Assignment 1: Exact diagonalization and quantum phase transitions

Instructor: Lesik Motrunich
TA: Liam O'Brien

Ph 121C: Computational Physics Lab, Spring 2024
California Institute of Technology
Due: Tuesday, April 16, 2024

1 Introduction to quantum phase transitions

In this assignment we will learn about the exact diagonalization method for quantum many-body physics and use it to investigate the phenomenon of quantum phase transitions in condensed matter systems. The analysis of phase transitions that occur at finite temperature should be familiar to you from statistical mechanics courses. These include transitions to superconductivity and superfluidity, as well as the transition to ferromagnetism at the Curie temperature. Generally speaking, a finite-temperature phase transition involves non-analyticity in some derivative of a system's free energy as a function of temperature. Continuous, or second-order, phase transitions fall into universality classes according to the power-law divergence of physical quantities like heat capacity and susceptibility, and the disappearance of the order parameter. All finite-temperature phase transitions are ultimately "classical," because above some length scale the phase change is primarily driven by thermal fluctuations.

Quantum phase transitions, instead, occur at $T = 0$ as one varies the parameters of the Hamiltonian specifying the system, and are signified by qualitative changes in the nature of the ground state which are driven by purely quantum mechanical fluctuations. However, they inherit the phenomenology of continuous phase transitions: at the critical point, physical properties still

develop singularities (in the thermodynamic limit) which allow us to define critical indices, and thus universality classes. The goal of this assignment is to provide you with experience in the phenomenology of phase transitions of quantum many-body Hamiltonians and to familiarize you with the exact diagonalization method for such problems. We will sometimes refer to a set of excellent lecture notes on the latter topic by Sandvik [1].

2 Quantum Ising model in one dimension

2.1 Many-body Hilbert space

Our example for this assignment is the quantum Ising model in one dimension. Consider a system living on a regular lattice of L sites. At each site $j = 1, \ldots, L$, we place a spin- 1/2 degree of freedom (qubit), whose state is described by a vector in a 2-dimensional local Hilbert space $\mathcal{H}_j$. Each local Hilbert space is spanned by orthonormal basis vectors we denote $|\uparrow\rangle$ and $|\downarrow\rangle$, corresponding to spin up and down, respectively, in the z direction. The local Hilbert space is a vector space over $\mathbb{C}$ having an inner product and standard norm, and a general (pure) state is of the form $|\psi_j\rangle = c_\uparrow|\uparrow\rangle + c_\downarrow|\downarrow\rangle$, where $c_{\uparrow,\downarrow} \in \mathbb{C}$ and $|c_\uparrow|^2 + |c_\downarrow|^2 = 1$. The full many-body Hilbert spacecontaining all possible states of the chain - is described by a vector in a much larger space which has a simple decomposition: it is the L-fold tensor product of all of the local Hilbert spaces. That is, $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \cdots \otimes \mathcal{H}_L \cong (\mathbb{C}^2)^{\otimes L}$. A convenient orthonormal basis for $\mathcal{H}$ consists of the following vectors:

$$
\begin{aligned}
|e_0\rangle &= |\uparrow_1 \otimes |\uparrow\rangle_2 \otimes |\uparrow\rangle_3 \otimes \cdots \otimes |\uparrow\rangle_L \equiv |\uparrow_1\uparrow_2\uparrow_3 \cdots \uparrow_L\rangle \\
|e_1\rangle &= |\downarrow_1\uparrow_2\uparrow_3 \cdots \uparrow_L\rangle\,, \\
|e_2\rangle &= |\uparrow_1\downarrow_2\uparrow_3 \cdots \uparrow_L\rangle\,, \\
|e_3\rangle &= |\downarrow_1\downarrow_2\uparrow_3 \cdots \uparrow_L\rangle\,, \\
&\vdots \\
|e_{2^L-1}\rangle &= |\downarrow_1\downarrow_2\downarrow_3 \cdots \downarrow_L\rangle\,.
\end{aligned}
\tag{1}
$$

Using this (or any other orthonormal) basis, the state of the system is written $|\psi\rangle = \sum_{\alpha=0}^{2^L-1} c_\alpha |e_\alpha\rangle$, where $c_\alpha \in \mathbb{C}$ for all α and $\sum_\alpha |c_\alpha|^2 = 1$. To see why the basis (1) is nice, relabel the local basis states as follows:

$|\uparrow\rangle \mapsto |0\rangle, |\downarrow\rangle \mapsto |1\rangle$. Then the many-body basis states are $|e_0\rangle = |000\cdots0\rangle$, $|e_1\rangle = |100\cdots0\rangle, |e_2\rangle = |010\cdots0\rangle, |e_3\rangle = |110\cdots0\rangle$, and so on. Using this choice, the spin configuration of basis state $\alpha = 0, \ldots, 2^L - 1$ is found easily from the base- 2 representation of α, with more significant bits toward the right ("little endian" order). As computers store integers in terms of their bits, to see the classical state of spin $j = 1, \ldots, L$ in $|e_\alpha\rangle$ one need only read off the bit $j-1$ in the number α, which can be done using fast bitwise operations 3. For example, in python or C the code alpha & $1(\ll j-1)$ evaluates to 0 if spin j is in state $|\uparrow\rangle$, or to 2^{j-1} for $|\downarrow\rangle$.

Two important notions for many-body quantum systems are correlations and entanglement. These concepts will appear in this and subsequent assignments and will be used frequently to characterize quantum ground states and phases. For example, according to the above description, consider the following state which is perfectly valid for a three-site system:

$$|\psi_{\text{ex}}\rangle = \frac{|\uparrow_1\uparrow_2\uparrow_3\rangle + |\downarrow_1\downarrow_2\uparrow_3\rangle}{\sqrt{2}}. \tag{2}$$

It is evident that in such a state it is impossible to answer a question like, "is spin 1 pointing up or down?" Instead, we could conceivably ask, "what is the probability that spin 1 is pointing up?", as is often done in single-particle QM. But it turns out that this type of question is not very useful in the many-body context. More interesting probes account for the correlated nature of the state: for example, "how correlated is spin 1 with spin 2?" (to which the answer would be, perfectly) or "how correlated is spin 1 with spin 3?" (to which the answer is, not at all). This distinction manifests itself in the ability to usefully decompose the state (2) as follows:

$$|\psi_{\text{ex}}\rangle = \frac{1}{\sqrt{2}}\left(|\uparrow_1\uparrow_2\rangle + |\downarrow_1\downarrow_2\rangle\right) \otimes |\uparrow_3\rangle \tag{3}$$

2.2 Hamiltonian

The Hamiltonian of a quantum system is a linear operator acting on the Hilbert space $\mathcal{H}$. It implements a measurement of energy and also controls dynamics via the Schrödinger equation, and thermodynamic equilibrium of the quantum system. Only eigenstates of $\mathcal{H}$ have definite energy, and because the Hamiltonian also determines the time evolution of the system, energy eigenstates are stationary in time. Thus the energy eigenstates specify a

preferred basis for $\mathcal{H}$. In particular, we will be mostly concerned with the ground state of H, which the system occupies with certainty in the zero-temperature limit due to the total suppression of thermal fluctuations.

The quantum Ising model is a generalization of the classical Ising model, and simulates the behavior of magnets by means of interactions in the z components of neighboring spin- $1/2$ degrees of freedom, subject to an applied magnetic field in the transverse x direction. The Hamiltonian on the one-dimensional chain of sites labeled $j = 1, \ldots, L$ as described above is

$$H = -J \sum_{j=1}^{L-1} \sigma_j^z \sigma_{j+1}^z - h \sum_{j=1}^{L} \sigma_j^x \tag{4}$$

Each h or J term is a Pauli spin matrix acting on the local Hilbert space of a particular site or a product of Paulis acting on two neighboring sites. As written here, the model has open boundary conditions. One can make the lattice periodic by adding the term $-J\sigma_L^z \sigma_1^z$, which we often do to mitigate boundary effects (we minimize finite-size effects because we are typically interested in the thermodynamic limit $L \to \infty$).

We have abused notation somewhat when writing the above lattice Hamiltonian. A priori, it does not make sense to take the sum of operators acting on different spaces, as written in (4). We have suppressed the trivial action of each operator on all other sites in the system, which is simply the identity in that local Hilbert space. That is,

$$\sigma_j^x = \mathbb{I}_1 \otimes \mathbb{I}_2 \otimes \cdots \otimes \mathbb{I}_{j-1} \otimes \sigma_j^x \otimes \mathbb{I}_{j+1} \otimes \mathbb{I}_{j+2} \otimes \cdots \otimes \mathbb{I}_L \tag{5}$$

$$\sigma_j^z \sigma_{j+1}^z = \mathbb{I}_1 \otimes \mathbb{I}_2 \otimes \cdots \otimes \mathbb{I}_{j-1} \otimes \sigma_j^z \otimes \sigma_{j+1}^z \otimes \mathbb{I}_{j+2} \otimes \cdots \otimes \mathbb{I}_L \tag{6}$$

with the natural definition of the tensor product of operators:

$$(A \otimes B \otimes \ldots)(|v\rangle \otimes |w\rangle \otimes \ldots) = |Av\rangle \otimes |Bw\rangle \otimes \ldots \tag{7}$$

That is, σ_j^x can change the state of only the j-th spin. For example,

$$\sigma_j^x \, |s_1 s_2 \ldots s_{j-1} \uparrow_j s_{j+1} \ldots s_L\rangle = |s_1 s_2 \ldots s_{j-1} \downarrow_j s_{j+1} \ldots s_L\rangle \, . \tag{8}$$

This notation, omitting operator tensor product symbols as well as identity operator factors, is clearly much more concise and is always used. Hamiltonians containing only terms of this form are described as " k-local," where

k is the maximum number of contiguous sites on which an individual term acts nontrivially (for example, (4) is 2-local). k-local Hamiltonians are the subject of widespread study, and are thought to be good approximations to many physical systems.

