Quantum Simulation

PART II: Introduction to tensor network methods

Wenyang Qian
Email: qian.wenyang@usc.es
All resources available at GitHub

November 17, 2025

Contents

Chapter 1

Introduction

1.1 Quantum many-body systems

Let us begin with the simplest setting in quantum mechanics: the **one-body problem**. A single particle is described by a wavefunction $\psi(x)$, whose evolution and stationary states are determined by the Schr"odinger equation

$$\hat{H}\psi(x) = E\psi(x), \tag{1.1}$$

where $\hat{H}$ is typically composed of a kinetic and potential term,

$$\hat{H} = -\frac{\hbar^2}{2m}\nabla^2 + V(x). \tag{1.2}$$

For many important cases—such as the hydrogen atom, harmonic oscillator, or a particle in a box—this equation can be solved exactly or efficiently numerically. The system involves a single degree of freedom, and the complexity of describing or computing its behavior is modest.

However, most systems in nature are not single particles but collections of many interacting constituents. Electrons in solids, spins in magnets, atoms in a lattice, or quarks and gluons inside hadrons—all are examples of **quantum many-body systems**. Their macroscopic behaviors—superconductivity, magnetism, confinement—arise from correlations among a large number of quantum degrees of freedom. Understanding such emergent phenomena requires solving the underlying many-body problem.

For a system of N interacting particles with coordinates $\{x_1, x_2, \ldots, x_N\}$, the stationary Schr"odinger equation generalizes to

$$\hat{H}\Psi(x_1, x_2, \ldots, x_N) = E\Psi(x_1, x_2, \ldots, x_N), \tag{1.3}$$

where the Hamiltonian can be written as

$$\hat{H} = \sum_{i=1}^{N} \hat{h}_i + \sum_{i<j} \hat{V}_{ij}. \tag{1.4}$$

Here, $\hat{h}_i$ acts locally on each particle and $\hat{V}_{ij}$ represents the interactions between them. Even if the form of $\hat{H}$ is simple, the wavefunction $\Psi(x_1, x_2, \ldots, x_N)$ depends simultaneously on all particle coordinates and thus encodes the full web of correlations among them.

This many-body wavefunction can be expressed in discrete form as a superposition of all possible many-body states,

$$|\Psi\rangle = \sum_{i_1,i_2,\ldots,i_N=1}^{p} C_{i_1 i_2 \ldots i_N} |i_1 i_2 \ldots i_N\rangle, \tag{1.5}$$

where $C_{i_1 i_2 \ldots i_N}$ is the complex coefficients that specifies the quantum state $|i_1 i_2 \ldots i_N\rangle$. The goal and challenge of quantum many-body physics is thus to understand and compute this highly correlated wavefunction. Unlike in the one-body case, the presence of interactions means that the total wavefunction cannot, in general, be factorized into single-particle parts. This inseparability—arising from entanglement—is what makes the many-body problem fundamentally richer and more difficult, and at the same time, the key to describing the collective phenomena that dominate the physical world.

1.2 The curse of dimensionality

The number of coefficients $C_{i_1 i_2 \ldots i_N}$ in Eq. (1.5) grows exponentially with N, as the total Hilbert space dimension is p^N. For example, a chain of N spin-$\frac{1}{2}$ particles has 2^N complex amplitudes. These two-level systems are called **qubits**. While small systems can be exactly represented and diagonalized, even modestly large systems quickly exceed any classical computational capacity. Storing the full state of $N = 40$ spins already requires $\mathcal{O}(10^{12})$ numbers, and simulating their dynamics is entirely impractical with conventional methods. Storing the state vector alone would require several terabytes of memory. See Fig. 2.11 for an estimate of memory cost to store quantum states on N qubits. From modern desktop memory of 32 GB to supercomputing cluster memory of 2 TB (on a single node), we are able to completely represent the quantum state around 30-40 qubits. For quantum many-body problems, we are still severely limited and such a limitation cannot be easily overcomed. This is known as the **exponential wall** or **curse of dimensionality**.

Overcoming this exponential explosion has been one of the central challenges of theoretical and computational physics. Several strategies (either exact or approximate methods) have been developed to approach the problem from different directions:

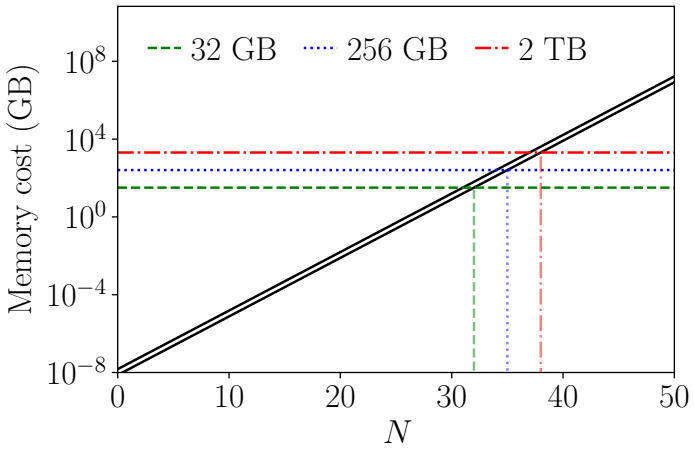


Figure 1.1: Memory cost (in GB) to store quantum state of N qubits on modern computers. The black bands shows the memory for both single- and double-precisions.

- **Exact methods** such as exact diagonalization (ED) or integrable models (e.g. Bethe ansatz) yield complete solutions but are restricted to very small systems or special Hamiltonians. Computational complexity for ED is typically $\mathcal{O}(N^3)$ and typically works well for 12-14 qubits (depending on machines).
- **Perturbative and mean-field approaches** provide analytical insight by simplifying interactions, but they often fail in strongly correlated regimes.
- **Stochastic methods** such as quantum Monte Carlo (QMC) can efficiently sample certain systems but suffer from the sign problem in fermionic or real-time settings.
- **Quantum computing** tackles the exponential complexity by brute-force representing the many-body wavefunction directly in a physical Hilbert space built from qubits. Quantum computer can evolve quantum states without storing all amplitudes explicitly, performing computations in parallel through quantum superposition.
- **Tensor network methods** exploit this physical structure to represent only the accessible corner of Hilbert space. Instead of treating $C_{i_1 i_2 \ldots i_N}$ as a single large tensor, it is factorized into a network of smaller tensors connected by shared indices. This network efficiently encodes correlations and entanglement patterns, providing a form of smart compression. Unlike quantum computing's brute-force representation, tensor networks achieve efficiency by understanding which parts of the wavefunction matter most.

Despite the exponential size of the Hilbert space, nature itself never explores all of it. The physically relevant states—such as ground states of local Hamiltonians or low-energy excitations—occupy only a small, structured corner of the full space. These states are typically constrained by locality and limited entanglement. The key question then becomes:

Can we find efficient representations that capture the essential structure of quantum many-body states without explicitly storing their exponentially many amplitudes?

In this lecture, we will focus on tensor network methods, beginning with their mathematical foundation and progressing toward practical algorithms for finding ground states and simulating dynamics. We will see how these techniques connect to renormalization group ideas, variational optimization, and time evolution schemes, and how they form a bridge between classical and quantum computational paradigms. This note is inspired in part by the following references [1, 2, 3, 4].

Chapter 2

Tensor networks

2.1 Tensor Network

Let us start from the beginning. A tensor is simply a multidimensional array of complex numbers. They are simply a generalization of vectors and matrices. The **rank** of a tensor is the number of indices. A rank-0 tensor is scalar (c), a rank-1 tensor is a vector (A_i), and a rank-2 tensor is a matrix (A_{ij}) and so forth.

An index contraction is the sum over all the possible values of the repeated indices of a set of tensors. For instance, the matrix product

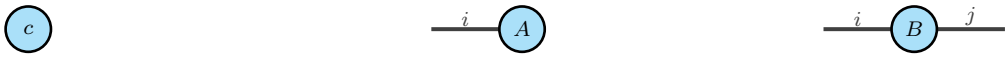
$$C_{ij} = \sum_{k=1}^{d} A_{ik} B_{kj} \tag{2.1}$$

is the contraction of index k, which amounts to the sum over its d possible values. One can also have more complicated contractions. Indices that are not contracted are called **open indices**, such as indices i and j here. One can also contraction that leaves no open indices and the result is a scalar, for example,

$$c = \sum_{i}^{d} A_i B_i \tag{2.2}$$

is the scalar product of two vectors.

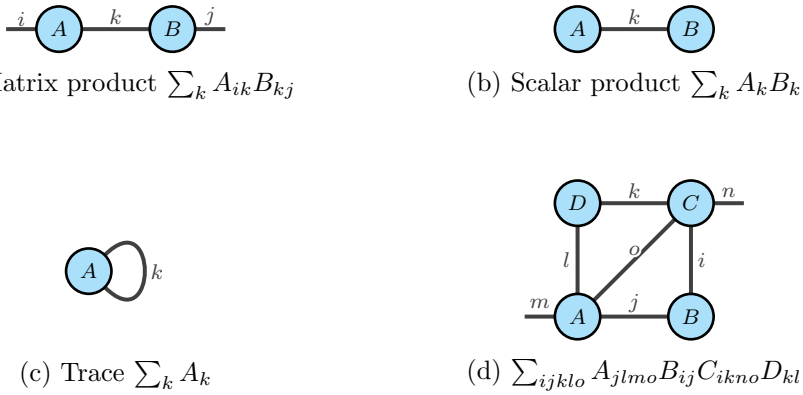
A Tensor network (TN) is a set of tensors where some, or all, of its indices are contracted according to some pattern. Contracting the indices of a TN is called, for simplicity, contracting the TN. The above equations are examples of TN, which is equivalent to a matrix product and a scalar product. Generally, one can more complicated TNs and once this point is reached, it is more convenient to introduce a diagrammic notation for tensors and TNs. For this purpose, **tensor network diagrams** are invented, see Fig. 2.1 and Fig. 2.2 for some examples.



(a) Scalar c (b) Vector tensor A_i (c) Matrix tensor B_{ij}

Figure 2.1: Examples of tensor network diagrams.

In these diagrams, tensors are represented by nodes, and indices in the tensors are represented by edges (legs). The edges connecting tensors correspond to contracted indices; edges that do not go from one tensor to another correspond to open indices in TN. Those uncontracted legs represent physical degrees of freedom. The letter in the node and indices on the edges are sometimes omitted for simplicity.



(a) Matrix product $\sum_k A_{ik}B_{kj}$ (b) Scalar product $\sum_k A_k B_k$

(c) Trace $\sum_k A_k$ (d) $\sum_{ijklo} A_{jlmo}B_{ij}C_{ikno}D_{kl}$

Figure 2.2: More examples of tensor network diagrams.

TN diagrams are intuitive to handle calculations with TN. For instances, operations such as matrix product, scalar product, trace, and so on can be represented by the diagrams in Fig. 2.2. Unlike plain equations, TN diagrams allow handling of complicated expressions in a visual way - you can compare the language of TN diagrams to that of Feynman diagrams in quantum field theory. Studying TN with their diagrams also helps understand and uncover their properties that are usually obscured in the equations.

2.2 Tensor network for quantum many-body state

Let us now go back to the beginning to explain the TN representation of quantum many-body states. First, we give a formal definition of quantum many-body states.

