DRSTP SPCTM school 2018

Introduction to tensor networks for quantum many-body systems

Lecture by Philippe Corboz (p.r.corboz@uva.nl) - March 2018

Abstract

Tensor networks are among the most powerful tools to study quantum many-body systems. One can even see them as a new language to think and talk about many-body physics. They have been developed by combining ideas from quantum information theory and condensed matter physics, where the notion of entanglement plays a central role. The main idea is to efficiently represent many-body states (or operators) by a trace over a product of tensors, with an accuracy that can be systematically controlled by the size of the tensors. A well-known example are matrix product states (MPS) - an efficient ansatz for ground states in one dimension - which has become the state-of-the-art approach to study low-dimensional systems. Progress in understanding the structure of entanglement in many-body systems has led to the generalization of this idea to higher dimensions (PEPS) and critical systems (MERA), offering a very promising route to solve challenging open problems for which previous approaches failed. This lecture provides a practical introduction to this fast-developing field, with topics covering the diagrammatic notation, the area law of the entanglement entropy, discussion of different tensor network ansaetze & algorithms, and more.

General information

Lecture material

The material of this lecture can be found in the following Dropbox folder:
`https://tinyurl.com/ycp9unaa`.

Exercises

Some of the exercises will involve programming. We will use Matlab as programming language since there exist useful tools for tensor network calculations (see dropbox folder). (You should be able to get a Matlab license from your university.)

Don't worry if you are new to Matlab. It's easy to learn and shares many similarities, e.g. with Python (Numpy/Scipy) or Fortran. We will go through some useful language features at the beginning of the exercise session. If you want to read more about Matlab, here are a few useful introductions:

- A short Matlab Primer: `http://faculty.olin.edu/bstorey/Notes/matlab.pdf`
- An even more compact overview: `http://www4.ncsu.edu/~pfackler/MPRIMER.htm`
- The official Matlab Primer: `https://www.mathworks.com/help/pdf_doc/matlab/getstart.pdf`

Useful references on tensor networks for further reading

- U. Schollwöck, *The density-matrix renormalization group in the age of matrix product states*, Annals of Physics 326, 96192 (2011).
  Open-access version: `http://arxiv.org/abs/1008.3477`
- F. Verstraete, V. Murg, and J. I. Cirac, *Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems*, Advances in Physics 57, 143 (2008).
  `https://arxiv.org/abs/0907.2796`
- G. Vidal, *Entanglement Renormalization: an introduction*, see `https://arxiv.org/abs/0912.1651v2`
- R. Orus, *A Practical Introduction to Tensor Networks: Matrix Product States and Projected Entangled Pair States*, Annals of Physics 349, 117158 (2014).
  `https://arxiv.org/abs/1306.2164v3`

1 Introduction

1.1 Motivation

Understanding the emergent behavior in quantum many-body systems is one of the grand challenges in modern condensed matter physics. In particular, strong correlation effects between the particles can give rise to a variety of remarkable phenomena, such as high-temperature superconductivity, quantum spin liquids, the fractional quantum Hall effect, and more. Despite the fact that many of these systems can be described by rather simple models on a lattice, it is extremely difficult to study them, because approximate analytical methods typically fail for these systems. This is why in many cases accurate and efficient numerical methods are crucial to get insight into the physics of these systems. Quantum Monte Carlo (QMC) is one of the most powerful methods to simulate these systems. However, it fails for important classes of models (fermionic and frustrated models) due to the infamous *negative sign problem*, which is the main reason why the physics of many relevant models is still controversial. A famous example is the two-dimensional (2D) fermionic Hubbard model - a very simple model of interacting electrons on a square lattice, which is believed to capture the physics of HTSC. Despite an enormous effort during the last 50 years the Hubbard model has still not been conclusively solved, and the mechanism leading to HTSC remains to be uncovered. Thus, in the past decades a lot of effort has been put into the development of new accurate numerical techniques for the study of strongly correlated systems.

For one dimensional systems (chains and ladder systems), tremendous progress has been achieved thanks to the so-called density matrix renormalization group (DMRG) method, which was invented by S.R. White in 1992.[1] The key idea of DMRG is to efficiently truncate the exponentially large Hilbert space down to $D = 100 - 10000$ states, where only important states are kept, and non-relevant states are discarded. The accuracy of this variational approach can be systematically controlled by D. In many cases this methods yields extremely accurate results for ground states for system sizes up to several hundreds (or even thousands) of sites. It was later found that DMRG has an underlying variational ansatz, so-called matrix product states (MPS), in which the wave function is represented by a product of matrices of size $D \times D$.

Progress in quantum information theory has helped to understand why DMRG and MPS are so successful, where the notion of *entanglement* plays a key role. It turns out that typical ground states of systems with local interactions give rise to states obeying the so-called *area law of the entanglement entropy*. Based on these new ideas, MPS have been generalized to two (and higher) dimensions, called tensor network states, in which the wave-function is represented by a product of tensors. These methods are still much newer and more challenging than in 1D, however, there have been several recent successes which show that these methods are very promising to solve challenging fermionic and frustrated systems.

1.2 Problem of finding the ground state

Consider a lattice of N sites where each site i is described by a local Hilbert space, $\mathcal{H}_i$ of dimension d. For example, in the case of a spin-1/2 system: $d = 2$ with basis states $\{|\uparrow\rangle, |\downarrow\rangle\}$. The total Hilbert space is given by $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \cdots \otimes \mathcal{H}_N$ and has dimension d^N. We consider a Hamiltonian which decomposes into a sum of local

[1] S. R. White, Phys. Rev. Lett. 69, 28632866 (1992).

terms, e.g. a Heisenberg model with nearest-neighbor interaction (for simplicity we will consider systems with open boundary conditions in the following). We would like to find the ground state (or low-energy states) which can in general be written as

$$|\Psi\rangle = \sum_{i_1 i_2 \ldots i_N} \Psi_{i_1 i_2 \ldots i_N} |i_1 \otimes i_2 \otimes \cdots \otimes i_N\rangle$$

where each i_k goes over the local basis states, e.g. $i_k \in \{|\uparrow\rangle, |\downarrow\rangle\}$. The expansion coefficients $\Psi_{i_1 i_2 \ldots i_N}$ are written here as a multidimensional array (tensor) containing d^N complex parameters, i.e. the number of parameters scales exponentially with N, which is not efficient. The aim is now to find a compact way to approximately represent the coefficients $\Psi_{i_1 i_2 \ldots i_N}$. Note that we could write the multidimensional array also as a vector, which is called reshaping the array into a vector.

1.3 Entangled states vs product states

The simplest way to approximate the coefficients is to write it as a product, i.e. $\Psi_{i_1 i_2 \ldots i_N} = \psi_{i_1} \psi_{i_2} \ldots \psi_{i_N}$, defined by vectors ψ_{i_k} on each site. These product states (or site-factorized states) are the ones which are obtained from a mean-field theory. However, in quantum systems the wave function can typically *not* be written as a simple product. These states are called *entangled*.

Consider two up-spins on two sites, $|\Psi\rangle = |\uparrow\rangle_1 |\uparrow\rangle_2$, which is clearly a product state. However, a singlet $|\Psi\rangle = (|\uparrow\rangle_1 |\downarrow\rangle_2 - |\downarrow\rangle_1 |\uparrow\rangle_2)/\sqrt{2}$ cannot be written as a product state of the two sites: this is an entangled state. We will later see how to quantify how strongly a state is entangled by defining an entanglement entropy.

The basic idea is now to represent the coefficients not by a product of vectors, but by a product of matrices (or more generally tensors), $\Psi_{i_1 i_2 \ldots i_N} \approx M_{i_1} M_{i_2} \ldots M_{i_N}$ which will allow us to represent entangled states. By varying the size of the matrices we can systematically approximate the expansion coefficients $\Psi_{i_1 i_2 \ldots i_N}$, i.e. the wave function $|\Psi\rangle$.

1.4 Diagrammatic notation

In tensor network methods we generally have to deal with tensors (multi-dimensional arrays) with many indices. Instead of writing out long formulas a VERY useful diagrammatic notation is often used, shown in Fig. 1. Each tensor is represented as a shape (here: a ball) and the open legs attached to the shape correspond to the indices. A connected index between two tensors denotes a sum over the index. It is not only a useful tool for tensor network methods, but also an alternative way to think about quantum mechanics!

1.5 Main idea of a variational tensor network ansatz

The main idea of a tensor network ansatz is an efficient representation of the expansion coefficients $\Psi_{i_1 i_2 \ldots i_N}$ by a trace over a product of tensors or matrices. Using the diagrammatic notation there is a simple graphical interpretation of a tensor network ansatz, shown in Fig. 2 for a 6-site system. The coefficients can be represented as one big tensor with N open legs. This big tensor is decomposed into small pieces, connected by lines: this is a tensor network. There are many different ways how this decomposition can be done. The particular ansatz shown in Fig. 2 is a matrix product state (MPS); we will see other examples later.

