量子物理计算方法选讲实验报告

李昊恩　2021010312

Exact Diagonalization, Task 3

目录

1 引言

本实验的任务是计算环状横场伊辛模型的色散关系。该模型是具有 N 个位点的环状格点模型，如图所示。其哈密顿量表达式为：

$$H = -J\sum_{i=1}^{N}(g\sigma_i^x + \sigma_i^z\sigma_{i+1}^z), \quad J, g \geq 0; \tag{1}$$

其中 σ_i^x 和 σ_i^z 分别表示 i 位点的 Pauli 算子，其定义为：

$$\sigma^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{2}$$

该问题中，量子态所在的 Hilbert 空间维数是 2^N。为了尽可能地缩减问题的规模，我们考虑该体系具有的平移对称性。我们定义平移算子 T，它的作用是将格点向某个方向平移一格。例如，对于 $N=4$ 的平移算子，它在组态 $|0011\rangle$ 上的作用为：

$$T|0011\rangle = |0110\rangle, \quad T^2|0011\rangle = |1100\rangle \quad T^3|0011\rangle = |1001\rangle, \quad T^4|0011\rangle = |0011\rangle. \tag{3}$$

显然，根据式 (1) 可知，将求和指标进行平移（轮换）并不会改变哈密顿量，也即：

$$[T, H] = 0. \tag{4}$$

因此，T 和 H 可同时对角化。若我们找出 T 的本征态，并构建 T-不变子空间的基，则 H 在这组基下为分块对角矩阵，从而降低了问题的规模。对于平移算符 T，由于具有周期性，我们可以

用“离散 Fourier 变换”找出其特征向量。具体来说，我们可以定义向平移算子 T 的特征空间的投影算子：

$$P_k := \frac{1}{N}\sum_{j=0}^{N-1} e^{i\frac{2\pi}{N}kj}T^j, \quad \text{where} \quad k=0,1,\cdots,N-1, \tag{5}$$

我们容易通过计算直接验证 P_k 的确是一个投影算子，也就是 $P_k^\dagger = P_k$，$P_k^2 = P_k$：

$$P_k^\dagger = \frac{1}{N}\sum_{j=0}^{N-1} e^{-i\frac{2\pi}{N}kj}(T^j)^\dagger = \frac{1}{N}\sum_{j=0}^{N-1} e^{-i\frac{2\pi}{N}kj}T^{-j} = \frac{1}{N}\sum_{j'=0}^{N-1} e^{i\frac{2\pi}{N}kj'}T^{j'} = P_k; \tag{6}$$

$$P_k^2 = P_k^\dagger P_k = \frac{1}{N^2}\sum_{\ell,j=0}^{N-1} e^{i\frac{2\pi}{N}(\ell-j)}T^{(\ell-j)} = \frac{1}{N}\sum_{j'=0}^{N-1} e^{i\frac{2\pi}{N}kj'}T^{j'} = P_k. \tag{7}$$

为了不重复、不遗漏地选出新的一组基中的所有向量，我们不妨规定一些“representative states”，具体来说，我们选出能够通过平移算子得到的每一族组态（即组态族 $\{T^0\left|r\right\rangle, T^1\left|r\right\rangle, \cdots, T^{N-1}\left|r\right\rangle\}$）中，其二进制数码对应的整数最小的组态。但是，对于有些 $\left|r\right\rangle$ 和 k，$P_k\left|r\right\rangle$ 可能会得零（事实上，只有当 $kR \equiv 0 \mod N$ 时，$P_k\left|r\right\rangle$ 才不得零）。剔除这些组态之后，剩下的向量 $\{P_k\left|r\right\rangle\}_{0\le k\le 4, \left|r\right\rangle \text{ 具有代表性}}$，构成了整个 Hilbert 空间的一族基，而且，H 在该组基下的表示矩阵为分块对角。这里，每个 k-sector 对应的能谱 $\sigma(K)$ 也被称为“色散关系”。