Quantum many-body state

Consider a generic quantum many-body system with N particles, each with p levels. Then a general wave function is

$$|\Psi\rangle = \sum_{i_1,i_2,\ldots,i_N} C_{i_1 i_2 \ldots i_N} |i_1 i_2 \ldots i_N\rangle \equiv \sum_{i_1,\ldots,i_N=1}^{p} C_{i_1 i_2 \ldots i_N} |i_1\rangle \otimes |i_2\rangle \otimes \cdots \otimes |i_N\rangle \quad (2.3)$$

where each basis $|i_r\rangle$ is the state of particle r taking $1, 2, \cdots, p$ possible values, the coefficient $C_{i_1 i_2 \cdots i_N}$ consists of p^N complex numbers.

The symbol $\otimes$ denotes the tensor product of individual quantum states for each one of the particles in the many-body system. Another way to put it, in quantum mechanics, one can use a column vector or ket of size p^N to describe the same wave function, where each element of this vector stores the corresponding complex coefficient.

In the language of TN, one can understand the p^N coefficients $C_{i_1 i_2 \ldots i_N}$ that describes the wave function $|\Psi\rangle$ as a tensor of C with N indices $i_1, i_2, \cdots, i_N$ where each of these indices take p different values. Diagrammatically, this tensor C can be represented as in Fig. 2.3 on the left. We can see that this is an exponentially expensive way to describe the quantum many-body states. The goal of TN is to reduce the exponential complexity in the representation of states with a controllable and accurate description of the entanglement of the state. This is typically achieved by replacing the "big" tensor C of high rank by a TN of "smaller" tensors of lower ranks, which are easy to manipulate. Various decomposition approaches are widely studied from the famous **Matrix Product State (MPS)** to the Projected Entangled Pair States (PEPS) and more. In Fig. 2.3, we show the example using MPS to decompose the high-rank tensor to low rank tensors. The MPS is a key focus in this note.

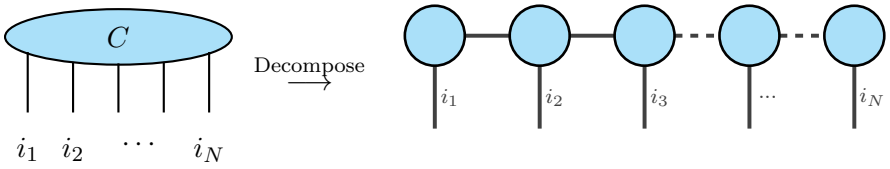


Figure 2.3: Decomposing a large rank-N tensor to products of smaller-rank tensors using the MPS representation with open boundary condition.

It is important to ensure the final representation of $|\Psi\rangle$ in terms of a TN depend on a polynomial number of parameters to make it computationally efficient and viable in describing the quantum state of the many-body system. Precisely, the total number of parameters P_{tot}

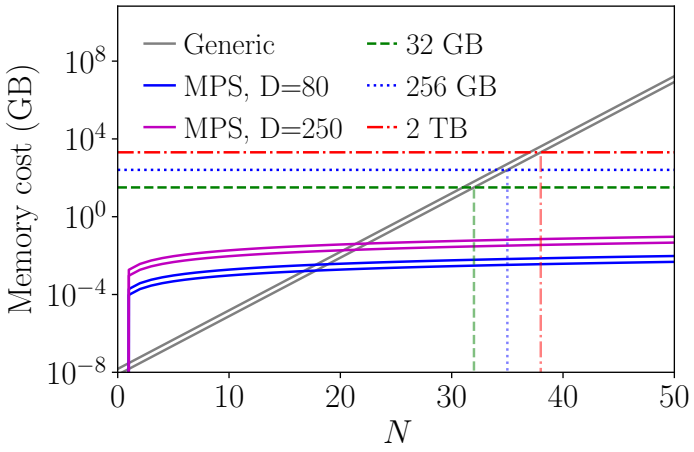


Figure 2.4: Memory cost (in GB) to store quantum state of N qubits using the MPS tensor network. The bands shows the memory for both single- and double-precisions.

in a TN is a sum of parameters in each individual tensor,

$$P_{\text{tot}} = \sum_{t=1}^{N_t} P(t), \tag{2.4}$$

where $P(t)$ is the number of parameters for tensor t in the TN and N_{t} is the number of tensors. Practically, N_t must be sub-exponential or constant in N, i.e., $N_t = \mathcal{O}(\text{poly}(N))$ or $\mathcal{O}(1)$. For each tensor t, its number of parameter

$$P(t) = \prod_{a_t=1}^{\text{rank}(t)} D(a_t), \tag{2.5}$$

where the product runs through all the indices of a given tensor t, and $D(a_t)$ is the different possible values of index a_t, and rank(t) is the number of indices of the tensor. Taking d_t the maximum of all the numbers $D(a_t)$ for tensor t, then

$$P(t) = \mathcal{O}(d_t^{\text{rank}(t)}). \tag{2.6}$$

Tensor network complexity

The total number of parameters in a TN is

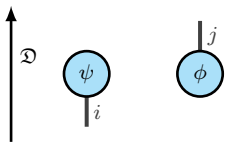
$$P_{\text{tot}} = \mathcal{O}(\text{poly}(N)\text{poly}(d)) \tag{2.7}$$

where N is the number of individual tensors, and d is the maximum possible values for any index of a tensor in TN. This assumed the rank of each tensor is bounded by a constant.

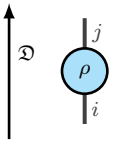
Scaling polynomially with system size gives the strength of tensor network representation of the quantum many-body states. Taking the example of a Matrix Product State (MPS) or tensor train with periodic boundary condition in Fig. 2.3. The MPS tensor network is a factorization of a tensor with N indices into a chain-like product of three-index smaller tensors. Assuming open indices in the MPS take p values and neighboring contracted indices take D values, the number of parameters in a MPS is merely $\mathcal{O}(NpD^2)$, an exponentially reduction from the original tensor of $\mathcal{O}(p^N)$ of the same rank N. The trick is that the MPS (or any other TN) involves extra degrees of freedom and those internal connecting indices represent structure of many-body entanglement in the quantum state $|\Psi\rangle$. These indices are usually called **bond**, and their number of possible values are referred to as **bond dimensions**, typically denoted by the variable D. Comparing with generic wave function, the MPS representation dramatically reduces the memory cost; see Fig. 2.4.

Exercise 2.1 *Calculate the explicit number of parameters in an MPS with open boundary (as in Fig. 2.3) and periodical boundary conditions (first and last nodes are connected).*

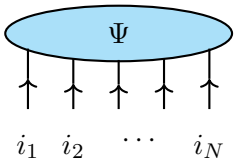
One can see that even with tensor network diagrams, it can still be complicated and confusing to understand the tensor networks. So far, we have just treated tensor networks as undirected graphs with nodes and edges. To make it more explicit, especially in the context of quantum many-body system and in the representation of quantum state, we need to make a few useful conventions.



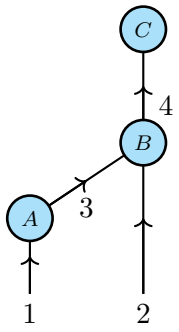
(a) Ket state $|\psi\rangle$ and bra state $\langle\phi|$



(b) Density matrix $\rho = \rho_i^j |i\rangle\langle j|$



(c) Rank-N wave function $|\Psi\rangle$



(d) Tensor network $A_{i_1}^{i_3} B_{i_3 i_2}^{i_4} C_{i_4}$

Figure 2.5: Tensor networks representation of quantum many-body states.

Tensor network graphical conventions

1. Tensor networks are oriented, and a direction of reference $\mathfrak{D}$ is specified. In this note, we choose $\mathfrak{D}$ as the down-up direction with respect to the page.

2. No edges representing indexes can be allowed to be orthogonal to the direction of $\mathfrak{D}$[a]. Let the direction of edge be from the tensor. Then, each edge of every tensor in the graph has a non-zero projection (either parallel or anti-parallel) with respect to $\mathfrak{D}$.

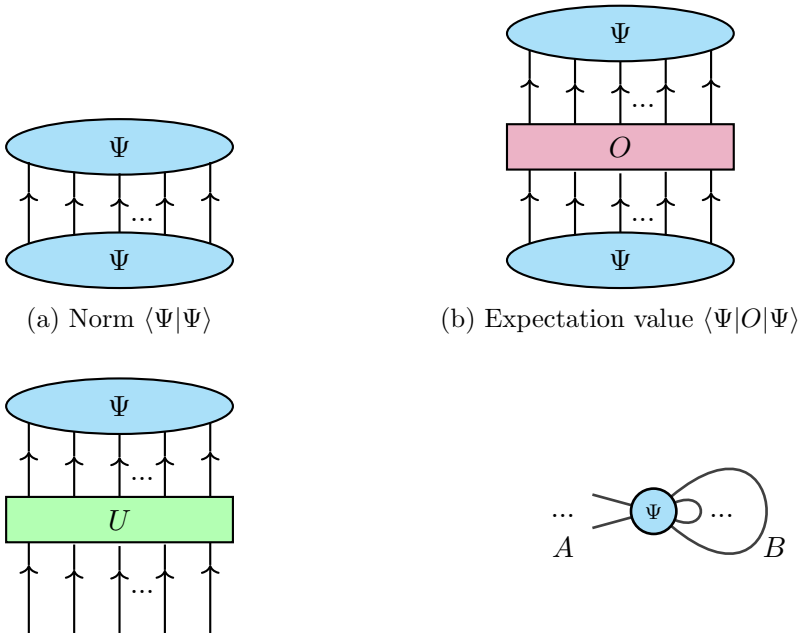
3. We use subscript (superscript) to represent edges that are anti-parallel (parallel) to $\mathfrak{D}$. The direction of $\mathfrak{D}$ on the edge can be optionally indicated with a directed arrow.

4. The ket states are labeled via subscripts, and the bra states defined in the dual state are labeled via superscripts. Superscripts contract only with subscripts and vice-versa.

5. The edges in the tensor network are numbered starting from the bottom to top (with respect to $\mathfrak{D}$) and left to right.

[a] While this is useful, we do not always reinforce this requirement for simplicty reasons on our diagrams when they are understood from the context.

With these rules in mind, we can define the tensor network representing the wave function more precisely, see Fig. 2.5 for examples on tensor network representation of ket, bra, and density matrix, and multi-dimensional wave functions. One should note that we are using labels for tensor network and quantum states interchangeably for convenience. In principle, the tensor network represents the coefficients in the multi-dimensional quantum statevector. The directional reference $\mathfrak{D}$, in this note, is always pointing in the upward direction (unless stated otherwise). As a consequence, the ket (bra) states have legs pointing in the downward (upward) direction. For simplicity reasons, we may sometimes drop the indices for the edges.



(a) Norm $\langle\Psi|\Psi\rangle$

(b) Expectation value $\langle\Psi|O|\Psi\rangle$

(c) Unitary evolution $U|\Psi\rangle = e^{iHt}|\Psi\rangle$

(d) Partial trace $\mathrm{Tr}_B(\Psi)$

Figure 2.6: Tensor network representation of useful operations on generic quantum states.

We are able to write and draw the operations on the wave functions, such as norm, expectation value, unitary evolution, and partial trace on the wave function, as in Fig. 2.6. In fact, all operations on the wave functions can be illustrated using tensor network diagrams. To further explain these operations, we need to introduce several useful tensor operations.