2.3 Ground states and phase transitions

Often quantum phase transitions are the result of competing ground-state preferences dictated by the various types of terms in the Hamiltonian. This is easily seen in the Ising model (4). On the one hand, the $-J\sigma_j^z\sigma_{j+1}^z$ interaction terms minimize energy by aligning neighboring spins along the z-direction identically in either of the symmetry-breaking $|\uparrow\uparrow\rangle$ or $|\downarrow\downarrow\rangle$ states. On the other, the
$-h\sigma_j^x$ magnetic field terms minimize energy with spins oriented in the perpendicular x-direction, that is, the $|+\hat{x}\rangle = (|\uparrow\rangle + |\downarrow\rangle)/\sqrt{2}$ state. This state maintains the Ising spin-flip symmetry of the Hamiltonian. Thus, the relative strengths of J and h determine which type of state is the ground state.

In fact, the ground state wavefunction depends only on the ratio h/J, which attains a critical value $(h/J)_c = 1$. In the quantum Ising model, for $h/J < (h/J)_c$ we expect a magnetically ordered phase, while for $h/J > (h/J)_c$ we expect a "disordered" phase (often called a quantum paramagnet). Approaching the critical point $(h/J)_c$ the correlation length of the ground state diverges, allowing quantum fluctuations at all length scales to take the system between the two phases. Accordingly, irrespective of the low-energy spectra of either phase around the transition, the excitation gap of the Hamiltonian vanishes at the critical point.

The ground state in the magnetically ordered phase is two-fold degenerate, roughly corresponding to the states $|\uparrow\uparrow\uparrow\cdots\rangle$ and $|\downarrow\downarrow\downarrow\cdots\rangle$. More precisely, there are two low-energy states with exponentially small splitting in system size, but these are actually linear combinations of the symmetry-breaking states. Any finite quantum system respects all of its symmetries; thus the true eigenstates are eigenstates of the Ising symmetry operator $U_x = \prod_j \sigma_j^x$, which are essentially $\frac{1}{\sqrt{2}}(|\uparrow\uparrow\uparrow\ldots\rangle \pm |\downarrow\downarrow\downarrow\ldots\rangle)$. All of these statements are correct in the limit $h/J \to 0$, and remain quite accurate elsewhere within the phase. The first excitation above this two-dimensional ground state manifold has a finite energy gap that persists in the thermody-

namic limit.

In the paramagnetic phase, the ground state is non-degenerate. In this case, it is easier to describe, being simply the state $|+\hat{x}\rangle \otimes |+\hat{x}\rangle \otimes \cdots$ in the limit $h/J \to \infty$. Here again there is a finite energy gap to the excited state spectrum even for $L \to \infty$.

2.4 Observables

To study the magnetic ordering within each phase and at the transition, we look for a quantity that behaves like an order parameter. The ground state expectation value of $M/L = \sum_{j=1}^{L} \sigma_j^z/L$ will be strictly zero in all phases due to the exact Ising symmetry of the Hamiltonian in finite systems. Instead, we can measure the expectation value of $(M/L)^2$:

$$\left\langle M^2 \right\rangle = \sum_{j,k} \left\langle \sigma_j^z \sigma_k^z \right\rangle \xrightarrow{\text{p.b.c.}} L \sum_{r=0}^{L-1} \left\langle \sigma_1^z \sigma_{1+r}^z \right\rangle . \tag{9}$$

In the second step we use the translation invariance of the system under periodic boundary conditions to reduce the complexity of computing this quantity. In the ferromagnetic phase, $\langle (M/L)^2 \rangle$ will saturate to a constant, but will decrease to zero (somewhat weakly) with system size L in the paramagnetic phase, providing a quantitative distinction.

Rather than calculating $\langle M^2 \rangle$, it is actually more instructive to study the correlation function $C^{zz}(r) \equiv \left\langle \sigma_1^z \sigma_{1+r}^z \right\rangle$ which is used in the second definition above. The asymptotic scaling of $C^{zz}(r)$ displays a characteristic form which is indicative of the phase: namely, the persistence of a strictly non-zero value even for $r \to \infty$ which identifies the magnetically ordered phase, contrasted with exponential decay to zero in the disordered phase. In addition, the decay of the correlation function takes a different form at the critical point itself, a fact which is closely related to the closing of the excitation gap (this topic is investigated in Sec. 4.6). From the scaling of the correlation function one can define another order parameter given by $\sqrt{\left\langle \sigma_1^z \sigma_{L/2}^z \right\rangle}$, which provides a single quantity which either saturates or monotonically decreases with increasing system size, corresponding to the presence or absence of magnetic order. (Note that in a periodic system the longest distance is $L/2$ because correlations can develop going in either direction on the chain, which is a circle.)

3 Exact diagonalization of many-body Hamiltonians

Exact diagonalization (ED) is a numerical method which entails constructing an explicit representation of the Hamiltonian operator in the Hilbert space $\mathcal{H}$, which one can diagonalize using standard numerical linear algebra tools. The primary advantage of ED is that it can be applied to any system in essentially the same way. The impediment to using ED is that it quickly becomes infeasible due to the exponential growth in dimension of the Hilbert space of a many-body system.

3.1 Dense diagonalization methods

The most straightforward approach to ED is to simply represent the entire Hamiltonian as a dense matrix. To do so requires storing $\dim(\mathcal{H})^2 = \left(2^L\right)^2$ elements, a cost which quickly becomes prohibitive: already for $L = 16, H$ requires 34 GB of memory in double precision floats. In order to explicitly generate the Hamiltonian one can simply compute all $\left(2^L\right)^2$ matrix elements as $H_{\alpha\beta} \equiv \langle e_\alpha | H | e_\beta \rangle$ using, e.g., the mapping from spin configurations to basis states described in (1). The Hermiticity of H $\left(H_{\beta\alpha} = H^*_{\alpha\beta}\right)$ can reduce the memory cost by a factor of roughly 2 . One can also use diagonalization routines optimized for Hermitian matrices, or for real symmetric matrices if all matrix elements $H_{\alpha\beta}$ are real, which significantly improves algorithmic performance.

3.2 Sparse matrix representation

Written in the basis (1), nearly all of the matrix elements of H will turn out to be 0 . To understand this, consider the structure of the terms comprising the Hamiltonian, described in Sec. 2.2. Only very specific types of states are connected by each of the local terms. Consider, for example, $\sigma^z_j \sigma^z_{j+1}$, whose explicit form is shown in (6). To have a nonzero matrix element for this term, two basis states $|e_\alpha\rangle$ and $|e_\beta\rangle$ must be identical on all sites where the operator acts trivially, in order to be connected by all $L-2$ identities $\mathbb{I}_k, k \neq j, j+1$. Because $\sigma^z_j \sigma^z_{j+1}$ acting in the local Hilbert space is also diagonal, it connects only basis states which are identical on all sites including j and $j+1$, but

with the crucial addition of a sign to the matrix element for certain two-spin configurations:

$$\sigma_j^z \sigma_{j+1}^z = \mathbb{I}_1 \otimes \cdots \otimes \mathbb{I}_{j-1} \otimes \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{j,j+1} \otimes \mathbb{I}_{j+2} \otimes \cdots \otimes \mathbb{I}_L \quad (10)$$

On the other hand, because the σ_j^x matrix is off-diagonal in this basis, these terms connect states related by a single spin-flip. Consider the state $|e_\beta\rangle$: we can ask with which state it has nonzero matrix element $\langle e_\alpha | \sigma_j^x | e_\beta \rangle$. One can quickly obtain α by using the bitwise XOR operation to flip the j th bit in the stored binary representation of β.

Evidently, the Hamiltonian in this basis is more appropriately represented as a sparse matrix, avoiding having to store and manipulate an overwhelming number of zeros. A local Hamiltonian like H has roughly $L \times 2^L$ nonzero terms, an enormous reduction compared with $\left(2^L\right)^2$. One can represent the sparse matrix by identifying the basis states connected by each term in H and

tabulating all 3 -tuples $(\alpha, \beta, H_{\alpha\beta})$ of nonzero matrix elements. The challenge of the sparse matrix approach lies predominantly in creating a bookkeeping system for all of the basis states and correctly identifying every nonzero matrix element of H.

There is no single standard format for sparse matrix representation. The table of triplets described above is known as "COO" format and is often convenient for initialization but inefficient when used in numerical routines. A common design pattern is to initialize a sparse matrix in COO
format, then convert it to a more computationally friendly option before calling a diagonalization routine. Ultimately one needs to consult the relevant documentation, e.g., Ref. [4].

3.3 Lanczos method

With this improvement in storage, we can tackle significantly larger systems, but of course the difficulty of computing the eigenstates continues to increase with the system size. It becomes impossible to solve for the entire spectrum of a sparse matrix while maintaining the improved storage, but one can find

a few eigenstates at either end of the spectrum or, with some more effort, from somewhere in the middle via "shift-and-invert" methods. A variety of iterative schemes exist, but they are not nearly as clean and robust as dense diagonalization. In particular, matrices with high condition number (the ratio of the largest-magnitude eigenvalue to the smallest) may present difficulties with convergence. One state-of-the-art sparse diagonalization library is ARPACK, which is written in Fortran but has interfaces in Python and Julia, and several in C++.

The algorithm generally favored in physics applications is one devised by Lanczos as an improvement over the basic power method. The power method is based on the observation that a random initial state $|\psi\rangle$ can be decomposed in the eigenbasis of H, and, if H is applied to $|\psi\rangle$, the component in the direction of the leading eigenvector of H (i.e., having the eigenvalue of largest magnitude) will increase relative to the other eigenvectors having eigenvalues of smaller magnitude. That is, if $H = \sum_\alpha \epsilon_\alpha |v_\alpha\rangle \langle v_\alpha|$ and $|\psi\rangle = \sum_\alpha c_\alpha |v_\alpha\rangle$,

$$H|\psi\rangle = \sum_\alpha \epsilon_\alpha c_\alpha |v_\alpha\rangle , \quad \text{and } \frac{|\epsilon_1 c_1|^2}{\sum_{\alpha>1} |\epsilon_\alpha c_\alpha|^2} > \frac{|c_1|^2}{\sum_{\alpha>1} |c_\alpha|^2} \tag{11}$$

with the eigenvalues sorted so that $|\epsilon_1| > |\epsilon_\alpha|$ for all $\alpha > 1$. This also requires that the leading eigenvector be unique. The power method can be described recursively by $|\psi_{k+1}\rangle = H|\psi_k\rangle / \|H|\psi_k\rangle\|$, which in the limit $k \to \infty$ converges to the extremal eigenvector. (This is the same argument made to claim the dominance of the maximal eigenvector of the transfer matrix in the thermodynamic limit of a 1d stat mech model.) However, in practice the convergence of this method turns out to be extremely slow, and its performance strongly depends on the gap to the first excited state.

