量子物理计算方法选讲实验报告

李昊恩　2021010312

Density Matrix Renormalization Group(DMRG), Task 2

目录

1 引言

本次实验的目的为，利用有限自动机（Finite automata）来快速构造一个具有一定对称性的简单哈密顿量的矩阵乘积算子（Matrix Product Operator, MPO）。此后，为了验证其正确性，我们对该 MPO 进行缩并，只留下物理指标，然后进行精确对角化，并与原哈密顿量的矩阵表示以及特征值、特征向量进行对比。

1.1 矩阵乘积态

我们考虑一个具有 N 个自旋的量子系统 $|\psi\rangle$，则该量子系统有如下的组态展开：

$$|\psi\rangle = \sum_{i_0,\cdots,i_{N-1}} c_{i_0,\cdots,i_{N-1}} |i_0 \cdots i_{N-1}\rangle , \tag{1}$$

在该表达式中，$c_{i_0,\cdots,i_{N-1}}$ 是一个 N 阶张量。我们考虑将该张量写成若干 2、3 阶张量的缩并形式，也就是：

$$c_{i_0,\cdots,i_{N-1}} = \sum_{u_1,\cdots,u_{N-1}} {}^{0}T_{i_0,u_1}\, {}^{1}T_{u_1,i_1,u_2} \cdots {}^{N-1}T_{u_{N-1}i_{N-1}} \tag{2}$$

以上表达式可以用张量网络图表示如1所示:

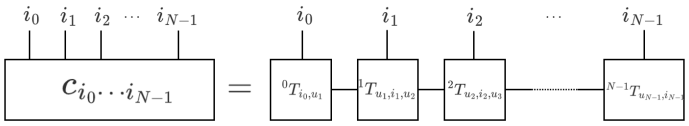


图 1: 开放边界矩阵乘积态

除了以上开放边界的形式之外，还有周期边界的矩阵乘积态（或 1D 环状矩阵乘积态等）。

在矩阵乘积态中引入的每个张量的新的指标 $u_1, \cdots, u_{N-1}$ 称为虚拟指标（或辅助指标），这些指标都被缩并掉，只剩下原有的指标 $i_0, \cdots, i_{N-1}$，它们称为物理指标。

1.2 矩阵乘积算子

1D 哈密顿量（或者其他的可观测量）可以写成矩阵乘积算子（MPO）的形式，例如，在开放边界条件中，类似于 MPS，H 可以写成一系列张量的缩并形式，这时候哈密顿量也就称为矩阵乘积算子（MPO）。每个张量此时有 2 个虚拟指标和 2 个物理指标。

与 MPS 相对应地，在 MPO 中，每个物理指标的维数就相当于每个局域 Hilbert 空间的维数。MPO 的形式为：

$$\rho = \sum_{i'_0 \cdots i'_{N-1} i_0 \cdots i_{N-1}} M^{[0]}_{i'_0, i_0, \alpha_1} M^{[1]}_{\alpha_1, i'_1, i_1, \alpha_2} \cdots M^{[N-1]}_{\alpha_{N-1}, i'_{N-1}, i_{N-1}} \bigotimes_{m=0}^{N-1} \bigotimes_{n=0}^{N-1} \left| i'_m \right\rangle\!\left\langle i'_m \right|_{i_n} . \tag{3}$$

MPO 的图形如图2所示，其中，每个张量 $M^{[i]}$ 的两个物理指标分别和局域 Hilbert 空间的左、右矢进行缩并。

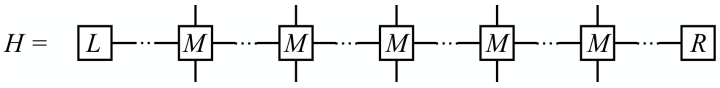


图 2: 开放边界矩阵乘积算子

1.3 有限自动机

在变分矩阵乘积态（varMPS）或密度矩阵重整化群（DMRG）算法中，我们需要得到哈密顿量对应的 MPO。我们可以写出算子的矩阵形式的 SVD 或者 QR 分解，从而得到其 MPO 形式。但是，对于一些简单的哈密顿量，以上过程并不是必须的，利用有限状态自动机（Finite automata）可以方便地生成对应的 MPO。

有限自动机是一种数学模型（事实上可以看成一种首先图灵机）。一个有限自动机可以用一个有向图来表示，其中有向图的节点（node）对应 “状态”，边（edge）则对应 “指令”。节点和边的连接关系则决定该有限自动机的 “状态转移函数”，我们用箭头来描述一个指令对应的始末状态。

