1D Harmonic Oscillator

一维谐振子的蒙特卡罗模拟

一、导言

在欧几里得量子场论中,物理可观量O的期望值形式上给出为:

$$\langle O \rangle = \frac{\int DxO(x)e^{-S(x)/\hbar}}{Z}$$

这里$Z = \int Dxe^{-S(x)/\hbar}$,$Dx$表示路径积分。通常除了像一维谐振子这样的简单物理问题,这个表达式很难求出解析结果。在本讲义中,我们的目标不是解析求解一维谐振子,而是引入基于格点的蒙特卡罗模拟方法,这是解决格子QCD中使用的数值方法的基础。作为一个教学性的练习,我们实现蒙特卡罗方法来求解一维谐振子问题。

根据方程(1),一个可观测量的期望值可以用该可观测量在N个样本配置$\{x^{(0)}, x^{(1)}, \cdots, x^{(N-1)}\}$上的平均值来逼近,这里样本配置的概率正比于$e^{-S(x^{(n)})/\hbar}$。当取足够多的配置时,平均值

$$\langle O \rangle \approx \frac{1}{N} \sum_{n=0}^{N-1} O(x^{(n)})$$

应该收敛到准确结果。在第二部分,我们解释蒙特卡罗方法生成所需概率分布的配置的过程。在第三部分,我们研究使用蒙特卡罗方法求解一维谐振子问题,给出基态和第一激发态的数值结果。

二、蒙特卡罗方法

A. 重要性采样

在路径积分中,不同的配置x由于作用$S(x)$的不同会给它们赋予不同的重要性。因此在求和所有配置时,包含较大权重因子$e^{-S(x^{(n)})/\hbar}$的配置比权重较小的配置更重要。也就是说,作用较小的配置比作用较大的配置更重要。重要性采样的思想是用比较少的配置子集来逼近完整求和,这里配分的概率分布正比于权重因子$e^{-S(x^{(n)})/\hbar}$。

B. 马尔可夫链

如何找到一组配置$\{x^{(0)}, x^{(1)}, \cdots, x^{(N-1)}\}$,其概率分布正比于权重因子$e^{-S(x^{(n)})/\hbar}$呢?思路是从任意一个起始配置出发,构造一个随机序列的配置,最终这个序列会趋向于平衡分布$P(x) \propto e^{-S(x^{(n)})/\hbar}$。这就是所谓的马尔可夫链:

$$x^{(0)} \to x^{(1)} \to x^{(2)} \to \cdots$$

在这个链中,每个配置$x^{(n)}$是从前一个配置$x^{(n-1)}$生成的。一个马尔可夫链由转移概率$T(x'|x)$描述,这是从配置x转移到配置x'的概率。这个概率只依赖于x'和x。$T(x'|x)$满足:

$$0 \leq T(x'|x) \leq 1, \quad \sum_{x'} T(x'|x) = 1$$

第一个关系限定了一个概率的范围。第二个关系意味着从配置x跳转到任何一个目标配置x'的概率等于1。

当一个马尔可夫链达到平衡时,跳转到配置x'的概率应该等于跳出这个配置的概率。平衡条件为:

$$\sum_{x} T(x'|x)P(x) = \sum_{x} T(x|x')P(x')$$

左边是从所有起始配置x跳转到x'的转移概率$T(x'|x)$,加权由系统在配置x的概率$P(x)$。右边给出从x'跳转到所有可能的终点配置x的概率。概率$P(x)$是我们要得到的平衡分布。这条关系称为平衡条件。可以证明,从任意起始配置x_0出发,反复应用满足平衡条件的转移概率$T(x'|x)$可以最终达到平衡分布$P(x)$。此外,为了得到正确的结果,马尔可夫链必须能够访问所有配置。如果对所有x和x',转移概率$T(x'|x)$都是严格正的,则可以最终达到每一个配置。

在实际计算中,我们在足够多次更新之后开始测量可观测量,以确保考虑的配置已经达到平衡分布。为了减少配置之间的自相关效应,我们通常在使用一个配置进行测量之前跳过一定数量的配置。

C. 美特罗波利斯算法

平衡方程(5)的一个充分条件是逐项要求等式成立:

$$T(x'|x)P(x) = T(x|x')P(x')$$

这称为详细平衡条件。详细平衡条件是一个充分但非必要条件。大多数蒙特卡罗算法使用详细平衡条件。最著名的算法是美特罗波利斯算法,下面我们解释这个算法。

从任意起始配置x出发,美特罗波利斯算法按以下步骤推进马尔可夫链:

1. 根据某个选择概率$T_0(x'|x)$选择一个候选配置x'作为试探更新,这里$T_0(x'|x)$对x和x'对称,即$T_0(x'|x) = T_0(x|x')$。
2. 计算候选新配置x'的作用$S(x')$。如果作用降低了,则接受x'作为新配置。否则以概率$e^{-(S(x')-S(x))/\hbar}$条件性接受。
3. 从头开始重复这些步骤。

可以简单证明美特罗波利斯算法满足详细平衡条件。

三、应用蒙特卡罗方法求解一维谐振子

在这一部分,我们给出使用蒙特卡罗模拟在格子上求解一维谐振子的详细过程。

A. 生成配置

引入一个时间格子,格子间隔$a = T/N$,这里T是时间方向的总长度,N是一个很大的整数。如图1所示。设x_n表示第n个时间片上的振子位置,一维谐振子的作用量离散化为:

$$S(x)=\sum_{n=0}^{N-1}\left(\frac{m(x_{n+1}-x_n)^2}{2a}+\frac{am\omega^2(x_{n+1}^2+x_n^2)}{4}\right)$$

采用周期性边界条件,$x_0=x_N$。

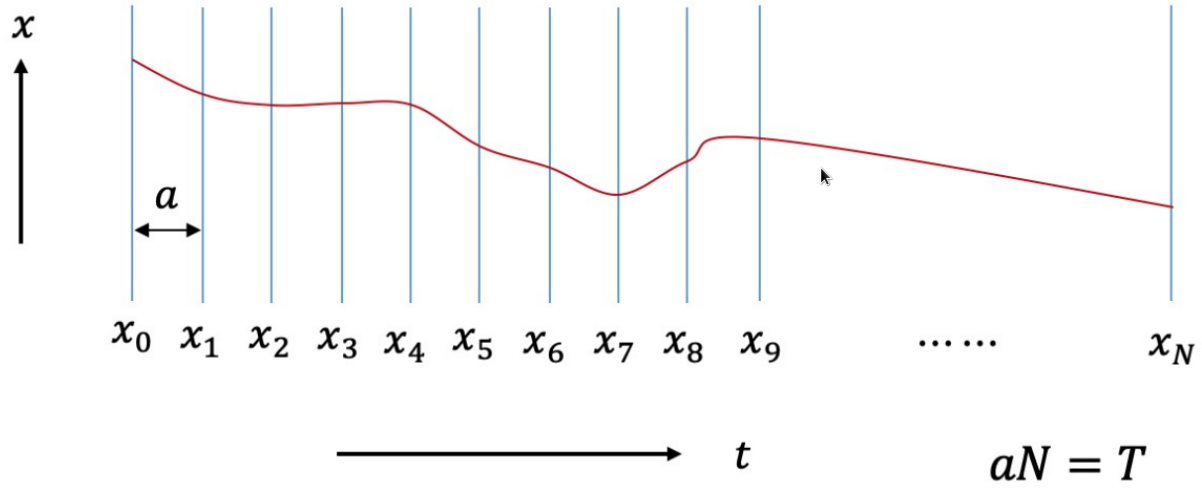

FIG. 1. Illustration of 1D harmonic oscillator on a time lattice.

A. Generating configurations

为了计算方便,我们定义无量纲化位置变量$u=x/b$,其中$b=\sqrt{\hbar/m\omega}$是谐振子的典型长度尺度。用u重写作用量:

$$S(u)/\hbar=\sum_{n=0}^{N-1}\left(\frac{(u_{n+1}-u_n)^2}{2a\omega}+\frac{a\omega(u_{n+1}^2+u_n^2)}{4}\right)$$

谐振子的经典周期是$2\pi/\omega$。在实际模拟中,格子间隔a应该远小于周期,而总长度T应该远大于周期。即$a\ll 2\pi/\omega\ll T$。为简单起见,在以下内容中我们取$\omega=1$。记住a和N的取值范围应满足$a\ll 2\pi\ll aN$,我们使用美特罗波利斯算法生成配置。

1. 选择一个任意起始配置$u=\{u_0,u_1,u_2,\cdots,u_{N-1}\}$。可以取所有$u_n=0$,这称为冷启动;也可以取$u_n$为一些随机数,称为热启动。
2. 选择候选配置$u_n'=u_n+\delta u\times r$,这里$r$是一个在[-1,1]间均匀分布的随机数,$\delta u$是一个经验选择的参数。
3. 如果新配分u'的作用$S(u')$比旧配分u的作用更小,则接受u'作为新配分。否则以概率$e^{-(S(u')-S(u))/\hbar}$条件性接受。如果u'被拒绝,则在马尔可夫链中再次考虑u。
4. 从第2步开始重复。