Because up to normalization the power method is simply the application of H^k - the k-th power of the Hamiltonian - to an arbitrary state, one might wonder whether instead any degree k polynomial of the Hamiltonian could be found having superior convergence, with no extra work since we already compute $H^l|\psi\rangle$ for $0 \le l \le k$. The span of these states in $\mathcal{H}$ is called the Krylov subspace, and the Lanczos algorithm entails finding the basis within this subspace that simplifies H as much as possible. For a precise description of the basic Lanczos method including pseudocode, see Sandvik's lecture notes, Ref. [1], or the textbook by Golub and van Loan, Ref. [2].

In this assignment you are not required to implement your own Lanczos algorithm and will make use of some standard sparse matrix library. How-

ever, if you are interested in this type of calculation, writing your own is an instructive experience. One further algorithmic simplification (described also by Sandvik) removes the necessity of directly representing the Hamiltonian at all. It turns out the Lanczos algorithm does not even need to know H : it's only necessary to be able to perform $|\psi\rangle \mapsto H|\psi\rangle$. Thus, most implementations allow you to provide a function pointer that computes the application of H to a vector rather than supplying a sparse matrix in any format. Using such a matrix-free specification of the Hamiltonian can be even more efficient.

3.4 Restriction to symmetry sectors

Global symmetries of a Hamiltonian can be used to reduce the effort required for ED, in some cases drastically. Here we discuss implementation of the global symmetry of the Ising model by again considering the states connected to each other by the Hamiltonian, as in Sec. 3.2.

The Ising symmetry operator is $U_x = \prod_j \sigma_j^x$, with eigenvalues ± 1. Accordingly, the total Hilbert space decomposes into the direct sum of the corresponding eigenspaces $\mathcal{H} = \mathcal{H}_- \oplus \mathcal{H}_+$. Since U_x commutes with H, the Hamiltonain preserves (i.e., acts without mixing) $\mathcal{H}_-$and $\mathcal{H}_+$. It is easiest to access these eigenspaces by performing a basis rotation, from the σ^z eigenstates in (1), which use local $|\uparrow\rangle$ and $|\downarrow\rangle$ basis, to the σ^x basis built from states we denote $|+\rangle = |+\hat{x}\rangle$ and $|-\rangle = |-\hat{x}\rangle$.

The accounting of the states that you have already done can be used as-is, as so far this is only a trivial relabeling. The distinction comes in the explicit form of the terms in the Hamiltonian (4), which no longer have the matrix elements described in Sec. 3.2. Instead, one finds that σ_j^x is diagonal in this basis (which is indeed why we chose it), and σ_j^z is a spin-flip operator. Thus, the Hamiltonian connects states that either have the same number of $|+\rangle$-spins and $|-\rangle$-spins, or differ by two. U_x counts the parity, and odd- and even-parity sectors are not connected by H.

The reduction of the Hilbert space is accomplished by separating the basis states according to their parity. Then one can set up an "odd" sector problem and an "even" sector problem by restricting the basis, each having Hilbert space dimension reduced by a factor of two. The energy eigenvalues in each sector will match those found in the full Hilbert space, with wavefunctions that are related by a unitary rotation. Because the cost of typical dense diagonalization routines scales as $O(N^3)$ in the linear dimension N of

the matrix, reducing N by half obtains an eightfold speedup, which gives a fourfold overall reduction (as the routine must run twice).

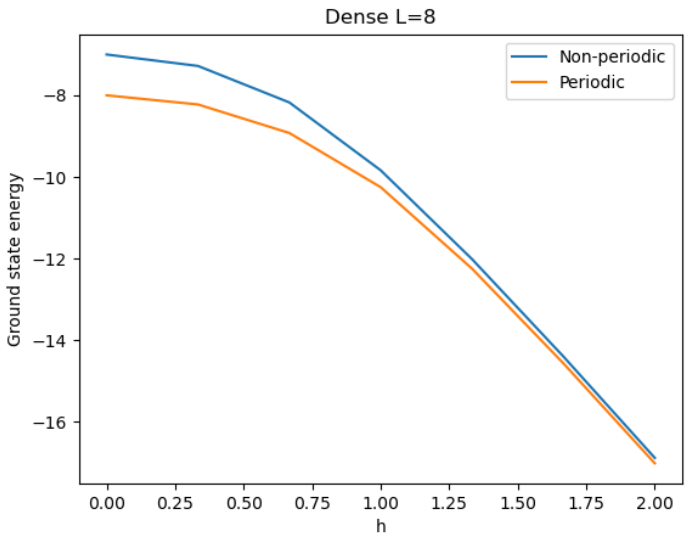
4 Assignment: ED study of quantum Ising model

For simplicity, in the following we set the parameter $J = 1$ in (4) and vary h.

4.1 Dense ED

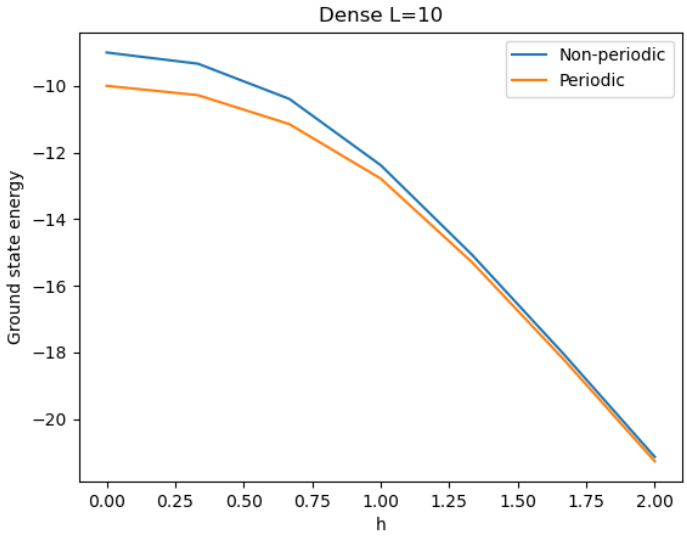
Generate the quantum Ising Hamiltonian (4) as a dense matrix and call an explicit diagonalization routine for the entire spectrum for system sizes $L = 8, 10, 12, 14$, and for a range of values of h. Plot the ground state energy as a function of h for the various L. Compare the open systems with periodic ones for the same parameters - how does each phase react to the boundaries?

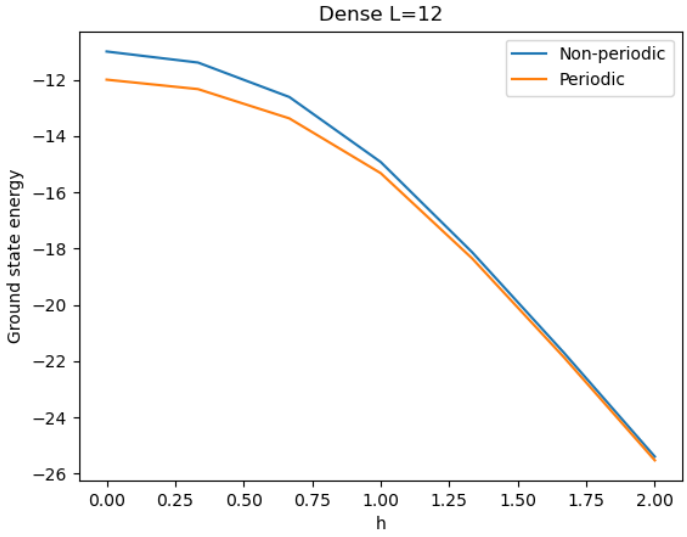
0.0.1

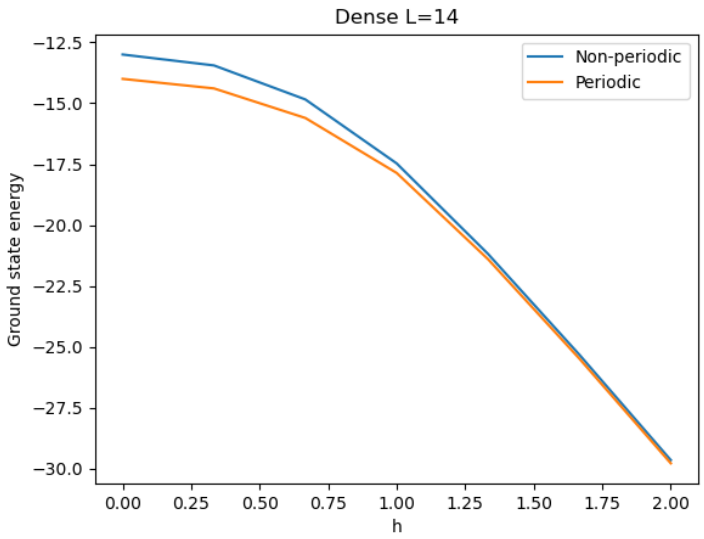


The paramagnetic phase (with the larger h value) does not care whether the boundaries are open or periodic because the assortment of spins is random and ways. For the ferromagnetic phase, there is a clear preference for the periodic boundary conditions, because this allows for an extra interaction term that minimizes the total energy even more.

```
sigma_x = np.array([[0, 1], [1, 0]])
sigma_z = np.array([[1, 0], [0, -1]])
I = np.identity(2)

def tensor_product(matrices):
    """Calculate the tensor product of a list of matrices."""
    result = matrices[0]
    for matrix in matrices[1:]:
        result = np.kron(result, matrix)
    return result

def open_dense_hamiltonian_explicit(L, h, J=1):
    # Initialize Hamiltonian to zero matrix
    H = np.zeros((2**L, 2**L))

    # Interaction term
    for i in range(L - 1):  # Add periodic term at the end if
    periodic
        matrices = [I] * L  # Start with identity matrices
```