矩阵乘法也可以用一个有限自动机表示。我们可以用有向图的节点表示矩阵指标的一个取值，带有箭头的边表示其连接的两个指标对应的矩阵元非零，指示方向为该非零矩阵元的行指标 → 列指标，该边的权（weight）则为矩阵元。如图3所示。

$$M = \begin{pmatrix} M(x_1, y_1) & M(x_1, y_2) \\ M(x_2, y_1) & M(x_2, y_2) \\ M(x_3, y_1) & M(x_3, y_2) \end{pmatrix} = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \\ M_{31} & M_{32} \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 4 & -3 \\ 0 & 2 \end{pmatrix}$$

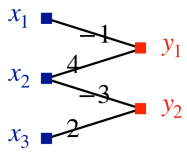
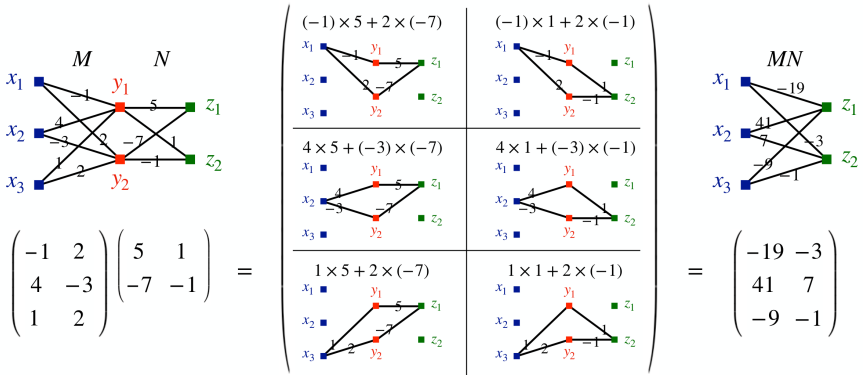


图 3: 矩阵的有向图表示

此时，矩阵乘法对应着沿着路径将始末状态的节点进行连接，如图4所示。



$$\begin{pmatrix} -1 & 2 \\ 4 & -3 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 5 & 1 \\ -7 & -1 \end{pmatrix} = \begin{pmatrix} -19 & -3 \\ 41 & 7 \\ -9 & -1 \end{pmatrix}$$

图 4: 矩阵乘法的有限自动机表示

1.4 画出有限自动机、构造矩阵乘积算子

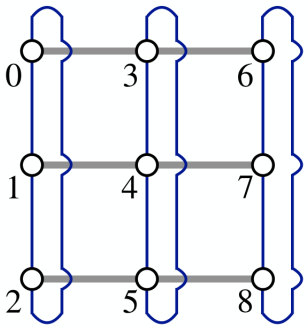
本实验考虑一个正方形自旋格点模型（图5），用于示例如何用有限自动机构造其矩阵乘积算子。该模型的哈密顿量为：

$$H = -J \sum_{\langle i,j \rangle_H} Z_i Z_j - g \sum_{\langle i,j,k \rangle_V} Z_i X_j Z_k - h \sum_i X_i. \tag{4}$$

其中 X 和 Z 分别表示 Pauli-X 和 Pauli-Z 算子。

该系统的哈密顿量为 “乘积-求和” 形式。因此，我们可以构造一个有限自动机，其中从头节点到尾节点的每一条路径与求和中的每一项一一对应。由此，我们可以将哈密顿量写成一些列 “矩阵” 的乘积，其中 “矩阵” 的每个元素是一个局部 Hilbert 空间上的算子，“矩阵元乘法” 则定义为算子之间的张量积。

有了这个思路，我们可以观察哈密顿量中每一项的特点，并且注意在有限自动机中将矩阵元（即边的权）相同的路径合一并不改变求和的结果，我们很容易将有限自动机写成如下的具有**平移对称性**的样子，如图6：



$$H = -J\sum_{\langle i,j\rangle_H}\sigma_i^z\sigma_j^z - g\sum_{\langle i,j,k\rangle_V}\sigma_i^z\sigma_j^x\sigma_k^z - h\sum_i\sigma_i^x$$

$$\langle i,j\rangle_H \in (0,3),(3,6),(1,4),(4,7),(2,5),(5,8)$$

$$\langle i,j,k\rangle_V \in (0,1,2),(1,2,0),(2,0,1),(3,4,5),(4,5,3),(5,3,4),(6,7,8),(7,8,6),(8,6,7)$$

