Chapter 2

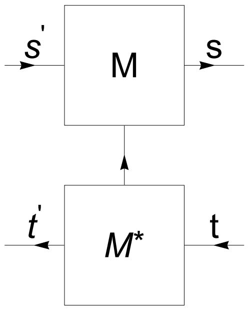
Manipulating Tensor Networks

A. Transfer Matrices

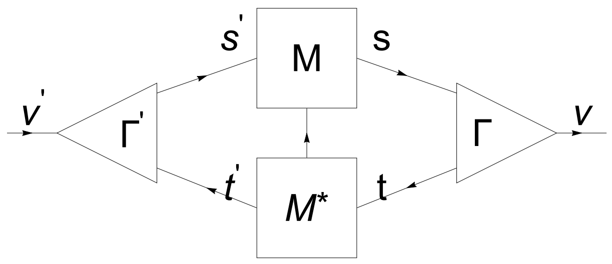
At the end of last day we began thinking about the matrix product state

$$|\psi\rangle = \cdots \begin{pmatrix} |\uparrow\rangle & \epsilon|\downarrow\rangle \\ \epsilon|\uparrow\rangle & |\downarrow\rangle \end{pmatrix} \begin{pmatrix} |\uparrow\rangle & \epsilon|\downarrow\rangle \\ \epsilon|\uparrow\rangle & |\downarrow\rangle \end{pmatrix} \begin{pmatrix} |\uparrow\rangle & \epsilon|\downarrow\rangle \\ \epsilon|\uparrow\rangle & |\downarrow\rangle \end{pmatrix} \cdots \tag{2.1}$$

We argued that the first step in finding the norm of this state was to construct the "transfer matrix"

$E_I =$  . (2.2)

This is a rank 4 tensor, each index taking on one of two values. It is however, natural to group the left two indices and the right two indices to write this as a 4×4 matrix. That is, we create a matrix whose rows correspond to $st = 00, 01, 10, 11$. One usually just does this in one's head, but one can also formally add graphical "combiners"

$E_I =$  (2.3)

where the rank 3 tensors have non-zero elements:

$$\Gamma^{s_0 t_0}_{v_{00}} = \Gamma^{s_1 t_0}_{v_{10}} = \Gamma^{s_0 t_1}_{v_{01}} = \Gamma^{s_1 t_1}_{v_{11}} \tag{2.4}$$

It is just a trivial relabeling of two indices as one composite index. In fact, when we write computer programs, we will make functions which exactly do that.

The reason for calculating E_I is that if we use periodic boundary conditions the norm is just $\langle\psi|\psi\rangle = \operatorname{Tr}(E_I)^N$. That means if λ is the largest eigenvalue of E_I, the norm is $\langle\psi|\psi\rangle = \lambda^N$. The state can then be normalized by dividing each M by $1/\sqrt{\lambda}$.

I find it is easiest to use Mathematica to calculate E_I – and later today we will begin to work on those tools. Just this one time we will also do it by hand. I like to first write down the matrix elements of $M^{\sigma}_{ss'}$:

$$M^{\uparrow}_{s_0 s_0'} = 1 \tag{2.5}$$
$$M^{\uparrow}_{s_0 s_1'} = \epsilon \tag{2.6}$$
$$M^{\downarrow}_{s_1 s_0'} = \epsilon \tag{2.7}$$
$$M^{\downarrow}_{s_1 s_1'} = 1 \tag{2.8}$$
$$\tag{2.9}$$

Then I consider the 16 matrix elements. There are four elements set by the $\sigma =\uparrow$ terms, ie those with $ss' = 00, 01$ and $tt' = 00, 01$. They yield

$$E^{s_0 s_0'}_{t_0 t_0'} = M^{\uparrow}_{s_0 s_0'}(M^{\uparrow}_{t_0 t_0'})^* = 1 \tag{2.10}$$
$$E^{s_0 s_1'}_{t_0 t_0'} = M^{\uparrow}_{s_0 s_1'}(M^{\uparrow}_{t_0 t_0'})^* = \epsilon \tag{2.11}$$
$$E^{s_0 s_0'}_{t_0 t_1'} = M^{\uparrow}_{s_0 s_0'}(M^{\uparrow}_{t_0 t_1'})^* = \epsilon^* \tag{2.12}$$
$$E^{s_0 s_1'}_{t_0 t_1'} = M^{\uparrow}_{s_1 s_1'}(M^{\uparrow}_{t_1 t_1'})^* = |\epsilon|^2. \tag{2.13}$$

There are then four set by the $\sigma =\downarrow$ terms,

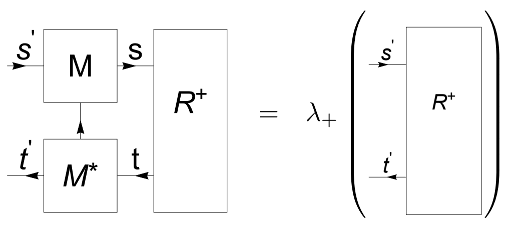
$$\begin{aligned} E^{s_1 s_1'}_{t_1 t_1'} &= 1 && (2.14)\\ E^{s_1 s_0'}_{t_1 t_1'} &= \epsilon && (2.15)\\ E^{s_1 s_1'}_{t_1 t_0'} &= \epsilon^* && (2.16)\\ E^{s_1 s_0'}_{t_1 t_0'} &= |\epsilon|^2. && (2.17)\end{aligned}$$