Dense L=10
Non-periodic
Periodic
Ground state energy
−10
−12
−14
−16
−18
−20
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
h


Dense L=12
Non-periodic
Periodic
Ground state energy
−12
−14
−16
−18
−20
−22
−24
−26
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
h




```
        matrices[i] = sigma_z  # Apply sigma_z at position i
        matrices[(i + 1)] = sigma_z  # Apply sigma_z at position (i+1) modulo L for periodic
        H += -J * tensor_product(matrices)

    # Transverse field term
    for i in range(L):
        matrices = [I] * L  # Start with identity matrices
        matrices[i] = sigma_x  # Apply sigma_x at position i
        H += -h * tensor_product(matrices)

    return H

def periodic_dense_hamiltonian_explicit(L, h, J=1):
    # Initialize Hamiltonian to zero matrix
    H = np.zeros((2**L, 2**L))

    # Interaction term
    for i in range(L):  # Add periodic term at the end if periodic
        matrices = [I] * L  # Start with identity matrices
        matrices[i] = sigma_z  # Apply sigma_z at position i
        matrices[(i + 1) % L] = sigma_z  # Apply sigma_z at position (i+1) modulo L for periodic
        H += -J * tensor_product(matrices)
```

```
41
42      # Transverse field term
43      for i in range(L):
44          matrices = [I] * L  # Start with identity matrices
45          matrices[i] = sigma_x  # Apply sigma_x at position i
46          H += -h * tensor_product(matrices)
47
48      return H
49
50 L = [8, 10, 12, 14]
51 h_range = np.linspace(0, 2, 7)  # Generates 8 values of h
       from 0 to 2
52
53 for l in L:
54      plt.figure()
55      plt.title(f'Dense L={l}')
56      open_ground_state_energies = []  # To store ground state
       energies for different h
57      periodic_ground_state_energies = []  # To store ground
       state energies for different h
58      for hi in h_range:
59          # for the given hi, comupte gs energy for both
       periodic and non-periodic
60          H = open_dense_hamiltonian_explicit(l, hi)
61          print(f'H: hi={hi}\n{H}')
62          H_periodic = periodic_dense_hamiltonian_explicit(l,
       hi)
63          # Compute the ground state energy using the lowest
       eigenvalue
64          open_ground_state_energies.append(min(np.linalg.
       eigvalsh(H)))
65          periodic_ground_state_energies.append(min(np.linalg.
       eigvalsh(H_periodic)))
66      # plot the ground state energy as a function of h for
       both periodic and non-periodic
67      plt.plot(h_range, open_ground_state_energies, label='Non-
       periodic')
68      plt.plot(h_range, periodic_ground_state_energies, label='
       Periodic')
69      plt.xlabel('h')
70      plt.ylabel('Ground state energy')
71      plt.legend()
72      # save a tout for the given value of L
73      plt.savefig(f'dense_ising_model_L{l}.png')
```

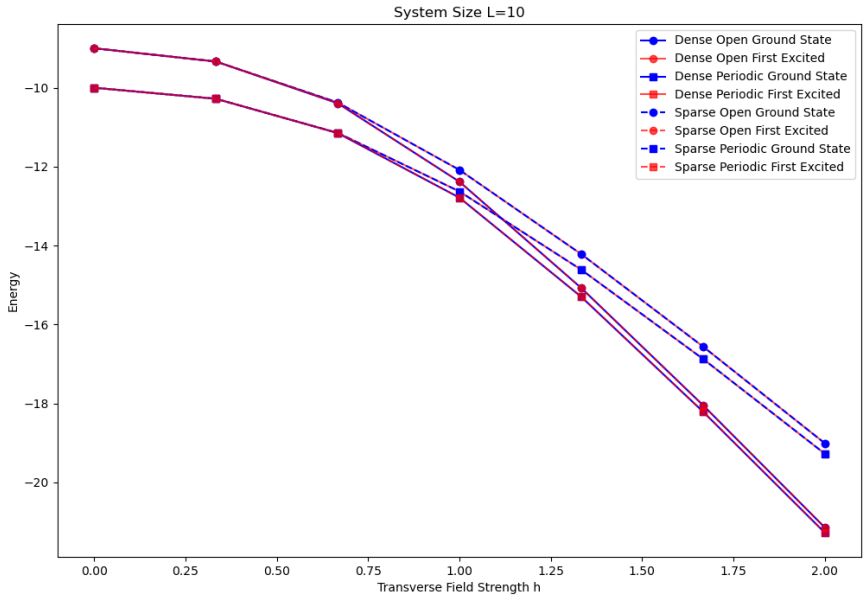
0.1 4.2 Sparse ED

Construct the same Hamiltonian as a sparse matrix and use a sparse matrix diagonalization routine provided by a standard library. Obtain the ground state and a few excited states, and for small system sizes verify your results against the dense ED solution. How large of a system can you push the sparse diagonalization routine to solve?

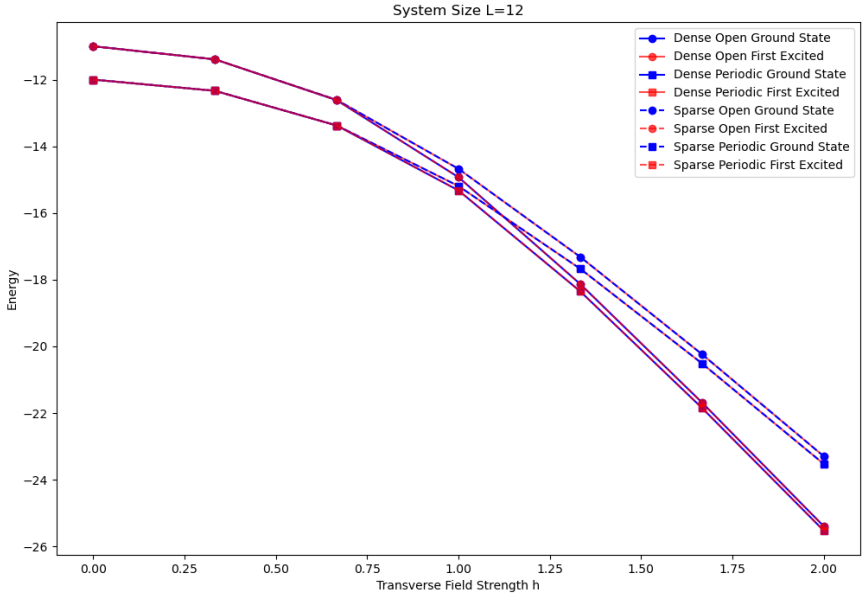
Optional. Try implementing your own Lanczos routine.

0.1.1 Answer

As can be seen from the plots 0.1.1 and 0.1.1, the energies for the GS and first excited states are identical for the L=10 and L=12 systems. The highest system size that I was able to get up to is L=20.




```python
def sparse_hamiltonian(L, h, J=1, periodic=False):
    row = []
    col = []
    data = []
    # loop over the index i
    for i in range(2**L):
        # turn that index into a binary string
        bi = binary_string(i, L)

        # to the diagonal element first
        total_interaction = 0
        # initialize the loop range based on the boundary conditions
        loop_range = L if periodic else L - 1
        for k in range(loop_range):
            # Handle periodic boundary by wrapping the index
            next_k = (k + 1) % L if periodic else k + 1
            # Add neighbor interaction based on the spin values
            total_interaction += -J * (1 if bi[k] == bi[next_k] else -1)
```

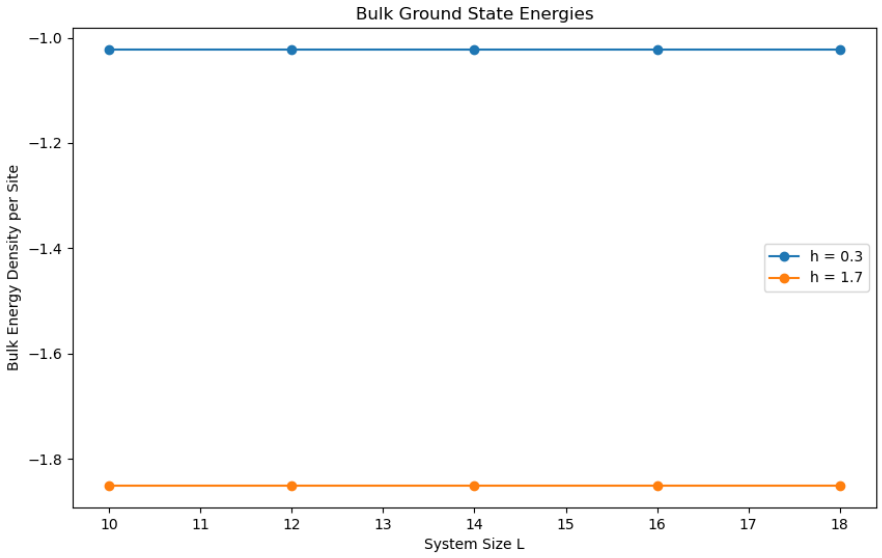

System Size L=12
Dense Open Ground State
Dense Open First Excited
Dense Periodic Ground State
Dense Periodic First Excited
Sparse Open Ground State
Sparse Open First Excited
Sparse Periodic Ground State
Sparse Periodic First Excited
Energy
−12
−14
−16
−18
−20
−22
−24
−26
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Transverse Field Strength h

```
        row.append(i)
        col.append(i)
        data.append(total_interaction)

        # now consider the off-diagonal elements connected by a single spin flip
        # make a function that takes the bit i and returns a list of basis states connected by a single flip
        def flip(i):
            connected_bases_states = []
            for j in range(L):
                flip_i = i ^ (1 << j)
                connected_bases_states.append(flip_i)
            return connected_bases_states

        # loop over the connected basis states
        for flip_i in flip(i):
            row.append(i)
            col.append(flip_i)
            data.append(-h)

    # Create a sparse matrix from the lists
    H = scipy.sparse.coo_matrix((data, (row, col)), shape=(2**L, 2**L))
    return H
```

0.2 4.3 Study of convergence with system size

For representative values of h inside each phase (e.g., $h = 0.3$ in the ferromagnetic phase and $h = 1.7$ in the paramagnet), study the L dependence of the ground state energy per site, $E_{\text{gs}}(L)/L$, for systems with both periodic and open boundary conditions. Comment on the approach to the thermodynamic limit $L \to \infty$ for the two types of boundaries.