$J = 1.0$, $g = 1.7$, $h = 0.7$, σ^x and σ^z are Pauli matrices

图 5: 正方形自旋格点模型

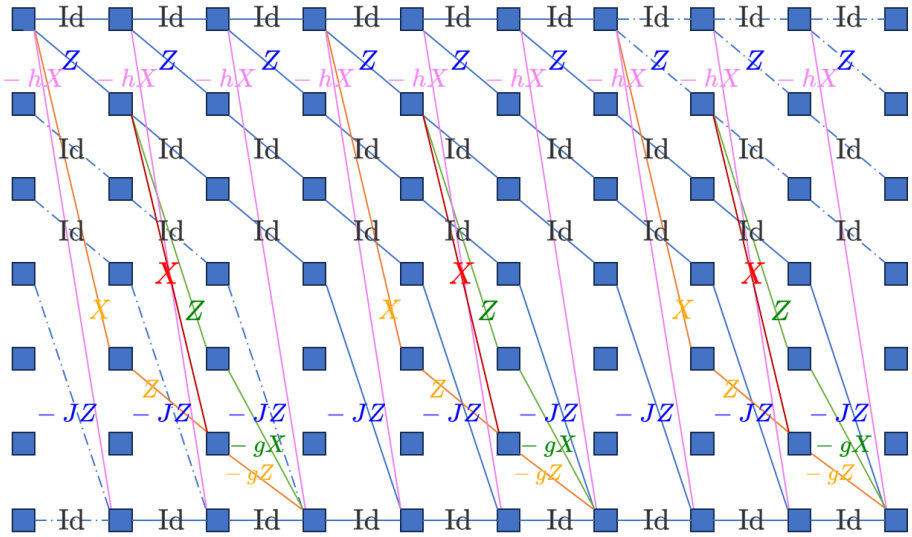


图 6: 有限自动机图

哈密顿量中的每一项最多涉及 4 个相邻自旋的相互作用（次次近邻），因此第一行和最后一行可以全部用恒等算子连接，表示每一项可以开始于若干步恒等算子，结束于若干步恒等算子。在这之间，可以是 $-J\sum_{\langle i,j\rangle_H} Z_iZ_j$ 的次次近邻相互作用，根据该项的平移对称性，我们重复地画出 $Z-\mathrm{Id}-\mathrm{Id}-(-JZ)$ 路径，如图6中的深蓝色折线。除此之外，也可以是如 $-h\sum_i X_i$ 项中，只经过一项 $-hX$，因此我们在图中平移对称地添加若干项 $-hX$，连接第一个指标和最后一个指标，如图6中粉色折线。对于 $-g\sum_{\langle i,j,k\rangle_V} Z_iX_jZ_k$，可以分成 $ZX(-gZ)$，$ZZ(-gX)$，$XZ(-gZ)$ 三种类型的项，每一项对应一种从第一个指标走到最后一个指标的方式。对于第一种类型，可在图中添加三条 $X-Z-(-gZ)$ 折线（黄色线条所在折线）；第二种类型对应图中三条 $Z-X-(-gZ)$ 折线（红色线条所在线）；第三种类型则对应图中三条 $Z-Z-(-gX)$ 折线（绿色线条所在线）。

至此我们已经添加了所有求和项对应的路径，再删掉一些不可能路径（图中用虚线表示），就得到整个系统的有限自动机。从有限自动机中可以快速读出矩阵乘积的每个因子，具体来说，在两列之间，若行指标 $\to$ 列指标有边，则在矩阵因子的对应位置上写下该边上的矩阵元，这就