模拟的输入参数有:

- a:格子间隔。a的值应远小于2π。
- N或T:时间的总长度。aN或T的值应远大于2π。
- δu:算法改变位置u的值的步长。这个参数影响接受率。
- N_{conf}:算法总共生成的配置数量。
- N_{dump}:前N_{dump}个配置会被保留。这意味着马尔可夫链在N_{dump}步后达到平衡。
- N_{skip}:在用一个配置进行测量之前跳过的配置数。这个数值应足够大以避免自相关。

注意:用于测量的配置总数是$(N_{conf} - N_{dump})/N_{skip}$,只需要将这些配置写到磁盘。

下面给出生成配置的示例代码:

```
# 配置生成代码

import numpy as np

N = 100         # 时间片数量
a = 0.1         # 格子间隔
delta_u = 0.1   # 更新步长
N_conf = 10000 # 总配置数
N_dump = 1000   # 丢弃前Ndump个配置
N_skip = 10     # 每Nskip个配置进行一次测量

u = np.zeros(N) # 冷启动

for i in range(N_conf):

    # 选择新配置
    for j in range(N):
        r = np.random.uniform(-1,1)
        u_new = u[j] + delta_u * r

    # 计算新旧配置对应的作用
    S_new = action(u_new)
    S_old = action(u)

    # 美特罗波利斯試探
    if S_new < S_old:
        accept = True
    else:
        p = np.exp(-(S_new - S_old))
        if np.random.rand() < p:
            accept = True
        else:
```

```
            accept = False

    if accept:
        u = u_new

    # 进行测量
    if i >= N_dump and (i-N_dump) % N_skip == 0:
        measure_observables(u)
```

```
/* compute the weight factor that involves three points x0, x1, x2. When updating x1, the change of
        the action only involves the neighbouring points x0 and x2.  */
double weight(double x0, double x1, double x2, double a){
    double S=0.0;
    S=(x1-x0)*(x1-x0)/(2.0*a) + a*(x0*x0+x1*x1)/4.0 + (x2-x1)*(x2-x1)/(2.0*a) + a*(x2*x2+x1*x1)/4.0;
    return exp(-1.0*S);
}

int main(){
    int Nt,Nconf;
    double a; // lattice spacing
    double delta;  //step size
    char output_path[200]; // output file path

// read in input
    double* x1 = new double[Nt];
    double* x2 = new double[Nt];
    double* x = new double[Nt*Nconf];

//generate random number
        std::default_random_engine seed;
        seed.seed(time(NULL));
        std::uniform_real_distribution<double> ran1(-1.0,1.0); // random number between [-1.0,1.0]
        std::uniform_real_distribution<double> ran2(0,1.0); // random number between [0,1.0]

//initialize the starting configuration to be 0.0
    for(int i=0; i<Nt; i++)
        *(x1+i) = *(x2+i) = 0.0;  // cold start

// update the configuration with metropolis algorithm
    std::vector<int> t_iter;
    for(int i=0; i<Nt; i++) t_iter.push_back(i);
    for(int n=0; n<Nconf; n++){
        std::random_shuffle (t_iter.begin(), t_iter.end()); // shuffle the order of updating
        for(int t=0; t<Nt; t++){
            int i=t_iter[t];
// chose a candidate configuration x2 by updating the ith points of x2
            *(x2+i) = *(x1+i) + delta*ran1(seed);
            if(ran2(seed) < weight(*(x2+(i-1+Nt)%Nt),*(x2+i),*(x2+(i+1)%Nt),aomega)/
                                    weight(*(x1+(i-1+Nt)%Nt),*(x1+i),*(x1+(i+1)%Nt), aomega))
                *(x1+i)=*(x2+i);  // accept the candidate
            else
                *(x2+i)=*(x1+i);  // reject the candidate
        }

        for(int i=0; i<Nt; i++)
            *(x + n*Nt + i) = *(x2 + i);
    }

// write out the configurations x to outfile
}
```