2.3 Tensor network manipulations

In tensor network algorithms, it is often necessary to manipulate tensors—reshaping, compressing, or transforming them—while preserving physical or algebraic meaning. Here we

introduce the most common operations include index fusion and splitting, compression via matrix factorization, differentiation with respect to tensor elements, local gauge transformations, and operator vectorization.

(1) Tensor product

Tensor product is a generalization of the outer product of vectors or Kronecker product of matrices. The values of the tensor product on a given set of indices is the element-wise product of the values of each constituent tensor. For example,

$$[A \otimes B]_{i_1,i_2,\cdots i_n,j_1,j_2,\cdots,j_m} = A_{i_1,i_2,\cdots i_n} \cdot B_{j_1,j_2,\cdots,j_m}, \tag{2.8}$$

and they are simply represented by two tensors placed together in tensor network diagrams, see Fig. 2.7.

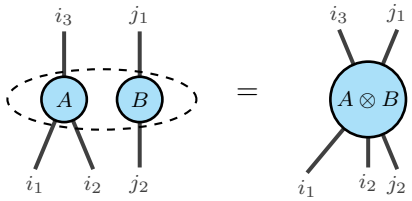


Figure 2.7: Tensor product.

(2) Fusion and splitting

Tensors are structures to organize information. A tensor can always be reshaped by fusing or splitting of indices. It is very much analogous to reshaping betwen matrices (rank-2 tensor) to vectors (rank-1 tensor) and vice-versa.

$$A_i = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} \Leftrightarrow B_{jk} = \begin{bmatrix} c_1 & c_2 \\ c_3 & c_4 \end{bmatrix} \tag{2.9}$$

No information is lost in the process, just rearrangement. In tensor networks, rank is a rather fluid concept, as long as the overall number of elements match (as in the case above). We adopt the symbol $\succ$ to represent fusion of indices and $\prec$ to represent splitting of indices. A rank-N tensor $C_{i_1 i_2 \ldots i_N}$ can be fused into a rank-1 tensor $D_{\boldsymbol{i}}$ by $i_1 i_2 \ldots i_N \succ \boldsymbol{i}$. The reverse can be done with splitting $\boldsymbol{i} \prec i_1 i_2 \ldots i_N$. We can also divide the indices into different subset and fuse them. See Fig. 2.8 for an example. It also works for contracted indices.

In fact, a rank-N tensor may be reshaped into any other ranks by these two operations (provided total dimensionality is sufficient). This also provides great shorthand to write operations such as

$$\langle\phi|\psi\rangle = \phi^{\alpha_1\alpha_2\ldots\alpha_N *}\psi_{\alpha_1\alpha_2\ldots\alpha_N} \equiv \phi^{\boldsymbol{i}*}\psi_{\boldsymbol{i}}. \tag{2.10}$$

Note this is purely a relabeling and introduces no numerical approximation, but it is essential for performing contractions and decompositions that require rank-2 tensors.

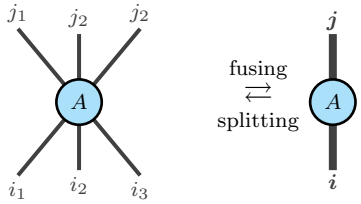


Figure 2.8: Tensor fusion and splitting.

(3) Liouville representation

Mathematically, it is perfectly fine to represent operators as vectors in an enlarged space — a mapping often called the Liouville or Choi–Jamiolkowski representation. Given an operator such as ρ_i^j, we define its Liouville representation as equivalent to lowering the superscripts,

$$L(\rho_i^j) = L(\rho_i^j|i\rangle\langle j|) = \rho_{i,j}|i,j\rangle\rangle = \rho_{i,j}|i\rangle|j\rangle = \rho_{i,j}, \tag{2.11}$$

which lives in a doubled Hilbert space $\mathcal{H}\otimes\mathcal{H}$. This mapping allows one to treat operators

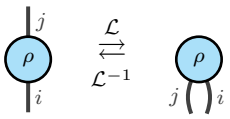


Figure 2.9: Liouville transform on tensors.

as states and to express superoperators (such as density matrix evolution) as matrix–vector multiplications. As a result, we can be a little relaxed in treating our indices as superscripts or subscripts. It is particularly useful when formulating tensor networks for mixed states or open quantum systems. See Fig. 2.9 for an example.

(4) Compression and truncation

The fusion operation essentially allows us to recast any rank-N tensor into a rank-2 matrix,

$$T_{\alpha_1\alpha_2...\alpha_N} \equiv T_{\boldsymbol{i},\boldsymbol{j}} \tag{2.12}$$

where $\alpha_1\alpha_2...\alpha_l \succ \boldsymbol{i}$, $\alpha_{l+1}\alpha_{l+2}...\alpha_N \succ \boldsymbol{j}$, and $\boldsymbol{i} \cup \boldsymbol{j} = \{\alpha_1\alpha_2...\alpha_N\}$. In matrix form, we can exploit many useful properties from linear algebra to act on the tensor. In particular, the singular value decomposition (SVD) plays a fundamental role.

Tensor Singular Value Decomposition

It is always possible to decompose a generic tensor T in three rank-2 tensors U, S, V using single value decomposition,

$$T_{\boldsymbol{i},\boldsymbol{j}} = \sum_{\boldsymbol{k}} U_{\boldsymbol{ik}} S_{\boldsymbol{kk}} V_{\boldsymbol{kj}}, \quad \text{i.e. } T = USV^\dagger, \tag{2.13}$$

where S is a diagonal and positive real matrix, U and V are isometries ($U^\dagger U = V^\dagger V = \mathbf{1}$)[a].

[a] In finite dim Hilbert space, an isometry is similar to a unitary matrix.

This is a remarkable result and has great benefits for us. See Fig. 2.10 for its representation in TN diagram. Since the diagonal elements of V are real and positive, they can be ordered and bounded from below by zero. One can rearrange the diagonal elements $\lambda_{\boldsymbol{i}} = V_{\boldsymbol{i},\boldsymbol{i}}$ of V in descending order, i.e., $\lambda_{\mathbf{1}} \geq \lambda_{\mathbf{2}} \geq \cdots \geq 0$. Then, it is possible to define a threshold cutoff $\epsilon \geq 0$ to discard all the $\lambda_{\boldsymbol{i}}$'s with $\lambda_{\boldsymbol{i}} < \epsilon$. Correspondingly the matrix S, V can be truncated, which may greatly reduce the computational complexity of the full tensor.

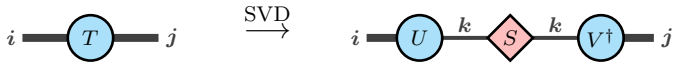


Figure 2.10: Tensor single value decomposition. To minimze space and keep it simple, we draw the diagrams horizontally, with $\mathfrak{D}$ pointing left to right.

This truncation is a key step in tensor network algorithms that will be introduced later, where it controls the bond dimension and computational cost. It also allows effective and systematic extrapolation to the exact case by taking this $\epsilon \to 0$. In particular, if ϵ drops to zero or the order of numerical precision (i.e., 10^{-15} for double floating precision), the truncation induces no error.

Exercise 2.2 *It is more intuitive to visualize the power of SVD. Taking black and white images for example. They are rank-3 tensors, where the free indices are horizontal position, vertical position, and gray scale (0-256)*[a]*. We can perform a SVD on the gray scale and observe the effect of truncation in singular value S on the image.*

[a]Colored images are rank-4 tensors, with additional dimension of color channels, r, g, b.

From the example below, one can see that SVD performs really well on images (that is essentially unstructured data set). For quantum many-body systems, which is usually much more structured, the SVD would be even more powerful.

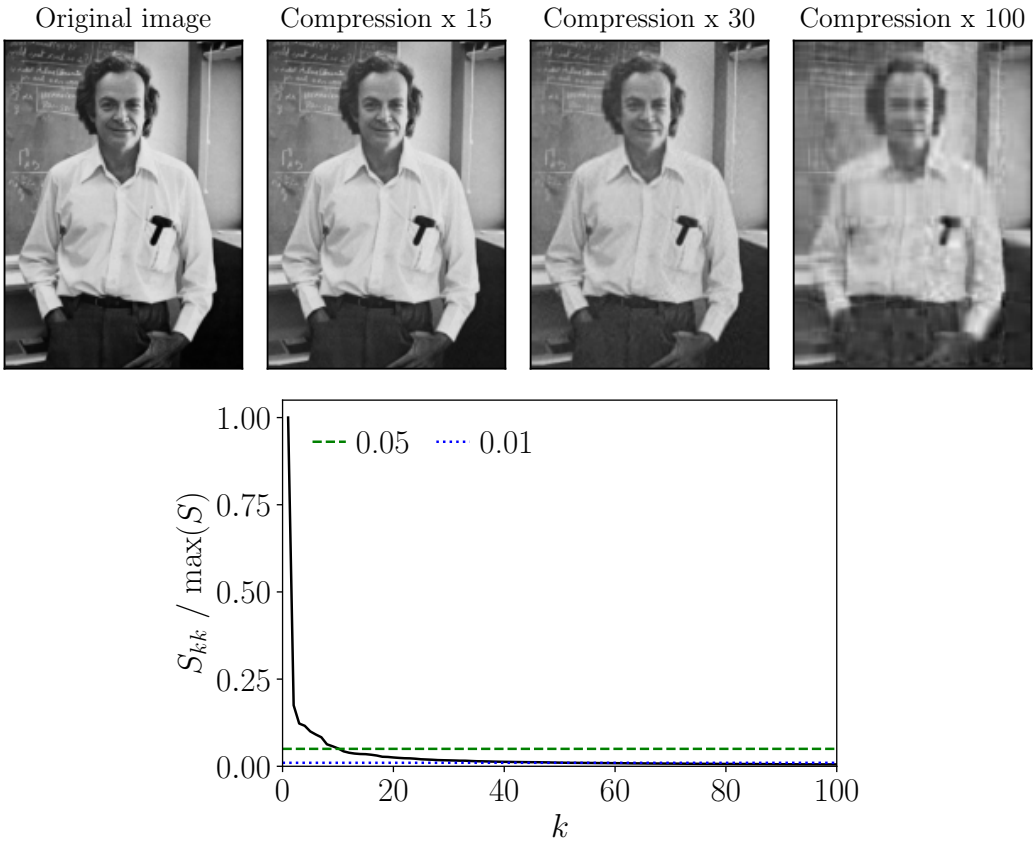


Figure 2.11: Performance of images with SVD on gray scale. The total number of non-zero elements in S is around 1500. A compression rate of 100 means we only keep the largest 15 singular values.

(5) Differentiation

In variational tensor algorithms, one frequently computes derivatives of scalar functionals with respect to tensor components, for example when minimizing the expectation value of the energy to find a tensor network representation of the ground state of the system. The computation of the energy is done by a quadratic function of two rank-N tensors,

$$E = \langle \Psi | \hat{H} | \Psi \rangle = \psi^{*\beta_1\beta_2\ldots\beta_N} H^{\alpha_1\alpha_2\ldots\alpha_N}_{\beta_1\beta_2\ldots\beta_N} \psi_{\alpha_1\alpha_2\ldots\alpha_N} \equiv \psi^{*\boldsymbol{i}} H^{\boldsymbol{j}}_{\boldsymbol{i}} \psi_{\boldsymbol{j}}. \tag{2.14}$$

where index fussion is used to simplify the expression. To find the minima of the energy, differential of the function should be set to zero,

$$\frac{\partial E}{\partial \psi_{\boldsymbol{j}}} = \psi^{*\boldsymbol{i}} H^{\boldsymbol{j}}_{\boldsymbol{i}} = 0. \tag{2.15}$$

In tensor network diagram, not surprisingly, as the said equation indicates, this corresponds to removing one tensor leg associated with $\psi_{\boldsymbol{j}}$ while contracting the rest of the network, see Fig. 2.12 for the representation with TN diagrams. This concept extends to automatic differentiation and gradient-based optimization in tensor networks.

