Содержание

1. Поиск минимума энергии **2**
1.1. Описание модели . . . 2
1.2. Задача . . . 2
1.3. Подходы к решению . . . 2

2. Алгоритм Метрополиса **4**
2.1. Описание модели . . . 4
2.2. Задача . . . 4
2.3. Подходы к решению . . . 4

3. Ускорение алгоритма Метрополиса, визуализация программы **5**
3.1. Описание модели . . . 5
3.2. Задача . . . 5
3.3. Подходы к решению . . . 6

4. Реплично-обменный Метрополис (параллельный отжиг) **7**
4.1. Описание модели . . . 7
4.2. Задача . . . 8
4.3. Рекомендации к выполнению . . . 9

5. Алгоритм Ванга-Ландау **10**
5.1. Описание модели . . . 10
5.2. Задача . . . 12
5.3. Рекомендации к выполнению . . . 12

Литература **15**

Лабораторная работа 1

Поиск минимума энергии

1.1. Описание модели

Дана квадратная решетка из взаимодействующих спинов, принимающих значения $S_i = +1/-1$. Спины взаимодействуют только с ближайшими соседними. В решетке действуют периодические граничные условия - виртуальные замкнутые края решетки. Например, левым соседом крайнего левого спина будет крайний правый спин.

Конфигурация системы определяется энергией, которая задана уравнением:

$$E = \sum_{<i,j>} -J \cdot S_i \cdot S_j, \tag{1.1}$$

где $J = \pm 1$ - тип взаимодействия, определяющий основные характеристики системы. Может быть ферромагнитным $(+1)$ либо антиферромагнитным (-1).

При инициализации программы система должна иметь случайную конфигурацию.

1.2. Задача

Необходимо реализовать алгоритм Монте-Карло для нахождения конфигурации с минимальной энергией системы.

Вывести на экран конфигурацию с минимальной энергией для

Размер решетки задается динамически - то есть, либо в коде выделена глобальная переменная, либо размер будет являться консольным параметром программы. В качестве размера задается L -количество спинов вдоль одной стороны квадрата, тогда $N = L \cdot L$ - суммарное число ячеек системы.

1.3. Подходы к решению

Задачу предлагается решать в два шага:

1. Можно представить все спины как биты двоичных чисел. Например -1 будет битом «0», а +1 битом «1». Тогда получится перебрать все конфигурации перечисляя все возможные варианты бинарных чисел, и считая энергию для каждой конфигурации. За счет выполнения этого шага можно построить и отладить модель системы. Однако, полный перебор - сложная для вычислителя операция, и таким образом получится посчитать системы с $L \leq 6$.

2. Случайным образом выбирать спины системы. Попробовать перевернуть спин. Если в результате переворота новая энергия понизилась, принять переворот. Иначе отменить переворот. Повторять выбор спина m раз. Итоговая конфигурация (теоретически) будет обладать минимальной энергией.

Лабораторная работа 2

Алгоритм Метрополиса

2.1. Описание модели

Модель аналогична лабораторной 1.

2.2. Задача

Необходимо модифицировать алгоритм поиска основного состояния из предыдущей работы, добавив в него температуру T - параметр, отвечающий за динамический беспорядок системы, и принимающий значения от 0.001 до 5. Вероятность принятия новой конфигурации будет определяться следующим уравнением:

$$P_{t\to(t+1)} = \min[1, e^{\frac{\Delta E}{T}}], \tag{2.1}$$

где ΔE - разница между текущей энергией текущей конфигурации и следующей.

Запустить алгоритм для разных температур. Для каждой температуры определить среднюю энергию, появляющуюся в системе на каждом шаге Метрополиса. Для каждой температуры выполнить как минимум $100 * N$ шагов Метрополиса. Ответить на следующие вопросы:

1. Как зависит средняя энергия от температуры?
2. Как ведет себя производная средней энергии по температуре? Испытывает ли она пик? При какой температуре?
3. Какой максимальный размер системы можно посчитать, если тратить на семплирование одной температуры не более 1 минуты?