I then fill in the matrix elements

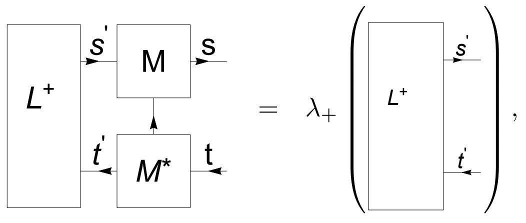
$$E_0 = \begin{pmatrix} E^{s_0 s_0'}_{t_0 t_0'} & E^{s_0 s_0'}_{t_1 t_0'} & E^{s_1 s_0'}_{t_0 t_0'} & E^{s_1 s_0'}_{t_1 t_0'} \\ E^{s_0 s_0'}_{t_0 t_1'} & E^{s_0 s_0'}_{t_1 t_1'} & E^{s_1 s_0'}_{t_0 t_1'} & E^{s_1 s_0'}_{t_1 t_1'} \\ E^{s_0 s_1'}_{t_0 t_0'} & E^{s_0 s_1'}_{t_1 t_0'} & E^{s_1 s_1'}_{t_0 t_0'} & E^{s_1 s_1'}_{t_1 t_0'} \\ E^{s_0 s_1'}_{t_0 t_1'} & E^{s_0 s_1'}_{t_1 t_1'} & E^{s_1 s_1'}_{t_0 t_1'} & E^{s_1 s_1'}_{t_1 t_1'} \end{pmatrix} = \begin{pmatrix} 1 & & & |\epsilon|^2 \\ \epsilon^* & & & \epsilon \\ \epsilon & & & \epsilon^* \\ |\epsilon|^2 & & & 1 \end{pmatrix} \qquad (2.18)$$

The eigenvalues are $0, 0, 1-|\epsilon|^2, 1+|\epsilon|^2$. Thus to normalize the state, we would divide each M by $\sqrt{1+|\epsilon|^2}$. We won't do that, but will instead work with a non-normalized state, and make sure to divide by the normalization when we take expectation values. Either way works.

In your homework you will work through calculating local expectation values, and correlation functions in this state. The math is straightforward, but ugly enough that I don't want to do it in front of you. Here is the basic approach. Let R_+ be the right eigenvector of M associated with the eigenvalue $\lambda_+ = 1+|\epsilon|^2$, and let R_- be the one associated with $\lambda_- = 1-|\epsilon|^2$. Similarly L_+ and L_- are the left eigenvectors. In our graphical notation, we would write



$$= \lambda_+ \qquad (2.19)$$

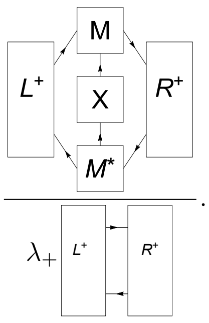


$$= \lambda_+ \qquad , \qquad (2.20)$$

and similar expressions with $+$ replaced by $-$.

Using standard linear algebra – which I will briefly review, we can now calculate the expectation value of some local operator X via

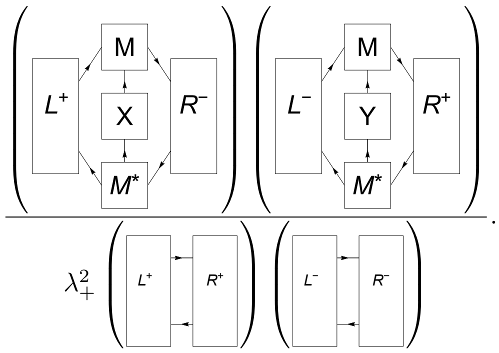
$$\langle X \rangle =$$



(2.21)

Two-point correlation functions $\langle X_i Y_j \rangle$ (with $i < j-1$) will depend on the distance $d = j - i$. We will find

$$\langle X_i Y_j \rangle = \langle X \rangle \langle Y \rangle + \left(\frac{\lambda_-}{\lambda_+}\right)^{j-i-1}$$



(2.22)

Before deriving this result, a few comments are in order. All correlation functions falls off exponentially with distance: the correlation length is $\xi = 1/\log|\lambda_+/\lambda_-|$. As will be clear from the derivation, this is true for any matrix product state. Making the M matrices bigger will add more terms, each of which will decay exponentially with a correlation length. Matrix product states are therefore potentially efficient for modeling states with exponentially decaying correlations.

You may be aware that correlations in metals are power laws. Therefore approximating some properties of metals will require using very large M's, and that at long enough distances the model will break down. The phrase you often hear is that matrix product states are good for describing gapped states. We will investigate this more next day.