It should be clear that periodic boundaries are preferred in order to minimize finite-size effects. However, sometimes open boundary conditions are preferred for technical reasons. An important example is the tensor network representation we will implement later in the course. One can mitigate the effects of the boundaries and better estimate the bulk energy per site using the following trick: for the system with open boundary conditions, plot the values $[E(L = 10) - E(L = 8)]/2$, $[E(L = 12) - E(L = 10)]/2$, etc. These quantities can be thought of as ground state energy per site for the two sites

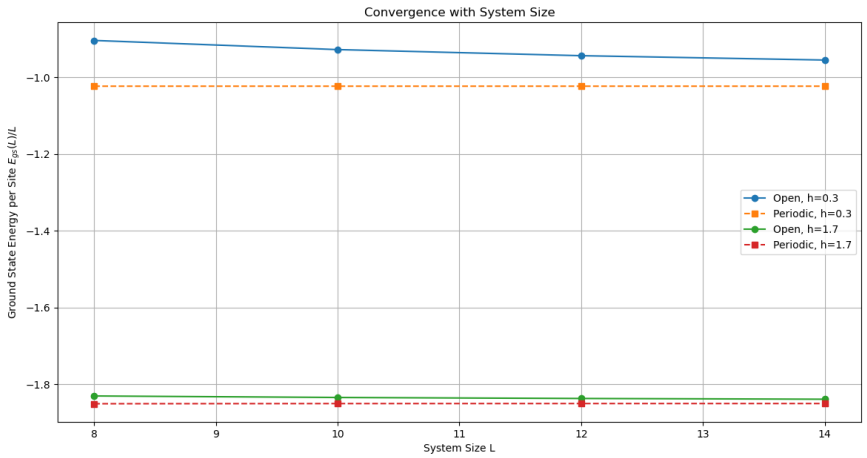


added in the middle, which feel reduced boundary effects.

0.2.1 Answer

As can be seen from the plot 0.2.1 this definition for the bulk energy per site is constant across L.

```
h_values = [0.3, 1.7]
L_range = range(8, 20, 2)  # From L=8 to L=18, in steps of 2

plt.figure(figsize=(10, 6))
plt.title('Bulk Ground State Energies')

# Dictionary to store energies for each h value
energies_per_h = {h: [] for h in h_values}

for L in L_range:
    for h in h_values:
        H_open = sparse_hamiltonian(L, h, periodic=False).
    asformat('csr')
        E_open = scipy.sparse.linalg.eigsh(H_open, k=1, which
    ='SA', return_eigenvectors=False)[0]
        energies_per_h[h].append((L, E_open))
```

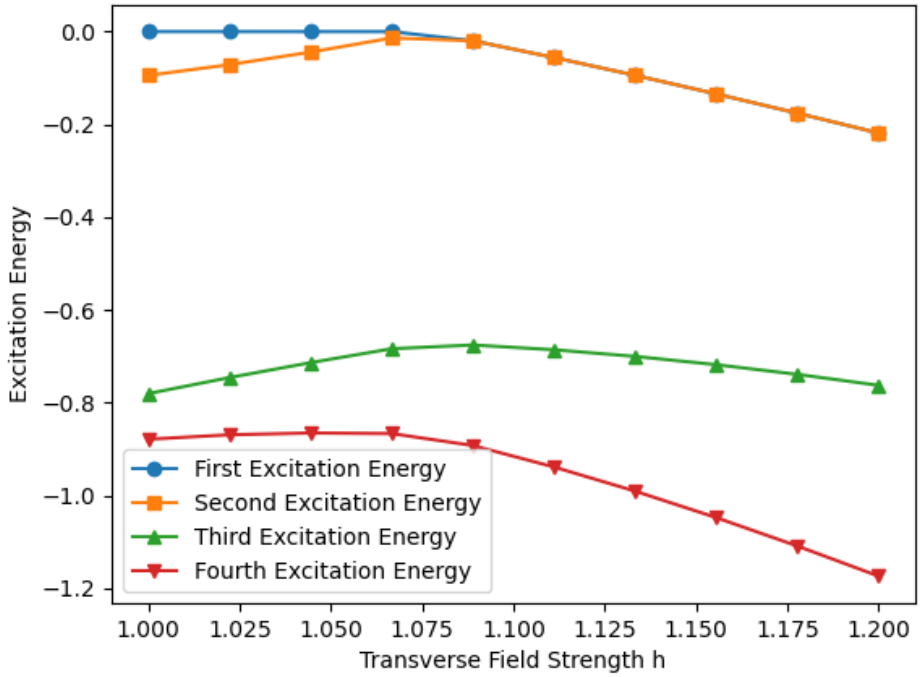



```
15
16 # Now, plot the energy differences for each h value
17 for h in h_values:
18     Ls, Es = zip(*energies_per_h[h])  # Unpack the energies
    and L values
19     # Calculate the differences and plot
20     energy_differences = [(Es[i] - Es[i-1]) / 2 for i in
    range(1, len(Es))]
21     plt.plot(Ls[1:], energy_differences, 'o-', label=f'h = {h
    }')
22
23 plt.xlabel('System Size L')
24 plt.ylabel('Bulk Energy Density per Site')
25 plt.legend()
26 plt.savefig('bulk_energy_density.png')
```

Both type of boundaries approach a thermodynamic limit as the system size increases. That is, the periodic systems should be more accurate, but this doesn't matter so much in the thermodynamic limit. One also notices in the convergence plot 0.2.1 that when we have a small transverse field h, the interaction terms dominatet, so the fined size effect is quite pronounced, but this is not the case when we have the larger value for h.

4.4 Finding the quantum phase transition

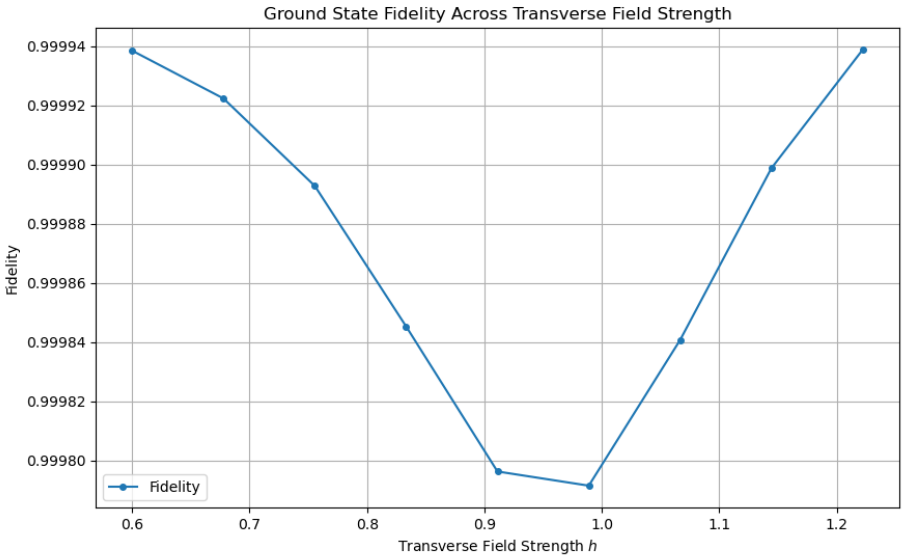
From now on, consider only the chain with periodic boundary conditions. The quantum phase transition can be identified via the closing of the energy gap, as described in Sec. 2.3. Using the sparse eigensolver, plot the excitation energies (relative to the ground state) for a few excited states vs. h for your largest system size. Estimate the critical field h_c from this study and compare with the known value of $h_c = 1$. The excitation gap is expected to vanish as $\Delta \sim |h - h_c|^{\nu}$. Can you determine the exponent ν from your data?



0.2.2 Answer

I estimate the critical field to be $h_c \approx 1.075$ for L=16 from 0.2.2. I noticed that this value h_c approaches 1 as the system size grows. The excitation gaps seem to vanish linearly as seen in the figure, so I estimate the exponent ν to be -1.

Another straightforward technique to find the phase transition is by studying so-called "fidelity." As you scan h, calculate the overlap between the ground states at consecutive h and $h+\delta h$ and plot the fidelity $|\langle\psi_{\rm gs}(h) \mid \psi_{\rm gs}(h+\delta h)\rangle|$ as a function of h. Deep inside either phase, the character of the ground state changes little with changing h, and the fidelity is close to 1 . However, near the transition the character of the ground state changes rapidly and the fidelity exhibits a cusp. The peak height you observe will depend on δh.



0.2.3 Answer

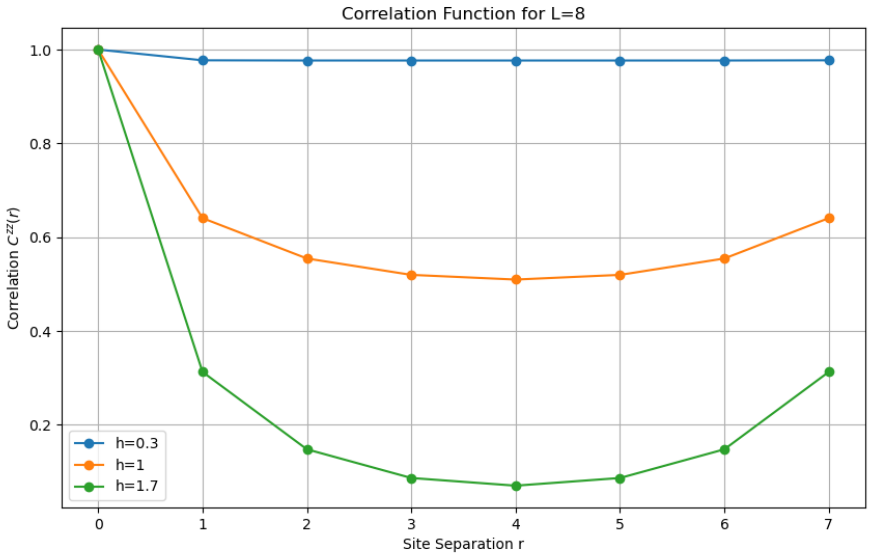
When I set $\delta h = 0.01$, I get the plot in 0.2.3. The fidelity is close to 1 deep inside either phase, but near the transition, the fidelity exhibits a cusp. The peak height I observe depends on δh; for a larger δh, the well is deeper.