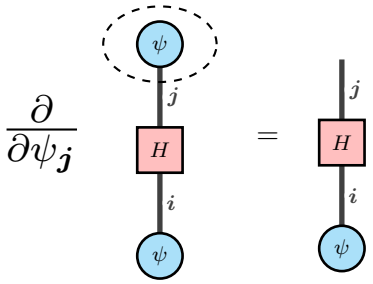


Figure 2.12: Partial differentiation on tensor network.

(6) Gauging

Tensor networks are not unique - one can change the tensor entries by means of local transformations, without affecting the global state to exploit some desired properties. This is known as gauging. To start, it is always possible to insert an identity operator $\mathbb{1} = U^\dagger U$ between any two contracted tensors with changing the overall represented physical state. Then, the tensor contraction becomes

$$A_{\boldsymbol{i}} B^{\boldsymbol{i}} = A_{\boldsymbol{i}} \mathbb{1}^{\boldsymbol{i}}_{\boldsymbol{j}} B^{\boldsymbol{j}} = A_{\boldsymbol{i}} U^{\dagger \boldsymbol{i}}_{\boldsymbol{k}} U^{\boldsymbol{k}}_{\boldsymbol{j}} B^{\boldsymbol{j}} = \tilde{A}_{\boldsymbol{k}} \tilde{B^{\boldsymbol{k}}}. \tag{2.16}$$

It is clear the local tensor entries have changed but the global tensor structures remained the same. This freedom, known as **gauge freedom**, can be exploited to simplify algorithms. Note that gauging transformation also applied to individual indices besides fused ones.

Another powerful tool to manipulate the tensors is the polar decomposition, which means that any matrix can be decomposed into product of a unitary and non-unitary part (scaling matrix):

$$A_{\boldsymbol{i}}^{\boldsymbol{j}} = U_{\boldsymbol{i}}^{\boldsymbol{k}} P_{\boldsymbol{k}}^{\boldsymbol{j}} \tag{2.17}$$

by taking $P = \sqrt{A^\dagger A}$ (unique positive-semidefinite square root) and $U = AP^{-1}$. It is easy to see U will be unitary (isometric). Geometrically, this relation tells us that any linear operation is equivalent to a rotation and subsequent rescaling. Pictorially, see Fig. 2.13.

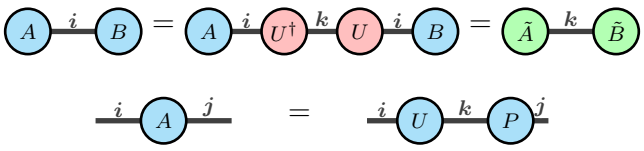


Figure 2.13: Two types of tensor gauging operations in TN diagram ($\mathfrak{D}$ pointing left to right).

(7) Contraction

Now, we proceed to discuss the computational complexity of tensor contractions. It is easy to see from earlier discussions that a scalar product requires $\mathcal{O}(d)$ operations where d is the dimension of the Hilbert space the wave function lives in. In comparison, a matrix-matrix product would require $\mathcal{O}(d^3)$ operations[1], since each element in the resulting matrix requires d operations. From these examples, one can easily generalize a rule for the complexity of any tensor network contraction that depends on the both the open indices and contracted ones.

[1]Note advanced linear algebra algorithm such as Strassen algorithm provides a better scaling of approximately $\mathcal{O}(n^{2.8})$.

Tensor network contraction complexity

Given a tensor network with n free indices $i_1, i_2, ..., i_n$ and m contracted ones, $j_1, j_2, ..., j_m$, its contraction complexity is given by the product of the dimensions of the free indices and the contracted ones,

$$\mathcal{O}\Big(\prod_{i=i_1}^{i_n} d_i\Big) \cdot \mathcal{O}\Big(\prod_{j=j_1}^{j_m} \bar{d}_j\Big) \to \mathcal{O}\Big(d^n \cdot d^m\Big) = \mathcal{O}\Big(d^{n+m}\Big) \tag{2.18}$$

where d_i ($\bar{d}_j$) is the dimension of free index i (contracted index j). For simplicity, taking d as the maximal (or average) dimension for all the indices, the expression simplifies exponential of the total number of indices, i.e., $n + m$.

This looks very bad, isn't it? Well, it means in general for a rank-N tensor network representing a underlying wave function, the complexity to perform contraction is exponential! However, this is very much a worst case scenario. In practice, we always utilize specialized structure of tensor network (such as the MPS) to reduce the complexity to perform useful operations down to polynomial.

(8) Bubbling

While tensor networks are defined in such a way that their values are independent of the order in which the constituent tenors are contracted, the order of contraction is crucial to the total complexity as it is determined by the largest intermediate tensor contraction. The order in which tensors are contracted is known as bubbling. For many tensor networks, there are both efficient and inefficient bubblings. Let us see through the following example of contraction of three tensors, in Fig. 2.14. Evidently, method (a) gives a total contraction complexity of $\mathcal{O}(d^4)$, while method (b) gives $\mathcal{O}(d^5)$, showing the importance of the order of tensor contraction.

Exercise 2.3 *What is the contraction complexity for simplifying a MPS (with open boundary)? Assuming all bond dimension is d for any open and contracted index.*

Exercise 2.4 *What is the contraction complexity for evaluate the norm $\langle\psi|\psi\rangle$ or overlap of two MPS (with open boundary)?*

By working through a few exercises, we should see that the order is really important. And if you want the full dense statevector by contracting all physical indices into an explicit vector, the cost is exponential $\mathcal{O}(d^N)$ in both storage (memory) and time — so the point of MPS (and other TNs) is to avoid doing it.

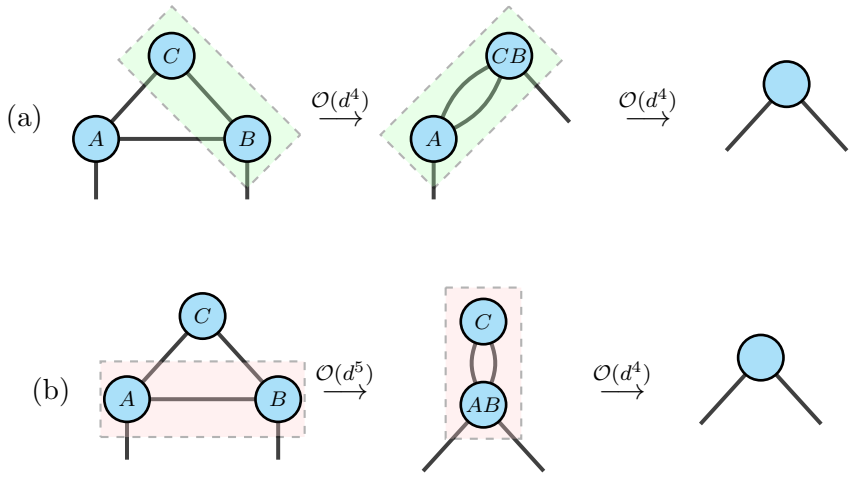


Figure 2.14: Two types of bubbling to contract three tensors.

Chapter 3

Matrix product states

3.1 Matrix Product States (MPS)

The MPS is probably the most famous example of TN states. This is because it is behind some very powerful methods to simulate 1-dimensional quantum many-body systems, most prominently the **Density Matrix Renormalization Group (DMRG)** algorithm [5].

Instead of giving the definition for the MPS, let us see how the form of the MPS is derived from a generic quantum many-body state. We already know a N-body wavefunction[1]

$$|\Psi\rangle = \sum_{i_1,\ldots,i_N} C_{i_1 i_2 \ldots i_N} |i_1 i_2 \ldots i_N\rangle \tag{3.1}$$

is represented by a rank-N tensor. To be precise and general, each state $|i_k\rangle$ in the tensor product is some d dimensional quantum system. They are basically qudits, i.e., generalized qubits with d levels. We now show it can be exactly rewritten as an MPS through successive matrix factorizations. The procedure is as follows (see Fig. 3.1 for a graphical representation[2]):

1. Fuse indices i_1 with the rest $(i_2 \ldots i_N)$ to form a matrix.
2. Perform SVD and truncate if desired.
3. Assign the first tensor A_1 from the product of the left unitary and singluar value.
4. Repeat recursively along the chain.

This procedure guarantees an exact decomposition if all singular values are kept, and a controlled approximation if truncation is used. In the exact case, we can keep all the singular

[1] We use the term "body" but it really refers to "sites", "subsystems", "mode", etc, as degrees of freedom of your choice.

[2] With Liouville transform, it is not so important to distinguish between bra and ket, so we write things horizontally; see Ref. [1] for a strict representation.

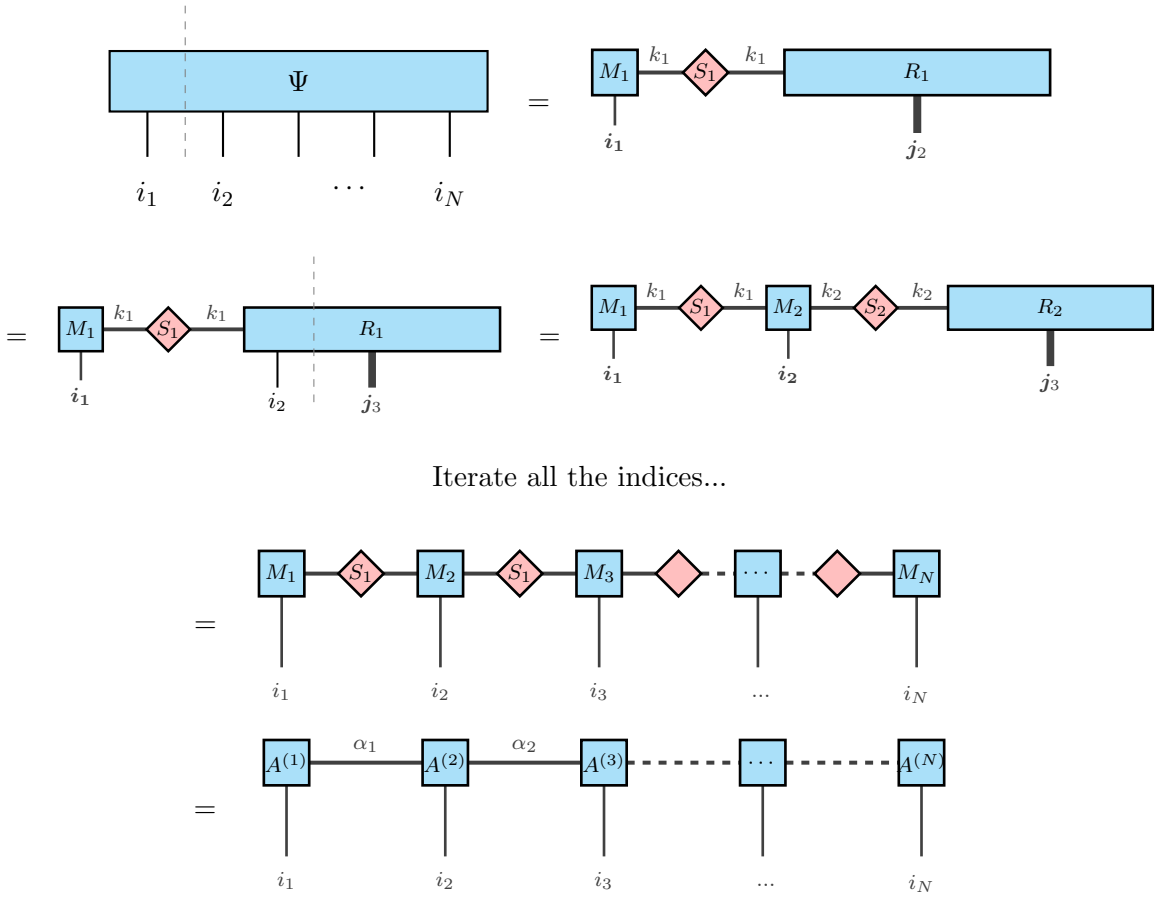


Figure 3.1: Decomposition of generic N-body quantum state $|\Psi\rangle$ to an MPS with open boundary.

