Diagonalizing QCNN Hamiltonian for Outputting Ground State Wave Functions

1 Decomposing the Hamiltonian

The Hamiltonian we need to simulate comes from the second equation of the paper, given by

$$H = -J\sum_{i=1}^{N-2} Z_i X_{i+1} Z_{i+2} - h_1 \sum_{i=1}^{N} X_i - h_2 \sum_{i=1}^{N-1} X_i X_{i+1} \tag{1}$$

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \text{and} \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

where X_i and Z_i are the Pauli operators for the spin at site i. For the purposes of the code, we want to output a Hamiltonian matrix of size $2^N \times 2^N$. We can't do this with equation 1, since the Pauli operators are 2×2 matrices. To output a matrix of size $2^N \times 2^N$ we can take a series of tensor products $\otimes$ between the identity matrix $\mathbb{1}$ and the Pauli operators. The structure of the tensor product between two matrices A and B is given below:

$$A \otimes B = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \otimes \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} & a_{12}\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \\ a_{21}\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} & a_{22}\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \end{bmatrix} \tag{2}$$

I believe the python package `np.kron(a,b)` can perform such computations. An example of $N = 3$ qubits is shown below ($J = 1$):

$$\begin{aligned} H &= \left(-1\sum_{i=1}^{3-2} Z_i X_{i+1} Z_{i+2}\right) - \left(h_1 \sum_{i=1}^{3} X_i\right) - \left(h_2 \sum_{i=1}^{3-1} X_i X_{i+1}\right) \\ &= -(Z_1X_2Z_3) - h_1(X_1 + X_2 + X_3) - h_2(X_1X_2 + X_2X_3) \\ &= -[(Z \otimes \mathbb{1} \otimes \mathbb{1}) \cdot (\mathbb{1} \otimes X \otimes \mathbb{1}) \cdot (\mathbb{1} \otimes \mathbb{1} \otimes Z)] \\ &\quad - h_1[(X \otimes \mathbb{1} \otimes \mathbb{1}) + (\mathbb{1} \otimes X \otimes \mathbb{1}) + (\mathbb{1} \otimes \mathbb{1} \otimes X)] \\ &\quad - h_2[(X \otimes \mathbb{1} \otimes \mathbb{1}) \cdot (\mathbb{1} \otimes X \otimes \mathbb{1}) + (\mathbb{1} \otimes X \otimes \mathbb{1}) \cdot (\mathbb{1} \otimes \mathbb{1} \otimes X)] \end{aligned} \tag{3}$$

Equation (3) is really what we're trying to automate for N qubits, as opposed to the example shown above for $N = 3$. That being said, we should run the code for smaller values of N to make sure it's working as expected before performing longer computations. For N qubits, the tensor product permutes through the Pauli operators in the following way:

$$\sum_{i=1}^{N} X_i = (X \otimes \cdots \otimes \mathbb{1}) + (\mathbb{1} \otimes X \otimes \cdots \otimes \mathbb{1}) + (\mathbb{1} \otimes \mathbb{1} \otimes X \otimes \cdots \otimes \mathbb{1}) + \cdots + (\mathbb{1} \otimes \cdots \otimes X) \tag{4}$$

As far as programming something like this, I'm not quite sure how we're going to automate the Pauli matrices permuting through the tensor products of identity matrices. We can take advantage of the fact that the index i determines the position of the Pauli matrix X or Z in the series of tensor products, as can be seen by the terms written out in equation (4).

2 Permutation of Pauli Operators