Diagonalizing QCNN Hamiltonian for Outputting Ground State Wave Functions

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

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

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 ($J = 1$):

$$H = \left(-1 \sum_{i=1}^{3-2} Z_i X_{i+1} Z_{i+2} \right) - \left(h_1 \sum_{i=1}^{3} X_i \right) - \left(h_2 \sum_{i=1}^{3-1} X_i X_{i+1} \right)$$

$$= -(Z_1 X_2 Z_3) - h_1(X_1 + X_2 + X_3) - h_2(X_1 X_2 + X_2 X_3)$$

$$\begin{aligned} &= -[(Z \otimes \mathbb{1} \otimes \mathbb{1}) \cdot (\mathbb{1} \otimes X \otimes \mathbb{1}) \cdot (\mathbb{1} \otimes \mathbb{1} \otimes Z)] \\ &\quad - h_1[(X \otimes \mathbb{1} \otimes \mathbb{1}) + (\mathbb{1} \otimes X \otimes \mathbb{1}) + (\mathbb{1} \otimes \mathbb{1} \otimes X)] \\ &\quad - h_2[(X \otimes \mathbb{1} \otimes \mathbb{1}) \cdot (\mathbb{1} \otimes X \otimes \mathbb{1}) + (\mathbb{1}) \cdot (\mathbb{1} \otimes \mathbb{1} \otimes X)] \end{aligned} \tag{3}$$

Equation (3) is really what we're trying to automate for N qubits. For N qubits, the tensor product permutes through the Pauli operators in the following way:

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