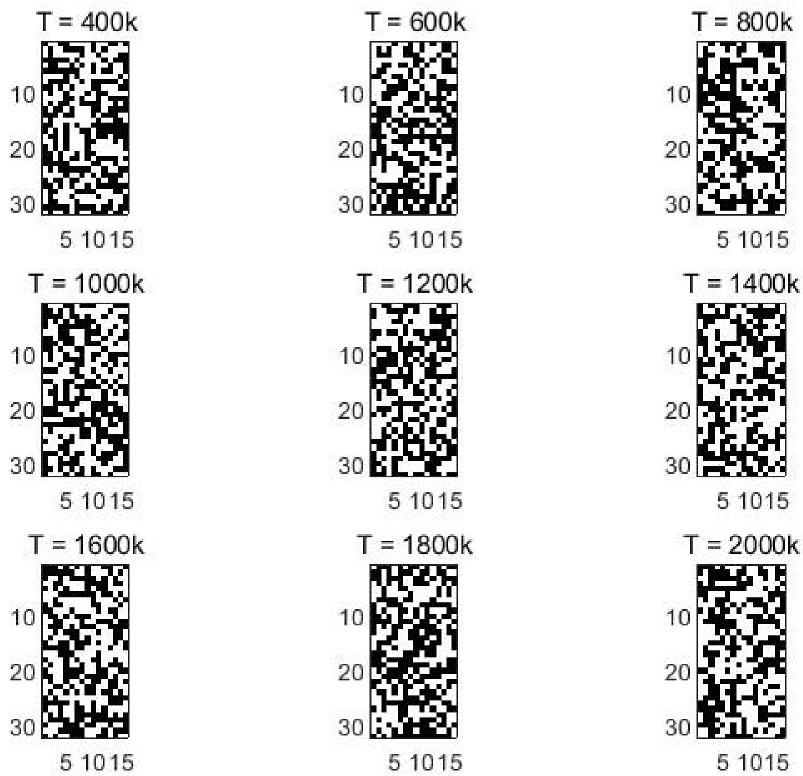
```
% Ising 模型模拟参数
Nrows = 32; % 格点的行数
Ncols = 16; % 格点的列数
NMaxSim = 100000;  % 最大模拟步数
N0 = 30000;        % 达到稳定状态之前的步数
n_verbose = 10000; % 每隔多少步输出一次信息
J = 1;             % 交换参数
H = 0.2;           % 外部磁场
Ncase = 9;         % 不同温度的模拟次数
x = [400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000]; % 模拟的温度值

% 初始化能量和图像存储变量
Energy = zeros(1, Ncase);
Image = zeros(Nrows, Ncols, Ncase);

% 启动 MATLAB 并行池
if isempty(gcp('nocreate'))
    parpool; % 如果还没有启动并行池，就启动它
end

% 使用 parfor 循环并行处理不同温度下的模拟
parfor k = 1:Ncase
    T = x(k); % 当前模拟的温度
    beta = 1 / T; % 逆温度参数
    E = 0; % 系统能量初始化
    S = 2 * (randi([0, 1], Nrows, Ncols) - 0.5); % 初始化自旋矩阵

    % 内循环：Metropolis 算法
    for t = 1:NMaxSim
        if (mod(t, n_verbose) == 0)
            fprintf("     Try #%d ...\n", t);
        end

        % 随机选择一个自旋
        i = randi([1, Nrows]);
        j = randi([1, Ncols]);

        % 计算能量变化
        deltaE = 2 * J * S(i, j) * (...
            S(mod(i - 2, Nrows) + 1, j) + S(mod(i, Nrows) + 1, j) + ...
            S(i, mod(j - 2, Ncols) + 1) + S(i, mod(j, Ncols) + 1)) + ...
            2 * H * S(i, j);

        % Metropolis 判据
        if (deltaE <= 0 || rand() < exp(-deltaE * beta))
            S(i, j) = -S(i, j); % 翻转自旋
            E = E + deltaE;
        end
    end

    % 计算并存储模拟结束时的平均能量
    Energy(k) = E / (NMaxSim - N0);

    % 存储模拟结束时的自旋配置
    Image(:, :, k) = S;
end

% 绘制能量分布
figure;
scatter(x, Energy, 'filled');
xlabel('Temperature (k)');
ylabel('Energy');
title('Energy Distribution');
```

```
% 绘制自旋配置图像
figure;
for icase = 1:Ncase
    subplot(3, 3, icase);
    imagesc(Image(:, :, icase));
    axis equal tight;
    colormap(gray);
    title(['T = ', num2str(x(icase)), 'k']);
end
```

得到结果： Starting parallel pool (parpool) using the 'Processes' profile ... Connected to the parallel pool (number of workers: 4).

```
Try #10000 ...
Try #10000 ...
Try #10000 ...
Try #20000 ...
Try #30000 ...
Try #40000 ...
Try #50000 ...
Try #60000 ...
Try #70000 ...
Try #80000 ...
Try #90000 ...
Try #100000 ...
Try #10000 ...
Try #20000 ...
Try #30000 ...
Try #40000 ...
Try #50000 ...
Try #60000 ...
Try #70000 ...
Try #80000 ...
Try #90000 ...
Try #100000 ...
Try #10000 ...
Try #20000 ...
Try #30000 ...
Try #40000 ...
Try #50000 ...
Try #60000 ...
Try #70000 ...
Try #80000 ...
Try #90000 ...
Try #100000 ...
Try #20000 ...
Try #30000 ...
Try #40000 ...
Try #50000 ...
Try #60000 ...
Try #70000 ...
Try #80000 ...
Try #90000 ...
Try #100000 ...
Try #10000 ...
Try #20000 ...
Try #30000 ...
```


Energy Distribution
$\times 10^{-4}$
6
4
2
0
-2
-4
-6
-8
-10
-12
Energy
400
600
800
1000
1200
1400
1600
1800
2000
Temperature (k)


T = 400k
T = 600k
T = 800k
T = 1000k
T = 1200k
T = 1400k
T = 1600k
T = 1800k
T = 2000k
10
20
30
5 10 15