下面我们讨论具体如何计算矩阵元。我们记 $P_k\left|r\right\rangle =: \left|r_k\right\rangle$，根据 P_k 是投影算子可知：

$$\left|r_k\right\rangle = \frac{P_k\left|r\right\rangle}{\sqrt{\left\langle r\middle|P_k^\dagger P_k\middle|r\right\rangle}} = \frac{P_k\left|r\right\rangle}{\sqrt{\left\langle r\middle|P_k\middle|r\right\rangle}}. \tag{8}$$

注意我们已经剔除了使得 $P_k\left|r\right\rangle = 0$ 的那些组态 r。此外，由于 $[T,H]=0$，所以 $[P_k,H]=0$，于是：

$$\left\langle r_k'\middle|H\middle|r_k\right\rangle = \frac{\left\langle r'\middle|P_k^\dagger H P_k\middle|r\right\rangle}{\sqrt{\left\langle r'\middle|P_k\middle|r'\right\rangle\left\langle r\middle|P_k\middle|r\right\rangle}} = \frac{\left\langle r'\middle|P_k H\middle|r\right\rangle}{\sqrt{\left\langle r'\middle|P_k\middle|r'\right\rangle\left\langle r\middle|P_k\middle|r\right\rangle}}. \tag{9}$$

由上述推导可知，我们只需要得到哈密顿量 H 作用在组态 $\left|r\right\rangle$，以及每个 P_k 作用在组态上所得的结果即可。

2 代码实现

我们同样先定义对整数的 0-1 串形式的操作函数如下：

```python
#flip nth bit
def FlipBit(i,n):
    return i^(1<<n)
#read nth bit
def ReadBit(i,n):
    return (i&(1<<n))>>n
#pick up n bits from kth bit
def PickBit(i,k,n):
```

```
    return (i&((2**n-1)<<k))>>k
#circuilar bit shift right
def RotLBit(i,L):
    return (PickBit(i,0,L-1)<<1)+(i>>(L-1))
#define a function to get all the configurations that can be obtained by translation of i
def transl_bit_patterns(i):
    transl_bits = [i]
    for _ in range(Ns-1):
        i = RotLBit(i,Ns)
        transl_bits.append(i)
    return transl_bits
```

随后，我们可以定义一个索引表，用于查询每个组态：1）是否为代表性组态，放在列表的第一列，其中 1 代表是，0 代表否；2）对应的代表性组态的整数（十进制）表示为何；3）从代表性组态出发，需要作用多少次平移算符才能得到该组态。代码实现如下：

```
#create the check list for whether it is a representative configuration and its corresponding representative config \r>
Check = np.zeros([2**Ns,3])
all_configs = []
for k in range(2**Ns):
    if k in all_configs:
        Check[k,1] = min(transl_bit_patterns(k))
        Check[k,2] = transl_bit_patterns(min(transl_bit_patterns(k))).index(k)
    else:
        all_configs += transl_bit_patterns(k)
        Check[k,0] = 1
        Check[k,1] = k
```

随后，我们可以用上述索引表生成所有的代表性组态，以及每个 k 对应的基向量的全体（剔除那些得零的向量）：

```
#create all the representative configurations using the checklist
rep_configs = []
for i in range(2**Ns):
    if Check[i,0] == 1:
        rep_configs.append(i)

#create the basis for every k, allow mod(kR,Ns) = 0, where R is the minimal translating times needed to obtain the original state
all_basis = []
for k in range(Ns):
    k_basis = []
```

```
    for r in rep_configs:
        if k*(Ns/transl_bit_patterns(r).count(r)) % Ns == 0:
            k_basis.append((k,r))
    all_basis.append(k_basis)
```