B. Linear Algebra

If you are like me, you are used to working with Hermitian matrices, and do not remember any thing about left and right eigenvectors, and may feel a bit nervous about eigendecompositions of non-Hermitian matrices. The Cliff's notes version is that suppose we have a $n \times n$ matrix M which can be diagonalized. That is, it can be written

$$M = Q^{-1} \Lambda Q \tag{2.23}$$

where Λ is a diagonal matrix, and Q is some matrix. Unlike the Hermitian case that you are used to, the matrix Q is generally *not* Unitary. We will denote the rows of Q as $L_1, L_2, \cdots L_n$. If Q was unitary, these would be orthogonal to each-other, but generically they are not. We will denote the columns of Q^{-1} as $R_1, R_2, \cdot R_n$. Since the product of a matrix and its inverse is the identity, $QQ^{-1} = I$ we have

$$L_i \cdot R_j = \delta_{ij}. \tag{2.24}$$

We can use this property to see that the R's are right eigenvectors, and the L's are left eigenvector. We begin by noting that the product QR_i is a vector where all components are 0, except for the i'th, which is 1. Borrowing quantum mechanics language, we would call this vector $|i\rangle$. Thus we can write

$$M|R_i\rangle = Q^{-1}\Lambda Q|R_i\rangle = Q^{-1}\Lambda|i\rangle = Q^{-1}|i\rangle\lambda_i, \tag{2.25}$$

but $Q^{-1}|i\rangle$ is identically the i'th column of Q^{-1}, which is R_i. Therefore R_i is an eigenvector of M with eigenvalue λ_i,

$$MR_i = \lambda_i R_i. \tag{2.26}$$

The same argument from the left shows that

$$L_i M = \lambda_i L_i. \tag{2.27}$$

These expression define what we mean by eigenvectors. Note that Q is not unique: I can multiply each row of Q by an independent number, and each column of Q^{-1} by the inverse of that number, and I have an equally good eigendecomposition.

Not all matrices are diagonalizable, but it turns out that the space of non-diagonalizable matrices has zero measure. For non-diagonalizable matrices, there is a normal form, known as the Jordan form, which we can use to do similar stuff – we will put of discussion of it to a future class..

Another way to interpret Eq. (2.23) is that

$$M = \sum_j |R_j\rangle\lambda_j\langle L_j|. \tag{2.28}$$

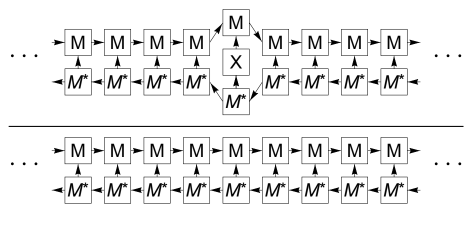
Actually, this form pre-supposes that I have normalized my left and right eigenvectors, so it is more convenient to write

$$M = \sum_j \frac{|R_j\rangle\lambda_j\langle L_j|}{\langle L_j|R_j\rangle}. \tag{2.29}$$

We know this form is right because it is "gauge invariant" – if I multiply L_j or R_j by an arbitrary number, it is unchanged.

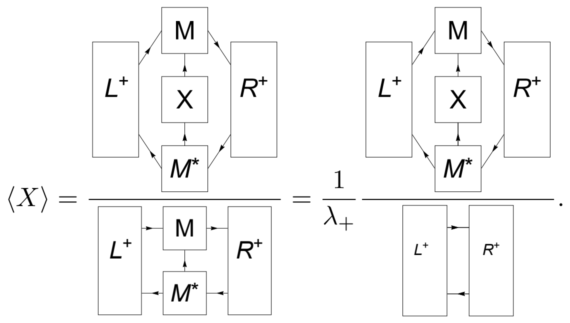
We can now derive our expressions. First, the expectation value of any local operator X is

$\langle X\rangle =$



(2.30)

If I put an arbitrary vector at infinity (which is not orthogonal to the dominant eignevectors) then all the E's on the right applied to it eventually turn it into the dominant right eigenvector. Similarly the E's on the left turn it into the dominant left eigenvector. There is some numerical factor, but the same factor appears in both the numerator and denominator, so



(2.31)

The argument for the correlator is similar, and is a homework problem.

C. Working with Tensors in Mathematica

We will now do a quick Mathematica Session to work on putting some of this technology into practice.

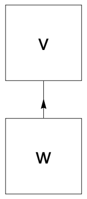
$$\boxed{\text{Complete the notebook “TensorIntro1.nb”.}}$$

D. Contracting Networks

As we saw last day, one can use tensor networks to describe quantum mechanical wavefunctions. A common operation one needs to do with tensor networks is contract them. In the examples we looked at so far, we did this in an ad-hoc way. We are now going to explore the computational cost of contracting tensors, and will see that the order in which one contracts a tensor network can make a big difference on the cost.

$$\boxed{\text{How many operations are needed to take a dot-product? Multiply a matrix and a vector? Multiply two matrices? Multiply two tensors?}}$$

Lets begin with the simplest contraction, a dot product between two vectors:

$\vec{v}\cdot\vec{w} =$ 

(2.32)

If the index being contracted can take on d separate values, then we need to perform d independent multiplication, and then add these numbers. This can be parallelized, but if we think about serial operations, this is of order d operations. Before the operation we are storing $2d$ numbers, after it we are storing 1 number.

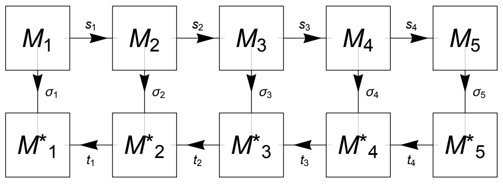
Now consider multiplying a matrix times a vector. If the matrix is $d_1 \times d_2$, and the vector is length d_2, we need to do $d_1 \times d_2$ multiplications, and a bunch of additions. Before the operation we are storing $(d_1 + 1) \times d_2$ numbers, and afterwards, just d_2.