2.3. Подходы к решению

С учетом модификации программы, полученной на предыдущей лабораторной, алгоритм работы будет следующий:

1. Выбираем случайный спин, и пробуем его перевернуть;
2. Вычисляем энергию системы с учетом одного перевернутого спина;
3. Принимаем новую конфигурацию с вероятностью из уравнения 2.1, иначе восстанавливаем предыдущую конфигурацию.

Лабораторная работа 3

Ускорение алгоритма Метрополиса, визуализация программы

3.1. Описание модели

Модель аналогична лабораторной 1. Как можно заметить, алгоритм из лабораторной работы 2 работает не особо быстро. Если проанализировать код программы, станет ясно что больше всего времени занимает вычисление энергии новой конфигурации. В системе из N спинов будет $2 \times N$ уникальных парных взаимодействий. Соответственно, требуется выполнить $2 * N$ шагов цикла для получения энергии новой конфигурации.

Это время можно сократить до 4 шагов. Из уравнения

$$E = \sum_{<i,j>} -J \cdot S_i \cdot S_j \tag{3.1}$$

видно, что если у спина всего 4 соседа, то он будет участвовать только в 4 парных взаимодействиях. Также, заранее известно, что при перевороте спина меняется только его знак. То есть, знаки всех парных взаимодействий, в которых участвует перевернутый спин, изменятся на противоположные. Соответственно, энергию новой конфигурации E_{new} системы можно вычислить зная энергию системы до переворота E_{old} и номер (k) перевернутого спина.

Энергия новой конфигурации будет:

$$E_{\text{new}} = E_{\text{old}} + 2JS_k \sum_{<j>} S_j, \tag{3.2}$$

где суммирование проходит только по значениям спинов, соседствующих с k-ым.

Таким образом, скорость алгоритма не будет более зависеть от N.

3.2. Задача

Ускорить код программы, полученной в результате выполнения лабораторной работы 3.

Визуализировать работу программы в виде картинки-анимации, где каждый спин представляет пиксель на картинке, цвет которого отображает значение спина. Например, черный (-1), и белый (+1). Новый кадр строить каждые N шагов Метрополиса. Пример визуализации доступен по ссылке http://mattbierbaum.github.io/ising.js/.

Провести наблюдения работы системы при различных температурах: как происходит упорядочение хаотичной системы? Как можно объяснить такой эффект?

3.3. Подходы к решению

Для ускорения подсчета энергии, нужно на каждом шаге хранить глобальную переменную E_{old}, в которой будет сохранена актуальная энергия текущей конфигурации системы. При обновлении системы в неё будет записана обновленная энергия. Стоит заметить, что в уравнениях 2.1 и 3.2 переменные ΔE и $2JS_k \sum_{<j>} S_j$ - это одни и те же значения. Соответственно, E_{new} нужно вычислять только в случае принятия конфигурации.

Для визуализации можно использовать любой удобный подход, исходя из собственных навыков программирования. Самый простой вариант - сохранить покадровые данные для анимации в файл, и отобразить используя существующие решения для визуализации научных результатов: *paraview* / *gnuplot* / *matplotlib*, и т.д.

Лабораторная работа 4

Реплично-обменный Метрополис (параллельный отжиг)

4.1. Описание модели

Данная работа является логическим продолжением предыдущей. Код программы можно разрабатывать путем модификации ранее разработанного и оттестированного кода. Физическая модель аналогична описанной в лабораторной работе 1.

При реализации работы 3 можно видеть, что требуется очень много времени для приведения системы из беспорядочного состояния к порядку. В системе появляются локальные «домены» - зоны с минимальной энергией, но различной намагниченности. Моделирование при низких температурах может погрузить систему в зону локального минимума энергии (заштрихованные области на рис. 4.1). При более высоких температурах симуляция может покрыть более широкие области пространства. Перестановки конфигураций между системами с более низкой и более высокой температурой позволяют системам с более низкой температурой выходить из одной области фазового пространства, где они были фактически «застряли», и отбирать репрезентативный набор минимумов с низкой свободной энергией.