values at every SVD step, which requires a bond dimension $D_k = \min(d^k, d^{N-k})$. Let $D_{\max}$ be the maximum bond dimension or equivalently the truncations on those singular values, then $D_{\max} = \exp(N/2)$ is sufficient to describe any quantum system. This representation is exact but exponentially large. For this reason, we typically truncate the bond dimension to some finite value $D_{\max}$ and systemically improved by increasing the bond dimension $D_{\max}$.

Formally, we put down the definition for the MPS below:

Matrix Product State

The Matrix Product State (MPS) of a N-body system with open boundary condition (OBC) is defined as

$$|\Psi^{\text{OBC}}\rangle = \sum_{i_1 i_2 ... i_N=0}^{d-1} A^{(1)}_{i_1} A^{(2)}_{i_2} ... A^{(N)}_{i_N} \, |i_1 i_2 ... i_N\rangle \, , \tag{3.2}$$

while the Matrix Product State (MPS) with periodic boundary condition (PBC) is defined as

$$|\Psi^{\text{PBC}}\rangle = \sum_{i_1 i_2 ... i_N=0}^{d-1} \text{tr}\big(A^{(1)}_{i_1} A^{(2)}_{i_2} ... A^{(N)}_{i_N}\big) \, |i_1 i_2 ... i_N\rangle \, , \tag{3.3}$$

where the d levels run from 0 to $d-1$ per site. Let the internal contraction be with bond D, the total number of parameters equal to $\mathcal{O}(NdD^2)$ in both cases.

The MPS is a representation of a many-body wavefunction as a product of smaller tensors arranged in a one-dimensional chain. Though the MPS is a chain of rank-3 tensors, It is useful to understand it from point of view of matrices. In the OBC case, $A^{(1)}_{i_1}$ is a $1 \times D$ matrix, $A^{(N)}_{i_N}$ is a $D \times 1$ matrix, and all intermediate $A^{(k)}_{i_k}$ are $D \times D$ matrices. In the PBC case, all $A^{(k)}_{i_k}$ for $k = 1, 2, ...N$ are $D \times D$ matrices.

Exercise 3.1 *Implement the MPS from scratch using matrices for both OBC and PBC.*

Now let us see a few examples of MPS for the common quantum states used in quantum computing.

(1) Product state

The product state is simply $|\Psi\rangle = |i_1\rangle \otimes |i_2\rangle \otimes ... \otimes |i_N\rangle$ for $i_k = 0$ or 1. It can be represented by an MPS with $D = 1$, by taking $A^{(k)}_{i_k} = 1$. For example, $A_0 = 1, A_1 = 0$ represents $|\Psi\rangle = |00...0\rangle$ which is typically used for an empty quantum circuit. Note $D = 1$ suggests there are no quantum entanglement between each partition, as expected.

(2) W state

The W state is another useful state in quantum information,

$$|W\rangle = \frac{1}{\sqrt{3}}\big(|100\rangle + |010\rangle + |001\rangle\big) \, , \tag{3.4}$$

useful for variational circuit initialization and error correction code. The generalized W state for N-qubits is $|W_N\rangle = \frac{1}{\sqrt{N}}\big(|100...0\rangle + |010...0\rangle + |00...01\rangle\big)$. In fact, the W state can also be represented easily with an MPS with $D = 2$ and OBC:

$$A_0^{(1)} = \begin{pmatrix} 1 & 0 \end{pmatrix}, \qquad A_1^{(1)} = \begin{pmatrix} 0 & 1 \end{pmatrix}, \tag{3.5}$$

$$A_0^{(k)} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad A_1^{(k)} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \tag{3.6}$$

$$A_0^{(N)} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \qquad A_1^{(N)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \tag{3.7}$$

where $k = 2, 3, ..., N-1$. The reason is that $\left(A_0^{(k)}\right)^2 = \mathbf{1}, \left(A_1^{(k)}\right)^2 = \mathbf{0}, A_1^{(k)} A_0^{(k)} = A_1^{(k)} = |0\rangle\langle 1|$.

(3) GHZ state

The other important state in quantum computing is the Greenberger–Horne–Zeilinger (GHZ) state,

$$|GHZ\rangle = \frac{|000\rangle + |111\rangle}{\sqrt{2}} \to \frac{|00...0\rangle + |11...1\rangle}{\sqrt{2}}, \tag{3.8}$$

which is typically a 3-qubit state but can also be generalized to N qubits. Using MPS with PBC, the GHZ states can be represented with $D = 2$ by

$$A_0 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad A_1 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \tag{3.9}$$

where the superscript is omitted for the same matrices are used for all sites. Importantly, a small value of bond dimension indicates a small entanglement entropy across a given cut location. But it does not mean the whole state is weakly entangled. The GHZ is the classical counterexample, as it is known as a maximally entangled state.

3.2 Quantum entanglement.

In quantum systems, entanglement is the key resource that determines how complex a state is. Consider dividing a system into two parts, left and right, at some cut, the **entanglement entropy** across that cut is

$$S = -\mathrm{Tr}(\rho_L \log \rho_L) \tag{3.10}$$

where ρ_L is the reduced density matrix of the left subsystem. A random highly entangled state in a Hilbert space of dimension d^N typically has $S = \mathcal{O}(N)$, that is a volume law: entropy scales with the number of degrees of freedom, i.e., the volume. By contrast, ground states of local Hamiltonians in 1D (and many low-energy states in general) satisfy an **area law**:

$S = \mathcal{O}(1)$, that is the entanglement entropy grows only with the boundary (area) between regions, not their volume.

One of the key features of MPS is that its bond dimension directly encodes the amount of entanglement a state can capture. When an MPS is partitioned across a bond connecting sites k and $k+1$, the bond index α_k corresponds exactly to the **Schmidt index** for that bipartition:

$$|\Psi\rangle = \sum_{\alpha_k=1}^{D_k} \lambda_{\alpha_k} |\phi_{\alpha_k}^{[1\ldots k]}\rangle \otimes |\phi_{\alpha_k}^{[k+1\ldots N]}\rangle. \tag{3.11}$$

where the Schmidts $\sum_{\alpha_k} \lambda_{\alpha_k}^2 = 1$ for normalized states and the bipartite states are

$$|\phi_{\alpha_k}^{[1\ldots k]}\rangle = \sum_{\boldsymbol{i}=0}^{d-1} U_{\boldsymbol{i}\alpha_k} |\boldsymbol{i}\rangle\,, \quad |\phi_{\alpha_k}^{[k+1\ldots N]}\rangle = \sum_{\boldsymbol{j}=0}^{d-1} V_{\boldsymbol{j}\alpha_k}^* |\boldsymbol{j}\rangle\,, \tag{3.12}$$

where $i_1 i_2 \ldots i_k \succ \boldsymbol{i}$ and $i_{k+1} i_{k+2} \ldots i_N \succ \boldsymbol{j}$. The Schmidt index α_k correspond exactly the singular values. Thus, the bond dimension D_k at a cut k limits the number of Schmidt coefficients, and therefore the maximum entanglement entropy that the MPS can encode at the cut k:

$$S_k = -\sum_{\alpha_k=1}^{D_k} \lambda_{\alpha_k}^2 \log \lambda_{\alpha_k}^2 \leq \log D_k. \tag{3.13}$$

This inequality shows that in general an MPS with moderate bond dimension D can accurately describe states with limited entanglement, i.e., $S \leq \log D$, such as ground states of 1D gapped Hamiltonians. When truncating singular values, it is useful to define discarded weight

$$\epsilon_{\text{trunc}} = \sum_{\alpha > D} \lambda_\alpha^2 \tag{3.14}$$

to quantify the accuracy of the MPS approximation.

Entanglement measures the correlations between subsystems that cannot be captured by product states. If $|\Psi\rangle$ is separable across the bipartition, $D_k = 1$ and $S_k = 0$. If $D_k > 1$, the state contains entanglement, and the Schmidt coefficients quantify its structure.

This is precisely why MPS are efficient for many 1D systems: physically relevant states often have area-law entanglement, meaning S does not scale with system size, and a modest bond dimension $D \sim \mathcal{O}(1)$ suffices. Another way to put it, many physical system obeys the so-called area law such that only finite number elements of each singular values are nonzero, suitable to be described by the MPS.

Exercise 3.2 *Prove $S_k \leq \log D_k$ and find the condition for the equality.*

3.3 Matrix Product Operator

A straightforward generalization of the MPS ansatz is the Matrix Product Operator (MPO), which represents the tensor network for the operators for those MPS that we just introduced. Using Liouville transform, an MPO is basically an MPS with double the number of indices, with one half for the space it is acting on and the other half for the dual space.

Matrix Product Operator

The Matrix Product Operator (MPO) of a N-body system with open boundary (loop-free) is defined as

$$\hat{O} = \sum_{\substack{i_1 i_2 ... i_N = 0 \\ j_1 j_2 ... j_N = 0}}^{d-1} B^{(1)}_{i_1,j_1} B^{(2)}_{i_2,j_2} ... B^{(N)}_{i_N,j_N} \, |i_1 i_2 ... i_N\rangle\langle j_1 j_2 ... j_N| \, , \tag{3.15}$$

where the d levels run from 0 to $d-1$ per site and the bond D is between the local indices, the total number of parameters equal to $\mathcal{O}(Nd^2D^2)$ in both cases.

The MPO provides an efficient representation of the Hamiltonian for those MPS along with many other observables such correlation operators. See Fig. 3.2 for some useful graphical representation of the MPO. By going through the exercises below, we can see that the MPO gives an efficient way to calculate these observables with polynomial scaling of system size $\mathcal{O}(N)$.

