Progress 2017

David Aceituno

Contents

2017

Half of January

- Started on exact diagonalization of the quantum Ising spin model, using C++ with the Eigen math-library
- Started the Many Particle Physics course.

February

- Worked with Many Particle Physics HW problems.
- Started a new course: FSI3000 History of physics and epistemology, 5 hp.
- Review of entanglement and entanglement measures
- Continued work on exact diagonalization. See dropbox paper.
- Learned about different software for diagonalization in C++. Nice find: Spectra. Main takeaway: large matrices are really slow to diagonalize, one is basically limited just a few eigenvalues (largest/smallest algebraic or absolute values), and even then it becomes intractable.

March

- Last work on Many Particle physics HW problems and its final presentation. Got an A! Yay! Incidentally, the last HW, a presentation-problem, was about the analytic solution to the quantum Ising spin model.
- Finished the review of entanglement by doing some exercies See dropbox paper.
- More serious look at Schollwöck's paper. Tried to grasp the notation and possible implementation of Schmidt decomposition.

- Tried doing minimal examples of successive Schmidt decompositions by hand, with two or three "bit"-particles. Confusion ensued. The explanations by Pollman's and Schollwöck lack some intermediate steps or are unclear, for what happens between each Schmidt-iteration. See GitHub paper. (It probably has some errors)
- Explored note-taking software. Result: Markdown+Latex (Mathjax/Katex) works great. Math can be copy-pasted into latex if needed. Dropbox paper has a limited version of this. Better to use Typora, ReText or StackEdit (online). Tried Jupyter briefly but I don't think it's the correct tool yet. Another promising programming notebook: Beaker, if it could only just work... check in the future.

April

- Looked at other software packages to see if I could reproduce results from these. ITensor stood out, but it is kind of slow according to Jens. Still, it is good to keep around to compare results and get hints on implementation.
- Started to implement a simplest case scenario of MPS decompositon. Basically this step:

$$c_{\sigma_1,\sigma_2...\sigma_n} = \sum_{a_1,a_2...a_{n-1}} A^{\sigma_1}_{a_1} A^{\sigma_2}_{a_1,a_2} ... A^{\sigma_n}_{a_{n-1}}$$

for an arbitrary tensor c. As usual this is done in C++ with the Eigen's math library. - A discussion with Jens sorted out the confusion I had about " tensor flattening". Also, between successive Schmidt decompositions the resulting matrices from the SVD need to be interpreted as tensors properly. Before continuing to the next step. This was the missing piece.

May

- Prepared a presentation for the course FSI3000
- Found a great book that clarified a lot. Basically I want the Vidal Canonical form $\Gamma\Lambda\Gamma\Lambda...$ so I can get any A (left-canonical) or B (right-canonical) form later.

- Succeeded in correctly interpreting U and V matrices as tensors after each SVD.
- Small victory: Truncating the SVD down to χ was quite simple.
- After correctly understanding storage orders and a related bug with RowMajor in Eigen's implementation of SVD, all my problems were gone. The MPS decomposition is finally correct for an arbitrary tensor state.
- Checked out the Julia programming language for. **If C++ becomes too cumbersome, Julia is a good option!!**
- Nice find: RedSVD can do SVD quickly on huge matrices using a randomized algorithm.
- Started working on implementing TEBD. Pollmann's paper is great at explaining this conceptually. I am mainly copying to C++ the steps in itebd.py by Jens and Frank Pollmann, while checking the steps in the paper.
- Somewhat distracted by the last solid mechanics exam.

June

Week 1

- Continued working on iTEBD work on C++. Some minor takeaways:

- Matrix exponentials are available in Eigen::Core for regular Matrix types, and not for Tensors. Luckily I've already written an interface to switch between these types.

- I finally understood all the reshapes and transposes (shuffles) of indices and why they were needed. Example: Say we want to obtain $\theta^{jk}_{\alpha\gamma}$ from the contraction $A^{j}_{\alpha\beta}B^{k}_{\beta\gamma}$. Then we do $A^{j}_{\alpha\beta}B^{k}_{\beta\gamma} = \{\text{contract}[2,1]\} = \tilde{\theta}^{j\alpha}_{k\gamma} = \{\text{shuffle}[0,2,1,3]\} = \theta^{jk}_{\alpha\gamma}$.
- Found that it was much clearer to work with the $\Gamma^A\Lambda^A\Gamma^B\Lambda^B$ -notation as in the paper, instead of the $B^AB^BB^C$ notation in the original iTEBD.py code.
- Finally got the same ground state result as in iTEBD.py! YAY!

Week 2

- C++ vs Python showdown! I did time comparisons using exactly the same settings. Subsections the code were timed as well. The settings for the table below were:
- On my laptop, 3 runs, 10000 iterations each, $\chi = 15$.

- C++ compiled with -O3 -march=native (gcc 6.3.0)
- SVD using Eigen::BCDSVD() (Bidiagonal Divide and Conquer SVD). This is expected to scale much better with increasing matrix size compared to Eigen::JacobiSVD().
- Python3, version 3.5.2.

The runtime differs by a factor ~4.3. However the SVD section, which differs by a factor ~1.6, is expected to scale worse than the others as χ increases, and it is not clear yet if this factor remains constant.

	C++ [s]	Python [s]
SVD	0.890 | 0.898 | 0.893	1.487 | 1.496 | 1.485
Time evolution $U(\delta t)\theta$	0.277 | 0.282 | 0.272	1.460 | 1.479 | 1.452
Reshaping SVD results	0.061 | 0.064 | 0.062	1.281 | 1.310 | 1.288
Update Γ (or B in python)	0.085 | 0.086 | 0.085	1.202 | 1.242 | 1.210
Total	**1.316 | 1.333 | 1.315**	**5.695 | 5.795 | 5.701**

- Tried out RedSVD. Specifically a header version on github. I can't get it to work properly. The usage is very simple, almost identical to Eigen, but I only get `nan` as a result. In any case, it seems like the performance is only excellent for low-rank approximations of sparse matrices, i.e. when the rank of the decomposition k << n, where n is a linear size of the matrix. Otherwise it performs similarly or worse, if I understand correctly.
- Discovered RStudio (free/open source/win-linux-osx) with support for R Markdown and R Notebook documents. While I don't intend to learn R language, RStudio is also a powerful (R)Markdown editor. Using knitr + pandoc in its back-end, it has full latex support (not only math!), as well as C++ and Python integration so that code chunks can be executed mid-text, similar to Jupyter.
- Relearned Tikz/PGF to make pretty tensor diagrams. Made a `tensorgraphics.sty` file to easily import them in latex preamble with `\usepackage{tensorgraphics}`. The usage is basically
- Tensor of rank 2: `\tensorII{name}{left index}{right index}`
- Tensor of rank 3: `\tensorIII{name}{physical index}{left index}{right index}`
- . . . And so on. See more in the code examples.
- Started work on iDMRG

Week 3

- My iDMRG code compiles and runs fine. However, the eigenvalue engine `Spectra` is not working correctly: it gives me completely wrong eigenvalues from the start, and arrives at an incorrect ground state energy. Perhaps I'm using it wrong. It's usage is very similar to scipy eigsh. Not all eigenvalue solvers allow one to define the matrix-vector product (most want you to supply a matrix), so it may be somewhat cumbersome to try another solver.
- Realized that the $B^A B^B B^C$ notation in `itebd.py` and `idrmg.py` was actually Hastings suggestion for improving numerical stability, mentioned in:

  [1] Hastings, M B. 2009. "Light Cone Matrix Product." Interpreting 87545: 15. doi:10.1063/1.3149556. Page 5-6

  [2] Schollwoeck, Ulrich. 2010. "The Density-Matrix Renormalization Group in the Age of Matrix Product States." Annals of Physics 326 (1): 96–192. doi:10.1016/j.aop.2010.09.012. Page 84-85.