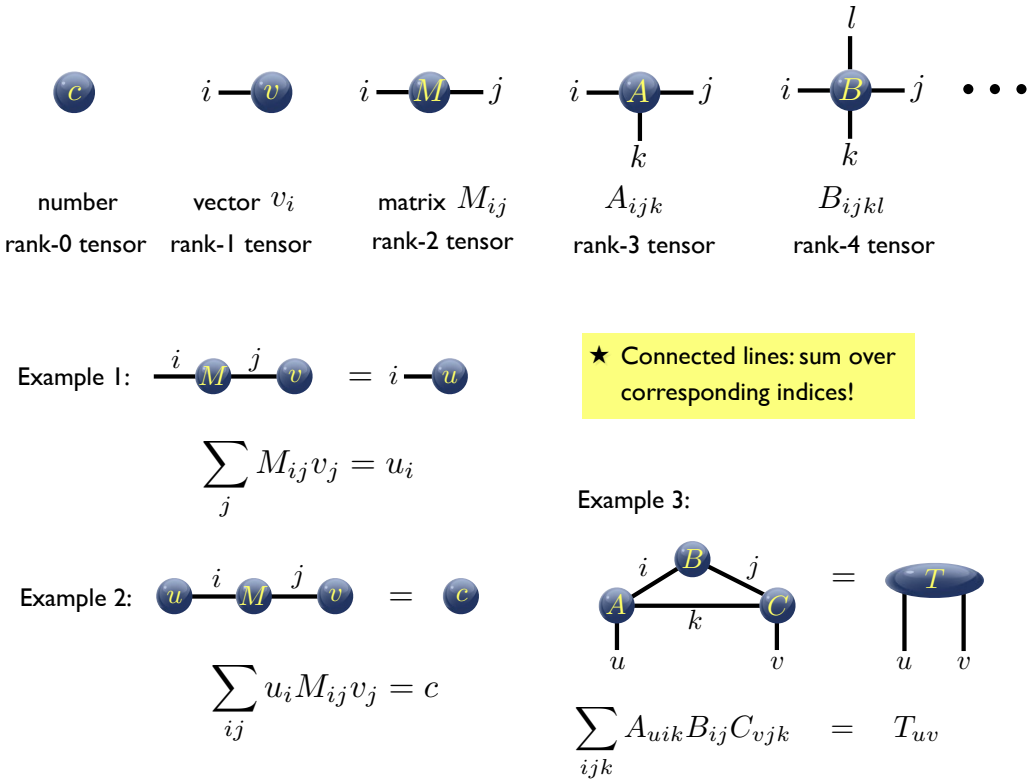


Figure 1: Diagrammatic notation used in tensor network methods.

Note that diagrammatically an MPS consists of rank-3 tensors (rank-2 tensors on the open boundaries). But if one computes a particular coefficient of a wave function, e.g. $\Psi_{\uparrow\downarrow\downarrow\uparrow\uparrow\downarrow}$, this boils down to a product of matrices (with vectors on the open boundaries), hence the name matrix product state. The open legs correspond to the local Hilbert spaces (each index i_k goes over d elements), whereas the horizontal connections are called "auxiliary space" with a bond dimension D, i.e. each auxiliary index goes over D elements. This bond dimension determines the size of the matrices, or in other words, the number of variational parameters contained in each tensor (which is dD^2 for a tensor away from the boundary, and dD for tensors A and F in the example). Thus, with D we can systematically control the accuracy the ansatz. We will later see how to find these variational parameters (e.g. by minimizing the energy as in other variational calculations) such that we have the best approximation of the ground state.

Now the crucial difference between the big tensor and the tensor network is the number of involved parameters. We started from d^N coefficients, however, in a tensor network the number variational parameters typically scales only polynomially with the bond dimension D an system size N. This is a very efficient representation of a state living in an exponentially large Hilbert space!

Of course, the crucial question is now how large does D need to be in order to have an accurate approximation of the ground state. In general, ANY state of the Hilbert space can be represented by choosing $D \propto 2^N$ (more precisely $D = 2^{N/2}$), however, then we have not gained anything, because then the exponential scaling is simply hidden in the D. It is true that not every state can be efficiently represented, but it turns out that the states we are interested in (ground states of local Hamiltonians) are very special: they

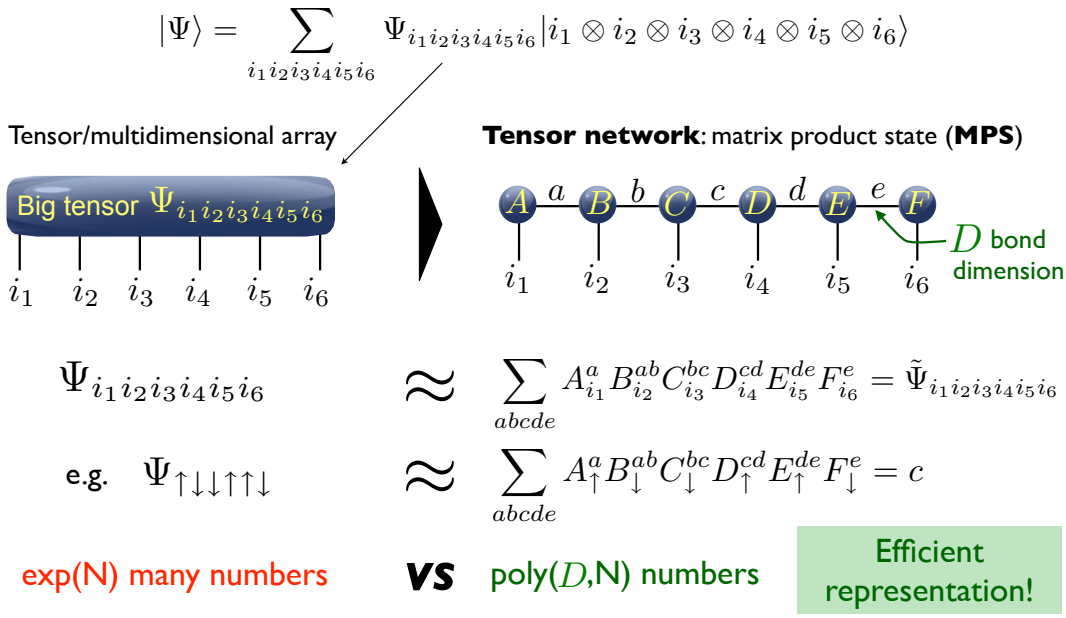


Figure 2: The main idea of a tensor network ansatz: decomposing the big tensor into small pieces connected by lines.

are not as much entangled as a random state in the Hilbert space. As a consequence, these states can be efficiently represented by a tensor network, as we will see later.

1.6 Schmidt decomposition

To gain further insight how large the D needs to be we consider a simpler problem than the MPS decomposition from the last section: we simply split the big tensor into two pieces, i.e. we cut the system in the middle. Let's call the left side A and the right side B, having both $L = N/2$ sites, and their Hilbert spaces $\mathcal{H}_A$ and $\mathcal{H}_B$ having dimension $M = 2^L$ each, with orthonormal bases $|i\rangle$ and $|j\rangle$ respectively. The wave function can be written as

$$|\Psi\rangle = \sum_{ij}^{M} \Psi_{ij} |i\rangle |j\rangle. \tag{1}$$

The Schmidt-decomposition is a special form of rewriting the wave-function:

$$|\Psi\rangle = \sum_{k}^{M} s_k |a_k\rangle |b_k\rangle, \tag{2}$$

with $s_k \geq 0$. Note that here we only have a single sum over k instead of a double sum. This form can be obtained by performing a singular value decomposition (SVD) of the matrix Ψ_{ij}:

$$\Psi = U\, s\, V^\dagger, \tag{3}$$

where U and V are unitary $M \times M$ matrices, and s a diagonal matrix with $s_{11} \geq s_{22} \geq \ldots s_{MM} \geq 0$, which are the singular values (Schmidt coefficients). By introducing $\Psi_{ij} = \sum_k U_{ik} s_{kk} V^*_{jk}$ in Eq. (1) we obtain Eq. (2), where we carried out the basis

transformation:

$$|a_k\rangle = \sum_i^M U_{ik}|i\rangle, \tag{4}$$

$$|b_k\rangle = \sum_j^M V_{jk}^*|j\rangle \tag{5}$$

The way to think of the state in Eq. 2 is to see it as a superposition of several states, where each term is given by a basis state living on A times a basis state living on B times a weight s_k. The question is now how many non-vanishing terms D do we have in this sum, i.e. how many of the s_k are non-zero, and how does the distribution of s_k look like. There are two extreme cases:

- Imagine that only $s_1 = 1$ and all other s_k are zero. This corresponds to a product state: $|\Psi\rangle = 1|a_1\rangle|b_1\rangle$ (no entanglement).
- The other extreme case is when all the singular values are non-zero and equal in magnitude: $|\Psi\rangle = \sum_k^M s_k|a_k\rangle|b_k\rangle$ with $s_k = 1/\sqrt{M}$: it's a superposition of all basis states with equal weight, which is called a maximally entangled state.