Exercise 3.3 *Find out the complexity for $\langle\psi|\psi\rangle$ with the MPS ψ.*

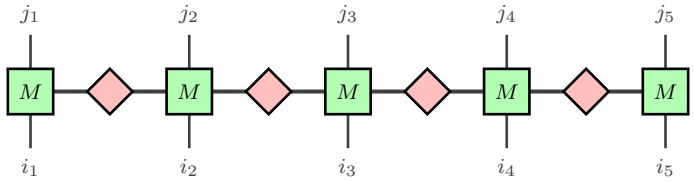
Exercise 3.4 *Find out the complexity for $\langle\psi|\hat{H}|\psi\rangle$ with the MPS ψ and the MPO $\hat{H}$.*

Exercise 3.5 *Find out the complexity for $\langle\psi|\hat{O}_i|\psi\rangle$ with the MPS ψ and a local operator $\hat{O}_i$.*

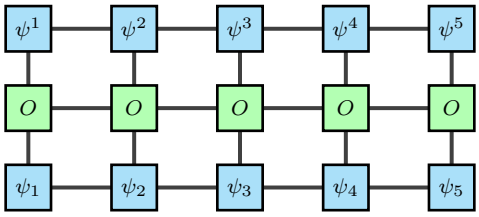
Exercise 3.6 *Find out the complexity for $\langle\psi|\hat{O}_i\hat{O}_{i+r}|\psi\rangle$ with the MPS ψ and a two-point correlator $\hat{O}_i\hat{O}_{i+r}$.*

From these exercises above, you might find that one would need to repeat the procession of contraction for different observables. Is there a better way to speed up the calculations? The answer is yes and it comes from the gauge freedom that we introduced earlier for tensor network.

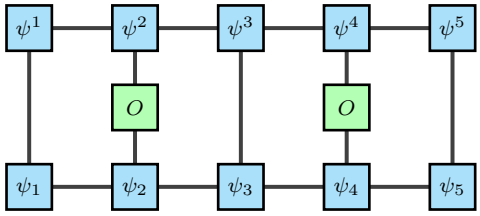
In terms of the MPS, we transform the MPS by putting down identities $I = M^{-1}M$ in between each site without changing its physical representation, leaving the physical states



(a) Representation of MPO $M^{j_1 j_2 j_3 j_4 j_5}_{i_1 i_2 i_3 i_4 i_5}$



(b) Computation of the energy value $\langle\psi|O|\psi\rangle$ via MPO and MPS.



(c) Computation of two-point correlation operator $\langle\psi|O_2 O_4|\psi\rangle$.

Figure 3.2: Matrix product operator and its operations. For simplicity the internal singular value matrices are often omitted.

invariant. This action is known as the gauge transformation. One can simply regard them as basis transformation on the virtual bonds. The MPS after gauging transformation is called the **canonical forms** of the MPS. They are various types of canonical forms and we illustrate the left-canonical or left-isometric form here, the resultant MPS obey

$$\sum_{j=0}^{d-1} A_j^\dagger A_j = \mathbf{1} \tag{3.16}$$

and pictorially this corresponds to Fig. 3.3. To obtain the left-canonical form, we can use the SVD or the QR[3] decomposition to recursively split each A_k into a isometric component and a remainder that is combined with the next A_k. We illustrate the steps to left-canonicalize an MPS of N sites with open boundary:

[3]Using the QR decomposition, any matrix can be decomposed into a product of isometry and a generic tensor.

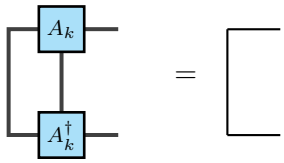


Figure 3.3: Left-canonical form of the MPS.

1. Start on the leftmost tensor $A^{(1)}_{i_1}$ and reshape it into a matrix say M_1 of shape $d \times D$.
2. Perform SVD or QR such that $M_1 = Q_1 R_1$ such that $Q_1^\dagger Q_1 = \mathbf{1}$ and R_1 is a generic tensor.
3. Replace $A^{(1)'}_{i_1} = Q_1$ and $A^{(2)'}_{i_2} = R_1 A^{(2)}_{i_2}$ such that site i_1 is contracted to identity.
4. Repeating this procedure for each site from left to right until the end.

Using QR or SVD, the typically scaling to do each is $\mathcal{O}(dD^3)$, so completing one full sweep still requires $\mathcal{O}(NdD^3)$. However, after canonicalization, many operations such as norm, local expectation values, overlaps with other canonical MPS become cheap, often $\mathcal{O}(1)$ or $\mathcal{O}(dD^2)$ per local operator, and that is the important reason for computing observable with MPS becomes efficient and practical.

3.4 Exercises

We will learn to use `ITensor` [6], a powerful library for tensor network calculations based on the `Julia` programming language. For installation, follow the official guide at julialang.org/install. For a quick walkthrough on `Julia`, follow any of the tutorials at julialang.org/learning.

To use `Juia` for tensor network, we need to install a few packages. This includes `ITensors`, `ITensorMPS`, `ITensorTDVP` for tensor network libraries, and `IJulia` for interactive `JupyterLab` notebook, and other necessary packages such as `PyPlot`, `LaTeXStrings`. See the "Julia-ITensor-Installation" guide within.

The hands-on exercise is available on Github. Exercises for this chapter are on the MPS, expectation values, entanglement entropy.

Chapter 4

Ground States via Tensor networks

4.1 Variational principle

The variational principle is one of the cornerstones of quantum mechanics and forms the foundation for many powerful variational algorithms in quantum computation. An incomplete list includes the variational quantum eigensolver (VQE), variational quantum deflation (VQD), quantum approximate optimization algorithm (QAOA) and so forth. It is also the central idea behind important algorithms in tensor network such as the Density Matrix Renormalization Group (DMRG). The variational principle provides a systematic way to approximate the ground state of a quantum system by minimizing the energy expectation value.

Variational Principle

For a Hamiltonian H that describes the studies system and any normalizable function ψ, the expectation value of the energy in this state is then

$$\varepsilon[\Psi] = \frac{\langle\Psi|\hat{H}|\Psi\rangle}{\langle\Psi|\Psi\rangle}\,. \tag{4.1}$$

The variational principle states that

- $\varepsilon \geq E_0$ where E_0 is the lowest energy eigenvalue of the Hamiltonian.
- $\varepsilon = E_0$ if and only if Ψ is exactly equal to the wave function of the ground state of $\hat{H}$.

Sometimes, this candidate state Ψ is also referred to as a "guess" wavefunction, or an **ansatz** wavefunction[1]. Note that we often minimize the energy functional without explicitly assuming the normalization for the ansatz. Taking the variational of the functional and setting

[1]The word "ansatz" is from German, "ansätze", whose original meaning is the initial placement of a tool at a work piece.

to zero leads directly to the time-independent Schrödinger equation,

$$\hat{H}|\Psi\rangle = E|\Psi\rangle\,, \tag{4.2}$$

further highlighting the connection between finding ground state and minimizing the energy functional.

In realistic many-body systems, the full Hilbert space is exponentially large and one cannot simply look for the Ψ blindly. Instead, we must restrict the search to a subset of the Hilbert space. We may assume some structure of the ansatz wave function that is expressible for the true eigenstate and parameterize it with sufficient number of parameters. Thus, we can write $|\Psi(\alpha)\rangle$ and iteratively minimize the energy $\varepsilon[\Psi(\alpha)]$ by trying out different αs. This is the main idea behind all the variational methods in quantum computation and tensor network.

Exercise 4.1 *Prove the variational principle.*

4.2 Mean field theory

From previous chapters, we have learned that the full representation of a many-body quantum system is complicated, having exponential number of degrees of freedom as in Eq.(2.3). For certain physical systems, one might be able to enforce strong approximation, for example, assuming the quantum correlations can be neglected, i.e., the many-body wave function is constrained to be a tensor product of independent single-body wave functions.

Mean Field Approximation

In the mean field approximation, the ansatz wave function becomes

$$|\Psi_{\mathrm{MF}}\rangle = |\Psi^{(1)}\rangle \otimes |\Psi^{(2)}\rangle \otimes \cdots \otimes |\Psi^{(N)}\rangle \tag{4.3}$$

$$= \sum_{i_1=1}^{d} C_{i_1}|i_1\rangle \otimes \sum_{i_2=1}^{d} C_{i_2}|i_2\rangle \otimes \cdots \otimes \sum_{i_N=1}^{d} C_{i_N}|i_N\rangle\,, \tag{4.4}$$

where each site is uncorrelated and the original exponential complexity is reduced to linear complexity $\mathcal{O}(Nd)$.

An additional assumption often used is that the wave function is translational invariant, meaning all $|\Psi^{(k)}\rangle$ are equal, so the original mean field approximation reduces to complexity to just $\mathcal{O}(d)$ independent of system size. Mean field approximation is useful in many scenarios, such as estimating the phase transition for transverse Ising model. The disadvantage of this approach is that the approximation is uncontrollable, i.e., there is no systematic way to parameterize the approximation to the true solution. More importantly, mean field theory should be used for any system where quantum correlation prevails.

Exercise 4.2 *The transverse field Ising model (TFIS) is known to exhibits quantum phase transition, where the energy density of the system becomes discontinuous at certain coupling of external fields. The Hamiltonian reads*

$$\hat{H}_{\text{TFIS}} = -\sum_{i=1}^{N-1} \sigma_i^x \sigma_{i+1}^x + \lambda \sum_{i=1}^{N} \sigma_i^z \,, \tag{4.5}$$

where σ's are Pauli matrices and λ is the external field strength. Find values of λ for quantum phase transition using mean field approximation.

To generalize the mean field theory to true solution, one can extend the mean field theory to **cluster mean field** theory,

$$|\Psi_{\text{CMF}}\rangle = \prod_{i=1}^{N/M} |\Psi_M^{(i)}\rangle = \prod_{i=1}^{N/M} \sum_{\beta_i=1}^{d^M} C_{\beta_i} |\beta_i\rangle \,, \tag{4.6}$$

where the N sites are divided into M clusters and the Hilbert space within each cluster is fully parameterized (with all sorts of quantum correlation). The complexity for the cluster mean field theory is $\mathcal{O}(Nd^M/M)$, i.e., exponential with cluster size, and taking $M = N$ reduces to the exact case. Obviously, this approximation can be useful when the physical system display clustered structure; however, the exponential scaling makes it inefficient and only applicable to small clusters.

4.3 Real-Space Renormalization Group

The real-space renormalization group method is a numerical approximation method based on a very powerful physical intuition — the ground state of a system is composed of low-energy states of the system's non-interacting bipartitions. Based on this principle, the full system can be understood by an iterative process involving truncations on smaller subsystems. The algorithm proceeds as follows:

1. Consider a small system of size N that can be studied exactly, meaning the Hamiltonian $H^{(N)}$ and its associated eigenvalues $e_i^{(N)}$ and eigenvectors $|v_i^{(N)}\rangle$ are explicitly known, i.e., $H^{(N)} = \sum e_i^{(N)} |v_i^{(N)}\rangle\langle v_i^{(N)}|$.

2. Construct a projector P onto the lowest m eigenstates, $P^{(N)} = \sum_{i=1}^{m} |v_i^{(N)}\rangle\langle v_i^{(N)}|$ such that the projected (renormalized) Hamiltonian is $\bar{H}^{(N)} = P^{(N)\dagger} H^{(N)} P^{(N)}$ and any projected (renormalized) observables are defined analogously, $\bar{O}^{(N)} = P^{(N)\dagger} O^{(N)} P^{(N)}$.

3. Construct Hamiltonian of size $2N$ using projected Hamiltonians, $H^{(2N)} = \bar{H}^{(N)} \otimes \mathbf{1} + \mathbf{1} \otimes \bar{H}^{(N)} + \bar{H}_{int}^{(N)}$ where the interaction Hamiltonian $\bar{H}_{int}^{(N)} = \bar{A}^{(N)} \otimes \bar{B}^{(N)}$ where $\bar{A}, \bar{B}$ are projected physical operators on each bipartite subsystem.