However, in the update there's still an inverse multiplication! Why?

```
X = np.transpose(np.tensordot(np.diag(s[ia] ** (-1)), X, axes=(1, 1)), (1, 0, 2))
B[ib] = np.tensordot(X, np.diag(s[ib]), axes=(2, 0))
```

Which essentially amounts to $B^A = (s^A)^{-1} X s^B$, equation 18 in [1], which is exactly what we were trying to avoid! Instead we should just have the lines

```
Z = np.transpose(np.reshape(Z, (chi2, d, chic)), (1, 0, 2))
B[ib] = np.tensordot(C, Z.conjugate(), axes=([1, 3], [0, 2]))
```

I.e., equation 19 in [1].

Unfortunately, this gives me completely inaccurate results... Have I misunderstood Hastings paper? Schollwöcks paper seems to agree with me, in eq. 274 in [2].

Week 4

- Solved the problem with Hastings trick for numerical stability. The wavefunction B has to be normalized using the same factor as for Λ^A. This is not explained clearly in Hastings or Schollwöcks papers, but was clear in TenPyLight.
- Resolved issue with eigensolver. Basically in the user-provided function `perform_op` for Spectra, the contents of my vector needed to be moved into the C-style array `double - y_out` using `std::move` and not the assignment operator `=` . iDMRG is now working.
- **I passed the solid mechanics exam!!!!!! YEEEHAW**

July

Week 1 (Vacations in Northern Sweden/Norway)

I've read some papers:

Paper 1:

> Schindler, Frank, Nicolas Regnault, and Titus Neupert. 2017. "Probing Many-Body Localization with Neural Networks," April, 1–11. http://arxiv.org/abs/1704.01578.

Very clear exposition on the MBL-ETH transition, and how neural networks work and how it aids the phase discrimination. There are some very interesting remarks here summarized:

- Several measures allow a [. . .] distinction of thermal and localized regimes. [. . .]These include energy level statistics, level statistics analyses of the entanglement spectrum and studies of the distribution of the entanglement entropy over a region of energy eigenstates. Necessarily, these methods make several implicit assumptions about the nature of either regime or about the transition. The neural network based method for identifying the ETH-MBL transition that we present here is a priori less biased, assuming only that the information for distinguishing the ETH from the MBL regime is – in some form – contained in the entanglement spectrum. This is useful in particular in situations where the physical characteristics of a phase are not fully understood, as one may certainly argue to be the case for MBL. Thus, the neural network approach also allows for the possibility of finding ways of characterizing the phase transition beyond established methods.
- In between the two limits, the behavior of a specific system being either ETH or in MBL depends on the specific disorder realization and the eigenstate that is considered. Averaging over disorder realizations removes these dependences, but the transition between ETH and MBL regimes may still depend on the energy density at which the system is probed, which amounts to the existence of a many-body mobility edge. In addition, an intermediate regime, the so-called Griffith phase may be present.
- Several possibilities have been explored to determine from the entanglement properties whether a state $|\psi\rangle$ at finite energy density and fixed disorder shares the character of the MBL or ETH regime.

1. The "Schmidt gap" $\lambda_1(\rho_A) - \lambda_2(\rho_A)$, where $\{\lambda_j(\rho_A); \lambda_j \geq \lambda_{j+1}\}$ denotes the spectrum of ρ_A.
2. ETH states have volume-law entanglement scaling, while MBL states have area-law entanglement scaling.
3. The standard deviation σ_E of a sample of entanglement entropies.
4. The level spacings in the entanglement spectrum follow distinct statistical distributions in the ETH and MBL regimes.

Paper 2

Carleo, Giuseppe, and Matthias Troyer. 2017. "Solving the Quantum Many-Body Problem with Artificial Neural Networks." *Science* 355 (6325): 602–6. doi:10.1126/science.aag2302.

Clear on certain aspects of the neural network approach. I do not understand Fig.2, which seems important. They call these "feature maps"... what is that?

Some interesting parts:

- In this case, RBM artificial networks are constituted by one visible layer of N nodes, corresponding to the physical spin variables in a chosen basis (e.g., $\mathcal{S} = \sigma_1^z, ..., \sigma_N^z$) and a single hidden layer of M auxiliary spin variables $(h_1, ..., h_M)$ (Fig. 1). This description corresponds to a variational expression for the quantum states

$$\Psi_M(\mathcal{S}; \mathcal{W}) = \sum_{\{h_i\}} e^{\sum_j a_j \sigma_j^z + \sum_i b_i h_i + \sum_{ij} W_{ij} h_i \sigma_j^z}$$

where $h_i = \{-1, 1\}$ is a set of M hidden spin variables and the network parameters $\mathcal{W} = \{a, b, W\}$ fully specify the response of the network to a given input state $\mathcal{S}$. Because this architecture features no intralayer interactions, the hidden variables can be explicitly traced out, and the wave function reads.

$$\Psi_M(\mathcal{S}; \mathcal{W}) = \sum_{\{h_i\}} e^{\sum_j a_j \sigma_j^z} \times \prod_{i=1}^{M} F_i(\mathcal{S})$$

where $F_i(\mathcal{S}) = 2\cosh[b_i + \sum_j W_{ij} \sigma_j^z]$. The network weights are, in general, to be taken complex-valued to provide a complete description of both the amplitude and the phase of the wave function.

- One of the practical advantages of Neural-network quantum states (NQS) is that its quality can, in principle, be systematically improved by increasing the number of hidden variables. The number M (or, equivalently, the density $\alpha = M/N$ then plays a role analogous to the bond dimension for the MPS. However, the correlations induced by the hidden units are intrinsically nonlocal in space and are therefore well suited to describe quantum systems in arbitrary dimension.
- [. . .] deep network architectures and convolutional neural networks, can constitute the basis of more advanced NQS and therefore have the potential for increasing their expressive power. Furthermore, the extension of our approach to treat quantum systems other than interacting spins is, in principle, straight- forward. In this respect, applications to answer the most challenging questions concerning inter- acting fermions in two dimensions can already be anticipated.
- Finally, at variance with tensor network states, the NQS feature intrinsically nonlocal correlations, which can lead to substantially more compact representations of many-body quantum states. A formal analysis of the NQS entanglement properties might therefore bring about substantially new concepts in quantum information theory.

Week 2

- After communication with Jens I'll focus on two goals in the near future: Finite DMRG, and Finite entanglement scaling. In the latter, reproduce the relevant plots in:

  Pollmann, Frank, Subroto Mukerjee, Ari M. Turner, and Joel E. Moore. 2009. "Theory of Finite-Entanglement Scaling at One-Dimensional Quantum Critical Points." *Physical Review Letters* 102 (25): 1–6. doi:10.1103/PhysRevLett.102.255701.

- Started work on finite-DMRG (fDMRG). It looked simple enough.
- To implement fDMRG I've looked at
- `simple-dmrg` : A program I found on Github that does **not** use MPS.
- Schollwöcks paper from 2011.
- A doctoral dissertation thesis by Michael Sekania.
- Density Matrix Renormalization Group for dummies (without MPS). Qualitative.
- Martin Jakob Ganahl's thesis
- New Trends in Density Matrix Renormalization (without MPS). Mostly for qualitative reading.
- An introduction to numerical methods in low-dimensional quantum systems. (without MPS).
- Got some mixed results. The algorithm makes sweeps, but behaves very strangely or crashes with large chi. . .