In the first case we only need $D = 1$ basis states to represent the wave function, in the second case we need all $D = M$ basis states to represent the state. In practice the states we are interested in fall somewhere in between: they are certainly entangled $D > 1$, however, the distribution s_k decays rapidly such that s_k becomes exponentially small for k larger than a certain $D << M$. Thus, in practice we can obtain an accurate approximation by keeping the D states with largest Schmidt coefficients s_k in the sum and discard the other states, i.e.

$$|\Psi\rangle \approx |\tilde{\Psi}\rangle = \sum_k^{D<<M} s_k|a_k\rangle|b_k\rangle \tag{6}$$

One can show that this choice minimizes the error $\epsilon = |||\Psi\rangle - |\tilde{\Psi}\rangle||^2$, i.e. it is the best approximation we can achieve by restricting ourselves to only D basis states. Choosing these particular basis states (and discarding the rest of the Hilbert space) is *the key idea of DMRG.*

1.7 The reduced density matrix

Originally DMRG was formulated in terms of the reduced density matrix and not the SVD as presented above, but the idea is the same. Let's recall some basic properties of density matrices/operators. The density operator of the (non-degenerate) ground state is a *pure state* since it contains only one state with probability one:

$$\hat{\rho} = |\Psi\rangle\langle\Psi| \tag{7}$$

This is in contrast to a *mixed state*

$$\hat{\rho} = \sum_k p_k|\phi_k\rangle\langle\phi_k| \tag{8}$$

which contains several states each appearing with probability p_k. A well known example of mixed state is a thermal state at temperature $T = 1/\beta$, given by $\hat{\rho} = \exp(-\beta\hat{H})/Z$.

The *reduced density matrix* of region A is obtained by taking the state of the full system $\hat{\rho}$ and tracing out the states of region B,

$$\hat{\rho}_A = Tr_B|\Psi\rangle\langle\Psi| = \sum_k \langle b_k|\Psi\rangle\langle\Psi|b_k\rangle = \sum_k s_k^2 |a_k\rangle\langle a_k|. \tag{9}$$

Here we introduced Eq. 2 which automatically yields a diagonal form of the reduced density matrix (we can also start from Eq. 1 and then perform a diagonalization). The reduced density matrix describes with which probability ($p_k = s_k^2$) the state $|a_k\rangle\langle a_k|$ can be found in region A. The probabilities simply correspond to the square of the Schmidt coefficients s_k.

Note that even though we started from a pure state for the entire system, the reduced density matrix typically corresponds to a mixed state, except if the ground state is a product state between region A and B, $|\Psi\rangle = |\phi_A\rangle|\phi_B\rangle$, then one can easily show that the reduced density matrix is a pure state, $\hat{\rho}_A = |\phi_A\rangle\langle\phi_A|$.

To reformulate the key idea of DMRG: keep the D states with largest eigenvalues of the reduced density matrix and discard the other states. This provides the most accurate way to truncate the Hilbert space.

1.8 Entanglement entropy and area law

In Sec. 1.6 we mentioned that the Schmidt coefficients s_k (or equivalently the eigenvalues of the reduced density matrix p_k) decay rapidly in the ground state, such that it is sufficient to keep only a "small" number $D << M$ of all states. Instead of looking at the distribution we can quantify the "amount" of entanglement by a number: the entanglement entropy given by

$$S_A = -\mathrm{Tr}\hat{\rho}_A \log \hat{\rho}_A = -\sum_k p_k \log p_k \tag{10}$$

Let us again consider the two extreme cases:

- In the case of a product state, $p_1 = 1, p_2 = p_3 = \cdots = p_M = 0$, $S_A = -1 \log 1 = 0$, i.e. no entanglement as expected for a product state.
- For a maximally entangled state all probabilities are equal, $p_k = 1/M$, and $S_A = \log M$.

We will now ask how S_A depends on the size L of regions A and B, and what it implies for the number of relevant states D which need to be kept. For a given $S_A(L)$ the number of relevant states scales as $D \sim \exp(S_A(L))$.

If one picks a random state from the Hilbert space most of the p_k will be non-vanishing, leading to an extensive entanglement entropy: $S_A(L) \sim L$ and thus a D growing exponentially with L. In d spatial dimensions this generalizes to

$$S_A(L) = L^d. \tag{11}$$

Now for ground states (and low-energy states) of local Hamiltonians $S_A(L)$ is not extensive but typically obeys an area law:

$$S_A(L) \sim L^{d-1}, \tag{12}$$

i.e. $S_A(L)$ does not grow with the volume of A, but with area of the boundary between A and B. Intuitively this has to do with the fact that the entanglement is mostly concentrated locally (for ground states of a local Hamiltonian).

The area law of the entanglement entropy has been proven for local, gapped Hamiltonians in 1D,[2] which implies that $S_A(L) \sim L^{1-1} = const.$ Thus, asymptotically (for system sizes much larger than the correlation length) the entanglement entropy $S_A(L)$ will saturate with increasing L. As a consequence, the number of relevant states D is also a constant! This explains the success of DMRG: Even though the Hilbert space is exponentially large the number of relevant states tends to a constant with increasing system size! This is why an efficient representation as an MPS is possible, and why 1D systems including thousands of sites can be simulated.

There are exceptions: Critical ground states in 1D exhibit a logarithmic correction $S_A(L) \sim logL$, leading to a number of states growing polynomially (not exponentially) with system size. Some critical ground states in higher dimension also exhibit a logarithmic correction (but not all of them). A general proof that an area law holds for gapped ground states in higher dimensions is still lacking.

1.9 Decomposing a state into an MPS

Previously we discussed how to split the system into two parts using an SVD. Instead of doing just one cut in the middle of the system, we can decompose an arbitrary state into an MPS by a sequence of SVDs, starting by splitting, e.g. the left outermost site from the rest of the system. After that we can split off the second site, and proceed similarly for the remaining sites, until the state is fully decomposed. The procedure is illustrated in Fig. 3.

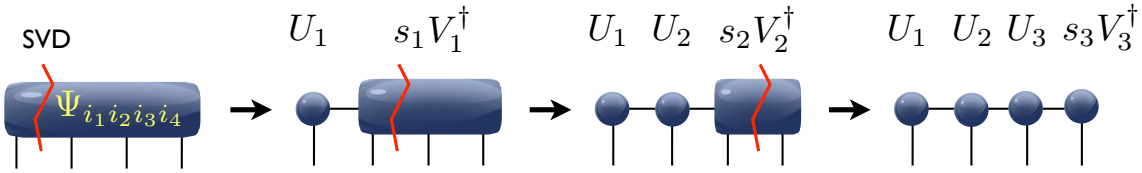


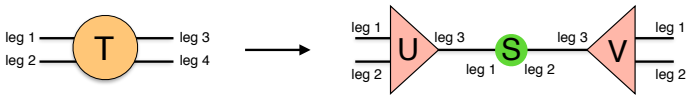
Figure 3: Decomposing a state into an MPS using several SVD's. Note that an SVD of a tensor is done by reshaping the tensor into a matrix (by grouping the indices on the left side and right side of the cut, respectively), then perform an SVD of the matrix, and then finally reshape the matrix back into a tensor.

If one keeps all singular values, the decomposition will be exact. If small singular values are discarded (keeping at most D singular values at each link) the decomposition will only be approximate. Instead of using a full SVD one can also use a QR decomposition, in which a matrix A is decomposed into a unitary matrix Q and R an upper triangular matrix.

Note that in practice one does not have the exact ground state (which then can be decomposed) but the goal is to directly find the (approximate) ground state written as an MPS. This will be the topic in the next sections.

In MATLAB we can conveniently use the tensorsvd function to perform an SVD of a tensor (see Dropbox), see example shown in Fig. 4.

[2] M. B. Hastings, J. Stat. Mech. 2007, P08024 (2007)



[U,s,V] = tensorsvd(T, [1,2], [3,4], D)

Figure 4: Example of the tensorsvd function in MATLAB to decompose a tensor using an SVD, keeping the D largest singular values. The resulting order of the legs is indicated.

2 Matrix product state algorithms

2.1 Overview: optimization and contraction

So far we have only discussed the MPS ansatz, but not the actual algorithm to find the ground state of a given Hamiltonian. Each MPS is a variational ansatz (i.e. it gives an upper bound to the true ground state energy). To obtain an approximation we can minimize the energy of the ansatz with respect to the parameters stored in the tensors (matrices). DMRG is nothing but an iterative procedure where one tensor after the other gets optimized in such a way that the energy is lowered. Another strategy to find the ground state is to start from a random initial MPS and then perform an imaginary time evolution, using a Trotter-Suzuki decomposition. We will discuss these ideas further below.

2.2 Contraction and computational cost

Once we have converged to the approximate ground state we can compute quantities of interest (expectation values of observables and other quantities like entanglement entropies and spectra). In order to do this we have to evaluate the corresponding tensor network, as for example shown in Fig. 5 for a local operator $\hat{O}$ acting on site 3. To contract (evaluate) this network we need to take the sum over all connected indices. This is typically done by a pairwise multiplication of tensors. Note that *the computational cost depends on the chosen sequence of pairwise multiplications*! Thus one needs to find the optimal sequence first in order to minimize the computational cost. What is most important is to try to keep the *leading* computational cost, i.e. the leading power in the bond dimension D, as low as possible, since this will be the bottleneck of the computation as we increase D more and more.

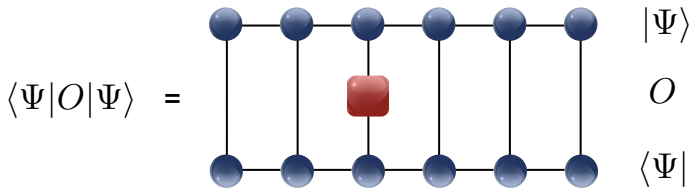


Figure 5: Tensor network diagram corresponding to the expectation value of an operator $\hat{O}$ acting on site 3.