Matrix multiplications: A $d_1 \times d_2$ matrix multiplied by a $d_2 \times d_3$ requires $d_1 \times d_2 \times d_3$ operations.

Tensor multiplication: A $d_1 \times d_2 \times \cdots d_n$ tensor where we contract the n'th index with at $d_n \times d_{n+1} \times \cdots d_{n+m}$ tensor requires $d_1 \times d_2 \times \cdot d_{n+m}$ operations, and result in $d_1 \times d_2 \times \cdot d_{n+m}/d_n$ numbers to be stored. Tensor multiplication for high rank tensors can become expensive – moreover, the number of numbers that you have to store goes up as you multiply the tensors together. In some sense this is why tensor networks are useful for encoding quantum states: they are some sort of compression method.

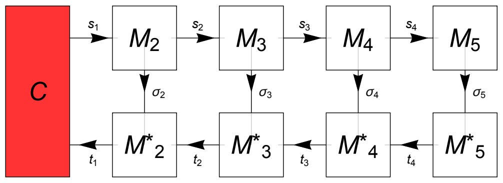
A consequence of this logic is that the order that you contract a tensor network changes the cost. Consider one of the contractions from last day:

$$\langle\psi|\psi\rangle = \qquad (2.33)$$

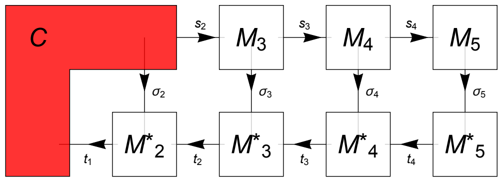


Suppose all of the s and t's take on $\chi = 3$ possible values, and all the σ's take on $d = 2$ possible values, what is the most efficient way to contract this network?

The cheapest one to contract is σ_1 – which takes $3 \times 3 \times 2$ operations. After the contraction, the network looks like



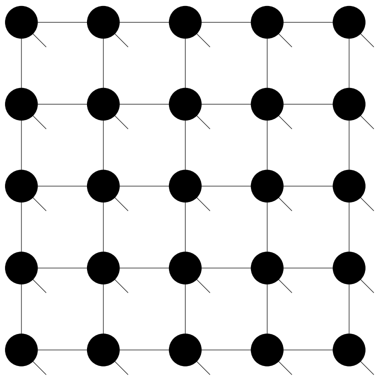
Next we do s_1, which takes $3 \times 3 \times 2$ operation, and yields



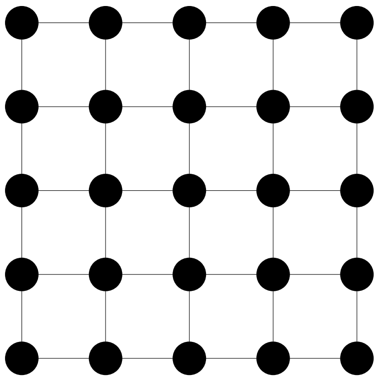
We then contract s_1 and σ_2. These can be done simultaneously, or one at a time. If we do then simultaneously, there are $3 \times 3 \times 3 \times 2$ operations. If we do them one at a time, the first one takes $3 \times 3 \times 3 \times 2$ operations, and the second $3 \times 3 \times 2$ operations. Clearly there is no gain from doing them one at a time.

In the end it takes of order $N \times \chi^3 \times d$ operations to take the overlap between two matrix product states. This is pretty good, considering the Hilbert space is exponential in N.

The most natural generalization of a matrix product state to two dimensions is a "PEPS" – which is a tensor network which look like

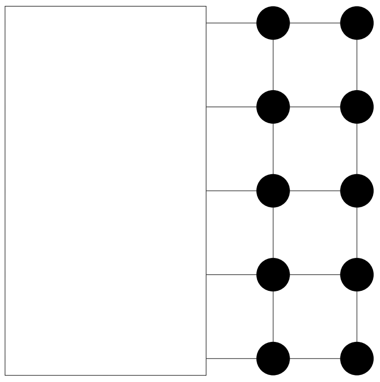


The dots represent rank 5 tensors, and the dangling bonds are the physical spin indices. Exactly contracting a tensor network like this is extremely expensive. Imagine we just need to contract something like



Estimate the number of operations needed to contract this network.

The problem is that at some point one needs to have a very high rank tensor, such as



Just storing this tensor, let alone operating with it, requires χ^L numbers, where L is the linear size of the system. Contracting such a network requires a number of operations which scales as the exponential of the square root of the total number of tensors. One important area is the development of algorithms to efficiently approximate such contractions. Another is finding tensor networks which are easy to contract, but do a good job of representing quantum mechanical states

E. Singular Value Decomposition

In addition to the operation of contraction, it is important to have the opposite operation: breaking a high ranking tensor into several lower ranking ones, for example,

The tool we will use for this is the singular value decomposition (SVD). This is an algorithm which lets you write a $n \times m$ matrix as the product

