HTML5 语言下的二维 Ising 模型模拟

摘要：本文介绍了一种在 HTML5 语言环境下对二维 Ising 模型的模拟和展示。本文使用 HTML5 语言，通过 HTML5 元素的设计实现了对 Ising 模型模拟的温度与外场的控制，通过 Metroplis 方法对二维 Ising 模型进行了模拟，得到并展示了计算结果，并且通过动画直观的体现了 Ising 模型的迭代过程与结果。

一、引言

HTML5 语言是一种网页开发技术，其相较与上一代 HTML 标准，增加了一些新特性，例如用于绘画的 Canvas 元素等。现代的浏览器都支持 HTML5，可以进行跨平台开发。网页通常使用 HTML、CSS、JavaScript 三种语言开发，其中 HTML 语言定义了网页的内容，如生成动画区以及控制杆，实现人机间的交互；CSS 语言则用于控制网页上各元素的位置，对各元素的样式及位置进行规定，如控制文字大小、样式和位置；JavaScript 语言将输入的值进行运算，并传递给 HTML 语言输出。

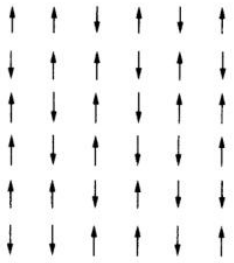
本文通过 JavaScript 语言实现了对二维 Ising 模型在 Metroplis 方法下的迭代模拟，并且通过 HTML 语言将整个模拟过程通过动画展现，并且通过 HTML 语言展示了模型的能量和磁矩在整个迭代过程中的变化。

二、原理

Ising 模型是热力学中较为经典的模型，是描述磁系统相变的一种最简单的模型。设共有 N 个自旋，分布在模型格点上，如图一所示，每个自旋只能取自旋向上或向下两个态。点阵可以是一维、二维、三维点阵，这种自旋体系称为 Ising 模型。假设只有近邻的自旋间有相互作用，系统的哈密顿量为：

$$H = -J\sum_{(i,j)} s_i s_j - \mu H \sum_{i=1}^{N} s_i$$

其中 s_i 表示格点 i 的自旋，值取+1 或-1，分别对应自旋向上与自旋向下的情况。J 为耦合系数，这里假设 J>0，表示铁磁体系。$\sum_{(ij)}$ 表示对近邻的自旋对求和，当近邻两个自旋同向时，相互作用能为-J，反向时相互作用能为 J。μ 为与自旋相联系的磁矩，H 为外场场强。[1]



图一.二维 Ising 模型示意图

二维 Ising 模型的严格解在 1944 年由昂萨格求得。在磁场为零时，二维 Ising 模型的相变温度为

$$T_c = 2.269J / k$$

在相变点附近，热力学极限下的热力学函数显现出奇异性。[1]

本文使用 Metroplis 方法对二维 Ising 模型进行取值抽样。假设初态为 S，某个格点上的自旋翻转带来的能量变化为ΔE，以 P 概率接受此次翻转。

$$P = \begin{cases} 1, \Delta E \le 0 \\ e^{-\Delta E / KT}, \Delta E > 0 \end{cases}$$

接受此次翻转则以翻转后自旋态 S'作为新的迭代初态，不接受则仍以 S 态作为新的初态。当对所有的格点做足够多次操作（实验中通常为$10^8 \sim 10^9$次操作）后，能量的平均偏差小于 0.01%，总磁矩的平均偏差小于 1%，可认为此时模型达到一个近似稳态。

为了在网页上实现对 Ising 模型的模拟和展示，我们需要分前端和后台两部分设计整个程序。前端用以实现模型的外部参数的输入和结果的展示。后台用以计算模型各项结果。前端的功能通过 HTML 实现，后台通过 JavaScript 语言实现。

外部参数的输入，如温度 T 与外场 H 的输入，点阵的行列数，这部分输入可以通过创建数据类，定义为“输入”，并且通过 ID 来获得输入值。输入的方式可以通过创建滑竿或者空格输入。展示结果，如迭代过程中模型的每一步的瞬时能量、平均能量、能量偏差，瞬时磁矩、平均磁矩及磁矩偏差，迭代步数，可以通过在网页上预留元素位置并定义数据的类“数据”，计算得到结果后通过 ID 赋值展示；较为复杂的结果展示，如能量、磁矩随迭代步数的变化，以及整个二维 Ising 点阵的展示，可以预留 Canvas 元素位置，通过 JavaScript 控制动画。本文还通过选择函数设计了控制点阵大小、图表种类的选择按键，以及控制动画更新的步进、暂停/开始、重启按键，以及重新对数据进行统计的按键。