Think about what the computational cost of multiplying two tensors is: you will find that it is given by the product of the dimensions of all the indices (counting the

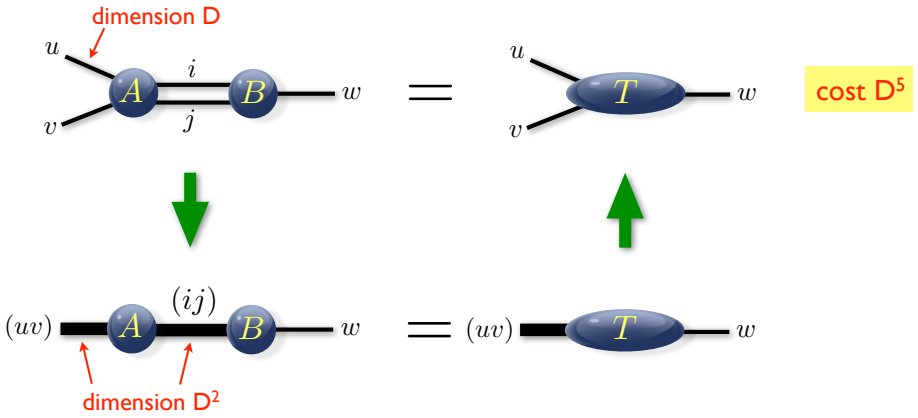
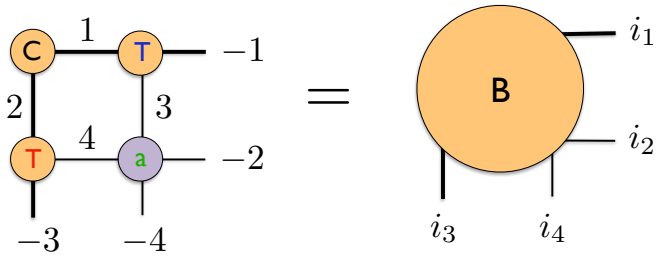


Figure 6: Multiplying two tensors together. The computational cost can be determined by counting the number of involved legs (here 5 legs each with a bond dimension D, thus the computational cost to contract the two tensors is D^5.) Instead of writing a for-loop over the contraction index (here i and j) one can also reshape the tensors into matrices and then call an efficient matrix-matrix multiplication routine (Fortran BLAS routines) and then reshape the resulting matrix back into a tensor. Note that depending on the order of the indices, one first needs to *permute* the indices before they can be combined into a big index.




```
B = ncon({C, T, T, a},{[1 2], [-1 1 3], [2 -3 4], [-2 3 4 -4]});
```

Figure 7: Example of the ncon function in MATLAB to contract a tensor network.

connected indices only once), see example in Fig. 6. In Matlab we can do contractions conveniently with the *ncon* [3] function, see Fig. 7 for an example.

2.3 Left-, right-, and mixed-canonical MPS

There is not a unique MPS for a given state. Consider for example two tensors A and B which are connected by a link with index i. We can always introduce an identity on that link, $\mathbb{I} = K\,K^{-1}$ with K a $D \times D$ invertible matrix. We can then absorb K into A, $\tilde{A} = AK$, and K^{-1} into B, $\tilde{B} = K^{-1}B$, and the tensor network would still represent the same state, since $\mathrm{Tr}_i \tilde{A}\tilde{B} = \mathrm{Tr}_i AB$. Thus there is a large "gauge" freedom in choosing our tensors.

The decomposition used in the last section creates a so-called left-canonical, or left-normalized MPS. This is because all the U_k-tensors are unitary matrices (obtained

[3] R. N. C. Pfeifer, G. Evenbly, S. Singh, and G. Vidal, arXiv:1402.0939 (2014).

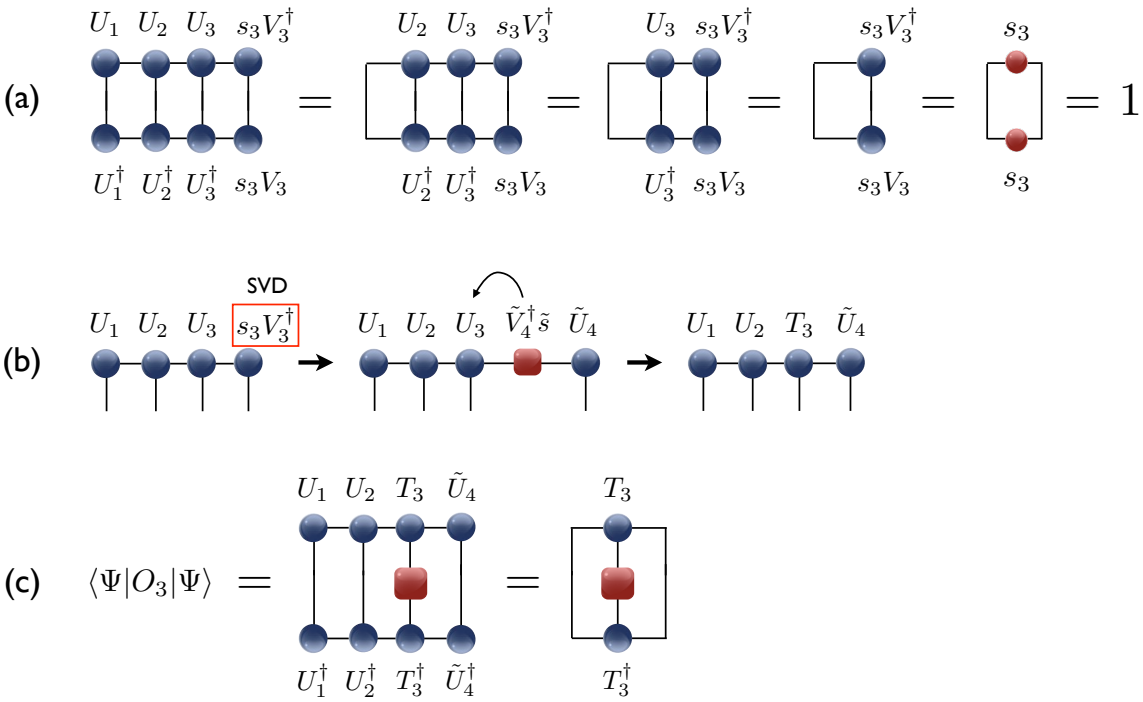


Figure 8: (a) The norm computed from a left-canonical MPS. (b) A right-canonical MPS can be obtained by performing a sequence of SVD's backwards from the right to the left. Here we started from a left-canonical MPS and stopped the procedure at site 3 which results in a *mixed-canonical* MPS with respect to site 3. The tensor T_3 is surrounded by unitary matrices U_1, U_2 on the left and $\tilde{U}_4$ on the right. (c) Computing a local expectation value of a local operator, here on site 3, simplifies a lot if the MPS is in the mixed-canonical form.

from the SVDs) which annihilate when they are multiplied with their corresponding conjugate $U_k^\dagger$. Thus, the network simplifies a lot when e.g computing the norm as shown in Fig. 8(a).

Similarly we can also create a right-canonical (right-normalized) MPS by performing the decomposition of the state Ψ starting from the right outermost site and proceed from right to left. Alternatively we can also start from a left-canonical MPS and perform consecutive SVD's from right to left. Instead of performing this procedure up to the first site on the left (which would yield a right-canonical MPS), we can also stop at a given site k. This results in a *mixed-canonical* MPS with respect to site k, see Fig. 8(b).

Having a MPS in a mixed-canonical form has several advantages: one example is shown in Fig. 8(c): if one wants to compute the expectation value of a local operator $\hat{O}$ on a site k it is very easy to evaluate this if the MPS is in the mixed-canonical form, or if one would like to compress the MPS to a smaller bond dimension then it is actually important to bring the MPS first into canonical form (see next section).

2.4 Compression of an MPS

We have seen how to select the best basis states when bi-partitioning a wave-function, namely by performing a Schmidt decomposition and then keeping the basis states with largest Schmidt values (i.e. singular values). We can use this idea to compress an

arbitrary bond in an MPS by bringing the MPS into a mixed canonical form with respect to that bond, i.e. with unitary tensors on the left and on the right of that particular bond, and a matrix of singular values in the middle of that bond. This is nothing but the wave function in the Schmidt decomposed form, i.e. we can just truncate the MPS to a smaller bond dimension $D' \to D$ by keeping only the D largest singular values out of the $D' > D$ singular values. Compression of a bond index is one of the key ingredients in a tensor network algorithm as we will see later.

2.5 Matrix product operators (MPOs)

In the previous sections we have learnt how to decompose a wave-function into an MPS. Similarly we can also write operators as a tensor network. A representation with one tensor per site (with two physical legs) is called a matrix product operator (MPO), with examples shown in Fig. 9. In principle we could obtain a representation of our Hamiltonian on an N-site system by performing a decomposition using a sequence of SVD's, similarly as when decomposing a wave-function. However, this is not done in practice since it would require to construct the N-site Hamiltonian explicitly which is of course highly inefficient. In practice the MPO of an arbitrary Hamiltonian can be constructed efficiently in a systematic way (see Schollwoecks review for details[4]).