得到了体系的 MPO。注意，由于此体系具有平移对称性，这里的 $M_0 = M_3 = M_6, M_1 = M_4 = M_7, M_2 = M_5 = M_8$，矩阵形式分别为：

$$M_0 = \begin{pmatrix} \mathrm{Id} & Z & 0 & 0 & X & 0 & -hX \\ 0 & 0 & \mathrm{Id} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathrm{Id} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -JZ \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{Id} \end{pmatrix} \tag{5}$$

$$M_1 = \begin{pmatrix} \mathrm{Id} & Z & 0 & 0 & X & 0 & -hX \\ 0 & 0 & \mathrm{Id} & 0 & Z & X & 0 \\ 0 & 0 & 0 & \mathrm{Id} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -JZ \\ 0 & 0 & 0 & 0 & 0 & Z & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{Id} \end{pmatrix} \tag{6}$$

$$M_2 = \begin{pmatrix} \mathrm{Id} & Z & 0 & 0 & X & 0 & -hX \\ 0 & 0 & \mathrm{Id} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathrm{Id} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -JZ \\ 0 & 0 & 0 & 0 & 0 & 0 & -gX \\ 0 & 0 & 0 & 0 & 0 & 0 & -gZ \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{Id} \end{pmatrix} \tag{7}$$

此时：

$$H = \mathbf{L}^T M_0 M_1 M_2 M_0 M_1 M_2 M_0 M_1 M_2 \mathbf{R}. \tag{8}$$

其中 $\mathbf{L} = (1,0,0,0,0,0,0)^T, \mathbf{R} = (0,0,0,0,0,0,1)^T$。

在下一部分我们编写程序来对该 MPO 进行缩并，只留下物理指标，然后进行精确对角化，并与原哈密顿量的矩阵表示以及特征值、特征向量进行对比。

2 代码实现

有了上一个部分的 MPO 形式，想要计算其缩并，实际上就是计算该 MPO 形式在物理指标的一组基底 $\{i_0 \cdots i_{N-1}\}$ 的矩阵表示，事实上，对于矩阵元 $\langle i'_0 \cdots i'_{N-1} | H_{\mathrm{MPO}} | i_0 \cdots i_{N-1} \rangle$，我

们有：

$$\langle i'_0 \cdots i'_8 | H_{\mathrm{MPO}} | i_0 \cdots i_8 \rangle = \mathbf{L}^T \langle i'_0 | M_0 | i_0 \rangle \langle i'_1 | M_1 | i_1 \rangle \cdots \langle i'_8 | M_2 | i_9 \rangle \mathbf{R}, \tag{9}$$

这里（例如）$\langle i'_0 | M_0 | i_0 \rangle$ 是一个形式记号，它表示一个分量是实数或复数的矩阵

$$\langle i'_0 | M_0 | i_0 \rangle := \left(\langle i'_0 | (M_0)_{k,l} | i_0 \rangle \right)_{k,l}, \tag{10}$$

也即，矩阵 $\langle i'_0 | M_0 | i_0 \rangle$ 的 k,l 元是对 M_0 的 k,l 元（注意 M_0 的每个元素是局部 Hilbert 空间上的一个算子）求其关于 $\langle i'_0 |$ 和 $| i_0 \rangle$ 的矩阵元所得的结果。

其代码实现如下，首先定义 $\langle i'_0 | M_0 | i_0 \rangle$ 矩阵：

```
#define contraction of physical index of MPOs, we only need to calculate M0, M1 and
    M2,
#(M0 = M3 = M6, M1 = M4 = M7, M2 = M5 = M8 due to translational invariance)
def PhysContractM0(lbit,rbit):
    i = eye[lbit,rbit]
    z = pZ[lbit,rbit]
    x = pX[lbit,rbit]
    M0 = np.array([[i,z,0,0,x,0,-h*x],
                   [0,0,i,0,0,0,0],
                   [0,0,0,i,0,0,0],
                   [0,0,0,0,0,0,-J*z],
                   [0,0,0,0,0,0,0],
                   [0,0,0,0,0,0,0],
                   [0,0,0,0,0,0,i]])
    return M0+0j

def PhysContractM1(lbit,rbit):
    i = eye[lbit,rbit]
    z = pZ[lbit,rbit]
    x = pX[lbit,rbit]
    M1 = np.array([[i,z,0,0,0,0,-h*x],
                   [0,0,i,0,z,x,0],
                   [0,0,0,i,0,0,0],
                   [0,0,0,0,0,0,-J*z],
                   [0,0,0,0,0,z,0],
                   [0,0,0,0,0,0,0],
                   [0,0,0,0,0,0,i]])
    return M1+0j

def PhysContractM2(lbit,rbit):
    i = eye[lbit,rbit]
    z = pZ[lbit,rbit]
```

```
    x = pX[lbit,rbit]
    M2 = np.array([[i,z,0,0,0,0,-h*x],
                   [0,0,i,0,0,0,0],
                   [0,0,0,i,0,0,0],
                   [0,0,0,0,0,0,-J*z],
                   [0,0,0,0,0,0,-g*x],
                   [0,0,0,0,0,0,-g*z],
                   [0,0,0,0,0,0,i]])
    return M2+0j
```