后台的计算，包括主要的从前端获得输入，生成初始格点，迭代计算格点自旋态，以及控制结果输出。JavaScript 语言可以通过输入时定义的元素 ID 来获得输入值，这些输入值可能在输入时为字符串，因此需要通过一定的函数处理将其转换为浮点数。迭代格点计算时，可以在行数与列数上生成随机数，然后将对应的自旋反向，计算能量差，用 Metroplis 方法对新自旋态进行判断，且对迭代次

数进行计数，将全格点均翻转一次记为一步，并且在新自旋态时重新计算能量与磁矩并将其赋值给 HTML 语言中预留的元素来显示。对于能量与磁矩随步数的变化，以及二维 Ising 点阵自旋态的展示，JavaScript 语言中可以通过动画更新函数来实现。在 HTML 中还设置了对点阵大小的控制、图表种类的控制以及对动画更新、数据统计的控制，JavaScript 通过元素的 ID 获得 HTML 中的输入值后，通过选择支对交互做出反应，实现对不同输入的反应。

CSS 语言可以对 HTML 语言中定义的字符串元素、浮点数以及 canvas 元素的样式，如大小、形状、颜色、位置进行定义，使得各元素在网页上排布更加有序。由此，JavaScript 语言在输出结果时，应对结果的精度做出一定的处理，以使得各元素不会因为精度显示打乱原排布。

三、实验：代码实现

本文重点介绍 HTML、JavaScript 语言对二维 Ising 模型模拟中的一些重点的代码实现，CSS 语言主要用于规划网页布置，不做特别的说明。

在 HTML 语言中，由于本文设计需要使用中文，以及在主体运行时需要引用 JavaScript 与 CSS 脚本，在<head>中需要做引用说明，具体代码如图二。

```
<head>
    <meta charset="utf-8"/>
    <script src="ising.js"></script>
    <link rel="stylesheet" href="style.css">

    <title>ising</title>
</head>
```

图二

为展示 JavaScript 计算反馈的结果，HTML 语言中需要为其预留元素位置，并且规定元素 ID 以赋值。具体代码如图 3.1、3.2 所示。

```
<!--显示屏开始-->
<canvas id='canvas' height=512 width=512></canvas>
<!--显示屏结束-->
```

图 3.1 canvas 元素预留

```
<pre class='数据' id='步数'>初始化</pre>
<pre class='数据' id='能量'>初始化</pre>
<pre class='数据' id='磁矩'>初始化</pre>
```

图 3.2 数据元素预留

为输入 T、H 等模型迭代参数，HTML 语言中需要创建滑竿与输入空格来输入值。本文代码设置了温度 T、外场强 H 以及动画更新速度的输入，其中温度 T 的滑竿输入处理稍稍与其他值不同：滑竿可以在一个较小的范围内输入值，如 -5~5，且输入的值精度较低。而 Ising 模型的相变温度为 2.267K，本实验中还希望能展示从 0~100K 的整个变化过程，因此本实验对温度输入进行了改进：输入

范围设置为-6~2，将输入值以 10 为底指数化作为输入温度，代码如图四所示。

```
<div>
    <label class='输入值' for="temp" title="Temperature">温度</label>
    <!--造一个滑竿-->
    <input style='width:150px;' type="range" id="temp" min=-6 max=2 step=0.000001 value=0 oninput='update_temp()' onchange="update_temp()"></input>
    <!--输入并显示值-->
    <span onclick='dotextbox(this.id)' class='输入' id="label_temp"></span>
    <input style='width: 80px; display: none;' id='label_temp_input' type='text' value='' onblur='undotextbox(this.id)'>
</div>
```

图四 温度参数输入代码

在设计中，本实验需要实现对格点点阵大小的调节，以及图表种类的调节。实验中使用 select 来创建选择支，规定选择支 ID 以便 JavaScript 语言进行识别，通过设置 value 来给后台处理时赋值，通过<option>设置选项，并且设置 onchange 事件在选择时改变赋值。以点阵大小的选择支设计为例，代码如图五所示。

```
<div class="选择支">
    <label class='输入值' for='num'>点阵规格</label>
    <select id='changenum' onchange='change_num()'>
        <option value="512">512 X 512</option>
        <option value="256" selected="selected">256 X 256</option>
        <option value="128">128 X 128</option>
        <option value="64">64 X 64</option>
        <option value="32">32 X 32</option>
        <option value="16">16 X 16</option>
    </select>
</div>
```