4. Repeating the steps 2-3 until convergence.

The nice feature of this approach is that the Hamiltonian dimension is kept constant $\mathcal{O}(m)$ while the renormalization procedure grows system exponentially. The numerical results can also be systematically improved with the effective Hamiltonian equal to the true solution by taking m to full problem size. However, it is apparent that truncation on the subspace for smaller system is not generally equivalent to truncation for larger system, $H_{eff} \neq P^{\dagger} H_{exact} P$. The RSRG effective Hamiltonian is generally only an approximation to the exact projected Hamiltonian and is likely to fail when discarded high-energy states become important in mediating effective interactions, or when the system is highly entangled.

Exercise 4.3 *Consider a Heisenberg chain of spins with only nearest-neighbor interactions, like*

$$\hat{H} = J \sum_{i=1}^{N-1} \boldsymbol{S}_i \cdot \boldsymbol{S}_{i+1} \tag{4.7}$$

where $\boldsymbol{S}_i = (\sigma_i^x, \sigma_i^y, \sigma_i^z)$. *Perform the RSRG procedure just once (increase the system size from* N *to* $2N$*) and compare the renormalized Hamiltonian with exact Hamiltonian. For this problem, the interacting Hamiltonian can be* $\bar{H}_{int} = \sum_\alpha \bar{\sigma}_N^\alpha \otimes \bar{\sigma}_1^\alpha$.

4.4 Density Matrix Renormalization Group

The density matrix renormalization group (DMRG) method is a numerical variational technique to solve low-energy physics of quantum many-body system with high accuracy. The original DMRG was invented by White in 1992 [7] and it is nowadays the most efficient approach for 1D system. It is often recognized as one of the most important algorithms in physics in the last 50 years. The DMRG is a powerful modification of the original RG algorithm by improving truncation rule on reduced density matrix and slowing down the growth of the system size. In DMRG, the system size increases linearly instead of exponentially with number of iterations.

Here is a brief sketch of the general procedure to the DMRG:

1. Start with a largest system of size N that can be studied exactly. Regroup the system into two single sites in the middle and two groups of M sites on the side. Construct the system's Hamiltonian as $H^{(N)} = H^{(L+1)} + H^{(int)} + H^{(R+1)}$ where $H^{(L+1)}$, $H^{(R+1)}$ are the left and right enlarged blocks, and $H^{(int)}$ is their interaction. The dimension of the full Hilbert space is then $(dm)^2$ where d (m) is dimension of single site (the grouped M sites). The diagonalization of $H^{(N)}$ yields the ground state,

$$|E_1^{(N)}\rangle = \sum_{\beta_1,\beta_2} C_{\beta_1\beta_2} |\beta_1\rangle |\beta_2\rangle \,, \tag{4.8}$$

where $|\beta_i\rangle$ spans the basis of the left and right half sites, with $\beta_i = 1, 2, ..., d^{M+1}$.

2. Compute the density matrix of the ground state, and it becomes

$$\rho_1^{(N)} = |E_1^{(N)}\rangle\langle E_1^{(N)}| = \sum_{\beta_1,\beta_2}\sum_{\beta_1',\beta_2'} C_{\beta_1\beta_2}C^*_{\beta_1'\beta_2'}|\beta_1\rangle|\beta_2\rangle\langle\beta_1'|\langle\beta_2'|\,, \tag{4.9}$$

and the reduced density matrix of the left half of the system (i.e. tracing out β_2) becomes

$$\rho_L = \mathrm{tr}_R\big(\rho_1^{(N)}\big) = |E_1^{(N)}\rangle\langle E_1^{(N)}| = \sum_{\beta_2} C_{\beta_1\beta_2}C^*_{\beta_1'\beta_2}|\beta_1\rangle\langle\beta_1'|\,. \tag{4.10}$$

3. Diagonalize $\rho_L = \sum_{i=1}^{md} w_i|w_i\rangle\langle w_i|$ and order the eigenvalues w_i in descending orders. Note w_i are non-negative real numbers such that $\sum w_i = 1$. Assuming left-right symmetric system, $\rho_L = \rho_R$. Taking only the first m eigenvectors of ρ_L, we define the projector $P = \sum_{i=1}^{m}|w_i\rangle\langle w_i|$ that truncates a Hilbert space from md to m states. The effective Hamiltonian is then $\bar{H}^{(L+1)} = P^\dagger H^{(L+1)}P$, $\bar{H}^{(R+1)} = P^\dagger H^{(R+1)}P$, and similar $\bar{H}^{(int)}$ for the interaction Hamiltonian. From here, we obtain the effective Hamiltonian of N sites of dimension m instead of md.

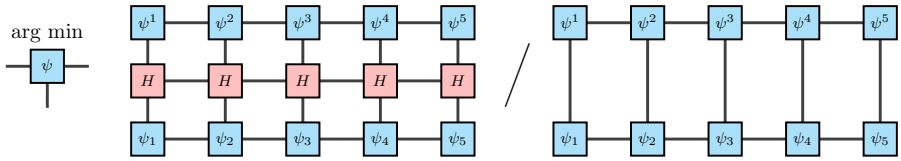
4. Iterate the steps 1-3 by replacing the left and right blocks by the newly constructed effective Hamiltonian. The net effect is that the system size increases from N to $N+2$ sites while the dimension stays as $(md)^2$.

The DMRG approach described above differs from the RSRG approaches by using a better truncation rule and a slower expansion procedure. Importantly, the DMRG and the RSRG algorithms we just described are targeted solving system as the thermodynamical limit (infinite system size). On fixed system size, instead of increasing the left and right block size, one can increase the size of the left block while decrease that of the right block, keeping N constant. At each iteration, the free sites are moved along the sites and when it is back for a second time, a DMRG **sweep** is completed. Additional sweeps will result in higher precisions and improved precisions.

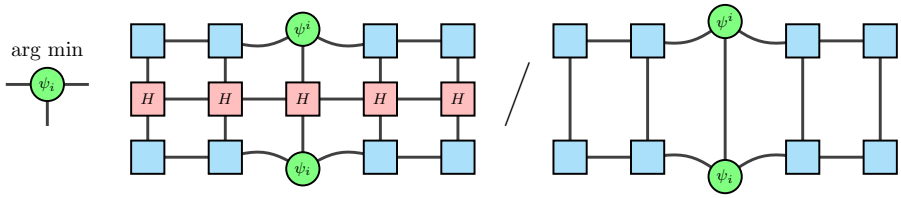
Both algorithms pre-dates the tensor network and their description is far more complicated technically. Within the tensor network paradigm, the original DMRG is now interpreted differently. We focus on the DMRG in the age of the MPS in the following sections; see Ref. [5] for a comprehensive review.

4.5 Ground state and excited states with MPS

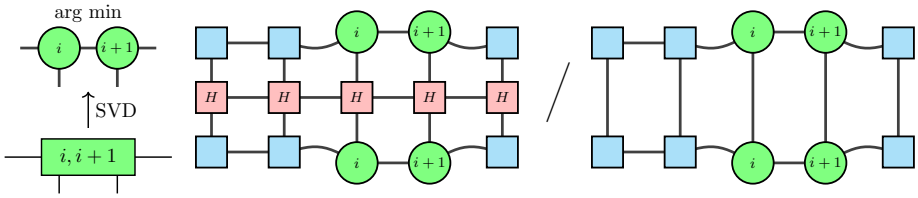
The goal of the DMRG for the MPS is simple: finding the lowest energy states by solving the optimization problem. Going back to Eq. (4.1), given a Hamiltonian MPO $\hat{H}$ and a MPS ansatz state $|\Psi\rangle = \sum_{i_1 i_2 ... i_N=0}^{d-1} A_{i_1}^{(1)}A_{i_2}^{(2)}...A_{i_N}^{(N)}\,|i_1 i_2 ... i_N\rangle$, we want to find all those coefficients $A_{i_k}^{(k)}$ such that the energy expectation $\varepsilon[\Psi] = \langle\Psi|\hat{H}|\Psi\rangle\ /\ \langle\Psi|\Psi\rangle$ is minimized. Pictorially, this task is represented in Fig. 4.1(a).



(a) Generic $\varepsilon[\Psi] = \langle\psi|H|\psi\rangle \,/\, \langle\psi|\psi\rangle$.



(b) One-site DMRG.



(c) Two-site DMRG.

Figure 4.1: Illustration of the DMRG methods for one-site and two-site MPS.

(a) The most naive approach is to calculate the contraction of the expectation and normalization for all possible MPS $|\Psi\rangle$ to find the minimum of the objective function. This is clearly not scalable.

(b) The DMRG approach to this optimization problem takes full advantage of canonical property of the MPS. Instead of variationally minimize the state everywhere, in the one-site DMRG, we move on the MPS sites in one direction (for example from left to right) and variationally minimize one tensor at a time. By putting the MPS on the left in the left-canonical forms and the MPS on the right in the right-canonical forms, the normalization reduces to just identity (recall Fig. 3.3). The expectation $\langle\Psi|H|\Psi\rangle$ can also be computed efficiently by contracting site-by-site using environment updates. See Fig. 4.1(b) for an illustration of the one-site DMRG.

The advantage of one-site DMRG is that the bond dimension in the MPS is fixed, since one only variationally modify the node itself. This can also be disadvantage as there is no way to understand if the bond dimension is sufficient. To address this problem, the two-site DMRG is introduced.

(c) In the two-site DMRG approach (commonly used nowadays and default in `iTensor` library), two-sites are variationally minimized at a time and their internal bond can be modified; see Fig. 4.1(c) for an illustration. The two tensors are blocked together first and minimized. Then using SVD, the individual tensors are determined with appropriate bond dimension set by the numerical threshold or maximum bond dimension. Using truncated singular values, we can quantify the numerical error associated with cutting off the bond dimension (quantum entanglement) in the system.

The advantage of two-site DMRG is clear. The bond dimension in the MPS may be dynamically adjusted in each sweep to better represent the problem of interests, though the computational cost is much higher.

Exercise 4.4 *Practice using DMRG from Julia on several useful model in 1D that are typically used as benchmark calculations.*

1. *Transverse-field Ising model*
2. *Heisenberg chain*
3. *1D Hubbard model*

Besides generic DMRG approach, one can also restrict the MPS to physical meaningful subspace such that the performance of the DMRG can be improved even further. This procedure is known as quantum number preserving DMRG. For example, if we want to only look for charge zero or spin zero states in the desired Hamiltonian. Another helpful aspects of the DMRG is that this approach can extended to either qubits, qudits, or Bosonic systems.

From the introduced DMRG method, we can find the ground state for 1D gapped system efficiently. Now, we extend the method to find excited states. The most straightforward approach targets specific excited states by modifying the optimization problem. Instead of minimizing $\langle\Psi|H\Psi\rangle$, we minimize

$$\varepsilon_n[\Psi] = \min_{\psi_n}\langle\Psi_n|H|\Psi_n\rangle \;/\; \langle\Psi_n|\Psi_n\rangle, \quad \langle\Psi_n|\Psi_m\rangle = 0,\, \forall m < n \tag{4.11}$$

where $\{\Psi_n\}$ are the previously found lower-energy states with $|\Psi_0\rangle$ the ground state. This orthogonality constraint ensures we find the next excited state.