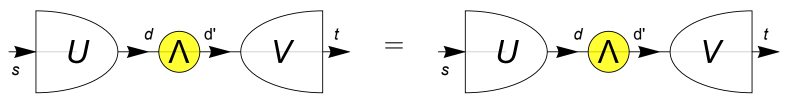
$$M = U\Lambda V \tag{2.34}$$

where U is $n \times d$ matrix, Λ is a diagonal $d \times d$ matrix, and V is $d \times m$, with $d \leq \min(m, n)$. The entries in Λ are the "singular values," the d rows of V are orthonormal, and the d columns of U are orthonormal:

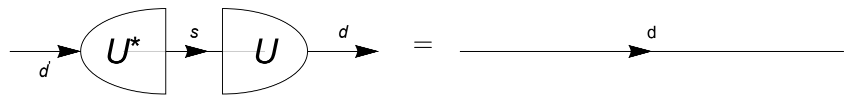
$$VV^\dagger = I \tag{2.35}$$

$$U^\dagger U = I \tag{2.36}$$

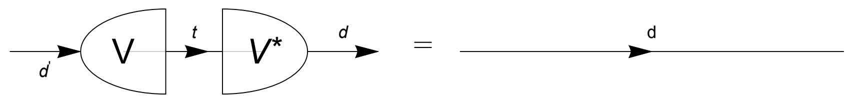
Graphically these relations can be written:



(2.37)



(2.38)



(2.39)

I used these curved shapes as a mnemonic about the orthonormality of the columns or rows of the U or V. The idea is that they are two half-circles, that get put together to make a circle – which is what I am using for a diagonal matrix.

For a Hermitian matrix, the singular value decomposition is identical to the eigenvalue decomposition. Generically, however, they are different, and hold different meanings.

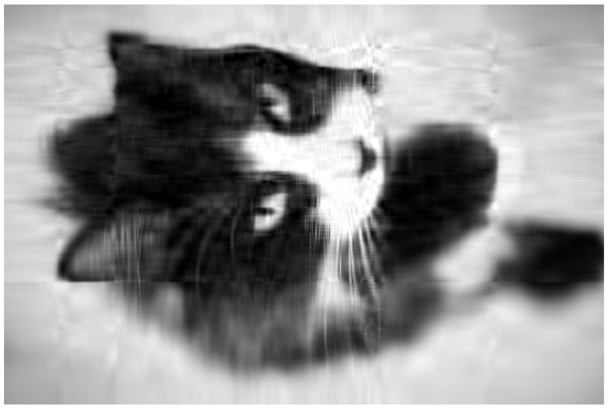
The way I like to think about unitary matrices, is that they are coordinate transforms. They take one orthonormal basis, and transform it into another, preserving all lengths and angles. In this way of thinking, a rectangular row-orthogonal matrix, such as V or $U^\dagger$ is a projector into a lower dimensional subspace. The operator $V^\dagger$ is the pull-back into that bigger space. The fact that $VV^\dagger = I$, tells us that $V^\dagger$ is an embedding of the small space into the big space.

What is the meaning of $VV^\dagger$?

The operator $VV^\dagger$ is the projector into the image of V.

The real power of the SVD comes from using it to approximate matrices. If we throw away the smallest singular values, we develop an approximation to our original matrix. We can see how this works with images. We can interpret this image of a cat as a matrix with $600 \times 400 = 240,000$ entries:

If we do a SVD, and keep only the 20 largest singular values, we need only store $600 \times 20 + 400 \times 20 + 20 = 20,020$ numbers, and get a reasonable cat:



The reason that this works is that the rows and columns of the cat picture are correlated.

E.1. Relationship to other decompositions

One useful relationship is that one can relate the singular value decomposition to the eigenvalue decomposition of $M^\dagger M$ and $MM\dagger$:

$$M^\dagger M = (V^\dagger \Lambda U^\dagger)(U\Lambda V) = V^\dagger \Lambda^2 V \tag{2.40}$$

$$MM^\dagger = (U\Lambda V)(V^\dagger \Lambda U^\dagger) = U\Lambda^2 U^\dagger. \tag{2.41}$$

Thus the singular values are the square roots of the eigenvalues of $M^\dagger M$ and $MM^\dagger$. The rows of V are the eigenvectors of $M^\dagger M$, while the rows of $U^\dagger$ are the eigenvectors of $MM^\dagger$.

The other common connection is to the "Polar Decomposition" – the matrix equivalent of writing $z = re^{i\theta}$. In particular

$$M = U\Lambda V = (UV)(V^\dagger \Lambda V) = (U\Lambda U^\dagger)(UV). \tag{2.42}$$

The combination UV is unitary, and the others are Hermitian (and even already diagonalized).

F. Entanglement

Suppose you divide a quantum system into two parts, A and B. Consider an arbitrary wavefunction $|\psi\rangle_{AB}$. The Schmidt decomposition of ψ is

$$|\psi\rangle_{AB} = \sum_j \lambda_j |\phi^j\rangle_A |\theta^j\rangle_B, \tag{2.43}$$

with

$$\langle\phi^i|\phi^j\rangle = \delta_{ij} \tag{2.44}$$

$$\langle\theta^i|\theta^j\rangle = \delta_{ij}. \tag{2.45}$$

This is nothing but the Singular Value Decomposition. The singular values, λ_j, are also referred to as the Schmidt values. If the original state was properly normalized then

$$\sum_j \lambda_j^2 = 1. \tag{2.46}$$

The Schmidt values tells you about how much subsystem A cares about subsystem B. If there is only one non-zero Schmidt value, you have a product state, and the two parts of the system are independent. Otherwise the parts are entangled. One quantifies the entanglement via the *entanglement entropy*

$$S_{ab} = -\sum_j \lambda_j^2 \log \lambda_j^2. \tag{2.47}$$

Another way to understand this decomposition, is to trace out part of the system:

$$\rho_A = \mathrm{Tr}\Big[|\psi\rangle_{AB}\ {}_{AB}\langle\psi|\Big] = \sum_\nu {}_B\langle\nu|\psi\rangle_{AB}\ {}_{AB}\langle\psi|\nu\rangle_B, \tag{2.48}$$

where we have explicitly stated which space each vector lives in. We can do the trace in any basis. In particular, we can use the $|\theta^i\rangle$ as a nice orthonormal basis, to get