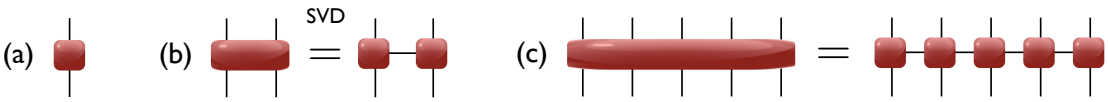


Figure 9: (a) A one-site operator with one incoming and one outgoing index (this is a matrix). (b) A two-site operator (Hamiltonian) can be split into two pieces by using an SVD. This corresponds to a 2-site MPO. (c) Similarly as for a wave-function, any N-site operator can be split into an N-site MPO. In practice, there are efficient ways how to directly construct the MPO for a given Hamiltonian (without using SVDs).

2.6 Energy minimization

Here we discuss how to optimize the variational parameters in the MPS to have the best approximation to the ground state for a given Hamiltonian $\hat{H}$. The idea is to iteratively optimize one tensor after the other while keeping the others fixed until convergence is reached. This iterative minimization ("sweeping") is one of the main ingredients of the DMRG method. (Originally DMRG was not formulated using MPS, but directly in the bases of the truncated Hilbert spaces.)

The minimization consists of the following steps:

1. Start with an initial MPS (randomly chosen, or obtained by some growth procedure starting from a small system).

2. Sweep from site $k = 1$ to site N and back. For each site k minimize the energy with respect to the tensor at this site T_k while keeping the other tensors fixed.

3. Repeat 2. until the energy is converged.

[4] U. Schollwöck, Annals of Physics 326, 96192 (2011).

Let's have a closer look at step 2, e.g., for $k = 2$. We want to minimize the energy with respect to the second tensor tensor T_2:

$$\frac{\partial}{\partial T_2^\dagger}\Big[\langle\psi|\hat{H}|\psi\rangle - \lambda\langle\psi|\psi\rangle\Big] = 0 \tag{13}$$

where λ is a Lagrange multiplier to keep the MPS wave function normalized. This expression is represented in Fig. 10(a).

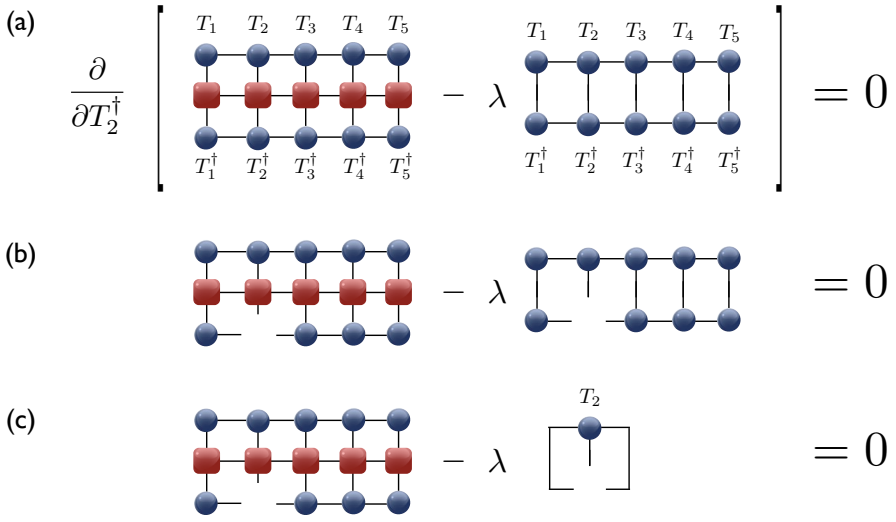


Figure 10: (a) Tensor network diagrams for the minimization of the energy with respect to the tensor $T_2^\dagger$. (b) The same expression as in (a) where we have taken the derivative with respect to $T_2^\dagger$. (c) The same as in (b) but in the case where the MPS is in a mixed-canonical form with respect to site 2.

We find that the derivative with respect to $T_2^\dagger$ simply corresponds to the network where we remove this tensor and leave the indices open, shown in Fig. 10(b). Thus, the minimization boils down to the following generalized eigenvalue problem,

$$A\vec{T_2} = \lambda B\vec{T_2}, \tag{14}$$

where A corresponds to the network on the left side of Fig. 10(b) without the T_2 tensor, where we reshape A into a matrix and T_2 into a vector. And similarly B corresponds to the network on the right side of Fig. 10(b) without the T_2 tensor (and also reshaped to a matrix). Solving this generalized eigenvalue problem for the lowest λ gives a new tensor T_2 which minimizes the energy (while keeping the others fixed).

The problem gets simplified to an ordinary eigenvalue problem,

$$A\vec{T_2} = \lambda\vec{T_2} \tag{15}$$

if the MPS is in a mixed-canonical form with respect to the site 2, shown in Fig. 10(c). Once the new tensor T_2 is found we replace it in the MPS and proceed with optimizing the next tensor T_3.

Above scheme is called the single-site update. Alternatively one can also perform a two-site update, in which two sites are updated at once (this is the scheme that has been originally used in DMRG), followed by an SVD of the two-site tensor to split it

into two independent tensors (which requires the MPS to be in mixed-canonical form with respect to that bond). The single-site update is computationally cheaper, but the two-site update has (typically) less problems in getting stuck in local minima during the optimization.

2.7 Imaginary time evolution

As an alternative to the energy minimization from the last section we can also find the ground state by performing an imaginary time evolution. If we act with $\exp(-\beta\hat{H})$ onto some initial state we obtain the ground state $|\psi_{GS}\rangle$ in the limit of β going to infinity:

$$e^{-\beta\hat{H}}|\psi_{init}\rangle \to |\psi_{GS}\rangle, \quad \beta \to \infty. \tag{16}$$

The reason is that for $\beta \to \infty$ this operator acts as a projector onto the state with the lowest eigenvalue, similarly as in a power method (i.e. a matrix A^k for $k \to \infty$ corresponds to a projection onto the extremal eigenvalue of A).

We first decompose the imaginary time evolution into M small time steps τ,

$$e^{-\beta\hat{H}} = \left(e^{-\tau\hat{H}}\right)^M, \quad \tau = \beta/M. \tag{17}$$

Here we assume that our Hamiltonian is given by a sum of nearest-neighbor terms, $\hat{H} = \sum_b \hat{H}_b$. For small enough time step τ we can do the following approximation, known as the (first-order) Trotter-Suzuki decomposition:

$$e^{-\tau\sum_b \hat{H}_b} \approx \prod_b e^{-\tau\hat{H}_b} \tag{18}$$

Note that this expression is only approximate because the terms $\hat{H}_b$ in general do not commute with each other!

Each term $e^{-\tau\hat{H}_b}$ corresponds to a two-body operator acting on two neighboring sites. We can therefore perform one imaginary time step onto our MPS by applying such a two-body operator between all the nearest-neighbor sites, as shown in Fig. 11(b). To evolve the MPS up to imaginary time $\beta = M\tau$ we have to perform M such steps.

In Fig. 11(c) it is shown how to proceed with one single two-body operator $e^{-\tau\hat{H}_b}$ acting on two sites: we can multiply it onto the two tensors, and retrieve two new tensors by performing an SVD. However, due to the application of the operator the bond dimension is typically increased from D to a certain D'. In order to keep the bond dimension bounded we perform a truncation, where we only keep the D largest singular values. We saw in Sec. 1.6 that keeping the largest singular values in of the wave-function in its Schmidt decomposed form minimizes the error of the truncation. In order to have the MPS in that form we need to bring it first into a mixed-canonical form with respect to one of two sites before we do the SVD. (If it is not in a mixed-canonical form then the singular values do not directly correspond to the Schmidt coefficients and thus the truncation would not be optimal).

A practical setup for an imaginary time evolution (which we will use in the exercises) is shown in Fig. 12(a). This scheme has the advantage that one can perform sweeps from left-to-right and back, while keeping the MPS automatically in the correct mixed-canonical form by absorbing the singular values either onto the right or left tensor, respectively. By reverting the order of the two-body operators one has automatically a second order Trotter-Suzuki decomposition, i.e. the Trotter error scales as $\mathcal{O}(\tau^2)$. For

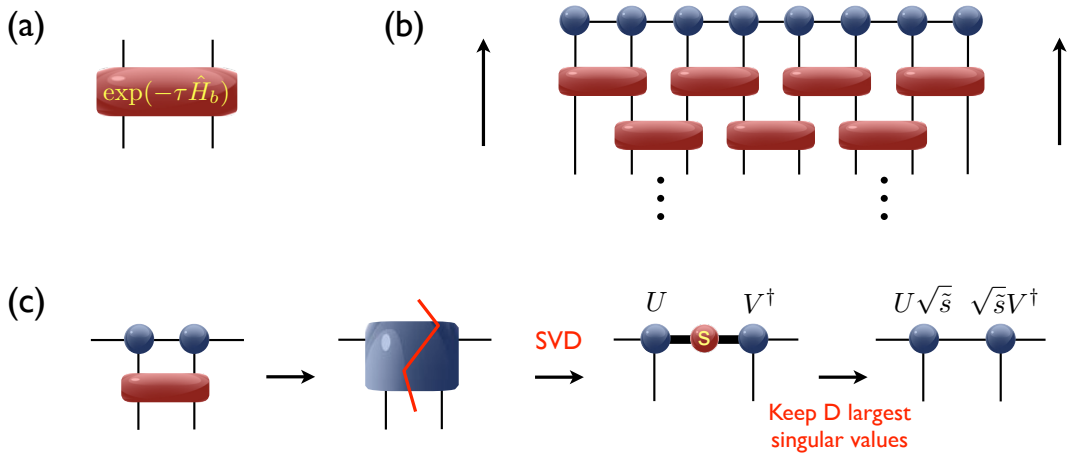


Figure 11: (a) The imaginary time evolution decomposes into a product of two-body operators, given by $\exp(-\tau \hat{H}_b)$. (b) One time step τ corresponds to applying the two-body operators between each pair of neighbors (here we have open boundary conditions). (c) Application of a two-body operator where an SVD is performed on the resulting tensor. Only the D largest singular values are kept in order to keep the bond dimension bounded. Note that the MPS needs to be in a mixed-canonical form with respect to the sites which are being updated (see text).

small time steps this error is in practice much smaller than the error due to the finite bond dimension D, but it is nevertheless advisable to carefully check the convergence of observables as a function of τ.