Общая идея параллельного отжига состоит в том, чтобы параллельно моде-

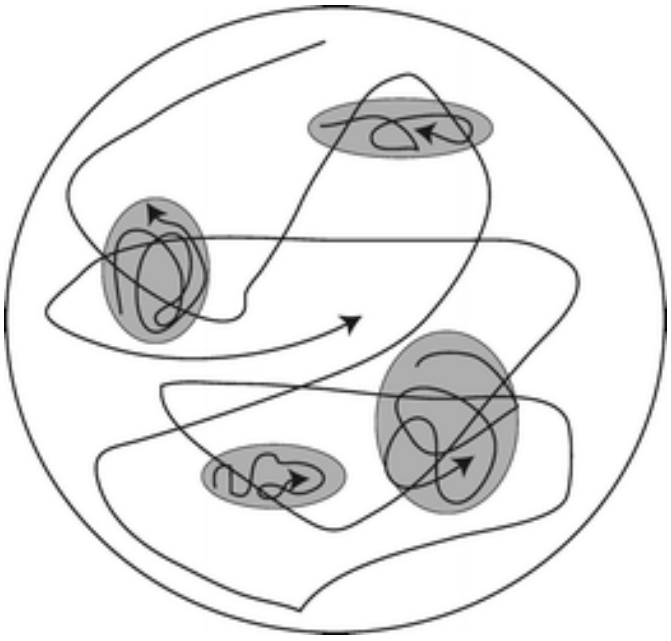

Рис. 4.1. 2D представление фазового пространства моделируемой системы.

Рис. 4.2. Схематическое представление процесса обмена конфигурациями для алгоритма параллельного отжига.

лировать M независимых реплик (копий) исходной системы. Каждая реплика обычно находится в каноническом ансамбле, и обычно каждая реплика имеет разную температуру. Высокотемпературные системы, как правило, способны семплировать большие объемы фазового пространства, тогда как низкотемпературные системы, хотя и имеют точную выборку в локальной области фазового пространства, могут попасть в локальные минимумы энергии во время типичного компьютерного моделирования.

В типичном моделировании параллельного отжига мы имеем M реплик, каждая в каноническом ансамбле, и каждая с разной температурой, T_i. В общем случае $T_1 < T_2 < \ldots < T_M$. Реплики не взаимодействуют энергетически. Обмен конфигурациями между репликами i и j выполняется с вероятностью:

$$P_{\text{acc}} = \min\left(1, e^{(E_i - E_j)(\frac{1}{T_i} - \frac{1}{T_j})}\right). \tag{4.1}$$

Попытки обмена применяются только для систем с соседними температурами: $j = i + 1$. Обмен конфигурациями может выполняться после любого числа шагов классического Метрополиса. На рисунке 4.2 показана классическая последовательность обмена конфигурациями для системы из пяти температур.

4.2. Задача

Реализовать алгоритм параллельного отжига с несколькими репликами. Для каждой реплики вычислить значения $\langle E \rangle$ и $\langle E^2 \rangle$, построить график теплоемкости. Построить аналогичный график для последовательного алгоритма Ванга-Ландау. Сравнить график с результатом предыдущей работы.

Ответить на следующие вопросы:

1. Как изменилась скорость работы программы?
2. Как алгоритм повлиял на точность результатов?

3. Можно ли с новым алгоритмом уменьшить число Монте-Карло шагов без потери точности результата?

4.3. Рекомендации к выполнению

Теплоемкость для термодинамических систем определяется как:

$$C(T) = \frac{d\langle E\rangle}{dT^2} = \frac{\langle E^2\rangle - \langle E\rangle^2}{T^2}. \tag{4.2}$$

Второй вариант более удобен при компьютерном моделировании, так как не требует численного дифференцирования. Значения теплоемкости прямо пропорциональны размеру системы, поэтому рационально приводить их на графиках в безразмерном виде, как $C(T)/N$.