Week 3

- Worked on a fluke, i.e. not really. Turns out it only worked with low chi because then the Gamma tensors are all the same sizes and can be contracted even if there's something fundamentally wrong.
- Started work on restructuring the code. I'm rewriting it object oriented, to make it clearer and more expandable.
- I realize some parts of the program I hadn't really understood properly. For instance, why and how the Hamiltonian MPO looks the way it does. Same goes for left/right environment tensors. Restructuring the code helped pinpoint these gaps in understanding. However I still don't exactly understand the way the Hamiltonian MPO matrix is reshaped and transposed into a useful tensor format. It simply works. [**Actually, this was later understood in December**]
- Program does the iDMRG-part well but the fDMRG part crashes. Somehow the legs of neighboring Γ^A and Γ^B aren't compatible when loading them from storage.

Week 4

- **fDMRG finally works!** There were issues when storing and loading the MPS and environments. The bookkeeping was tricky. Also, for the two-site MPS $\Theta = \Lambda^B_{n-1}\Gamma^A_n\Lambda^A_n\Gamma^B_{n+1}\Lambda^B_{n+1}$ I had forgotten that the first Λ is from position $n-1$ when doing finite DMRG, as opposed to $n+1$ as in iDMRG.

The algorithm should be clear from the commented code below:

```
// superblock contains the MPO, Left/Right environments
// and two-site MPS at the current position on the chain.

// S contains previous MPS and environments created during
// infinite-DMRG for chain lengths up to max_length.

// direction == 1 means sweep right, -1 means sweep left.

void finite_DMRG(class_superblock &superblock, class_storage &S, long sweeps){
    long direction  = 1;
    long sweep = 0;
    while(sweep < sweeps) {

      // Load the two-site MPS (i.e., GA, LA, GB, LB) and Left/Right
      // environments of the current position into the superblock.
      S.load(superblock);

      // Read the external bond dimensions of the superblock,
      // i.e. d, chi_a, and chi_b for a tensor Theta.
      //                      d     d
      // Theta =    chi_a__.__|___.__|__.__chi_b
      //                   LB GA LA GB LB
      superblock.update_bond_dimensions();

      // Find the lowest eigenvalue and eigenstate, corresponding to ground
      // state MPS, by diagonalising the tensor network defined by Theta,
      // Left/Right environments and the Hamiltonian MPO.
      superblock.find_ground_state();

      // Truncate the found eigenstate using SVD decomposition, keeping "chi" singular
      // values at the bond LA depicted above.
      superblock.truncate();

      // Update GA, LA and GB.
      superblock.update_MPS();

      // Compute the energy density and compare with exact value.
      superblock.print_error();

      // Store MPS and environments of the current superblock to "disk"
      S.overwrite(superblock);

      // Move one step according to "direction" by
      //    1) enlarging left OR right environment (contraction with current MPS)
      //    2) Storing the enlarged environment to disk
      //    3) position += direction.
      S.move(superblock, direction);
    // Turn around when reaching the end of the chain.
    if (S.position_L <= 1 || S.position_R >= S.max_length - 1) {
      direction *= -1;
    }

    // Increase sweep when passing by the middle of the chain.
    if(S.position_L == S.max_length/2 -1 && S.position_R == S.max_length/2){
      sweep++;
    }
  }
```

- Starting work on batch processing to study finite entanglement scaling.
- Set up a Github repository

August

Week 1

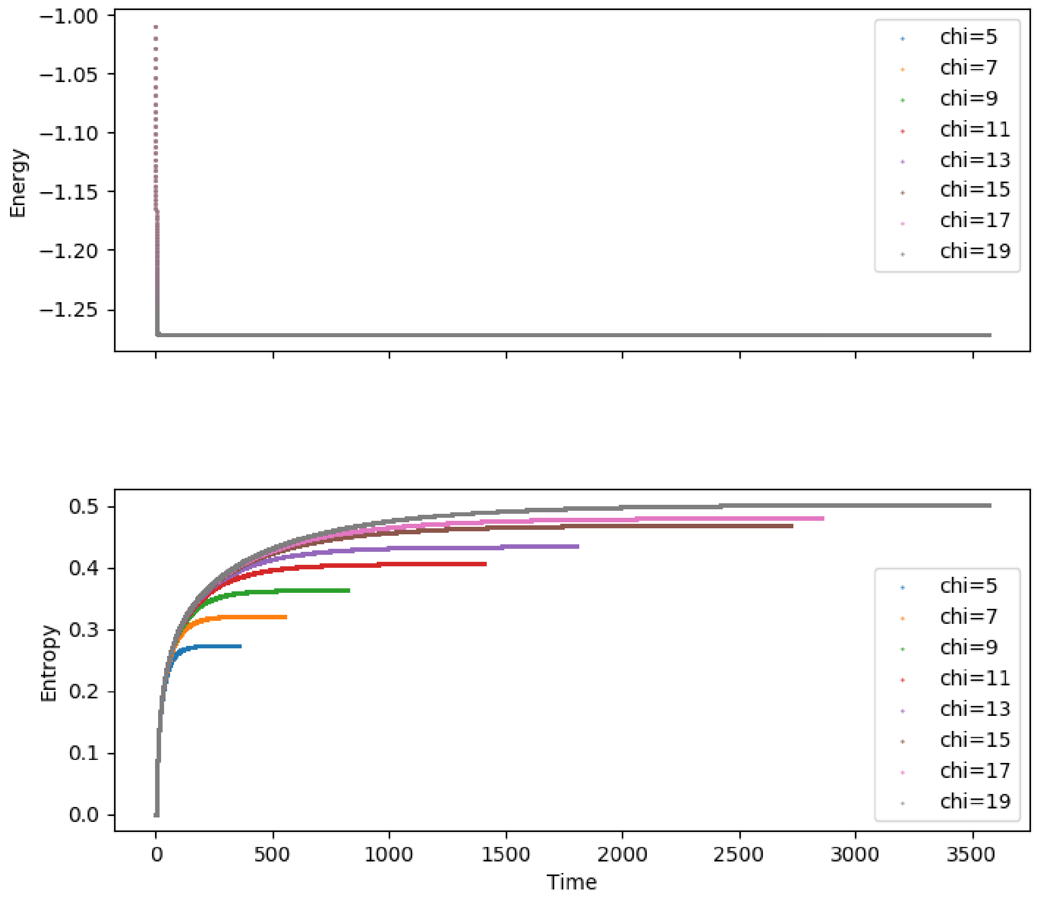
- HDF5
- Documentation, Doxygen.
- **Finally got my Bachelors, Masters and Civil Engineering diplomas!!! Weeee!**

Week 2

- Continued HDF5 and polishing up code structure.
- Continued on documentation.
- Wrote code to get data out of the program in hdf5 format and plot it conveniently in python. First plots are starting to emerge.

Week 3

- Got some plots for (non-)convergence of iTEBD, that looked like log-divergence. After discussion with Jens I realized I just needed to simulate a while longer. Then it converged



properly.

- Started to check if Finite-entanglement scaling results from itebd and idmrg agree. Initially, they do not! This is very puzzling. The ground state energies agree but entropies do not!