B. 测量基态能量

有了上一部分生成的配置,直接计算u^2的期望值即可:

$$\langle u^2 \rangle = \frac{1}{M} \sum_{m=0}^{M} u^2(m)$$

这里M是配置数量,$u^2(m)$是配分$u^{(m)}$上的所有时间片u^2的平均值。$\langle u^2 \rangle$的误差由标准误方程给出:

$$\sigma_{u^2} = \sqrt{\frac{\sum_m (u^2(m) - \langle u^2 \rangle)^2}{M(M-1)}}$$

$\langle u^2 \rangle$的精确理论值由公式给出:

$$\langle u^2 \rangle_{theory} = \frac{1}{2} \sqrt{1 + \frac{a^2}{4}} \frac{1 + R^N}{1 - R^N}, \quad R = 1 + \frac{a^2}{4} - a\sqrt{1 + \frac{a^2}{4}}$$

在图2中,我们给出数值结果和理论值的比较。蓝点和误差棒是数值结果,红点是理论值。红线是连续极限$a \to 0$,体系大小极限$N \to \infty$下的理论值0.5。在左图中,我们在固定$N = 500$的条件下比较不同的格子间隔a下的结果。可以看出当a很小时,数值结果和理论值有较大偏差,这是因为总长度T不够大。当a较大时,数值结果也偏离了连续极限下的理论值0.5。在右图中,我们在固定$a = 0.01$的条件下比较不同N的值。可以清楚看到N必须取足够大才能得到正确结果。

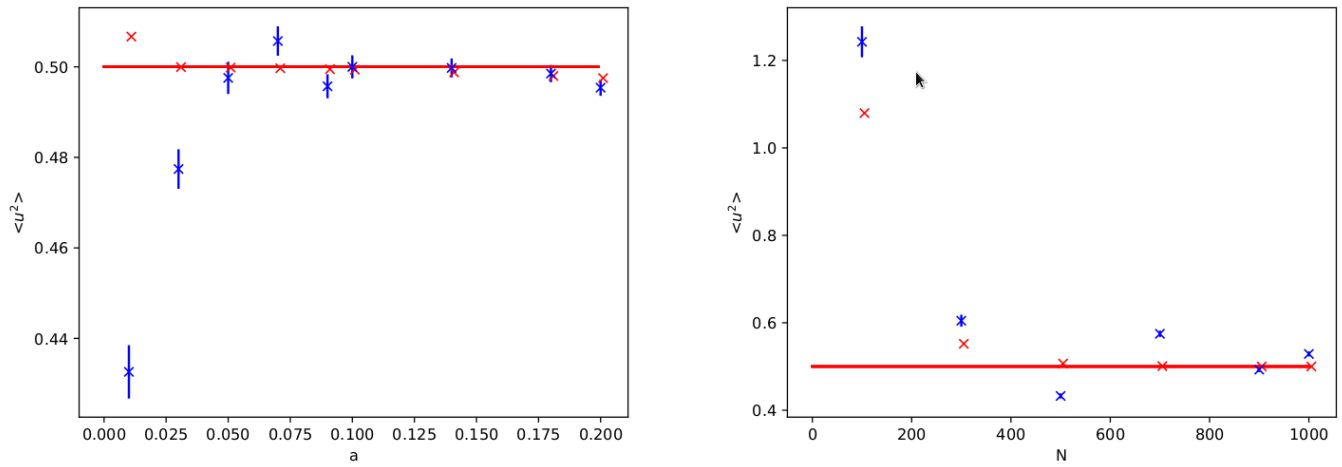

图2: u^2的期望值。蓝点和误差棒是数值结果,红点是理论值。左图:在固定$N = 500$时比较不同a。右图:在固定$a = 0.01$时比较不同N。

C. 测量第一激发态能量

为了得到激发态能量,我们计算相关函数:

$$C(t) = \langle x(t)x(0) \rangle = \frac{1}{Z} \int Dx\, x(t)x(0)e^{-S(x)}$$

通过对大t范围的相关函数进行指数拟合可以得到第一激发态的能量。在图3中画出相关函数随t的关系,相关函数是在$a=0.1, N=500$条件下计算的。在计算相关函数时,我们不仅在配分上平均,而且在时间片上平均。实际计算的是:

$$\langle x(t)x(0)\rangle = \frac{1}{M}\sum_{m}\left(\frac{1}{N}\sum_{t_0=0}^{N-1} x(t+t_0)^{(m)}x(t_0)^{(m)}\right)$$