4.5 Study of magnetic ordering

One can also use the wavefunctions obtained from diagonalization to measure any observable in the ground state. For representative points in the two phases and also at the critical point, plot $C^{zz}(r)$ as defined in Sec. 2.4 as a function of r for multiple system sizes L and comment on the behavior at large separation r. Also compute the quantity $\langle (M/L)^2 \rangle$ and compare the behavior of this order parameter with the half-chain correlation function values.

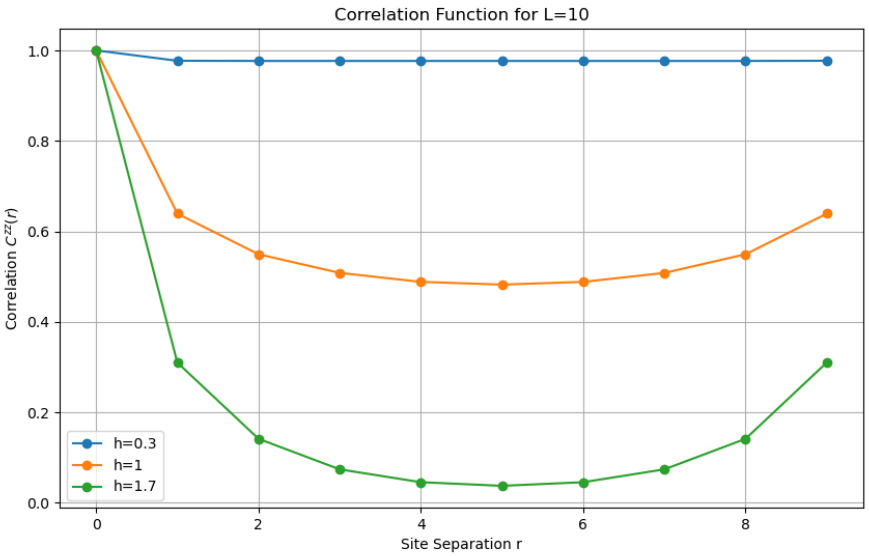
0.2.4 Answer

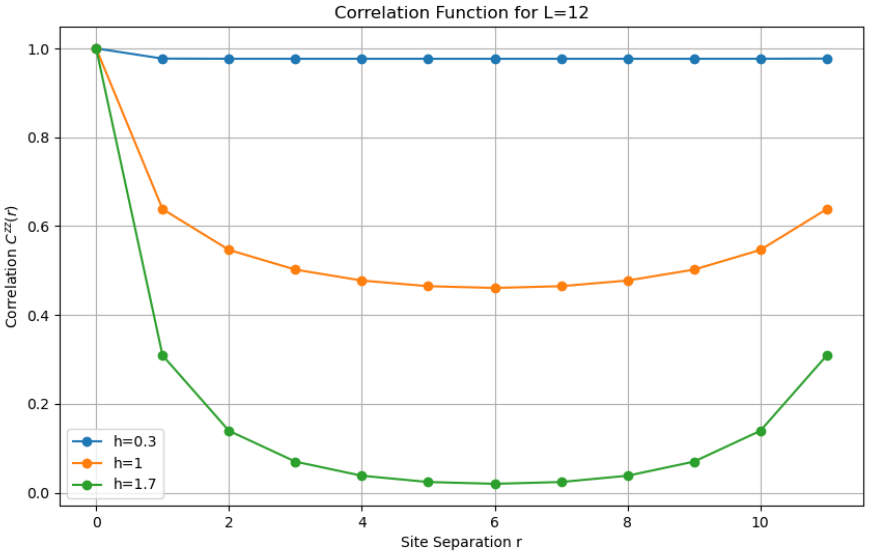
As can be seen in the figure 0.2.4, the correlation function $C^{zz}(r)$ is close to 1 statically for the ordered phase and dynamically undergoes an exponential



decay for the disordered phase. It should be noted that in the disordered phase, the correlation length exponentially decreases until the farthest point from the "origin" of the periodic system is reached and then it starts to increase again.

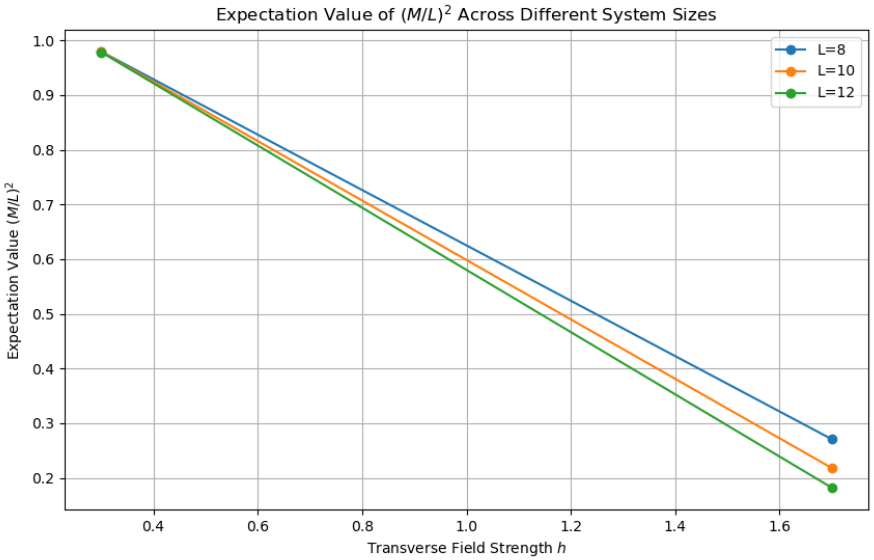
```
def correlation_function(psi_gs, L):
    """Calculate the correlation function of a state."""
    correlations = np.zeros(L)
    for i in range(2**L):
        # Convert the state index to binary
        binary_state = format(i, '0{}b'.format(L))
        for r in range(L):
            # Get the spins at site 1 and site 1+r from the binary representation
            spin1 = 1 if binary_state[0] == '1' else -1  # First site spin
            spin2 = 1 if binary_state[r % L] == '1' else -1  # (1+r)th site spin considering periodic boundary
            # Add the product of the spins to the correlation value, weighted by the probability of this basis state
            correlations[r] += spin1 * spin2 * np.abs(psi_gs[i])**2

```


Correlation Function for L=10
Correlation $C^{zz}(r)$
Site Separation r
h=0.3
h=1
h=1.7
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8


Correlation Function for L=12
Correlation $C^{zz}(r)$
Site Separation r
h=0.3
h=1
h=1.7
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
10

```
    # The correlations array now contains the expectation value of the product of spins at positions 1 and 1+r for all r
    return correlations
# choose multiple system sizes
L_values = [8, 10, 12]
# choose some representative values for h
h_values = [0.3, 1.7]
ground_states = {}  # Dictionary to store ground states and energies for each (h, L) pair

# Calculate ground states and correlation functions for various L and h
for L in L_values:
    ground_states[L] = {}  # Initialize a sub-dictionary for each L
    for h in h_values:
        H = sparse_hamiltonian(L, h, periodic=True)
        psi_gs, E_gs = calculate_ground_state(H)
        ground_states[L][h] = (psi_gs, E_gs)
        # Compute the correlation function for the given ground state
        correlations = correlation_function(psi_gs, L)
        ground_states[L][h] += (correlations,)  # Add correlations to the tuple

# Now ground_states is a nested dictionary with structure {L: {h: (psi_gs, E_gs, correlations)}}
for L in L_values:
    plt.figure(figsize=(10, 6))
    for h in h_values:
        correlations = ground_states[L][h][2]  # Retrieve correlations from the stored tuple
        r_values = range(L)  # Site separations
        plt.plot(r_values, correlations, 'o-', label=f'h={h}')

    plt.title(f'Correlation Function for L={L}')
    plt.xlabel('Site Separation r')
    plt.ylabel('Correlation $C^{zz}(r)$')
    plt.legend()
    plt.grid(True)
    plt.savefig(f'4-5_correlation_L{L}.png')
```

When we plot $\langle (M/L)^2 \rangle$ for some characteristic values of the transverse field



h, we see that for a larger system size, the value is 1 for the ordered phase and close to 0 for the disordered phase, as would be expected. This can be seen in the figure 0.2.4. Inspection shows that at the critical field value of 1, the correlation length value is the same as that of $\langle (M/L)^2 \rangle$; in the L=8 case, this falls around a value of 0.5.

```
def expectation_value_M_squared(psi_gs, L):
    """
    Calculate the expectation value of (M / L)^2 for a ground
    state.
    """
    # Calculate the correlation function
    correlations = correlation_function(psi_gs, L)

    # Compute the expectation value <M^2>
    # Under periodic boundary conditions, <M^2> simplifies to
    L times the sum of all correlations
    M_squared = L * np.sum(correlations)
    return M_squared
# Plot setup
plt.figure(figsize=(10, 6))

# Loop over each L and plot the expectation values for all h
```

```
for L in L_values:
    M_squared_per_L = []  # To store the calculated <(M/L)^2> for each h
    for h in h_values:
        psi_gs, E_gs, _ = ground_states[L][h]
        M_squared = expectation_value_M_squared(psi_gs, L)
        M_squared_per_L.append(M_squared / L**2)

    # Plot for the current L
    plt.plot(h_values, M_squared_per_L, 'o-', label=f'L={L}')

# Enhance the plot
plt.title('Expectation Value of $(M / L)^2$ Across Different System Sizes')
plt.xlabel('Transverse Field Strength $h$')
plt.ylabel('Expectation Value $(M / L)^2$')
plt.legend()
plt.grid(True)
plt.savefig('4-5_M_squared.png')
```

4.6 Optional. Study of critical correlations

Measure the correlator $C^{xx}(r) = \langle \sigma_1^x \sigma_{1+r}^x \rangle$ as a function of r in the ground state of both phases. Observe the long-distance behavior of this quantity; is it what you would have expected based on the discussion of Sec. 2.4. Why do you think you obtained this result?