Week 4

- Continued checking and bug-checking to see if I can make itebd and idmrg entanglement scaling results agree. They still do not!
- Tips from Jens: Check orthogonality conditions, i.e. canonicalization. Right now the iTEBD algorithm converges to the correct entanglement entropies while iDMRG does not. The correct values can be found in Pollmanns paper, following the formula $S_E = \frac{1}{\sqrt{\frac{12}{c}}+1} \log \chi$. See picture below:
- Reading up on Vidal's paper on how to "canonicalize", i.e., convert any iMPS into canonical form.
- Began to prepare QM assistant exercise lecture 1.

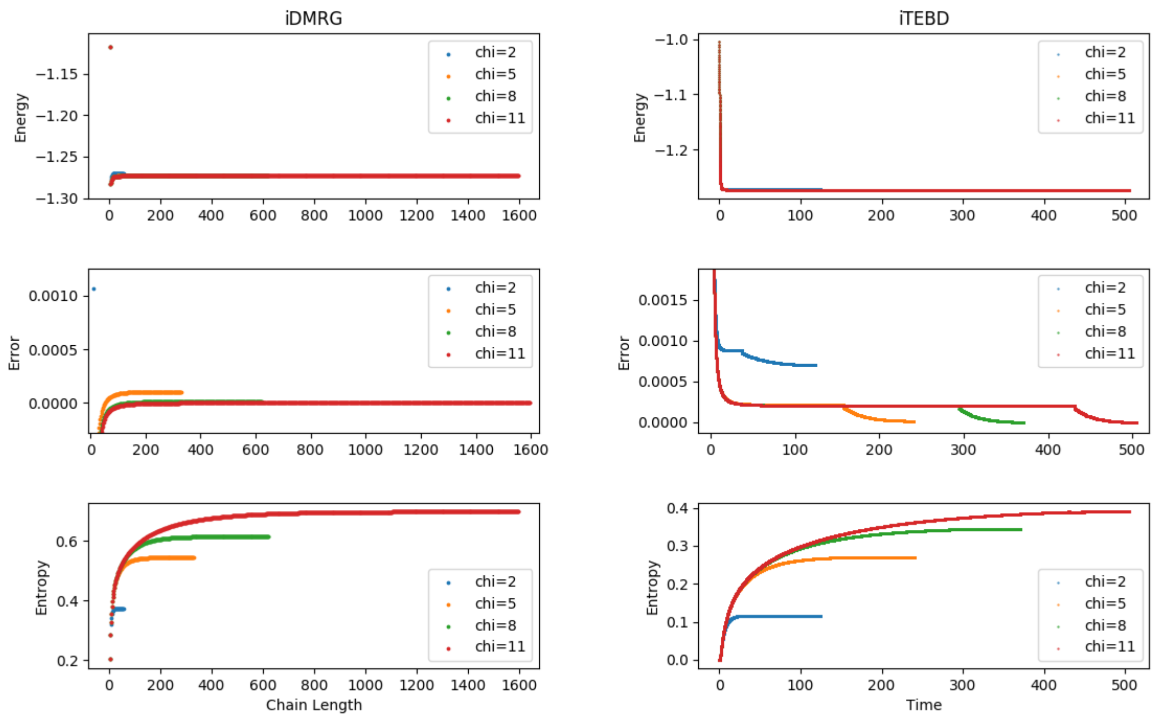


Figure 1: iDMRG vs iTEBD

September

Week 1

- **Got officially admitted to the PhD program!!! Finally!!!!!**
- Prepared QM exercise lecture 2 and 3.
- Continued implementation of orthogonalization for iMPS. Something is really wrong. The canonicalization destabilizes the computation, and it shouldn't, as far as I can tell. There are 4 suggested implementations for doing this and I've done them all: Vidal, McCulloch & Vidal 2014, McCulloch 2008, and the clearest yet by Ran, Shi-Ju, Emanuele Tirrito, Cheng Peng, Xi Chen, Gang Su, and Maciej Lewenstein. 2017. "Review of Tensor Network Contraction Approaches," September. http://arxiv.org/abs/1708.09213

Week 2

- Prepared QM exercise lecture 4 and 5
- Started correcting QM homework problem set 1
- Continued work on orthogonalization. I've understood many things that I've been doing wrong. For instance I hadn't understood the meaning of computing **left** eigenvectors, in a way that differs from the right ones fundamentally. This, among some misunderstandings were really hindering progress. Now I feel that I'm close to solving the issue.

Week 3

- QM exercise 6 and 7.
- Correcting QM homework problem set 1. This is taking a lot of time.
- Writing down answers to chapter 2 in Griffiths to put up on course webpage

Week 4

- QM exercise 8 and 9 + computer lab. Tough week.
- Went to meet Jonas Kjäll to discuss a project idea to study the MBL/TH (mobility edge?) transition. In short, his idea is to do the smallest possible injection of a TH state into one that is clearly MBL, by swapping a column-row pair from the respective Hamiltonians. More precisely we discussed:
- Create a Hamiltonian that is clearly in the non-localized (ergodic) regime (perhaps a GOE matrix?) and a Hamiltonian that is clearly in MBL (perhaps an Anderson-like model with random field strengths h_i at each site), and swap a column-row pair between these two Hamiltonians. Explore the characteristics of such a system, and study the possibility of getting a phase in between ergodic and localized regime.
- Note that you should check the energy levels in the middle, if you order them after diagonalization. This is where the level spacing should be smallest.
  - Using entanglement entropy $S_E = -\mathrm{Tr}\rho_A \log \rho_A$ as order parameter. Here $\rho_A = \mathrm{Tr}_B \rho_{AB} = \mathrm{Tr}_B |n\rangle\langle n|$ is the density matrix of subsystem A , and $|n\rangle = |A> |B> = |$ďďď... $> |$ďďď... $>$ is the full many body state vector of L particles “¤”, composed of subsystems A and B .
  - Apparently, according to D. Page (1993) the entropy of the thermal phase is $S_{Th} \approx \frac{1}{2}(L \log(2) - 1)$ or something like that, whereas $S_{MBL} =$ constant due to area law scaling.
  - We can probably look into using the Rényi entropy as an order parameter also, which to second order ($q = 2$) may be less computationally expensive than Neumann entropy (Modak, R., & Mukerjee, S. (2015)). It is defined as $S_q = \frac{1}{1-q} \log \sum_{i=1}^{n} p_i^q$, where p_i is a probability distribution, $p_i = \Pr(X = i)$. In practice the choice $q = 2$ is often used, and the studied quantity is $S_2 = -\log_2(\mathrm{Tr}_A \rho_A(t)^2)$. It says in Luitz, Huveneers & de Roeck (2017) that $S_2 \propto L \log 2$ for ergodic systems while $S_2 \propto$ constant if localized.