根据第1部分的相应推导，我们需要计算出 P_k 以及哈密顿量 H 作用在每个组态 $|f\rangle$ 后所得的结果，我们将结果记录为一个列表，每个列表由若干二元组组成，每个二元组的第一个元素为组态前的系数，第二个元素则为组态的整数（十进制）表示。随后，根据组态之间的正交归一性，我们可以计算以上所得结果对应量子态之间的内积，进而得到我们需要的矩阵元 $\langle r|P_k|r\rangle$ 和 $\langle r'|P_k H|r\rangle$。代码实现为：

```
#applying Hamiltonian to the representative configurations |r>
def apply_Hamil(r):
    Ham_r = []
    couplings = 0
    for i in range(Ns):
        Ham_r.append((-J*g,FlipBit(r,i)))
        couplings += (-1)**(ReadBit(r,i%Ns)+ReadBit(r,(i+1)%(Ns)))
    Ham_r.append((-J*couplings,r))
    return Ham_r

#applying Pk to the representative configurations |r>
def apply_proj(k,r):
    Proj_r = []
    for j in range(Ns):
        Proj_r.append((np.exp(1j*2*math.pi*k*j/Ns),transl_bit_patterns(r)[j]))
    return Proj_r

#calculating <s|P_k H|r>
def mel_Pk_H(k,r,s):
    mel = 0
    for entries in apply_Hamil(r):
        for m in range(len(apply_proj(k,s))):
            if apply_proj(k,s)[m][1] == entries[1]:
                mel += apply_proj(k,s)[m][0]*entries[0]
    return(mel)

#calculating <r|P_k|r>
def ev_Pk_r(k,r):
    ev = 0
    for entries in apply_proj(k,r):
        if entries[1] == r:
            ev += entries[0]
```

```
    return(ev)
```

最后，我们对每个 k-sector 生成矩阵并进行对角化，即可得到每个 k-sector 的能谱，即色散关系：

```
#generate each k-sector using the function defined above and diagonalize each block
for i in range(Ns):
    H_matrix = np.zeros([len(all_basis[i]),len(all_basis[i])])+0j
    for row_idx in range(len(all_basis[i])):
        for col_idx in range(len(all_basis[i])):
            r = all_basis[i][col_idx][1]
            s = all_basis[i][row_idx][1]
            mel = mel_Pk_H(i,r,s)/np.sqrt(np.real((ev_Pk_r(i,s))*(ev_Pk_r(i,r))))
            H_matrix[row_idx,col_idx] = mel
    evals,evecs = np.linalg.eigh(H_matrix)
    print("(1)_Momentum_Sector_ki={}:{}".format(i,np.sort(evals)))
```