然后，缩并求出 H_{MPO} 的每个矩阵元，得到哈密顿量，随后进行精确对角化：

```
Hamil_contractmpo = np.zeros([2**9,2**9])+0j
Lvec = np.array([1,0,0,0,0,0,0])+0j
Rvec = np.transpose(np.array([0,0,0,0,0,0,1]))+0j

#Contract mpo to get the matrix elements of Hamiltonian
for i in tqdm(range(2**9)):
    for j in range(2**9):
        mel = Lvec @ PhysContractM0(ReadBit(i,0),ReadBit(j,0)) @ PhysContractM1(
    ReadBit(i,1),ReadBit(j,1)) @ \
        PhysContractM2(ReadBit(i,2),ReadBit(j,2)) @ PhysContractM0(ReadBit(i,3),
    ReadBit(j,3)) @ PhysContractM1(ReadBit(i,4),ReadBit(j,4))\
        @ PhysContractM2(ReadBit(i,5),ReadBit(j,5)) @ PhysContractM0(ReadBit(i,6),
    ReadBit(j,6)) @\
        PhysContractM1(ReadBit(i,7),ReadBit(j,7)) @ PhysContractM2(ReadBit(i,8),
    ReadBit(j,8)) @ Rvec
        Hamil_contractmpo[i,j] = mel

# Diagonalize the Hamiltonian
evalsmpo, evecsmpo = np.linalg.eigh(Hamil_contractmpo)

# Find the 20 lowest eigenvalues
lowest_evals = np.sort(evalsmpo)[:20]

print("The Lowest 20 Eigenvalues:{}".format(lowest_evals))

```

为了验证正确性，我们直接构造哈密顿量的矩阵形式并进行对角化，首先定义多体算子的矩阵形式：

```
#define many_body_operator
def many_body_operator(idx, oprts, size = 9):
    "Tensor product of `orts` acting on indexes `idx`. Fills rest with Id."
    matrices = [eye if k not in idx else oprts[idx.index(k)] for k in range(size)]
```

```
    prod = matrices[0]
    for k in range(1, size):
        prod = np.kron(prod, matrices[k])
    return prod
```

随后，对多体算子求和直接得到哈密顿量的矩阵形式，并对角化：

```
Hamil = np.zeros([2**9,2**9])+0j
for hh in Hlist:
    Hamil += -J*many_body_operator(hh, [pZ,pZ])
for v in Vlist:
    Hamil += -g*many_body_operator(v, [pZ,pX,pZ])
for i in range(9):
    Hamil += -h*many_body_operator([i], [pX])
# Diagonalize the Hamiltonian
evals, evecs = np.linalg.eigh(Hamil)

# Find the 20 lowest eigenvalues
lowest_evals = np.sort(evals)[:20]

print("The Lowest 20 Eigenvalues:{}".format(lowest_evals))
```

3 结果

采用 $(J, g, h) = (1.0, 1.7, 0.7)$，用 MPO 缩并得到的哈密顿量精确对角化的结果为：

```
The Lowest 20 Eigenvalues:[-16.98738797 -16.33765141 -14.20762975 -13.75785526
    -13.51871353 -13.51871353 -12.92917989 -12.47041377 -12.47041377 -12.35546297
-12.35546297 -12.24793588 -11.50149999 -11.50149999 -11.22721333
-11.22721333 -11.047253   -11.047253   -10.86124774 -10.55957422]
```

对直接构造的哈密顿量矩阵进行精确对角化的结果为：

```
The Lowest 20 Eigenvalues:[-16.98738797 -16.33765141 -14.20762975 -13.75785526
    -13.51871353 -13.51871353 -12.92917989 -12.47041377 -12.47041377 -12.35546297
-12.35546297 -12.24793588 -11.50149999 -11.50149999 -11.22721333
-11.22721333 -11.047253   -11.047253   -10.86124774 -10.55957422]
```

可见其能谱完全一致。事实上，我们运行如下代码可以看出两种方法得到的哈密顿量矩阵也是完全相同的：

```
#determine whether the contraction of mpos equals to the true Hamiltonian
print(np.array_equal(Hamil_contractmpo, Hamil))
```

输出为 `True`。