October

Week 1

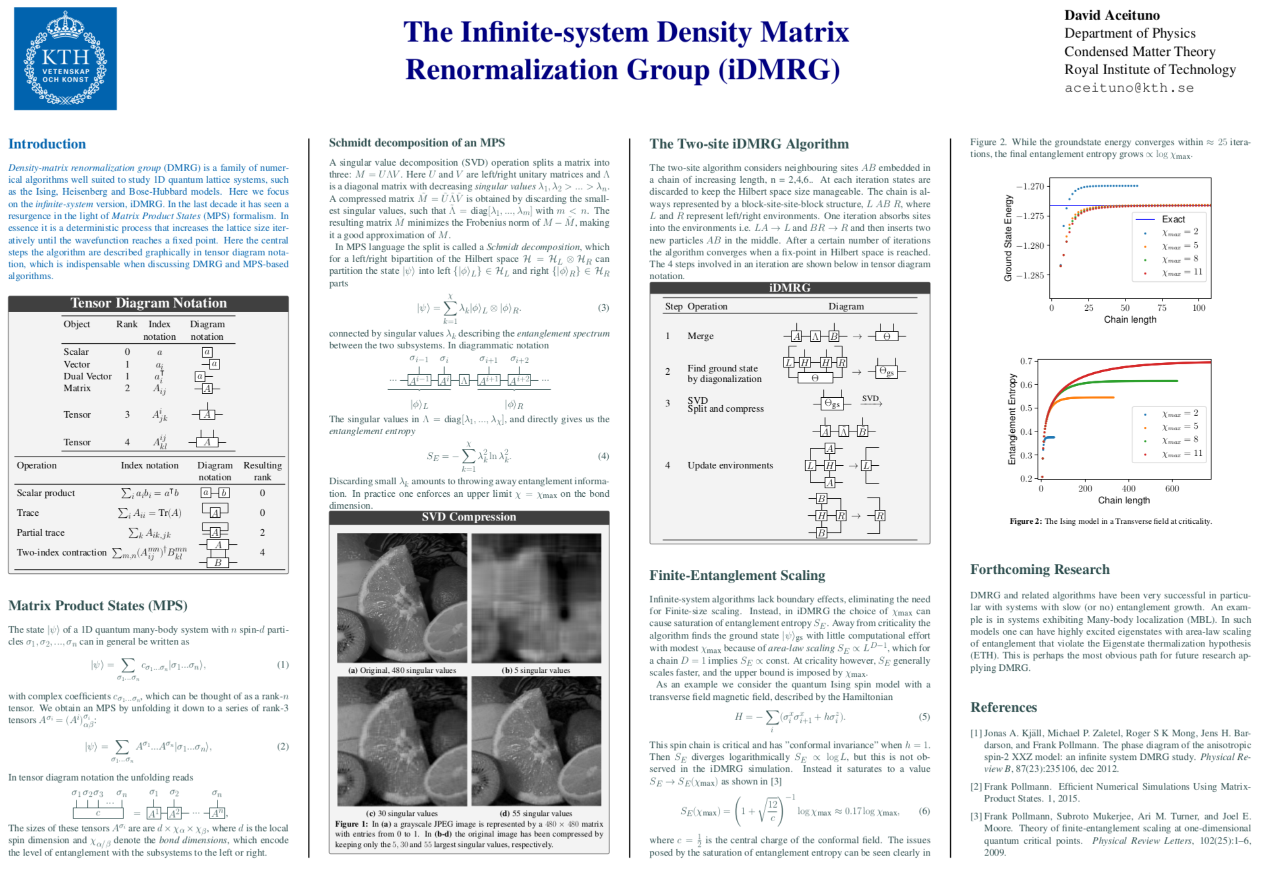
- QM excercise 10
- Continued work on canonicalization. This week I'm giving it everything I've got. I have tried
- Orus and Vidal's original approach by coarse graining (2008)
- Phien and Vidal's iterative approach (2014)
- Ran, Tirrito and Peng's approach that does not coarse grain but still explains how to canonicalize a two-site system.

None of them work.

- Decided with Jens and Jonas to read two papers until next friday: Luitz, D. J., Huveneers, F., & de Roeck, W. (2017). How a small quantum bath can thermalize long localized chains. and Thiery, T., Huveneers, F., Müller, M., & De Roeck, W. (2017). Many-body delocalization as a quantum avalanche

Week 2

- QM excercise 11 and 12, the last ones!!



The Infinite-system Density Matrix Renormalization Group (iDMRG)

David Aceituno
Department of Physics
Condensed Matter Theory
Royal Institute of Technology
aceituno@kth.se

Introduction

Density-matrix renormalization group (DMRG) is a family of numerical algorithms well suited to study 1D quantum lattice systems, such as the Ising, Heisenberg and Bose-Hubbard models. Here we focus on the *infinite-system* version, iDMRG. In the last decade it has seen a resurgence in the light of *Matrix Product States* (MPS) formalism. In essence it is a deterministic process that increases the lattice size iteratively until the wavefunction reaches a fixed point. Here the central steps the algorithm are described graphically in tensor diagram notation, which is indispensable when discussing DMRG and MPS-based algorithms.

Tensor Diagram Notation

Object	Rank	Index notation	Diagram notation
Scalar	0	a	$\boxed{a}$
Vector	1	a_i	$-\boxed{a}$
Dual Vector	1	$a_i^\dagger$	$\boxed{a}-$
Matrix	2	A_{ij}	$-\boxed{A}-$
Tensor	3	A^i_{jk}	$-\boxed{A}-$
Tensor	4	A^{ij}_{kl}	$-\boxed{A}-$

Operation	Index notation	Diagram notation	Resulting rank
Scalar product	$\sum_i a_i b_i = a^\intercal b$	$\boxed{a}-\boxed{b}$	0
Trace	$\sum_i A_{ii} = \mathrm{Tr}(A)$	$\boxed{A}$	0
Partial trace	$\sum_k A_{ik,jk}$	$\boxed{A}$	2
Two-index contraction	$\sum_{m,n}(A^{mn}_{ij})^\dagger B^{mn}_{kl}$	$\boxed{A}$ $\boxed{B}$	4

Matrix Product States (MPS)

The state $|\psi\rangle$ of a 1D quantum many-body system with n spin-d particles $\sigma_1, \sigma_2, ..., \sigma_n$ can in general be written as

$$|\psi\rangle = \sum_{\sigma_1...\sigma_n} c_{\sigma_1...\sigma_n}|\sigma_1...\sigma_n\rangle, \tag{1}$$

with complex coefficients $c_{\sigma_1...\sigma_n}$, which can be thought of as a rank-n tensor. We obtain an MPS by unfolding it down to a series of rank-3 tensors $A^{\sigma_i} = (A^i)^{\sigma_i}_{\alpha\beta}$:

$$|\psi\rangle = \sum_{\sigma_1...\sigma_n} A^{\sigma_1}...A^{\sigma_n}|\sigma_1...\sigma_n\rangle, \tag{2}$$

In tensor diagram notation the unfolding reads

$$\overset{\sigma_1\sigma_2\sigma_3 \cdots \sigma_n}{\boxed{\quad c \quad}} = \overset{\sigma_1}{\boxed{A^1}}-\overset{\sigma_2}{\boxed{A^2}}-\cdots-\overset{\sigma_n}{\boxed{A^n}},$$

The sizes of these tensors A^{σ_i} are are $d \times \chi_\alpha \times \chi_\beta$, where d is the local spin dimension and $\chi_{\alpha/\beta}$ denote the *bond dimensions*, which encode the level of entanglement with the subsystems to the left or right.

Schmidt decomposition of an MPS

A singular value decomposition (SVD) operation splits a matrix into three: $M = U\Lambda V$. Here U and V are left/right unitary matrices and Λ is a diagonal matrix with decreasing *singular values* $\lambda_1, \lambda_2 > ... > \lambda_n$. A compressed matrix $\tilde{M} = \tilde{U}\tilde{\Lambda}\tilde{V}$ is obtained by discarding the smallest singular values, such that $\tilde{\Lambda} = \mathrm{diag}[\lambda_1, ..., \lambda_m]$ with $m < n$. The resulting matrix $\tilde{M}$ minimizes the Frobenius norm of $M - \tilde{M}$, making it a good approximation of M.