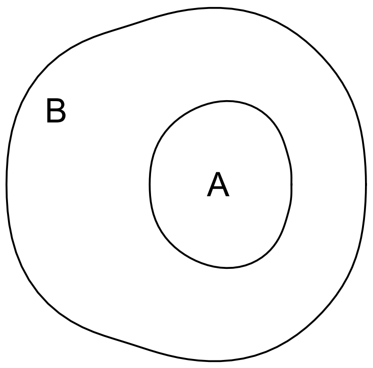
$$\rho_A = \sum_j \lambda_j^2 |\phi_j\rangle\langle\phi_j|. \tag{2.49}$$

We therefore see that λ_j^2 is the j'th eigenvalue of the reduced density matrix – and $|\phi_j\rangle$ is the associated eigenvector. This is the same construction we found for relating the SVD of M to the eigenstates of $M^\dagger M$. We can similarly get $|\theta_j$ as the j'th eigenvector of the reduced density matrix for subsystem B.

We therefore see that the entanglement entropy is the entropy of the ensemble generated by tracing over part of the system. We will find that tensor networks are a natural way to encode the geometry of entanglement in a quantum state. Further we will find that by making arguments about the structure of entanglement, will lead us to approximations which help us solve the many-body problem.

G. Geometry of Entanglement

Consider a typical bipartition of a system:



We will imagine that it is a lattice system. We will call V the total number of sites – the "volume" of the system – but it may not be a 3D system. We will call V_A the number of sites in A, and V_B the number of sites in B.

G.1. Generic States

For a generic state – the state of B will depend on all details of A. The Hilbert space of A has a size which is exponential in V_A, so the entanglement entropy, which will be the *log* of this, should scale as

$$S \sim V_A. \tag{2.50}$$

G.2. Gapped Ground States

The situation is very different for a "gapped" ground state. Think for example of a ferromagnet. There is a finite correlation length, and fluctuations in region B, only depend on the parts of A which are within ξ of the boundary. Therefore the entropy should scale as

$$S \sim \Omega_A \xi, \tag{2.51}$$

where Ω is the surface area/ perimeter of the region. This is known as the "Surface law."

Now you might argue that gapped states are boring. While there is some truth to that claim, there is actually a lot of interesting physics with gapped states. In fact, these are conventionally "ordered" state – like magnets and superconductors. We will develop tools which take advantage of the surface-law scaling of gapped states in 1D. Of course, the "surface" in 1D is just a pair of points. So in 1D with a gapped system, the entanglement entropy is independent of the size of the region you are looking at. This is good for doing numerics.

G.3. Free Particles

For non-interacting particles, the only way that the states of subsystem A and B are correlated is by number conservation.

Suppose region A has V_A sites, and the system has an average density of n particles per site. The mean number of particles in region A is $N_A = nV_A$. The fluctuations are $\sqrt{N_A}$. Therefore one expects the entanglement entropy to scale as

$$S_A \sim \log\sqrt{N_A} = \frac{1}{2}\log V_A + \cdots \tag{2.52}$$

Free particles (at fixed density) have an entropy which scales as the logarithm of the subsystem.

On the flip side, if one instead has N total particle, and a total volume V, and divide the system in half. By the same argument

$$S \sim \sqrt{N} \tag{2.53}$$

is independent of system size.

G.4. Topological Order

There is a special case of gapped states, namely those which are "topologically ordered". Topological order was a concept which I believe was first developed in thinking about spin systems, and contemplating ground states which had no local order parameter, but were gapped. The name "topological" has a number of meanings: First, these systems can be distinguished from ordinary "topologically trivial" states, and from one-another, by asking if one can continuously deform one into another without opening a gap. Topology in the mathematics of continuity. There is another way that "topology" enters into topologically ordered states, namely that the ground-state degeneracy depends on the topology of the space. These ground states cannot be distinguished by any local measurement.

A related feature of topologically ordered states is that there exist bulk excitations which cannot be created by any local operator. There may, however, be local operators which can create a finite number of them. The ground state degeneracy is related to the properties of these "quasiparticles." If you create a pair of them, move one around a non-contractable loop, then recombine them, you typically move in the space of ground states.

If you have a topologically ordered state with boundaries, there will generically be gapless edge states. The hand-waving picture is that you are spatially moving between two topologically distinct regions – which is only possible if you locally close a gap.

There are also weaker flavors of "topological order," where you ask about continuously connecting states without breaking any symmetries. These are interesting, but do not necessarily have all of the features of the more generic kind of topological order.