For correlation functions such as the above, it is useful to instead study the connected correlator $C^{xx}_{\text{conn}}(r) \equiv \langle \sigma_1^x \sigma_{1+r}^x \rangle - \langle \sigma_1^x \rangle \langle \sigma_{1+r}^x \rangle$, which ignores the classical correlations between the two operators. The connected correlations display quasi-long range order at the critical point, meaning that they decay algebraically as a power law with distance. The set of exponents of these power laws is an inherent property of the critical theory. Try to fit the decay of the connected correlations at the critical point to a power-law form and extract the critical exponent associated with $C^{xx}_{\text{conn}}(r)$. Importantly, because you are studying a relatively small periodic system, you will need to plot

$C^{xx}_{\text{conn}}(r)$ against the chord length $\left|\frac{L}{\pi} \sin\left(\frac{\pi r}{L}\right)\right|$ instead of simply r. Repeat this process for $C^{zz}_{\text{conn}}(r)$, extracting this critical exponent as well.

0.3 4.7 Making use of Ising symmetry

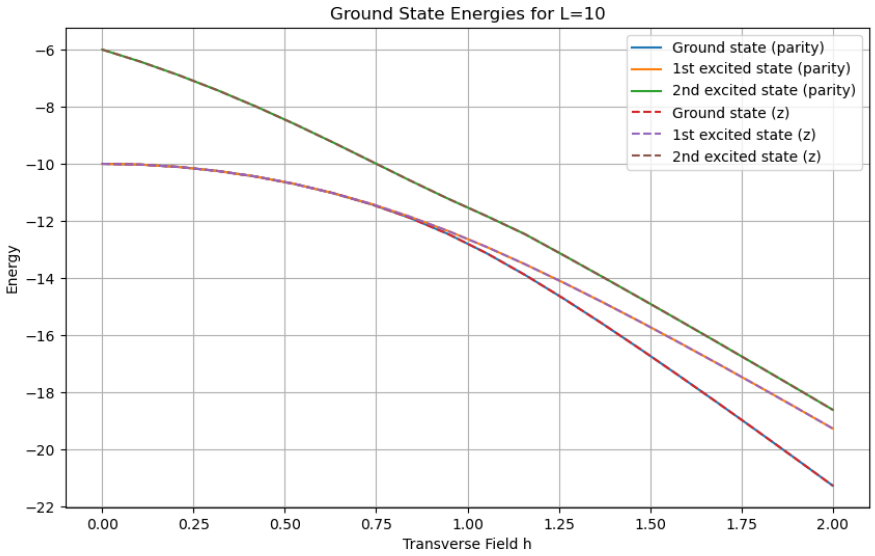
In the previous steps, no use has been made of the Ising symmetry: the fact that unitary $U_x = \prod_j \sigma_j^x$ commutes with the Hamiltonian. Correctly implementing the symmetry reduces the size of the diagonalization problem by a factor of two, which saves memory and gives a fourfold speedup.

Set up a sparse diagonalization of the Ising Hamiltonian in the σ^x basis, utilizing the Ising symmetry as described in Sec. 3.4. As you will no longer be able to exploit the convenient basis (1), the bookkeeping is more challenging here. Obtain a few states in each sector and compare with your previous results. Use the symmetric solver to add a single site to the largest size you were able to solve previously and find the ground state and a few excited states at this size for representative values of h in each phase.

Utilizing symmetries can also improve the accuracy of certain interesting features of the spectrum. For example, in Sec. 2.3 the two ground states in the ordered phase are described as having an energy splitting that is exponentially small in system size. By solving for the ground state within each symmetry sector, try to obtain the dependence of this splitting on system size L.

0.3.1 Answer

As can be seen in figures like 0.3.1 there is a near exact matching of energies obtained from the symmetric solver and the non-symmetric solver.




```python
def sparse_hamiltonian_x_basis(L, h, J=1, periodic=False):
    size = 2**L
    index_even = []  # To track indices of even parity states
    index_odd = []   # To track indices of odd parity states

    # Initialize lists for even and odd parity sectors
    row_even, col_even, data_even = [], [], []
    row_odd, col_odd, data_odd = [], [], []

    def count_ones(n):
        """Helper function to count '1's in binary
   representation."""
        return bin(n).count('1')
    # Populate even and odd indices
    for i in range(size):
        if count_ones(i) % 2 == 0:
            index_even.append(i)
        else:
```

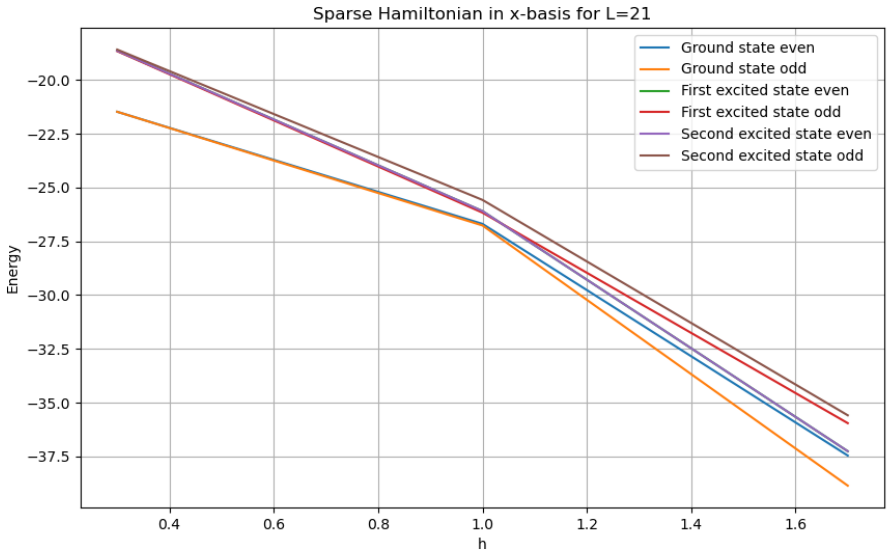
```
            index_odd.append(i)

    # Map original indices to new compacted indices
    map_even = {idx: n for n, idx in enumerate(index_even)}
    map_odd = {idx: n for n, idx in enumerate(index_odd)}

    # Construct the Hamiltonian for each state
    for i in range(size):
        x_basis_state = binary_string(i, L)
        parity = count_ones(i) % 2  # Calculate parity of the
    state

        # Select the correct lists and mapping based on the
    parity
        row = row_even if parity == 0 else row_odd
        col = col_even if parity == 0 else col_odd
        data = data_even if parity == 0 else data_odd
        mapping = map_even if parity == 0 else map_odd

        # Diagonal contributions from ^x (magnetic field)
        row.append(mapping[i])
        col.append(mapping[i])
        data.append(-h * (x_basis_state.count('1') -
    x_basis_state.count('0')))

        # Off-diagonal contributions from ^z ^z interaction
        loop_range = L if periodic else L - 1
        for j in range(loop_range):
            flipped_index = i ^ (1 << j) ^ (1 << ((j + 1) % L
    ))
            if flipped_index in mapping:  # Check if flipped
    index is in the same parity
                row.append(mapping[i])
                col.append(mapping[flipped_index])
                data.append(-J)

    # Create sparse matrices for each parity sector
    H_even = sp.coo_matrix((data_even, (row_even, col_even)),
    shape=(len(index_even), len(index_even)), dtype=float).
    tocsr()
    H_odd = sp.coo_matrix((data_odd, (row_odd, col_odd)),
    shape=(len(index_odd), len(index_odd)), dtype=float).tocsr
    ()

    return H_even, H_odd
```

```
54
55 # Example usage:
56 L = [8, 10, 12]
57 h_values = np.linspace(0, 2.0, 20)  # Range of h values to
       scan
58
59 for L_val in L:
60     plt.figure(figsize=(10, 6))
61     plt.title(f'Ground State Energies for L={L_val}')
62
63     # Lists to store the lowest three unique energies for
       each h value
64     lowest_three_parity = []
65     lowest_three_z_basis = []
66
67     for h_val in h_values:
68         H_even, H_odd = sparse_hamiltonian_x_basis(L_val,
       h_val, periodic=True)
69         H_z = sparse_hamiltonian(L_val, h_val, periodic=True)
70
71         # Diagonalize and collect the lowest three energies
       for even sector
72         eigvals_even = scipy.sparse.linalg.eigsh(H_even, k=3,
        which='SA', return_eigenvectors=False)
73
74         # Diagonalize and collect the lowest three energies
       for odd sector
75         eigvals_odd = scipy.sparse.linalg.eigsh(H_odd, k=3,
       which='SA', return_eigenvectors=False)
76
77         # Combine and sort the eigenvalues from even and odd
       sectors, then take the lowest three
78         combined_parity_eigvals = np.union1d(eigvals_even,
       eigvals_odd)
79         combined_parity_eigvals.sort()
80
81         # Append to the list of lowest three energies for the
        parity sectors
82         lowest_three_parity.append(combined_parity_eigvals
       [:3])
83
84         # Diagonalize and collect the lowest four energies
       for the z-basis for comparison
85         eigvals_z = scipy.sparse.linalg.eigsh(H_z, k=4, which
       ='SA', return_eigenvectors=False)
```

```
        eigvals_z.sort()

        # Append to the list of lowest three energies for the z-basis
        lowest_three_z_basis.append(eigvals_z[:3])

    # Reshape the lists for plotting
    lowest_three_parity = np.array(lowest_three_parity).T  # Transpose to match h_values shape
    lowest_three_z_basis = np.array(lowest_three_z_basis).T

    # Plot for parity sectors
    plt.plot(h_values, lowest_three_parity[0], label='Ground state (parity)')
    plt.plot(h_values, lowest_three_parity[1], label='1st excited state (parity)')
    plt.plot(h_values, lowest_three_parity[2], label='2nd excited state (parity)')

    # Plot for z-basis; using dashed lines for distinction
    plt.plot(h_values, lowest_three_z_basis[0], label='Ground state (z)', linestyle='--')
    plt.plot(h_values, lowest_three_z_basis[1], label='1st excited state (z)', linestyle='--')
    plt.plot(h_values, lowest_three_z_basis[2], label='2nd excited state (z)', linestyle='--')

    plt.xlabel('Transverse Field h')
    plt.ylabel('Energy')
    plt.legend()
    plt.grid(True)
    plt.savefig(f'4-6_L{L_val}_energies.png')
```

We also did this for L=21 in 0.3.1.