3 结果

我们对 $N=8, J=1, g=1.4$ 以及 $N=8, J=1, g=0.4$ 分别运行程序可得：

(1) $J=1, g=1.4$;

```
(1)_Momentum_Sector_ki=0:[-1.26962251e+01 -1.18661078e+01 -1.02528265e+01
    -7.90610579e+00
 -7.19933670e+00 -4.98424762e+00 -4.75593806e+00 -4.66477333e+00
 -3.27551373e+00 -2.97575442e+00 -2.80000000e+00 -2.80000000e+00
 -2.22137469e+00 -1.44093011e+00 -1.44093011e+00 -1.02424558e+00
 -8.32115082e-01 -1.31493976e-15 -5.09943552e-16  5.91697979e-17
  5.27120947e-16  8.32115082e-01  9.84247622e-01  2.22137469e+00
  2.80000000e+00  2.80000000e+00  3.27551373e+00  3.90610579e+00
  4.66477333e+00  4.75593806e+00  5.44093011e+00  5.44093011e+00
  7.19933670e+00  7.86610784e+00  1.02528265e+01  1.26962251e+01]
(1)_Momentum_Sector_ki=1:[-1.06861068e+01 -8.72608159e+00 -6.44517671e+00
    -5.93205502e+00
 -3.98000102e+00 -3.97014353e+00 -3.80424660e+00 -3.48865637e+00
 -2.62093113e+00 -2.44517671e+00 -1.79575340e+00 -1.52674488e+00
 -6.94629802e-01 -1.99989771e-02 -6.09513127e-16  9.70370695e-16
  1.99989771e-02  6.94629802e-01  1.33907092e+00  1.52674488e+00
  2.44517671e+00  3.48865637e+00  3.97014353e+00  3.98000102e+00
  4.26092908e+00  5.08610681e+00  5.93205502e+00  6.44517671e+00
  8.22093113e+00  8.72608159e+00]
(1)_Momentum_Sector_ki=2:[-9.22517773e+00 -8.72608159e+00 -7.45879990e+00
    -5.44093011e+00
```

```
 -5.26517568e+00 -5.23742522e+00 -3.97014353e+00 -3.62517773e+00
 -2.80000000e+00 -2.80000000e+00 -2.79402657e+00 -1.96191149e+00
 -1.52674488e+00 -1.44093011e+00 -6.94629802e-01 -3.34824317e-01
  3.41546956e-16  3.34824317e-01  6.94629802e-01  1.44093011e+00
  1.52674488e+00  1.96191149e+00  2.79402657e+00  2.80000000e+00
  2.80000000e+00  3.62517773e+00  3.97014353e+00  5.23742522e+00
  5.26517568e+00  5.44093011e+00  7.45879990e+00  8.72608159e+00
  9.22517773e+00]
(1)_Momentum_Sector_ki=3:[-8.22093113e+00 -7.45879990e+00 -6.76417010e+00
    -6.44517671e+00
 -5.23742522e+00 -5.08610681e+00 -4.26092908e+00 -3.98000102e+00
 -2.79402657e+00 -2.44517671e+00 -1.96191149e+00 -1.33907092e+00
 -1.26728169e+00 -1.99989771e-02 -3.94739579e-16  8.39854935e-16
  1.99989771e-02  1.26728169e+00  1.79575340e+00  1.96191149e+00
  2.44517671e+00  2.62093113e+00  2.79402657e+00  3.80424660e+00
  3.98000102e+00  5.23742522e+00  6.44517671e+00  6.76417010e+00
  7.45879990e+00  1.06861068e+01]
(1)_Momentum_Sector_ki=4:[-7.86610784e+00 -6.76417010e+00 -6.76417010e+00
    -5.93205502e+00
 -5.93205502e+00 -5.44093011e+00 -5.44093011e+00 -3.90610579e+00
 -3.48865637e+00 -3.48865637e+00 -2.80000000e+00 -2.80000000e+00
 -1.26728169e+00 -1.26728169e+00 -9.84247622e-01 -1.33319677e-16
  4.85996381e-16  1.02424558e+00  1.26728169e+00  1.26728169e+00
  1.44093011e+00  1.44093011e+00  2.80000000e+00  2.80000000e+00
  2.97575442e+00  3.48865637e+00  3.48865637e+00  4.98424762e+00
  5.93205502e+00  5.93205502e+00  6.76417010e+00  6.76417010e+00
  7.90610579e+00  1.18661078e+01]
(1)_Momentum_Sector_ki=5:[-8.22093113e+00 -7.45879990e+00 -6.76417010e+00
    -6.44517671e+00
 -5.23742522e+00 -5.08610681e+00 -4.26092908e+00 -3.98000102e+00
 -2.79402657e+00 -2.44517671e+00 -1.96191149e+00 -1.33907092e+00
 -1.26728169e+00 -1.99989771e-02 -1.53839540e-15  1.55693489e-15
  1.99989771e-02  1.26728169e+00  1.79575340e+00  1.96191149e+00
  2.44517671e+00  2.62093113e+00  2.79402657e+00  3.80424660e+00
  3.98000102e+00  5.23742522e+00  6.44517671e+00  6.76417010e+00
  7.45879990e+00  1.06861068e+01]
(1)_Momentum_Sector_ki=6:[-9.22517773e+00 -8.72608159e+00 -7.45879990e+00
    -5.44093011e+00
 -5.26517568e+00 -5.23742522e+00 -3.97014353e+00 -3.62517773e+00
 -2.80000000e+00 -2.80000000e+00 -2.79402657e+00 -1.96191149e+00
 -1.52674488e+00 -1.44093011e+00 -6.94629802e-01 -3.34824317e-01
  1.95234614e-16  3.34824317e-01  6.94629802e-01  1.44093011e+00
```

```
  1.52674488e+00  1.96191149e+00  2.79402657e+00  2.80000000e+00
  2.80000000e+00  3.62517773e+00  3.97014353e+00  5.23742522e+00
  5.26517568e+00  5.44093011e+00  7.45879990e+00  8.72608159e+00
  9.22517773e+00]
(1)_Momentum_Sector_ki=7:[-1.06861068e+01 -8.72608159e+00 -6.44517671e+00 -5.93205502e+00
 -3.98000102e+00 -3.97014353e+00 -3.80424660e+00 -3.48865637e+00
 -2.62093113e+00 -2.44517671e+00 -1.79575340e+00 -1.52674488e+00
 -6.94629802e-01 -1.99989771e-02 -6.04040053e-15  6.50332847e-15
  1.99989771e-02  6.94629802e-01  1.33907092e+00  1.52674488e+00
  2.44517671e+00  3.48865637e+00  3.97014353e+00  3.98000102e+00
  4.26092908e+00  5.08610681e+00  5.93205502e+00  6.44517671e+00
  8.22093113e+00  8.72608159e+00]
```