In the context of these lectures, the interesting thing about topologically ordered states is that there is some sort of very diffuse entanglement in the system: The degenerate ground states are distinguished by some "loop-correlations" which extend over non-contractible spatial loops. Consequently, in a bipartition, subsystems B and A are entangled not only by physics happening at the boundary, but also by this non-local physics. This gives a sub-dominant contribution to the entanglement entropy:

$$S \sim \Omega_A \xi - \gamma, \tag{2.54}$$

where γ is a constant which is independent of the size of A. Of course there is some ambiguity here: How exactly do you define the area? If you use two definitions which differ by an additive constant, then that changes the perceived γ. Further there are corrections due to "corners" and other features of the boundary geometry. There are also some more technical subtleties: For example, if A is non-contractable, then the entanglement entropy actually depends on which ground state you are in. Nonetheless physicists have figured out those details, and one can actually use the scaling of the entanglement entropy as a mechanism for distinguishing topological phases. Our tensor network ideas give a framework for thinking about this entanglement.

Quantum Info

There is an equivalent definition of topological order which is to say that two wavefunctions are in the same topological class if a finite depth quantum circuit can transform one into the other. Topologically distinct wavefunctions cannot be connected by a finite depth circuit. Symmetry protected phases add constraints on the gates.

G.5. Critical States

Critical systems have correlation functions which fall off as power laws. In 1D this leads to a

$$S_A \sim \log V_A. \tag{2.55}$$

In higher dimensions there is typically an area law with logarithmic contribution. So in 2D, you would get $S \sim L \log L$, where L is the perimeter.

Do the second notebook on using tensors in Mathematica.

H. Homework 2 – Due Jan 31

Problem 6. For Credit: Take any image. Use the SVD on it. Make a plot of the singular values λ_j as a function of j. Truncate to a small number of singular values. How many do you need to keep for the image to look good? By how much did you compress the image. [Note that you won't do as well as *.jpg*, unless you do an image which has a lot of horizontal and vertical lines – like a flag or a Mondrian painting.]

Problem 7. For Credit: A gas of fermions hopping on a 1D lattice can be mapped onto a spin model – and then analyzed using the techniques in this module. The approach for doing this is the "Jordan-Wigner" transform. For simplicity we will work with spinless Fermions, but adding spin is straightforward. Note, it is a non-local transformation, so local order in the spin language may be non-local in the Fermi language, and vice-versa.
Consider a single spin 1/2, and define the down-spin state to be $|0\rangle$ – which will map onto the absence of a Fermion, while the up-spin state $|1\rangle$ will be the presence of a fermion. It is natural to define

$$f^\dagger = \sigma^- \tag{2.56}$$
$$f = \sigma^+ \tag{2.57}$$

where $\sigma^\pm$ are Pauli operators on the spin basis.

7.1. Express σ_z in terms of f and $f^\dagger$.

7.2. Show that f and $f^\dagger$ obey fermion anticommutation relations.

7.3. If we have more sites, we run into the problem that the f's defined this way commute on different sites. To fix this, we define

$$a_j^\dagger = (-1)^{\sum_{k<j} f_k^\dagger f_k} f_j^\dagger \tag{2.58}$$
$$a_j = (-1)^{\sum_{k<j} f_k^\dagger f_k} f_j, \tag{2.59}$$

Show that these a's obey fermionic commutation relations.

7.4. Map the XXZ model onto a Fermi model,

$$H = \sum_j J_x(\sigma_j^x \sigma_{j+1}^x + \sigma_j^y \sigma_{j+1}^y) + J_z(\sigma_j^z \sigma_{j+1}^z). \tag{2.60}$$

7.5. Map the transverse field Ising model onto a Fermi model, using the basis perpendicular to the one we used in the last homework

$$H = \sum_j -h\sigma_j^z - J\sigma_j^x \sigma_{j+1}^x. \tag{2.61}$$

The resulting Hamiltonian has the structure of a model of superconductivity. (It is a 1D p-wave superconductor, sometimes refered to as the "Kitaev chain" – Alexi Kitaev has a famous paper where he shows that it has "Majorana" edge modes.

7.6. In the last homework we distinguished the phases of the transverse field Ising model by the order parameter $\langle\sigma^x\rangle$. Show that this is a non-local operator in the Fermi language. Hence the Fermi analog of the ordered phase is topological – in fact it is an example of a symmetry protected topological phase.

Problem 8. (For Credit) Derive Eq. (2.22).

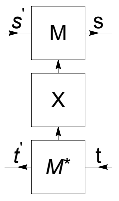
Problem 9. (For Credit) In the lecture notes we produced an expression for $\langle X_i Y_j \rangle$ when $|i-j|>1$. Find a similar result for the cases where $j=i+1$ and $j=i$.

Problem 10. (For Credit) Here you will complete the calculation of the properties of the matrix product state described by Eq. (2.1). We will take ϵ to be real throughout. I recommend using a computer algebra system.

10.1. Show that the right eigenvectors of the E_I in Eq. (2.18), corresponding to $\lambda_- = 1-\epsilon^2$ and $\lambda_+ = 1+\epsilon^2$ are $(1-\epsilon^2, 0, 0, \epsilon^2-1), (1+\epsilon^2, 2\epsilon, 2\epsilon, 1+\epsilon^2)$. Similarly, show that the left eigenvectors are $(-1,0,0,1), (1,0,0,1)$.