Лабораторная работа 5

Алгоритм Ванга-Ландау

5.1. Описание модели

Физическая модель аналогична описанной в лабораторной работе 1, однако подход к реализации алгоритма существенно изменится.

Как и метод Метрополиса [1], ВЛ-метод принадлежит группе МК-методов. В основу работы таких методов положено условно случайное блуждание по пространству состояний. Однако, алгоритмы, используемые этими методами отличаются способом выборки или подходом к семплированию, т.е. способом составления выборочной совокупности, части генеральной совокупности состояний, которая охватывается одним экспериментом, серией последовательных или параллельных экспериментов. Для построения выборки в ВЛ-методе используется равновероятное семплирование. Распределение вероятностей энергетических уровней, или ПВС, представляется в виде гистограммы $g(E)$, рис. 5.1.

Также, см., напр. [2, 3], существует возможность вычислить многомерное распределение ПВС любой измеряемой в численном эксперименте термодинамической величины $X(E)$ и получить ее среднее значение

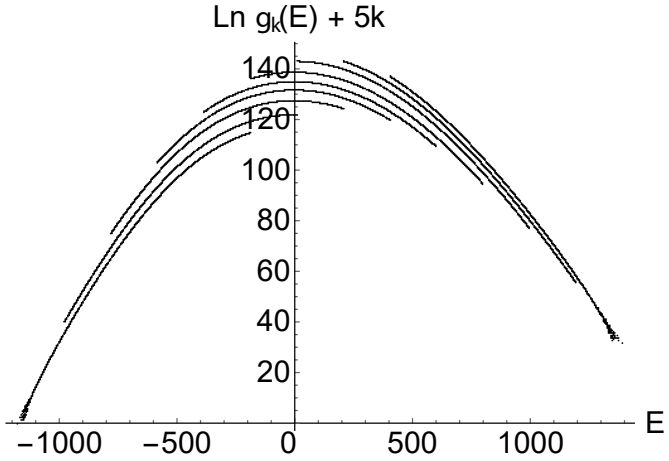

Рис. 5.1. Пример нормированной гистограммы $\ln[g(E)]$, полученной для модели квадратного спинового льда 180 спинов параллельным ВЛ-методом с дальнодействующим взаимодействием между диполями. Энергетическое пространство разделено на 9 равных интервалов с перекрытием 75%. Части $g_k(E)$ сдвинуты вдоль оси ординат для наглядности.

$$\langle X \rangle = \frac{\sum_E X(E) g(E) \exp\left[-\dfrac{E}{kT}\right]}{\sum_E g(E) \exp\left[-\dfrac{E}{kT}\right]}. \tag{5.1}$$

Гистограмма $g(E)$ обновляется каждый шаг независимо от принятия или отмены конфигурации. Такой подход позволяет значительно уменьшить время работы алгоритма.

Принцип работы и детали реализации последовательного ВЛ-алгоритма изложены в основополагающих работах [4, 5], см. также [6, 7]. Дополнительно к $g(E)$ формируется гистограмма $H(E)$, которая служит индикатором равномерного обхода всех возможных энергетических уровней системы. Изначально задаются следующие начальные значения: $g(E) = 1$, $H(E) = 0$, $\forall E$, модификационный фактор $f = e^1 \simeq 2.7182818$, влияние точности которого на результат и скорость сходимости будет обсуждаться ниже.

Процесс работы алгоритма заключается в пошаговой генерации цепочки состояний системы. На каждом МК-шаге происходит выдвижение кандидата на новую конфигурацию C'_{i+1}, отличающуюся от предыдущей одним перевернутым спином. С учетом вероятности P_{ac} либо принимается новая конфигурация ($C_{i+1} = C'_{i+1}$), либо возвращается старая $C_{i+1} = C_i$:

$$C_1 \xrightarrow{P_{ac}^{1,2}} C_2 \xrightarrow{P_{ac}^{2,3}} \ldots \xrightarrow{P_{ac}^{(n-1),n}} C_n. \tag{5.2}$$

Вероятность обновления

$$P_{ac}(E_{old} \to E_{new}) = \min\left[1, \frac{g(E_{old})}{g(E_{new})}\right] \tag{5.3}$$

зависит от распределения вероятности энергетических состояний, полученного на предыдущих итерациях алгоритма, где E_{old} и E_{new} — энергии старой и новой конфигураций, соответственно.