In practice, one can add a penalty term to the Hamiltonian

$$\hat{H}^{'} = \hat{H} + \sum_{n=0}^{m} \omega_n |\Psi_n\rangle\langle\Psi_n| \tag{4.12}$$

where ω_n are large positive penalties. Minimizing $\langle\hat{H}^{'}\rangle$ naturally avoids previously found states. Importantly, the penalty weight needs to be chosen as least as large as the estimated gap between the ground and first excited states, or generally $\omega_n > \varepsilon_n - \varepsilon_{n-1} = \Delta\varepsilon_n$, $\forall n > 0$. Otherwise, the optimization will return the ground state when calculating the excited states. In practice, using $\omega_n \simeq 1.5\Delta\varepsilon_n$ is desirable.

Exercise 4.5 *Prove the criterion of ω used for successful optimization of the excited states. One can assume there are only two distinct eigenstates in the system ground state $|\Psi_0\rangle$ with eigenvalue ε_0 and the first excited state $|\Psi_1\rangle$ with eigenvalue ε_1, and discuss the following three cases:*

- $\omega_1 < \Delta\varepsilon$
- $\omega_1 > \Delta\varepsilon$
- $\omega_1 = \Delta\varepsilon$

where $\Delta\varepsilon = \varepsilon_1 - \varepsilon_0$ is the energy gap.

4.6 Exercise

The hands-on exercise is available on Github. Exercises for this chapter focus on using DMRG to find MPS round states of one-dimensional system.

Chapter 5

Time Evolution via Tensor networks

5.1 Time-Evolving Block Decimation

The Time-Evolving Block Decimation (TEBD) is an efficient algorithm for simulating real-time and imaginary-time evolution of quantum many-body systems using MPS introduced by Vidal in 2007 [8]. The idea is to approximate the time evolution operator using Trotter decomposition and apply it sequentially to an MPS while controlling entanglement growth through SVD truncation.

Trotter Decomposition

For a local Hamiltonian $H = \sum_i h_{i,i+1}$ (nearest-neighbor), the evolution operator is

$$e^{-iHt} = e^{-it\sum_i h_{i,i+1}}. \tag{5.1}$$

To approximation the evolution operator, various Trotter decomposition can be used. Using Lie–Trotter decomposition (first order approximation),

$$e^{-iH\delta t} \approx e^{-ih_{1,2}\delta t} e^{-ih_{2,3}\delta t} \cdots e^{-ih_{N-1,N}\delta t} + \mathcal{O}(\delta t^2). \tag{5.2}$$

Using the Suzuki-Trotter (second-order) yield more accurate results:

$$e^{-iH\delta t} \approx e^{-ih_{\text{odd}}\delta t/2} e^{-ih_{\text{even}}\delta t} e^{-ih_{\text{odd}}\delta t/2} + \mathcal{O}(\delta t^3) \tag{5.3}$$

where

$$h_{\text{odd}} = \sum_{\text{odd } i} h_{i,i+1} \quad \text{and} \quad h_{\text{even}} = \sum_{\text{even } i} h_{i,i+1}. \tag{5.4}$$

The TEBD Algorithm uses the following steps:

1. Initialize with MPS $|\Psi(t=0)\rangle$ in canonical form.

2. Apply gates for a trotter timestep δt. Specifically applying two-site gates $U_{i,i+1} = e^{-ih_{i,i+1}\delta t}$ sequentially depending on Trotter order.

3. Perform SVD and truncate to maximal bond dimension D.

4. Update the MPS by absorbing singular values.

5. Repeat steps 2-4 for a total time $t = N_{\text{step}}\delta t$.

Though we described the algorithm and the trotterization in a real-time simulation setting. It is obvious to see that these algorithmic steps pay no regard to real-time or imaginary-time evolution. While real-time evolution can deal with simulation of quantum dynamics such as studying scattering and quenching phenomena, imaginary-time evolution can be used to study thermal equilibrium properties. Alternative to the DMRG method, imaginary-time evolution can also be used to use ground state, for $\exp(-\tau H)|\Psi(t=0)\rangle \to |\Psi_0\rangle$ taking $\tau \to \infty$.

For long-range interactions, the TEBD is modified with swap gates or MPO-based evolution. The total error in a TEBD calculation comes from two sources, the trotter error and the truncation error (discarded singular values). The trotter error can accumulate over time. Another key challenge is that the entanglement typically grows exponentially, $S(t) = \mathcal{O}(e^t)$, which limits long-time evolution. Overall, the total scaling for the TEBD is $\mathcal{O}(Nd^3D^3)$ per time step using second-order trotterization for N site d-level systems of bond dimension D.

Exercise 5.1 *Implement the TEBD method to simulate the transverse Ising model. See Eq. (4.5) for the Hamiltonian.*

1. *Compare the result with exact diagonalization for $N = 12$.*
2. *Experiment with the number of lattice sites N and the bond dimension D to see the accuracy as well as limit of the TEBD algorithm.*

5.2 Time-Dependent Variational Principle

The time-dependent variational principle (TDVP) [9, 10] is a more sophisticated approach to time evolution that treats the MPS manifold as a Riemannian manifold and evolves states along geodesics, avoiding Trotter errors. As the name suggests it is a variational method. The time-dependent Schrödinger equation $i\partial_t|\Psi(t)\rangle = \hat{H}|\Psi\rangle$ becomes a variational problem on the MPS manifold $\mathcal{M}_D$:

$$\text{Find } |\Psi(t)\rangle \in \mathcal{M}_D \text{ such that } \langle\delta\Psi(t)|i\partial_t - \hat{H}|\Psi(t)\rangle = 0 \tag{5.5}$$

for all tangent vector $|\delta\Psi(t)\rangle$ to the manifold. Since exact evolution may leave the MPS manifold, the TDVP needs to project the time derivatives back onto the tangent space

$$i\partial_t|\Psi(t)\rangle = \mathrm{P}_\perp(H|\Psi(t)\rangle) \tag{5.6}$$

where $\mathrm{P}_\perp$ is the projector onto the tangent vector of $\mathcal{M}_D$ at $|\Psi(t)\rangle$.

Just like the DMRG algorithms, the TDVP also has a one-site version and a two-site version. As expected, the one-site version evolves a single tensor at a time with fixed bond-dimensions. It is fast and preserves the MPS structure, but requires good initial guess and cannot increase the bond dimension. The two-site TDVP is more often used, which evolves pairs of neighboring tensors, allowing the bond dimension to grow adaptively according to the physical problem. Overall, the one-site (two-site) TDVP algorithm scales as $\mathcal{O}(Nd^2D^4)$ ($\mathcal{O}(Nd^4D^4)$) per time step for N site d-level systems of bond dimension D.

Compared with the TEBD, the TDVP is generally more preferred, as there is no trotter error and the long-time stability is better. The TDVP also exactly conserves energy and normality, while the TEBD violates it; see the following exercise.

Exercise 5.2 *Use TEBD and TDVP algorithms to compute the energy expectation the Heisenberg chain starting with the ground state. Comparing to the exact solution, show the error in both approaches.*

5.3 Simulating quantum circuit with tensor network

In general, there is no theorem stating that quantum computing and tensor networks are equivalent, but there is a very precise and useful relationship between them. Every quantum circuit can be exactly represented as a tensor network:

- Each quantum gate corresponds to a rank-4 tensor
- Each qubit "wire" corresponds to an index
- Composing gates corresponds to contracting indices

This mapping is completely general and shows that quantum computation is a special case of tensor-network computation. Since we only need two-qubit basic gates in quantum computing to universally describe any circuit, a rank-4 tensor is only required. In this sense, one may view a quantum circuit simply as a particular tensor network with unitarity constraints and a fixed geometry. Notably, this mapping does not imply equivalence of computational power.

The reverse direction is not true in general. Tensor networks form a much broader class of mathematical objects: they may have arbitrary bond dimensions, arbitrary geometries, and tensors that need not correspond to physical unitary operations. As a consequence, contracting a general tensor network is a **P-hard** (polynomial-hard) task, believed to be much harder than simulating a quantum circuit. On the other hand, quantum circuit simulation is generally considered to be **BQP** (Bounded-error Quantum Polynomial)[1]. Only special subclasses—such as the MPS with polynomial bond dimension and a few others correspond

[1] BQP is the class of all computational problems that can be solved efficiently on a quantum computer with a bounded probability of error.

closely to efficiently preparable quantum states. Nonetheless, it is still under debate which of the two complexity class is larger and under what context.

To simulate a quantum circuit using tensor networks, we simply translate each gate into its corresponding tensor and then contract the network according to the circuit's wiring pattern. For example, a single-qubit state $|\Psi\rangle$ is represented as a rank-1 tensor; applying a gate U corresponds to contracting U_{ij} with Ψ_j; and applying a two-qubit gate G_{ijkl} corresponds to contracting its indices with two qubit state tensors. The entire circuit becomes a large tensor network whose final contraction yields the output amplitudes $\langle x|U\Psi\rangle$. This formulation not only provides a clear mathematical picture of quantum circuits but also forms the basis of classical tensor-network simulators for quantum computation. See `Qiskit Aer` tutorial for a concise demonstration.

Exercise 5.3 *Simulating the quantum circuit with tensor network backend using `Qiskit Aer`'s MPS simulator.*

5.4 Exercise

The hands-on exercise is available on Github. Exercises for this chapter focus on using TEBD and TDVP algorithms to perform various real-time evolutions.

Bibliography

[1] S. Montangero, *Introduction to Tensor Network Methods: Numerical simulations of low-dimensional many-body quantum systems*, Springer International Publishing (2018).

[2] R. Orus, *A Practical Introduction to Tensor Networks: Matrix Product States and Projected Entangled Pair States*, *Annals Phys.* **349** (2014) 117 [arXiv:1306.2164].

[3] J.C. Bridgeman and C.T. Chubb, *Hand-waving and Interpretive Dance: An Introductory Course on Tensor Networks*, *J. Phys. A* **50** (2017) 223001 [arXiv:1603.03039].

[4] M.C. Bañuls, *Tensor Network Algorithms: A Route Map*, *Ann. Rev. Condensed Matter Phys.* **14** (2023) 173 [arXiv:2205.10345].

[5] U. Schollwöck, *The density-matrix renormalization group in the age of matrix product states*, *Annals of physics* **326** (2011) 96.

[6] M. Fishman, S.R. White and E.M. Stoudenmire, *The ITensor Software Library for Tensor Network Calculations*, *SciPost Phys. Codebases* (2022) 4.

[7] S.R. White, *Density matrix formulation for quantum renormalization groups*, *Phys. Rev. Lett.* **69** (1992) 2863.

[8] G. Vidal, *Classical simulation of infinite-size quantum lattice systems in one spatial dimension*, *Phys. Rev. Lett.* **98** (2007) 070201.

[9] J. Haegeman, C. Lubich, I. Oseledets, B. Vandereycken and F. Verstraete, *Unifying time evolution and optimization with matrix product states*, *Phys. Rev. B* **94** (2016) 165116 [arXiv:1408.5056].

[10] J. Haegeman, J.I. Cirac, T.J. Osborne, I. Pizorn, H. Verschelde and F. Verstraete, *Time-Dependent Variational Principle for Quantum Lattices*, *Phys. Rev. Lett.* **107** (2011) 070601 [arXiv:1103.0936].