In MPS language the split is called a *Schmidt decomposition*, which for a left/right bipartition of the Hilbert space $\mathcal{H} = \mathcal{H}_L \otimes \mathcal{H}_R$ can partition the state $|\psi\rangle$ into left $\{|\phi\rangle_L\} \in \mathcal{H}_L$ and right $\{|\phi\rangle_R\} \in \mathcal{H}_R$ parts

$$|\psi\rangle = \sum_{k=1}^{\chi} \lambda_k |\phi\rangle_L \otimes |\phi\rangle_R. \tag{3}$$

connected by singular values λ_k describing the *entanglement spectrum* between the two subsystems. In diagrammatic notation

$$\cdots \underbrace{-\overset{\sigma_{i-1}}{\boxed{A^{i-1}}}-\overset{\sigma_i}{\boxed{A^i}}-}_{|\phi\rangle_L}\boxed{\Lambda}\underbrace{-\overset{\sigma_{i+1}}{\boxed{A^{i+1}}}-\overset{\sigma_{i+2}}{\boxed{A^{i+2}}}-}_{|\phi\rangle_R} \cdots$$

The singular values in $\Lambda = \mathrm{diag}[\lambda_1, ..., \lambda_\chi]$, and directly gives us the *entanglement entropy*

$$S_E = -\sum_{k=1}^{\chi} \lambda_k^2 \ln \lambda_k^2. \tag{4}$$

Discarding small λ_k amounts to throwing away entanglement information. In practice one enforces an upper limit $\chi = \chi_{\max}$ on the bond dimension.

SVD Compression

(a) Original, 480 singular values (b) 5 singular values

(c) 30 singular values (d) 55 singular values

Figure 1: In **(a)** a grayscale JPEG image is represented by a 480×480 matrix with entries from 0 to 1. In **(b-d)** the original image has been compressed by keeping only the $5, 30$ and 55 largest singular values, respectively.

The Two-site iDMRG Algorithm

The two-site algorithm considers neighbouring sites AB embedded in a chain of increasing length, n = 2,4,6.. At each iteration states are discarded to keep the Hilbert space size manageable. The chain is always represented by a block-site-site-block structure, $L\ AB\ R$, where L and R represent left/right environments. One iteration absorbs sites into the environments i.e. $LA \to L$ and $BR \to R$ and then inserts two new particles AB in the middle. After a certain number of iterations the algorithm converges when a fix-point in Hilbert space is reached. The 4 steps involved in an iteration are shown below in tensor diagram notation.

iDMRG

Step	Operation	Diagram
1	Merge	$-\boxed{A}-\boxed{\Lambda}-\boxed{B}- \rightarrow -\boxed{\Theta}-$
2	Find ground state by diagonalization	$\boxed{L}-\boxed{H}-\boxed{H}-\boxed{R}$, $\boxed{\Theta}$ $\rightarrow -\boxed{\Theta_{gs}}-$
3	SVD Split and compress	$-\boxed{\Theta_{gs}}- \xrightarrow{\text{SVD}} -\boxed{\tilde{A}}-\boxed{\tilde{\Lambda}}-\boxed{\tilde{B}}-$
4	Update environments	$\boxed{A}$, $\boxed{L}-\boxed{H}-$, $\boxed{A}$ $\rightarrow \boxed{L}-$; $\boxed{B}$, $-\boxed{H}-\boxed{R}$, $\boxed{B}$ $\rightarrow -\boxed{R}$

Finite-Entanglement Scaling

Infinite-system algorithms lack boundary effects, eliminating the need for Finite-size scaling. Instead, in iDMRG the choice of $\chi_{\max}$ can cause saturation of entanglement entropy S_E. Away from criticality the algorithm finds the ground state $|\psi\rangle_{\mathrm{gs}}$ with little computational effort with modest $\chi_{\max}$ because of *area-law scaling* $S_E \propto L^{D-1}$, which for a chain $D = 1$ implies $S_E \propto$ const. At cricality however, S_E generally scales faster, and the upper bound is imposed by $\chi_{\max}$.

As an example we consider the quantum Ising spin model with a transverse field magnetic field, described by the Hamiltonian

$$H = -\sum_i (\sigma_i^x \sigma_{i+1}^x + h\sigma_i^z). \tag{5}$$

This spin chain is critical and has "conformal invariance" when $h = 1$. Then S_E diverges logarithmically $S_E \propto \log L$, but this is not observed in the iDMRG simulation. Instead it saturates to a value $S_E \to S_E(\chi_{\max})$ as shown in [3]

$$S_E(\chi_{\max}) = \left(1 + \sqrt{\frac{12}{c}}\right)^{-1} \log \chi_{\max} \approx 0.17 \log \chi_{\max}, \tag{6}$$

where $c = \frac{1}{2}$ is the central charge of the conformal field. The issues posed by the saturation of entanglement entropy can be seen clearly in

Figure 2. While the groundstate energy converges within ≈ 25 iterations, the final entanglement entropy grows $\propto \log \chi_{\max}$.

Figure 2: The Ising model in a Transverse field at criticality.

Forthcoming Research

DMRG and related algorithms have been very successful in particular with systems with slow (or no) entanglement growth. An example is in systems exhibiting Many-body localization (MBL). In such models one can have highly excited eigenstates with area-law scaling of entanglement that violate the Eigenstate thermalization hypothesis (ETH). This is perhaps the most obvious path for future research applying DMRG.

References

[1] Jonas A. Kjäll, Michael P. Zaletel, Roger S K Mong, Jens H. Bardarson, and Frank Pollmann. The phase diagram of the anisotropic spin-2 XXZ model: an infinite system DMRG study. *Physical Review B*, 87(23):235106, dec 2012.

[2] Frank Pollmann. Efficient Numerical Simulations Using Matrix-Product States. 1, 2015.

[3] Frank Pollmann, Subroto Mukerjee, Ari M. Turner, and Joel E. Moore. Theory of finite-entanglement scaling at one-dimensional quantum critical points. *Physical Review Letters*, 102(25):1–6, 2009.

Figure 2: Fig

- Meeting with Jonas and Jens about the MBL mobility edge project.
- Decided that I could start by checking what happens if we replace the GOE ergodic grain in the paper by Luitz, with a simple spin operator σ_0^z, so the Hamiltonian becomes: $H = \epsilon_0\sigma_0^z + \sum \epsilon_i \sigma_i^z + \sum e^{-i/\xi}\sigma_i^z\sigma_0^z$, i.e. where ϵ_0 could simply be 1, and $\epsilon_i \in [-W, W]$, either with or without bias as explained in the same paper.
- Try to reproduce the plots in Figure 2, basically calculate the gap ratio $r = \frac{\min(\Delta E_k, \Delta E_{k+1})}{\max(\Delta E_k, \Delta E_{k+1})}$, with $\Delta E_k = E_{k+1} - E_k$.
-

Week 3

- Correct HW problem set 3 in QM
- Create a poster presentation for next week's P@K boat-conference

Week 4

- Monday to Wednesday on P@K conference. See my poster below.
- Kept working on canonicalization and some refactoring on code.