图五 温度参数输入代码

实验的设计中需要对动画的更新与统计做出调控，因此设计中包含了步进、暂停/开始、重启，以及重新对数据进行统计的按键。HTML 语言通过<input type=button>生成按键控件，并且设置事件 onclick。具体代码如图六所示。

```
<div>
    <input type='button' id='dostep' value='步进'   onclick='update_step()'></input>
    <input type='button' id='pause' value='暂停'   onclick='update_pause()'></input>
    <input type='button' id='restart' value='重启' onclick='update_restart()'></input>
    <input type='button' id='resetdata' value='重新统计' onclick='init_measurements()'></input>
    <br>
</div>
```

图六 按钮控制代码

在 JavaScript 语言中，代码实现的是对输入的反馈，计算 Ising 模型的能量和磁矩并运用 Metroplis 方法进行迭代，最后输出结果。

外部输入的实现在此实验中通过 getElementById 方式获得。需要注意的是，HTML 输入的与输出的数据均为字符串，而计算中所需的均为浮点数，因此需要将输入的字符串转换为浮点数，需要用 parseFloat 函数。以外磁场的输入为例，将 getElementById 得到的输入磁场的字符串通过 parseFloat 转换为浮点数，通过变量 gfield 运用于计算中，代码如图七所示。

```
gfield = parseFloat(document.getElementById('field').value);
```

图七 外部输入磁场代码

而输出也需要通过字符串输出，同时对各个数据的输出的精度应做一定的规定，以便在网页上显示时不会打乱布局。因此此实验中使用自定义函数来将计算得到的浮点数进行精度处理并转换为字符串。这个函数的思路是将浮点数扩大精度的倒数倍取整，然后将整数部分、小数部分与零的部分分别转换为字符串，连接起来最后输出。具体代码如图八所示。

```
//处理值精度
function toFixed(value, precision) {
    var precision = precision || 0;
    var sneg = (value < 0) ? "-" : " ";
    var neg = value < 0;
    var power = Math.pow(10, precision);
    var value = Math.round(value * power);
    var integral = String(Math.abs((neg ? Math.ceil : Math.floor)(value/power)));
    var fraction = String((neg ? -value : value) % power);
    var padding = new Array(Math.max(precision - fraction.length, 0) + 1).join('0');
    return sneg + (precision ? integral + '.' +  padding + fraction : integral);
}
```

图八 输入字符串函数 toFixed 代码

而输出数据至网页的方法则是先通过 getElementById 定位需要展示数据的预留位置，然后使用 innerHTML 将数据放在位置上。有时需要在一个位置上放多个数据，则可以使用+=赋值。具体代码如图九所示。

```
//显示返回数据
function update_measurements_labels(){
    t = document.getElementById('步数');
    e = document.getElementById('能量');
    m = document.getElementById('磁矩');

    t.innerHTML = "步数 = "+toFixed(gt, 4)+"     每秒步数 = "+toFixed(sps, 3);
    e.innerHTML = "能量 = "+toFixed(genergy, 5);
    m.innerHTML = "磁矩 = "+toFixed(gmag, 5);

    t.innerHTML += "     全晶格均运算一次为一步"
    e.innerHTML += "     平均能量 = "+toFixed(ge_avg, 5);
    m.innerHTML += "     平均磁矩 = "+toFixed(gm_avg, 5);

    e.innerHTML += "     能量偏差 = "+toFixed(ge_var, 9);
    m.innerHTML += "     磁矩偏差 = "+toFixed(gm_var, 9);
}
```

图九 输出计算结果代码

对 Ising 模型的迭代计算较为简单，将每个格点自旋以序号标记后计算总能量，使用 Metroplis 方法判断迭代。在计算时，通过 push 函数将每步计算得到的数据，如平均能量等，组合成为一个数组，以便画图时使用。二维 Ising 模型的