Problem 10. cont...

10.2. Calculate

$E_X =$  (2.67)

as a 4×4 matrix, where $X=\sigma_x$ is the Pauli matrix. (Same indices as we used for E_I).

Problem 10. cont...

10.3. Show that

$$\langle L^+|E_X|R^+\rangle = 8\epsilon^2 \tag{2.77}$$

Problem 10. cont...

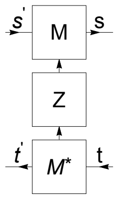
10.4. Show that $\langle L^+|R^+\rangle = 2(1+\epsilon^2)$.

Problem 10. cont...

10.5. Calculate $\langle \sigma_x \rangle$.

Problem 10. cont...

10.6. Calculate

$E_Z =$ 

(2.81)

as a 4×4 matrix, where $Z = \sigma_Z$ is the Pauli matrix.

Problem 10. cont...

10.7. Show that

$$\langle L_+|E_Z|R_+\rangle = 0 \tag{2.83}$$
$$\langle L_+|E_Z E_Z|R_+\rangle = 2(1+\epsilon^2)^2(1-\epsilon^2) \tag{2.84}$$
$$\langle L_+|E_Z|R_-\rangle = 2(1-\epsilon^4) \tag{2.85}$$
$$\langle L_-|E_Z|R_+\rangle = -2(1-\epsilon^4) \tag{2.86}$$
(2.87)

10.8. Calculate the correlation function $\langle \sigma_z^i \sigma_z^j \rangle$ in this state. Separately consider the case $|i-j| = 1$ and $|i-j| > 1$.

Problem 10. cont...

10.9. Calculate the expectation value $\langle H \rangle$ in this state, where H is given by Eq. (1.28). Minimize with respect to ϵ to optimize the wavefunction. Plot the resulting $E/(NJ)$ as a function of h/J. Compare it with the mean field prediction. Note this ansatz has the opposite problem of the product state we previously used – it overestimates the stability of paramagnetic state – so you should find the phase transition at smaller h/J.
Hint: Define $y = \epsilon^2/(1+\epsilon^2)$. The minimization is easier in terms of y. Note, $0 < y < 1$.

Problem 11. (Challenge – not for Credit) We can describe both the ordered and disordered phases with the two-parameter matrix product state:

$$|\psi\rangle = \cdots \begin{pmatrix} |\uparrow\rangle & \epsilon\eta|\downarrow\rangle \\ \epsilon|\uparrow\rangle & \eta|\downarrow\rangle \end{pmatrix} \begin{pmatrix} |\uparrow\rangle & \epsilon\eta|\downarrow\rangle \\ \epsilon|\uparrow\rangle & \eta|\downarrow\rangle \end{pmatrix} \begin{pmatrix} |\uparrow\rangle & \epsilon\eta|\downarrow\rangle \\ \epsilon|\uparrow\rangle & \eta|\downarrow\rangle \end{pmatrix} \cdots \tag{2.96}$$

Where ϵ and η are real and positive.
It would be nice to repeat the previous problem with this state. Unfortunately the algebra gets too messy for paper and pencil – so this will be a partially numerical problem. I used Mathematica for it – which was pretty efficient. If you don't already have good familiarity with Mathematica, it is probably more trouble than it is worth. I also recommend naming your variables x and y instead of ϵ and η. Will make entering easier in the computer.

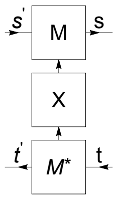
11.1. Generate the transfer matrix E_I, using the same basis as the lecture.

Problem 11. cont...

11.2. Write six different functions in some computer language that take ϵ and η produce each of: $\lambda_+(\epsilon,\eta), \lambda_-(\epsilon,\eta), R_+(\epsilon,\eta), R_-(\epsilon,\eta), L_+(\epsilon,\eta), L_-(\epsilon,\eta)$. The latter 4 are length 4 vectors. You can debug this by noting that when $\eta = 1$ you should get the same results as for the last question (with the caution that the eigenvectors are only defined up to a multiplicative constant).

Problem 11. cont...

11.3. Calculate

$E_X =$  (2.98)

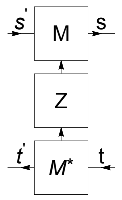
as a 4×4 matrix, where $X = \sigma_x$ is the Pauli matrix.

Problem 11. cont...

11.4. Write a computer program that will calculate Calculate $\langle\sigma_x\rangle$ as a function ϵ and η.

Problem 11. cont...

11.5. Calculate

$E_Z =$  (2.108)

as a 4×4 matrix, where $Z = \sigma_Z$ is the Pauli matrix.

Problem 11. cont...

11.6. Write functions that will calculate $\langle \sigma_z^i \rangle$ and $\langle \sigma_z^i \sigma_z^{i+1} \rangle$ as a function of ϵ and η.

11.7. Combine your results to make a function that will give the energy $\bar{E} = \langle H \rangle / (JN)$ from Eq. (1.28, when given ϵ, η, and h/J. Feed this into a minimization routine, so that you optimize the parameters. Plot the resulting $\langle \sigma_z \rangle$ as a function of h/J.

Problem 11. cont...

11.8. Plot the correlation length as a function of h/J.