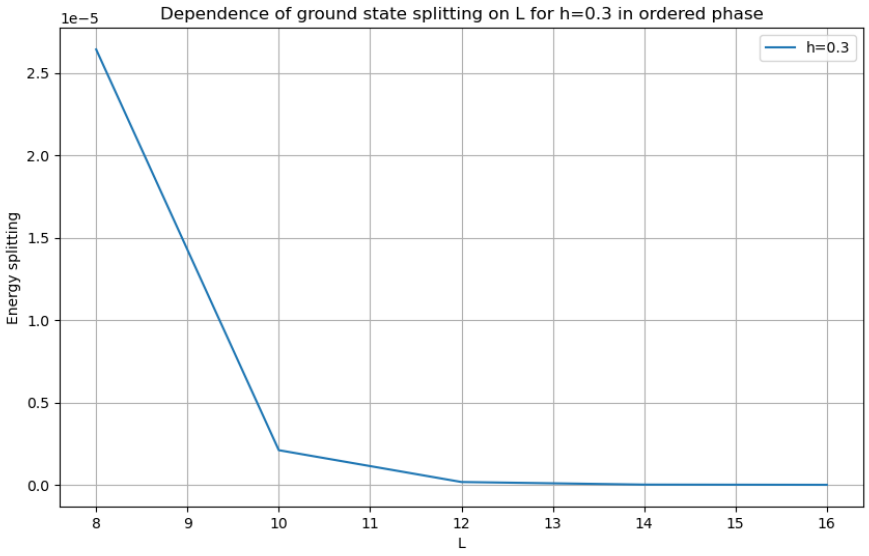
```
L = 21
h_vals = [0.3, 1, 1.7]  # This should probably be h_vals to avoid confusion with h in plt.plot
energies_ground_even = []
energies_ground_odd = []
energies_first_excited_even = []
energies_first_excited_odd = []
energies_second_excited_even = []
energies_second_excited_odd = []

plt.figure(figsize=(10, 6))
```




```
plt.title(f'Sparse Hamiltonian in x-basis for L={L}')
for h_val in h_vals:
    H_even, H_odd = sparse_hamiltonian_x_basis(L, h_val, periodic=True)
    eigvals_even, _ = scipy.sparse.linalg.eigsh(H_even, k=3, which='SA')
    eigvals_odd, _ = scipy.sparse.linalg.eigsh(H_odd, k=3, which='SA')

    energies_ground_even.append(eigvals_even[0])
    energies_ground_odd.append(eigvals_odd[0])

    energies_first_excited_even.append(eigvals_even[1])
    energies_first_excited_odd.append(eigvals_odd[1])

    energies_second_excited_even.append(eigvals_even[2])
    energies_second_excited_odd.append(eigvals_odd[2])

# Now plot using the accumulated lists
plt.plot(h_vals, energies_ground_even, label='Ground state even')
plt.plot(h_vals, energies_ground_odd, label='Ground state odd')
```

```
plt.plot(h_vals, energies_first_excited_even, label='First excited state even')
plt.plot(h_vals, energies_first_excited_odd, label='First excited state odd')
plt.plot(h_vals, energies_second_excited_even, label='Second excited state even')
plt.plot(h_vals, energies_second_excited_odd, label='Second excited state odd')

plt.xlabel('h')
plt.ylabel('Energy')
plt.legend()
plt.grid(True)
plt.savefig('4-6_parity_resolved_energies.png')
```

Then we made a plot for the splitting as a function of system size in the ordered phase.




```
L =[8, 10, 12, 14, 16]
h = 0.3
plt.figure(figsize=(10, 6))
plt.title(f'Dependence of ground state splitting on L for h={h} in ordered phase')
```

```
energies = []
excitation_energy = []
for L_val in L:
    H_even, H_odd = create_sparse_hamiltonian_x_basis(L_val, 1, h)
    eigvals_even, _ = scipy.sparse.linalg.eigsh(H_even, k=2, which='SA')
    eigvals_odd, _ = scipy.sparse.linalg.eigsh(H_odd, k=2, which='SA')
    # append all of the energies to a list
    energies.append([eigvals_even[0], eigvals_odd[0]])
    # energies.append([eigvals_even[1], eigvals_odd[1]])
    energies.sort()
    # determine the excitation energy
    excitation_energy.append(energies[0][1] - energies[0][0])

# fought the excitation and energy for each the value of L
plt.plot(L, excitation_energy, label=f'h={h}')
plt.xlabel('L')
plt.ylabel('Energy splitting')
plt.legend()
plt.grid(True)
plt.savefig('4-6_L_dependence.png')
```

5 Optional. Heisenberg antiferromagnetic chain

The Heisenberg model also simulates the magnetic behavior of spin- 1/2 degrees of freedom on a lattice, but with perhaps a more realistic spin-spin term adding $\sigma^x - \sigma^x$ and $\sigma^y - \sigma^y$ interactions to that of the Ising model. Here we do not apply the transverse magnetic field. The quantum Heisenberg Hamiltonian is

$$H = J \sum_{j=1}^{L-1} \boldsymbol{\sigma}_j \cdot \boldsymbol{\sigma}_{j+1} = J^{L-1} \left(\sigma_j^x \sigma_{j+1}^x + \sigma_j^y \sigma_{j+1}^y + \sigma_j^z \sigma_{j+1}^z \right) \tag{12}$$

This model has a continuous O(3) spin-rotation symmetry (because H is symmetric in the x, y, and z directions, we may rotate however we want in this 3 -dimensional space and the theory is invariant), so one might expect that the model hosts gapless degrees of freedom. This is correct, but we owe to Haldane the surprising discovery that if the local degrees of freedom have integer total spin, the ground state of H is actually gapped.

5.1 Diagonalization within sectors of fixed $S^z_{\mathbf{tot}}$

For the purposes of numerical solution, the Heisenberg model has a very useful symmetry, which is conservation of total spin in the z-direction: $S^z_{\text{tot}} = \sum_j S^z_j$. It is not immediately evident that this operator commutes with the Hamiltonian (12). To make the symmetry manifest, one rewrites the exchange interaction term using the spin-raising and lowering operators $S^+_j$and S^-_j, where

$$2S^{\pm}_j = \sigma^x_j \pm i\sigma^y_j \tag{13}$$

Then $S^+|\uparrow\rangle = 0, S^+|\downarrow\rangle = |\uparrow\rangle$, and $S^-|\uparrow\rangle = |\downarrow\rangle, S^-|\downarrow\rangle = 0$. Some algebra demonstrates that

$$2\left(S^+_j S^-_{j+1} + S^-_j S^+_{j+1}\right) = \sigma^x_j \sigma^x_{j+1} + \sigma^y_j \sigma^y_{j+1}. \tag{14}$$

Replacing the terms in the Hamiltonian (12) gives a form which manifestly commutes with S^z_{tot} . (As an aside, this is also the first step in the famous Jordan-Wigner transformation, which maps these qubit degrees of freedom to interacting spinless fermions hopping on a one-dimensional wire.)

Contrary to the case of the Ising model, here the σ^z basis is appropriate for implementation of the symmetry. Additionally, there are many more eigenvalues of S^z_{tot} , and the dimension of each
eigenspace is greatly reduced, by about a factor of L. Solving for the ground state in multiple selection sectors gives a straightforward way of computing more diverse excited states than would otherwise be possible. However, in this example the bookkeeping required when identifying basis states of the full Hilbert space with those specific to a particular sector is more detailed than in the comparatively simple Ising case. To check your solutions, compare your results against a simpler setup that does not use S^z conservation. The reduction of the dimension of the Hilbert space greatly increases the performance of the eigensolver and will allow you to reach yet larger system sizes. (You can simply set $J = 1$ here, and the Hamiltonian 12 has no parameters.)

5.2 $J_1 - J_2$ Heisenberg chain

One can modify the Heisenberg model to include second-neighbor interactions,

$$H = J_1 \sum_j \boldsymbol{\sigma}_j \cdot \boldsymbol{\sigma}_{j+1} + J_2 \sum_j \boldsymbol{\sigma}_j \cdot \boldsymbol{\sigma}_{j+2}. \tag{15}$$

This model undergoes a transition from a gapless "Bethe" phase to a gapped so-called "valence bond solid" (VBS) phase that spontaneously breaks the translational symmetry. Roughly, spins pair up into singlets on either the even or the odd links of the lattice. Such singlets are similar to chemical valence bonding, and the "bonds" form a crystal, hence the name VBS. The transition happens at $J_2/J_1 \approx 0.25$, while $J_2/J_1 = \frac{1}{2}$ is the special Majumdar-Ghosh point, where the above qualitative picture for the ground state is quantitatively exact.

One can apply similar techniques to find and characterize the transition as in the previous examples (e.g., monitor fidelity and observe opening of the gap as one varies J_2/J_1; study spinspin correlation functions and observe power law decay in the Bethe phase as compared with exponential decay in the VBS phase). One can also devise an order parameter for detecting VBS: $\sum_j (-1)^j \langle \boldsymbol{\sigma}_j \cdot \boldsymbol{\sigma}_{j+1} \rangle$ will roughly detect the "wave" at wavevector π in the energy density of the valence bond crystal.

References

[1] Sandvik, Anders W. "Computational studies of quantum spin systems." In AIP Conference Proceedings, vol. 1297, no. 1, pp. 135-338. American Institute of Physics, 2010.

[2] Golub, Gene H., and Charles F. van Loan. Matrix computations. Vol. 3. JHU press, 2013.

[3] https://en.wikipedia.org/wiki/Bitwise_operation

[4] https://docs.scipy.org/doc/scipy/reference/sparse.html