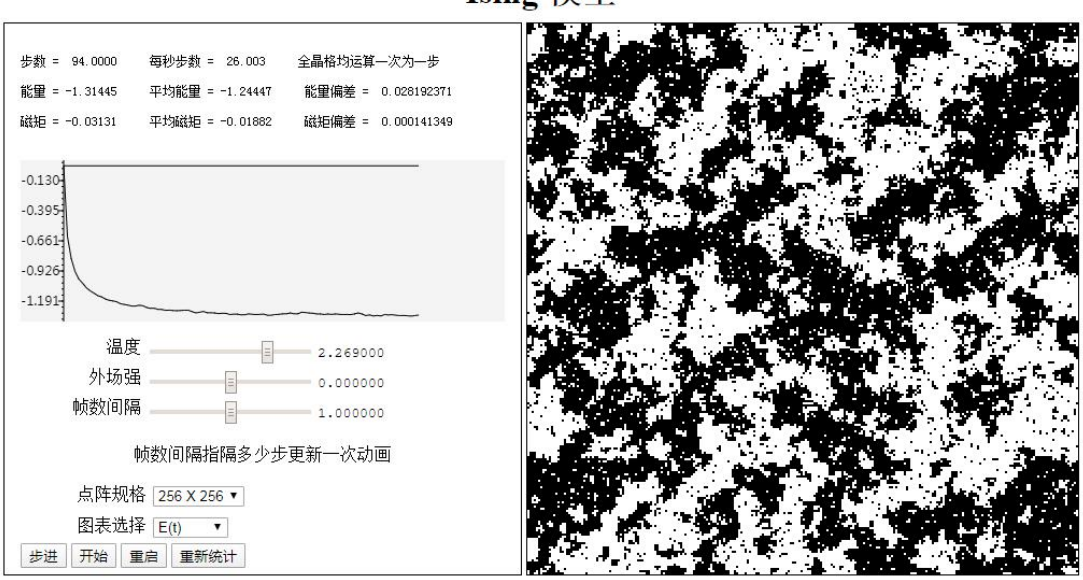
迭代方式在原理中介绍的较详细且代码的实现较为简单，这里仅展示用以画能量-步数与磁矩-步数图的代码以及平均能量、平均磁矩、能量偏差、磁矩偏差的代码，如图十所示。

```
function push_measurement(t, e, m){
    times.push(t);
    gtimeseries_energy.push(e);
    gtimeseries_mag.push(m);

    n = times.length;
    ge0 = ge_avg;
    gm0 = gm_avg;

    ge_avg = ge_avg + (e - ge_avg)/n;
    ge_var = ((n-1)*ge_var + (e - ge_avg)*(e - ge0)) / n;
    gm_avg = gm_avg + (m - gm_avg)/n;
    gm_var = ((n-1)*gm_var + (m - gm_avg)*(m - gm0)) / n;
    gtimeseries_eavg.push(ge_avg);
    gtimeseries_mavg.push(gm_avg);


    sps = 1000.0*sweeps/(Date.now() - lasttime);
}
```

图十 计算数据代码

四、实验结果

运行代码即可得到图十一所示网页，可以看到设计的部分基本上均实现了。



图十一 最终结果

左上角为模型计算结果的展示，左中为动画展示图表，左下为各项参数的输入，点阵大小、图表的选择以及控制按钮。右边为 Ising 自旋点阵的动画展示。

从最终运行结果可以看出，设计的效果均有实现。步数、能量、磁矩等统计

量均在设计的位置展示出来了，且均根据所设计的精度进行了修整。图表也达到了预期，能够画出各统计量随步数的变化，温度、磁场与帧数间隔都能如设计调整，其中温度可以从 0.000001~100.000000 调整。点阵规格选择栏可以选择多个点阵规格，图表选择栏可以选择图表种类，步进按钮可以暂停 Ising 自旋点阵更新，暂停/开始按钮可以控制动画的暂停和开始更新，重启按钮可以将自旋点阵还原至初态并重新迭代，重新统计按钮可以将统计数组清空重新计数。右侧由黑白格子代表不同的两种自旋态，整个点阵表示 Ising 模型自旋格点态。

五、结论

通过 HTML、JavaScript、CSS 语言，本实验成功实现了二维 Ising 模型在 Metroplis 方法下的迭代模拟，并通过设计输入滑竿，实现了外部对模型的参数控制，并且将能量等统计量在网页上展示。通过动画代码，实现了各统计量随步数的变化展示，以及模型自旋点阵随步数的迭代展示，并且通过代码设计实现了对动画的开始、暂停、重启调节。