Гистограммы обновляются независимо от принятия или отмены новой конфигурации согласно правилу

$$\begin{cases} g(E) \to g(E) * f, \text{ где } f > 1, \\ H(E) \to H(E) + 1. \end{cases} \tag{5.4}$$

В практической реализации для достижения пределов точности, которые вычислительное устройство может поддержать, желательно использовать $\ln g(E)$ вместо $g(E)$. Тогда условие обновления $g(E)$ будет

$$\ln g(E) \to \ln g(E) + \ln(f). \tag{5.5}$$

Семплирование выполняется до момента, пока гистограмма $H(E)$ не станет равномерной, с определенной точностью. На этом шаг ВЛ заканчивается, устанавливаются новые значения $f = \sqrt{f}$ и $H(E) = 0$, $\forall E$. Таким образом алгоритм начинает новый цикл семплирования с большей точностью обновления $g(E)$. Точность равномерности определяется максимальным отклонением каждого элемента гистограммы $H(E)$ от ее среднего значения. Обычно это 80%. Согласно результатам, приведенным в работе [4], увеличение порогового значения приводит к ухудшению сходимости алгоритма. Увеличение точности результата не наблюдается. Ответ на вопрос, как влияет равномерность распределения вспомогательной гистограммы на точность вычислений, требует дополнительных исследований. Однако, можно утвердительно сказать, что увеличение равномерности распределения приведет к снижению скорости работы алгоритма.

Новое распределение $g(E)$ формируется на основе предшествующего. Модификационный фактор f является одновременно и критерием завершения, и показателем скорости вычисления. Алгоритм продолжается до тех пор, пока $f > f_{min}$, т.е. пока не достигнуто определенное минимальное значение. Учитывая, что каждый МК-шаг $g(E)$ увеличивается в f раз, путем изменения f_{min}, мы фактически варьируем точность изменения $g(E)$, за счет изменения числа полных ВЛ-циклов. Эта закономерность определяет баланс между точностью $g(E)$ и скоростью работы алгоритма.

5.2. Задача

Реализовать алгоритм Ванга-Ландау для модели Изинга. Сохранить в файл результирующую гистограмму $G(E)$. На основе гистограммы построить теплоемкость, получаемую согласно уравнению (4.2). Однако, в отличие от метрополиса, для получения термодинамических средних величин $\langle E \rangle$ и $\langle E^2 \rangle$, необходимо использовать уравнение (5.1).

Сравнить данные теплоемкости с результатами предыдущей лабораторной работы (должны совпадать). Сравнить скорость работы программы с предыдущими.

5.3. Рекомендации к выполнению

Ниже приведен листинг с кодом на c++ и python для выполнения термодинамического усреднения. В коде реализована защита от переполнения переменных при экспонировании больших значений. Для работы кода достаточно любым способом определить переменные, заданные в начале. В результате работы программа выдаст на экран 4 столбца с данным:
1 - температура; 2 - средняя энергия; 3 - теплоемкость; 4 - энтропия.

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

//****** Переменные ниже нужно указать самостоятельно ******//
#define nlamax (100000) //Сколько возможных значений будет в гистограмме g(E)
int nla;                        //сколько спинов в системе
double    iener[nlamax+1]; // массив энергий
double    g[nlamax+1];     // массив ln(g(E)).
// Например, если энергия E=5 вырождена 10 раз в системе, то iener[i]=5; g[i]=ln(10)=2.302585093;
double tmin=0.001; //минимальная и максимальная температура, в интервале которых строить
double tmax=5;     //термодинамическое усреднение. Для модели Изинга можно оставить как есть
//****** Переменные выше нужно указать самостоятельно ******//