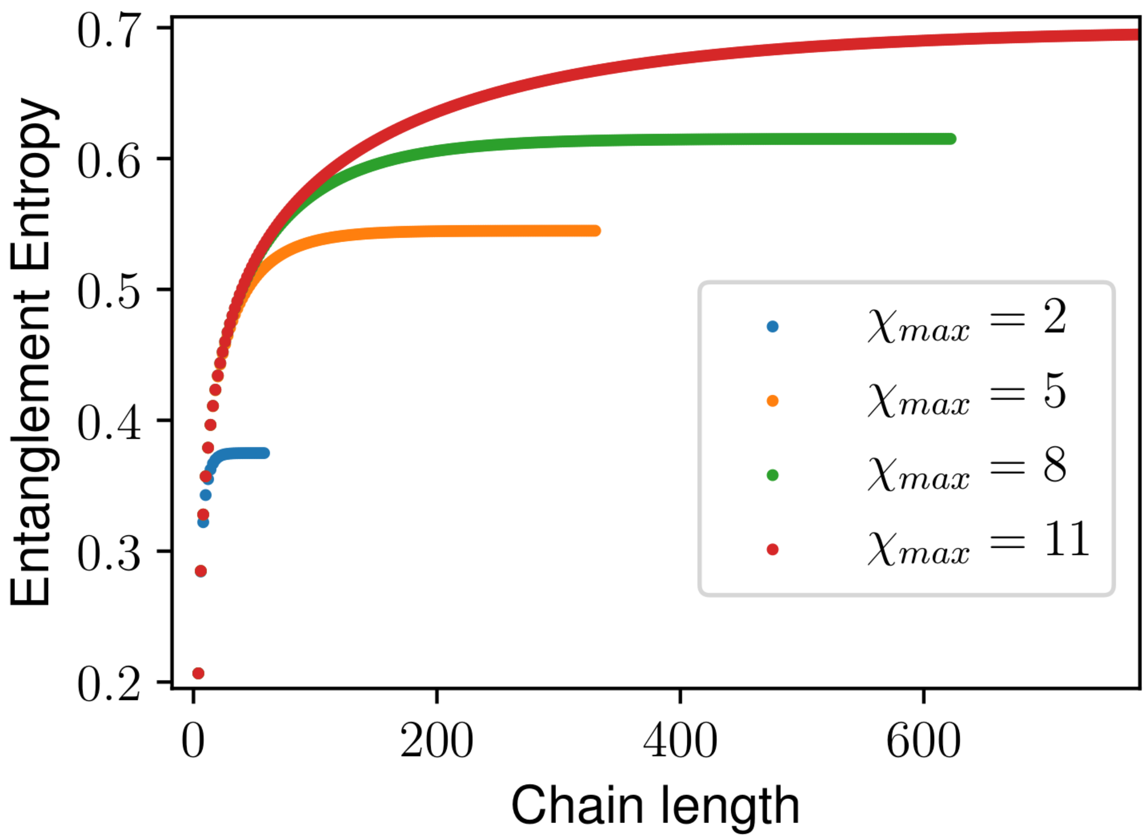


Figure 3: Wrong entanglement entropy

November

Week 1

- Had a first meeting with Axel where we discussed DMRG, delayed convergence and canonicalization. We agreed on the following

1. Make a plot of S_E vs iterations with $J = \pm 1$
2. Start environments in a different way, by basically contracting into them a couple of relevant initial MPS states. This could fix delayed convergence, see the next points below.
3. Check the iterative method
4. Check out the robust SVD
5. Check Clang + Cmake on ny own code.

- Recall that the whole reason for starting to look at canonicalization is that Jens and I believe that the singular values are wrong due to not having properly left and right canonical MPS. The correct values should be given by $S_E = \frac{1}{\sqrt{\frac{12}{c}}+1} \log \chi$, with $c = 1/2$ for the TFQ Ising model at criticality.
- However, I noticed earlier that there is a delayed convergence!! I just happened to run the simulation much longer and this happened:
- Axel suggested fixing this delayed convergence by either contracting some MPS into the environment

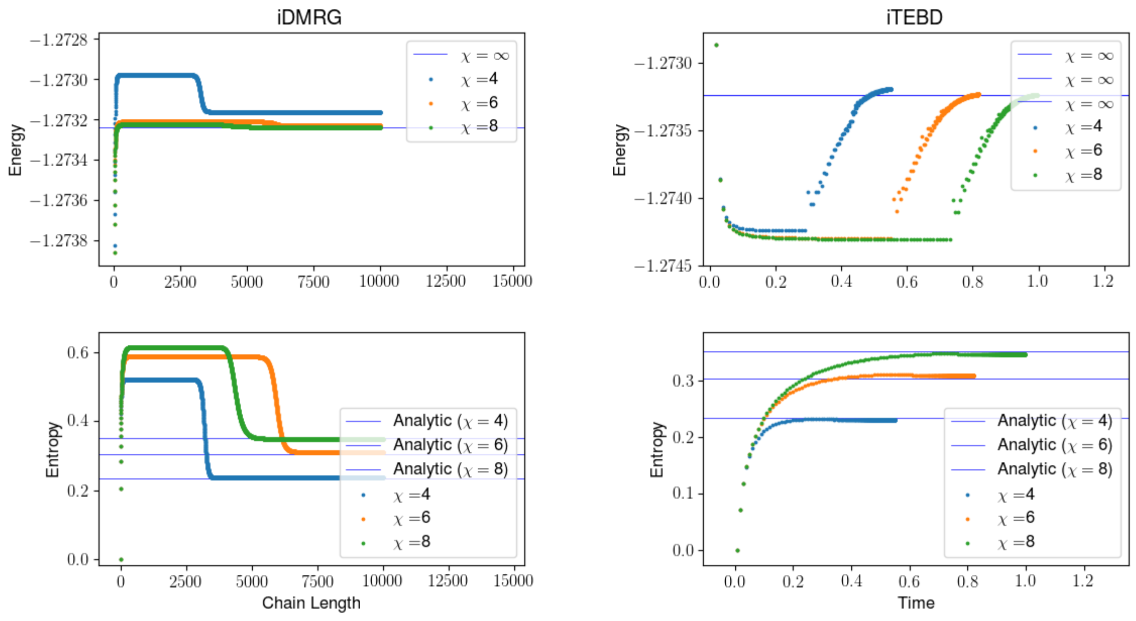


Figure 4: Delayed Convergence

before starting the simulation, or by starting with another initial MPS with greater overlap to the ground state.

- **UPDATE**: Axels first suggestion worked!! Contracting some MPS'es into the environment. Why though? And could canonicalization help convergence at all?

Week 2 Belgium Tensor Network School!

- Had a lot of fun in Belgium. Had workshops on DMRG and PEPS.
- One afternoon I got Canonicalization working. But then the next day it didn't work again. Very strange. In any case, I've realized that I probably don't need Canonicalization to begin with.
- I had been under the impression that the truncation of singular values destroyed orthogonality. I asked Ulrich Schollwöck and he said no, it does not. In fact, the effect of the truncation will be to make the identity matrix a smaller matrix, when computing the left/right eigenmatrix of the transfer matrix. This is more in line with what I've already seen. Thus, because SVD already spits out left/right orthogonal matrices, I don't need to "re-orthogonalize" them. Perhaps if I ever need to have canonical form on the full 2-site transfer matrix, that will be needed (and then you may take the iterative approach! Otherwise, no… just don't bother.
- PEPS and MERA was quite interesting theoretically but there are just so many challenges in implementation. I suppose after trying PEPS with Roman Orus it doesn't feel too distant anymore, but there be dragons in the deep end.
- One other PhD student had made a "MERA-light" approach to capture long-range correlations in 1D system by doing coarse graining of the chain. I was fairly impressed.