这里M是配分数量。拟合采用最小χ^2方法。拟合函数是$f(t;\Delta E)=e^{-\Delta E t}$。通过最小化$\chi^2$得到$\Delta E$的值。$\Delta E$的误差可以用Jackknife或Bootstrap方法估计。图3中的拟合值是$\Delta E = 0.995 \pm 0.009$,与理论值1非常吻合。

strong text

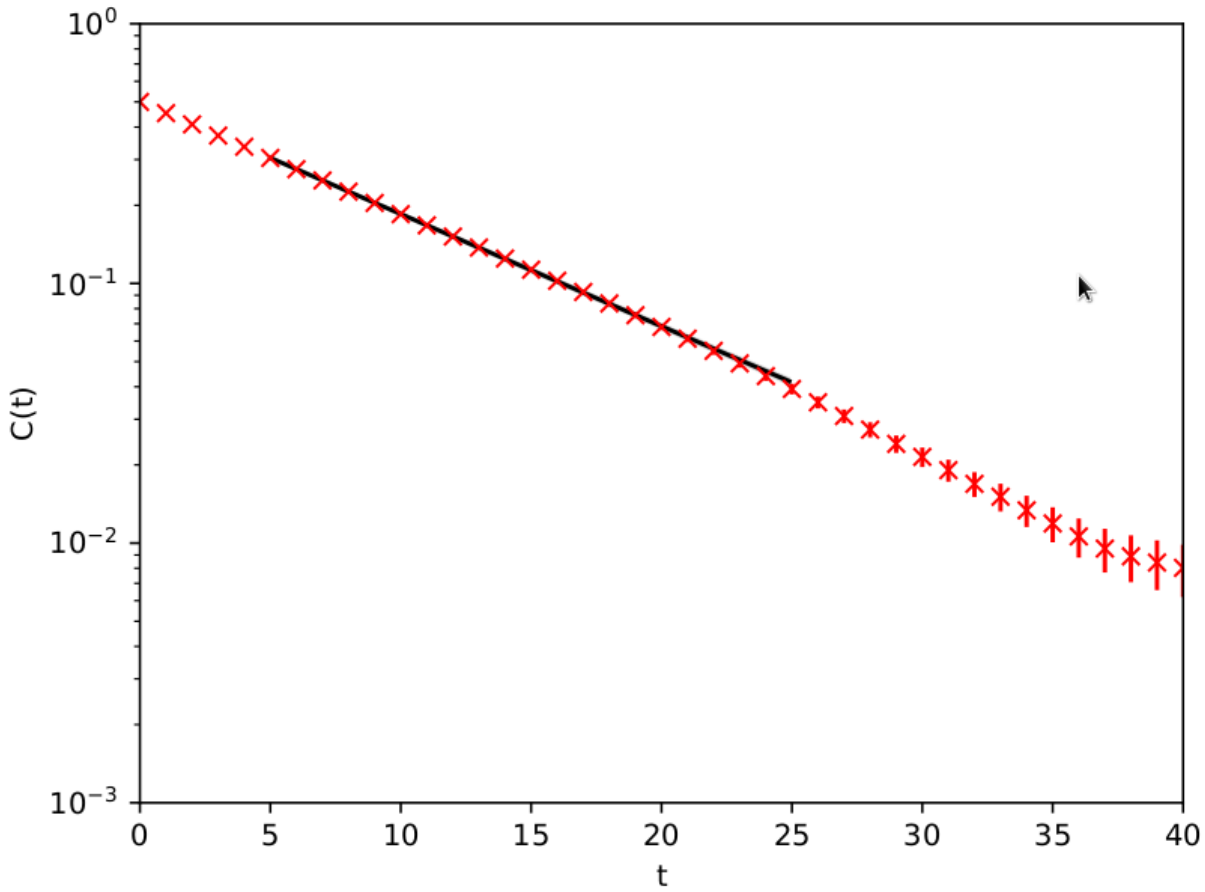

图3: 相关函数随时间的变化,黑线是指数拟合。

四、总结

本讲义总结如下:

1. 引入了蒙特卡罗方法的基本思想 - 重要性采样和马尔可夫链。解释了平衡条件、详细平衡条件的物理意义。
2. 介绍了著名的蒙特卡罗算法 - 美特罗波利斯算法。给出了算法的具体实现步骤。
3. 以一维谐振子为例,详细阐述了如何应用蒙特卡罗方法在格子上进行量子模拟。

- 生成配置的过程,给出示例代码。

- 测量基态能量的方法。比较数值结果与解析结果。
- 计算相关函数的方法。提取第一激发态的能量。

4. 通过这个简单的练习,让读者对蒙特卡罗方法有直观的了解,为学习更复杂的格子规范场论奠定基础。

本讲义内容全面系统地介绍了蒙特卡罗方法及其在一维谐振子问题中的应用。既包含了理论方法的细致阐释,也给出了编程实现的具体示例,是学习蒙特卡罗量子蒙特卡罗量子模拟方法的良好教材。