Alternatively, a full time evolution step can also be represented as an MPO, as illustrated in Fig. 12(b)-(c). The MPO can then be multiplied onto an MPS, followed by a compression of all the bonds of the MPS.

Instead of perform an imaginary time evolution we can also perform an evolution in real-time in a very similar way, by decomposing and applying $\exp(-it\hat{H})$ onto an initial MPS. However, it is known that a real-time evolution will typically create highly-entangled states over time, which require a (exponentially increasingly) large bond dimension. Thus, real-time evolutions at long time scales are much more challenging than ground state calculations (where the entanglement entropy remains bounded thanks to the area law).

2.8 Finite temperature simulations

In the previous section we have seen how to express $\hat{\rho} = \exp(-\beta \hat{H})$ as a tensor network, we can also directly evaluate observables $\hat{O}$ at finite temperature, $\langle \hat{O} \rangle = \mathrm{Tr}[\hat{\rho}\hat{O}]/\mathrm{Tr}\hat{\rho}$, by contracting the corresponding networks of the numerator and denominator. Besides this there are also other approaches to compute quantities at finite temperature (see Schollwoeck's review if you want to learn more).

2.9 Infinite matrix product states (iMPS)

We can also use an MPS to represent an infinite 1D system. If the system is translational invariant, we can parametrize the MPS simply by one single tensor A which is repeated infinitely many times in the ansatz, see Fig. 13(a). If the state breaks translational

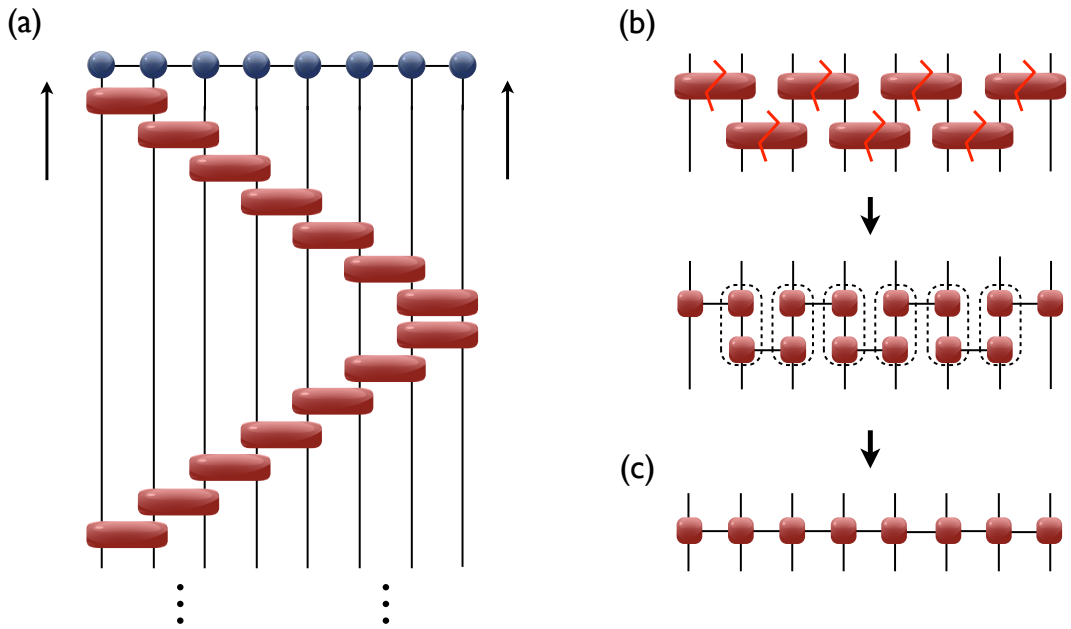


Figure 12: (a) Useful setup to perform an imaginary time evolution where the MPS can be kept in canonical form on-the-fly and having a second order Trotter-Suzuki decomposition at the same time. (b)-(c) Rewriting a full time evolution step as an MPO.

invariance, e.g. if it has a 2-sublattice structure, one can use a larger unit cell of tensors which is periodically repeated, see e.g. Fig. 13(b). For simplicity let's consider 1-site unit cells in the following.

To contract the norm of an infinite MPS consists of multiplying the transfer matrix T shown in Fig. 13(d) infinitely many times together, i.e. T^k applied to some boundary vector on the left and on the right end. In the limit $k \to \infty$ the only relevant contribution will come from the extremal left- and right- eigenstate of the matrix T, which we call v_L and v_R, respectively. Thus to compute a local expectation value we can compute v_L and v_R and contract the network shown in Fig. 13(c) (if the tensors are not normalized we also need to divide by the norm).

The iMPS can also be put into a canonical form shown in Fig. 13(e), where one keeps diagonal matrices (singular values) on the bonds. Also in this case one can enlarge the unit cell as shown in Fig. 13(f). The Γ tensors are such that if one absorbs s into Γ on the right, one obtains a right-unitary (i.e. a right-canonical MPS), and if one absorbs s into Γ on the left, one obtains a left-unitary (i.e. a left-canonical MPS), see Fig. 13(g), up to a normalization constant η. With the iMPS in this form it becomes particularly easy to bring it into mixed-canonical form with respect to a bond, e.g. to do a compression (see later), or to compute a local expectation value, see Fig. 13(h).

2.10 Imaginary time evolution with iMPS

We can do an imaginary time evolution algorithm for an iMPS very similarly as in the finite case. Using the canonical form makes the algorithm particularly simple. Due to the Trotter-Suzuki decomposition into even and odd bonds we use a unit cell of (at least) 2 different Γ tensors and s tensors. In Fig. 14 the approach is summarized. This

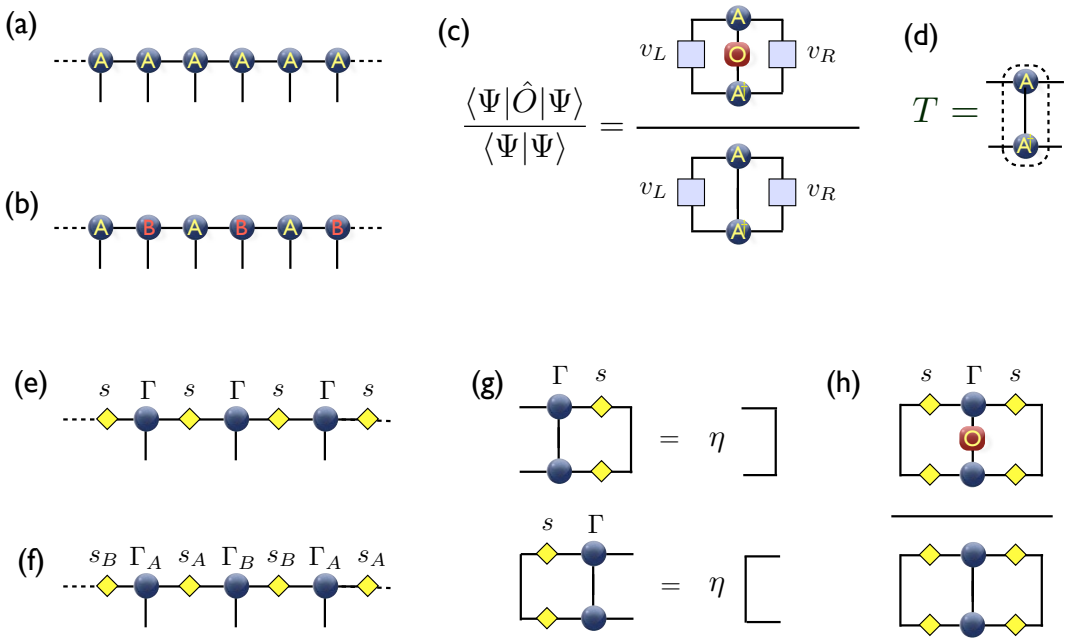


Figure 13: (a) Infinite MPS ansatz for a state with translational invariance. (b) iMPS with a 2-site unit cell. (c) Expectation of a local observable requires computation of the extremal left and right dominant eigenstates of the transfer matrix T shown in (d). (e) Canonical form of a translational invariant MPS with diagonal matrices (singular values) on the bonds. (g) Absorbing the s into Γ on the right (left) creates a right (left) canonical MPS (up to a constant factor η). (h) Thanks to the canonical form local expectation values of the iMPS can be easily computed.

method is also known as the iTEBD method (infinite time-evolving block decimation method). [5]

Strictly speaking the application of an imaginary time evolution gate destroys the canonical form, because it is a non-unitary evolution. In practice, however, it turns out that because the evolution operators are close to an identity (because the time step τ is small), the iMPS remains approximately in a canonical form throughout the evolution.