int main(void){
    int count=nlamax;
    double x,beta;
    int ix,itmin,itmax;
    double gmax;
    double a0,ae,ae2,aecp,cv,as,as2,ddummy;
    double weight;

    itmin=tmin*1000; itmax=tmax*1000;

    for(ix=itmin; ix<=itmax; ix++){
        x=0.001*ix;
        beta=1./x;
        gmax=-1000.;
        for(ie=0; ie<count; ie++){
            if(g[ie]-beta*(iener[ie]-iener[0])>gmax){
                gmax=g[ie]-beta*(iener[ie]-iener[0]);
            }
        }

        a0=0; ae=0; ae2=0; aecp=0; as2=0;

        for(ie=0; ie<count; ie++){
            weight=exp(g[ie]-beta*(iener[ie]-iener[0])-gmax);
            a0  += weight;
            ae  += weight*iener[ie];
            ae2 += weight*iener[ie]*iener[ie];
        }

        aecp=ae;

        ae  /= (double)nla; ae2 /= (double)nla*(double)nla;
        ae  /= a0; ae2 /= a0; aecp /= a0;

        as2 = (log(a0) + gmax - beta*iener[0] + aecp*beta)/(double)nla;
        cv = beta*beta*(ae2-ae*ae)*(double)nla;

        printf("%f %e %e %e \n",x,ae,cv,as2);
    }
}
```

Полная версия кода на языке C (с функциями считывания из файла) доступна по ссылке https://pastebin.com/LZa7sA9e.

```
#//****** Переменные ниже нужно указать самостоятельно ******//
nla = 25.;                          #сколько спинов в системе
g={0:1,2:3,4:5}      # массив ln(g(E)).  Ключем является энергия, значение – ln(g(E))
# Например, если энергия E=5 вырождена 10 раз в системе, то g[5]=ln(10)=2.302585093;
tmin=0.001; #минимальная и максимальная температура, в интервале которых строить
tmax=5;      #термодинамическое усреднение. Для модели Изинга можно оставить как есть
#//****** Переменные выше нужно указать самостоятельно ******//


iener0 = next(iter(g))

for ix in range(int(tmin*1000),int(tmax*1000)):
    x=0.001*ix
    beta=1./x
    gmax=-1000.

    for ie in g:
        if g[ie]-beta*(ie-iener0)>gmax:
            gmax=g[ie]-beta*(ie-iener0)

    a0=0; ae=0; ae2=0; aecp=0; as2=0;

    for ie in g:
        weight=math.exp(g[ie]-beta*(ie-iener0)-gmax)
        a0  += weight
        ae  += weight*ie
        ae2 += weight*ie*ie

    aecp=a

    ae  /= nla; ae2 /= nla*nla;
    ae  /= a0; ae2 /= a0; aecp /= a0;

    as2 = (math.log(a0) + gmax - beta*iener0 + aecp*beta)/nla;
    cv = beta*beta*(ae2-ae*ae)*nla;

    printf("%f %e %e %e \n" % (x,ae,cv,as2));
```

Копия кода на python доступна по ссылке https://pastebin.com/P4F2zFg7.

Литература

[1] D. P Landau and K. Binder, Cambridge Univ. Press, 384 (2000).

[2] G. Brown, Kh. Odbadrakh, D. M. Nicholson et al., Phys. Rev. E **84**, 065702 (2011).

[3] M. S. Kalyan, R. Bharath, V. S. S. Sastry et al., J. Stat. Phys. **163**, 197 (2016).

[4] F. Wang and D. P. Landau, Phys. Rev. Lett. **86**, 2050 (2001)

[5] D. P. Landau, S. H. Tsai and M. Exler, Am. J. Phys. **72**, 1294 (2004).

[6] И. А. Силантьева, П. Н. Воронцов-Вельяминов, Вычислительные методы и программирование **12**, 397 (2011).

[7] Л. Н. Щур., Механика, управление и информатика, Институт космических исследований Российской академии наук, Москва (2014), с. 160.