(2) 对 $J = 1, g = 0.4$;

```
(1)_Momentum_Sector_ki=0:[-8.32346445e+00 -8.32320887e+00 -5.72840338e+00 -5.23953684e+00
 -4.62729724e+00 -4.01507702e+00 -3.48008258e+00 -3.06859501e+00
 -2.81114570e+00 -2.03223617e+00 -9.31404990e-01 -8.85021510e-01
 -8.00000000e-01 -8.00000000e-01 -2.16084628e-01 -1.54065923e-01
 -1.54065923e-01 -5.74868673e-16 -2.90876444e-16 -6.80028190e-17
  2.53736334e-16  1.50770223e-02  2.16084628e-01  8.00000000e-01
  8.00000000e-01  8.85021510e-01  1.23953684e+00  2.03223617e+00
  2.81114570e+00  3.48008258e+00  4.15406592e+00  4.15406592e+00
  4.32320887e+00  4.62729724e+00  5.72840338e+00  8.32346445e+00]
(1)_Momentum_Sector_ki=1:[-5.58137285e+00 -5.17785031e+00 -4.62730693e+00 -4.05368991e+00
 -3.54183602e+00 -3.14561414e+00 -2.89590194e+00 -1.45862884e+00
 -1.27324101e+00 -6.27306929e-01 -5.50553069e-01 -4.58163984e-01
 -3.34468441e-01 -3.26758994e-01 -5.93872565e-16  5.69445835e-16
  1.87770093e-01  3.34468441e-01  4.58163984e-01  5.50553069e-01
  6.27306929e-01  1.41222991e+00  1.45862884e+00  3.14561414e+00
  3.54183602e+00  3.98137285e+00  4.05368991e+00  4.49590194e+00
  4.62730693e+00  5.17785031e+00]
(1)_Momentum_Sector_ki=2:[-5.17785031e+00 -4.96914295e+00 -4.60424298e+00 -4.15406592e+00
 -3.71922147e+00 -3.36914295e+00 -3.14561414e+00 -1.88547091e+00
 -1.12416040e+00 -9.08075769e-01 -8.00000000e-01 -8.00000000e-01
 -5.50553069e-01 -3.34468441e-01 -2.85470913e-01 -1.54065923e-01
 -1.15299214e-16  1.54065923e-01  2.85470913e-01  3.34468441e-01
  5.50553069e-01  8.00000000e-01  8.00000000e-01  9.08075769e-01
  1.12416040e+00  1.88547091e+00  3.14561414e+00  3.36914295e+00
  3.71922147e+00  4.15406592e+00  4.60424298e+00  4.96914295e+00
```

```
 5.17785031e+00]
(1)_Momentum_Sector_ki=3:[-4.62730693e+00 -4.60424298e+00 -4.49590194e+00 -4.26977454e+00
 -3.98137285e+00 -3.71922147e+00 -3.54183602e+00 -1.41222991e+00
 -1.12416040e+00 -9.08075769e-01 -6.27306929e-01 -5.73607328e-01
 -4.58163984e-01 -1.87770093e-01 -3.35915893e-16  3.27091788e-16
  3.26758994e-01  4.58163984e-01  5.73607328e-01  6.27306929e-01
  9.08075769e-01  1.12416040e+00  1.27324101e+00  2.89590194e+00
  3.54183602e+00  3.71922147e+00  4.26977454e+00  4.60424298e+00
  4.62730693e+00  5.58137285e+00]