2.11 Infinite DMRG

As in the finite case we can also use an iterative energy minimization algorithm to obtain an iMPS representing the ground state of an infinite system. Here we describe the iDMRG algorithm which starts from small system and lets the system grow by adding 2 new sites in the middle of the system until one reaches the desired system size. This is actually the very first DMRG method that was introduced by S. White in 1992.[6]

The algorithm is most conveniently implemented using an MPO representation of the Hamiltonian and the steps are shown in Fig. 15. At each iteration one introduces two new sites in the middle and performs a two-site update, i.e. one finds a new tensor T_m on two sites by solving the corresponding eigenvalue problem, and then split T_m

[5] G. Vidal, Phys. Rev. Lett. 98, 070201 (2007); G. Vidal, Phys. Rev. Lett. 91, 147902 (2003).
[6] S. R. White, Phys. Rev. Lett. 69, 28632866 (1992).

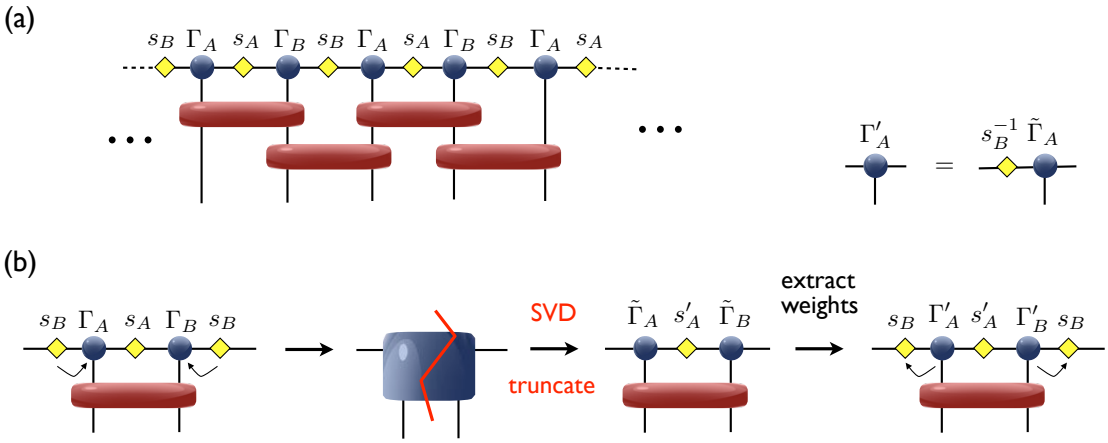


Figure 14: Imaginary time evolution of an iMPS. (a) One time step of the evolution corresponds of applying the imaginary-time evolution operators between odd and even bonds onto the iMPS, here in canonical form. Note that the neighboring weights (here s_B) need to be included in order to bring the iMPS into mixed-canonical form. These weights can be extracted again from the resulting tensors $\tilde{\Gamma}_A$ and $\tilde{\Gamma}_B$ by multiplying them by the inverse of the weights (here s_B^{-1}).

into two tensors using an SVD. The MPS is always kept in a mixed-canonical form with respect to the center, i.e local expectation values can be easily computed using the T_m tensor. After each iteration the representation of the Hamiltonian of the left (right) side of the system H_L (H_R) is updated, as shown in Figs. 15)(e)-(f).

The steps shown in Figs. 15)(c)-(f) are repeated until convergence is reached. If we reach a system size which is much bigger than the correlation length of the system then local expectation values in the middle of the system will be approximately equal to the values in the thermodynamic limit.

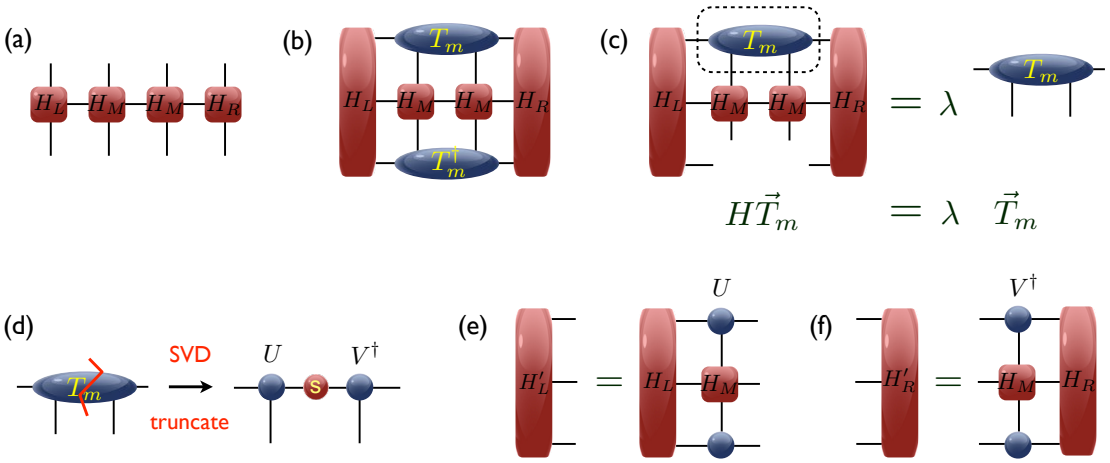


Figure 15: Steps of the iDMRG method. (a) Start with the MPO representation of a small system. H_L, H_M and H_R are the left, middle, right MPO tensors, respectively. (b) Expectation value of the Hamiltonian with respect to the state represented by T_m. Note that on the left and right boundary sites we have simply put MPS tensors corresponding to an identity. (c) We optimize tensor T_m by taking the derivative with respect to $T_m^\dagger$ and solve the resulting eigenvalue problem (see Sec. 2.6) for the lowest eigenvalue (energy). (d) The resulting tensor T_m is split by using an SVD and keeping the D largest singular values. (e)-(f) Update of the left and right Hamiltonian terms. Since we always absorb unitaries to the left and to the right we can keep the MPS automatically in canonical form with respect to the center.

3 Tree tensor networks and the MERA

There exist other types of tensor networks, as for example tree tensor networks (TTN) and the multi-scale renormalization ansatz (MERA)[7] which are inspired by ideas from the real-space renormalization group,[8] see Fig. 16. Instead of having one tensor per lattice site as in an MPS, there are tensors appearing on different length scales. Here we only briefly outline some of the main ideas. For a detailed and pedagogic introduction see e.g. [G. Vidal, arXiv:0912.1651 (2009)].

The TTN consists of tensors (isometries) that map a block of sites into a new coarse-grained site where not all the states of the blocks are kept, but only χ states.[9] One can easily show that upon bipartition of a system the entanglement entropy of a TTN is also bounded by $\log(\chi)$ as in a matrix product state, i.e. independent of system size, corresponding to the area law in 1D.

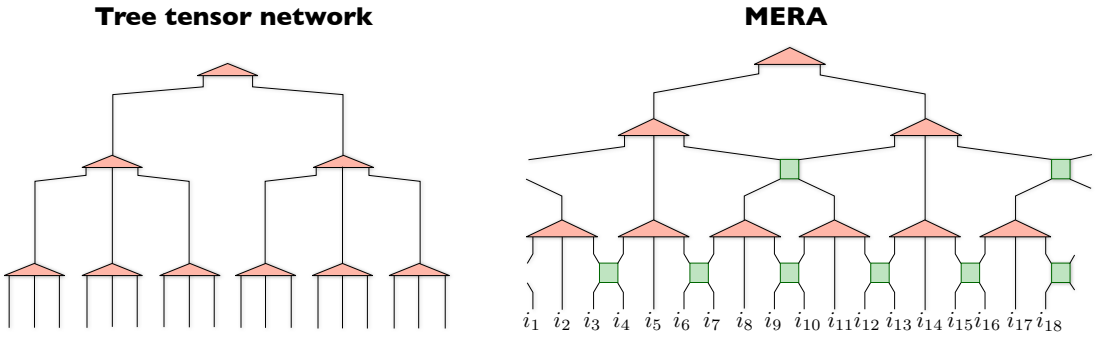


Figure 16: Tree tensor network and the multi-scale renormalization ansatz (MERA) in one dimension with a 3-to-1 coarse graining. The structure of a TTN and the MERA are not unique, e.g. one can also perform a 2-to-1 coarse-graining scheme.

The MERA has an additional type of tensors, called disentanglers, which are unitary transformation acting across the boundary of two blocks to reduce the short-range entanglement between the blocks before they are coarse-grained. Thanks to the disentanglers one can show that the MERA can reproduce a logarithmic correction of the entanglement entropy, i.e. $S(L) \sim log(L)$ (see Fig. 18), and algebraically decaying correlations, making this a natural ansatz for critical states in 1D.

Expectation values of local observables (also two-point correlation functions) can be efficiently computed thanks to the fact that isometries and unitaries cancel with their conjugates, see Fig. 17 which shows the evaluation of a 2-site local operator. All tensors lying outside the so-called *causal cone* of the operator cancel, and one is left with a tensor network involving only a few tensors in each layer. It is a defining property of the MERA that it has a causal cone with a *bounded width*, i.e. at each level of coarse-graining only a few tensors ($\mathcal{O}(1)$) lie inside the causal cone.

There exist not a unique TTN/MERA, but one can design different coarse-graining schemes, e.g. a 2-to-1 coarse graining scheme instead of a 3-to-1 scheme. Depending on the structure, the computational cost is different. The 3-to-1 structures shown in Fig. 16 have a computational cost of χ^4 (TTN) and χ^8 (MERA) to contract the networks, i.e. larger than in MPS algorithms (D^3). The standard 2-to-1 scheme for the

[7] G. Vidal, Phys. Rev. Lett. 101, 110501 (2008).
[8] G. Vidal, Phys. Rev. Lett. 99, 220405 (2007)
[9] One usually denotes the bond dimension by χ instead of D here.

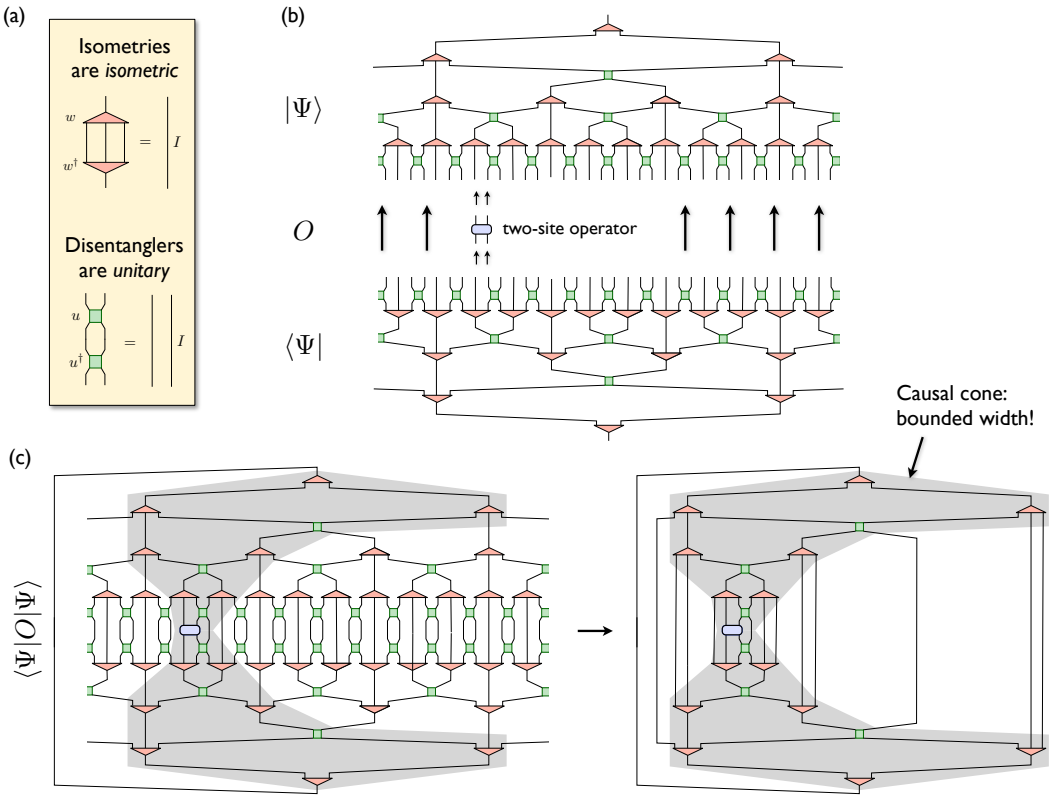


Figure 17: (a) The tensors in the MERA fulfill special properties: both isometries and unitaries annihilate their corresponding conjugates, resulting in an identity, represented by straight lines. This can be used to simplify contractions a lot, e.g. when computing an expectation value of a local 2-site operator shown in (b) and (c). (c) Tensor network representing the expectation value of a 2-site operator which simplifies into the network shown in (d).

MERA has a slightly higher cost of χ^9, but there exist also cheaper versions with χ^7. In general there is a tradeoff between the efficiency of the coarse-graining structure and the involved computational cost.

As in an infinite MPS (or periodic MPS) one can exploit translational invariance by using one isometry and disentangler repeated in the spatial direction (but different pairs of tensors in different layers). For a critical state one can exploit the fact that the wave-function has a scale-invariant structure. In this case one can make an ansatz with one isometry and one unitary which are the same in each layer. It has been pointed out [10] that the scale invariant MERA for the ground state of a quantum spin chain can be interpreted as a discrete realization of the AdS/CFT correspondence - a connection which is currently being explored also in high-energy physics.

The optimization of the MERA is usually done by performing an iterative energy minimization, sweeping over all the tensors in the ansatz, in a similar way as in DMRG, however, with the difference that one restrains the resulting tensor to be unitary or isometric. For details on the algorithm we refer to Ref. [G. Evenbly and G. Vidal, Phys. Rev. B 79, 144108 (2009)].

[10] B. Swingle, arXiv:0905.1317; G. Evenbly and G. Vidal, J Stat Phys 145, 891918 (2011)

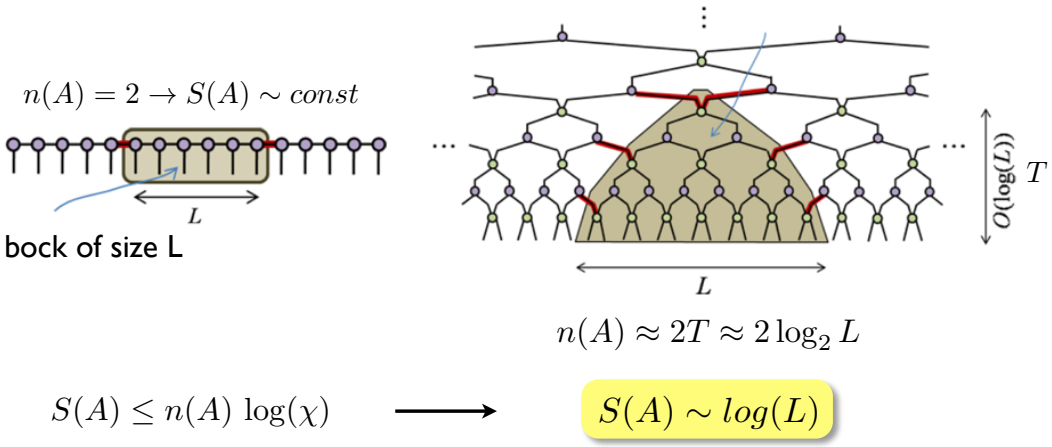


Figure 18: Logarithmic scaling of the entanglement entropy in the MERA. Figures are taken from Ref. [11] below. The maximal entanglement entropy between a region A and the rest of the system is bounded by $n(A)\log(\chi)$, with χ the bond dimension (D in the MPS) and $n(A)$ the (minimal) number of connections between A and the rest of the system. In the MPS case $n(A) = 2$, i.e. independent of L giving rise to the 1D area law. In the MERA, however, the minimal number of connections grows with L: there are two connections in each layer, and the number of layers involved is given by $\approx log_2 L$ (in the 2-to-1 coarse graining scheme). Thus $S(A) \sim log(L)$ as in a critical state in 1D.

The MERA can also be extended to 2D systems, as e.g. shown in Fig. 19 which however has a pretty high computational cost ($\mathcal{O}(\chi^{16})$). One can show that this ansatz can reproduce an area law in 2D, i.e. $S(L) \sim L$. There exist critical states in 2D which still obey an area law (typically systems with point nodes in k-space), whereas other critical states have a logarithmic correction, i.e. $S(L) \sim L \log L$ (typically systems with a 1D Fermi surface in k-space, e.g. free fermions). The latter scaling cannot be reproduced by a conventional MERA, however, there exist generalizations of the MERA, called branching MERA, which can reproduce it (and other scalings). [11]

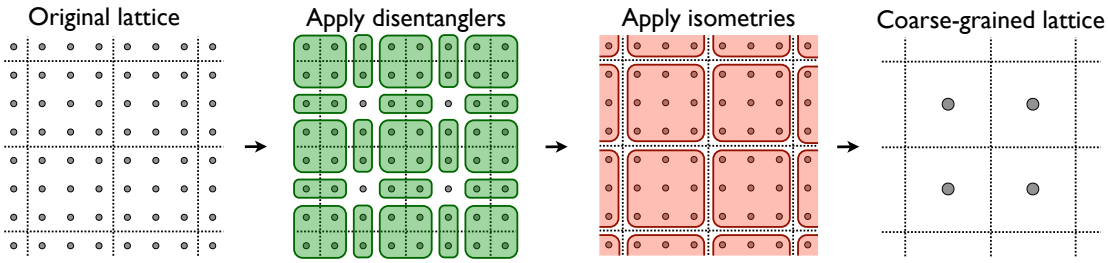


Figure 19: 2D MERA (top view) with a 3-to-1 coarse graining scheme.

[11] G. Evenbly and G. Vidal, J. Stat. Phys. 145, 891918 (2011)

4 Tensor networks in 2D

An introduction to 2D tensor networks will be given on slides (see dropbox folder for a copy of the slides).