(1)_Momentum_Sector_ki=4:[-4.32320887e+00 -4.26977454e+00 -4.26977454e+00 -4.15406592e+00
 -4.15406592e+00 -4.05368991e+00 -4.05368991e+00 -1.45862884e+00
 -1.45862884e+00 -1.23953684e+00 -8.00000000e-01 -8.00000000e-01
 -5.73607328e-01 -5.73607328e-01 -1.50770223e-02 -3.62636144e-16
  1.37793938e-16  1.54065923e-01  1.54065923e-01  5.73607328e-01
  5.73607328e-01  8.00000000e-01  8.00000000e-01  9.31404990e-01
  1.45862884e+00  1.45862884e+00  3.06859501e+00  4.01507702e+00
  4.05368991e+00  4.05368991e+00  4.26977454e+00  4.26977454e+00
  5.23953684e+00  8.32320887e+00]
(1)_Momentum_Sector_ki=5:[-4.62730693e+00 -4.60424298e+00 -4.49590194e+00 -4.26977454e+00
 -3.98137285e+00 -3.71922147e+00 -3.54183602e+00 -1.41222991e+00
 -1.12416040e+00 -9.08075769e-01 -6.27306929e-01 -5.73607328e-01
 -4.58163984e-01 -1.87770093e-01 -3.27794285e-16  1.45711456e-16
  3.26758994e-01  4.58163984e-01  5.73607328e-01  6.27306929e-01
  9.08075769e-01  1.12416040e+00  1.27324101e+00  2.89590194e+00
  3.54183602e+00  3.71922147e+00  4.26977454e+00  4.60424298e+00
  4.62730693e+00  5.58137285e+00]
(1)_Momentum_Sector_ki=6:[-5.17785031e+00 -4.96914295e+00 -4.60424298e+00 -4.15406592e+00
 -3.71922147e+00 -3.36914295e+00 -3.14561414e+00 -1.88547091e+00
 -1.12416040e+00 -9.08075769e-01 -8.00000000e-01 -8.00000000e-01
 -5.50553069e-01 -3.34468441e-01 -2.85470913e-01 -1.54065923e-01
  3.17070848e-16  1.54065923e-01  2.85470913e-01  3.34468441e-01
  5.50553069e-01  8.00000000e-01  8.00000000e-01  9.08075769e-01
  1.12416040e+00  1.88547091e+00  3.14561414e+00  3.36914295e+00
  3.71922147e+00  4.15406592e+00  4.60424298e+00  4.96914295e+00
  5.17785031e+00]
(1)_Momentum_Sector_ki=7:[-5.58137285e+00 -5.17785031e+00 -4.62730693e+00 -4.05368991e+00
 -3.54183602e+00 -3.14561414e+00 -2.89590194e+00 -1.45862884e+00
```

```
-1.27324101e+00 -6.27306929e-01 -5.50553069e-01 -4.58163984e-01
-3.34468441e-01 -3.26758994e-01 -1.79519130e-15  1.58128402e-15
 1.87770093e-01  3.34468441e-01  4.58163984e-01  5.50553069e-01
 6.27306929e-01  1.41222991e+00  1.45862884e+00  3.14561414e+00
 3.54183602e+00  3.98137285e+00  4.05368991e+00  4.49590194e+00
 4.62730693e+00  5.17785031e+00]
```

由此可见，不论是对于 $g = 1.4$ 还是 0.4 的数据，能谱都具有一定的对称性：

- $(1,7),(2,6),(3,5)$ 的能谱分布相同；
- $(2,6)$ 的能谱本身的正负分布式对称的。