His thesis can be found here https://esc.fnwi.uva.nl/thesis/centraal/files/f749277872.pdf

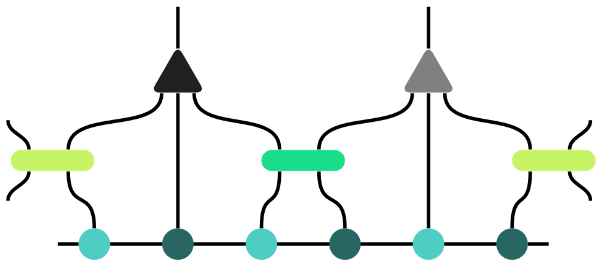


FIGURE 3.6: Full coarse graining and disentangling of the Hamiltonian operator. All three possible placements of the operator are taken into account.

Note the even-odd structure off all tensors, symbolized by different shades.

Figure 5:

Week 3

- Learned how to compute the number of operations, or computational cost, of contracting tensor networks. Implemented the fastest way I could find and enjoyed a speedup.

In summary, always contract the tensor network in such a way as to keep the order of χ the lowest, if possible. For instance, when contracting L - H - H - L with θ (in the Lanczos ground state search step), a stupid thing would be to contract L and R first, thereby getting a cost $\propto \chi^4$ in the next step. Instead start by contracting one of the χ legs of L or R, with θ, and then the subsequent steps will all be $\propto \chi^3$ or less.

- Also, the order of the legs given to Eigen/Numpy, to contract in a multiple-leg contraction, is also important. Always start with the largest first! For larger tensor networks it is probably best to order the indices before contracting!

- Started computing the variance of the MPS. There are two suggestions:

1. Find the left dominant eigenmatrix of the transfer matrix for $A - H - H - A^\dagger$, and similarly the right dominant eigenmatrix of the transfer matrix $B - H - H - B^\dagger$, where H is in MPO-form. These eigenmatrices can probably be used as left/right environment blocks when computing the variance.
2. Keep track of a larger left/right environment from the beginning, just as the regular left/right environments that are used for computing energy.

- **UPDATE!** The suggestion with transfer matrices above does not work!

Week 4

- Computing the variance turns out to be hard. The transfer matrices do not help much.
- Started computing variance with moment generating functions and characteristic functions instead.
- I can get correct energy with the method above, but the variance seems wrong, way too large. I'll try computing the variance in the slow way to compare (basically keeping track of another set of environments that can be contracted with two MPO's).

December

Week 1

- Held a presentation in the course Research Methodology in Physics, "Life as a PhD student". That was fun. Perhaps I can earn a module credit for the course SH3000 Width and ethics something something. Ask Pär.

- Continued implementing variance using moment generating functions. Did some restructuring of code to have clearer file management during the weekend.

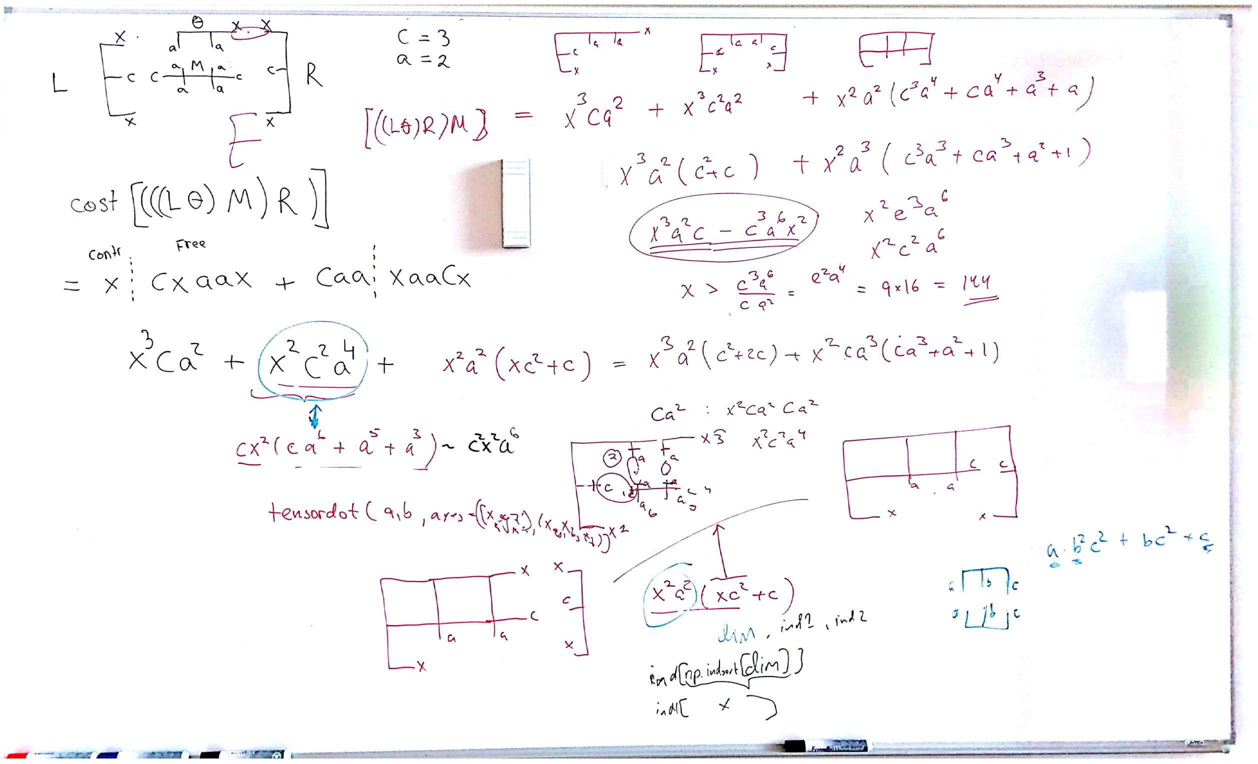


Figure 6: Computational cost

- Discussed a new project with Loic for studying MBL with TEBD algorithms. The idea is to look at a paper by Cirac for implementation of Ancillas /auxiliary particles to simulate random models that give MBL on systems that are still translationally invariant, such that we can use TEBD/DMRG.

1. Let's say we work with the XXZ model $H_0 = \sum_i J(S_i^X S_{i+1}^X + S_i^Y S_{i+1}^Y + \Delta S_i^Z S_{i+1}^Z)$. We add an additional term $H_1 = \sum_i h_i S_i^Z$ where h_i is a random field at each site, such that the total Hamiltonian becomes $H = H_0 + H_1$.

2. Now we follow the prescription by Cirac, where we discretize $h_i \in \{-D, ..., D\}$ in n levels. In what follows let's consider $n = 2$, so that we get binary disorder $h_i = [-D, D]$

3. We add auxiliary spins to replace $\sum_i h_i S_i^Z \to D \sum_i \tau_i^Z S_i^Z$. Here $\tau_i^Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$. Note that τ_i^Z and S_i^Z live in different Hilbert spaces.

4. The structure of our wavefunction becomes $|\Psi\rangle = |\phi\rangle \otimes \bigotimes_i^L [\sqrt{p_\uparrow}|\uparrow\rangle + \sqrt{p_\downarrow}|\downarrow\rangle]$ where $|\phi\rangle$ is the physical part of the wavefunction, and the rest corresponds to the auxiliary spins with coefficient satisfying $p_\uparrow + p_\downarrow = 1$. By this construction we can see that the average of an observable $\hat{O}$ is $\langle\Psi|\hat{O}|\Psi\rangle = \langle\phi|\hat{O}|\phi\rangle$ times some unimportant phase, because the p coefficients will all add up to $|1|$.

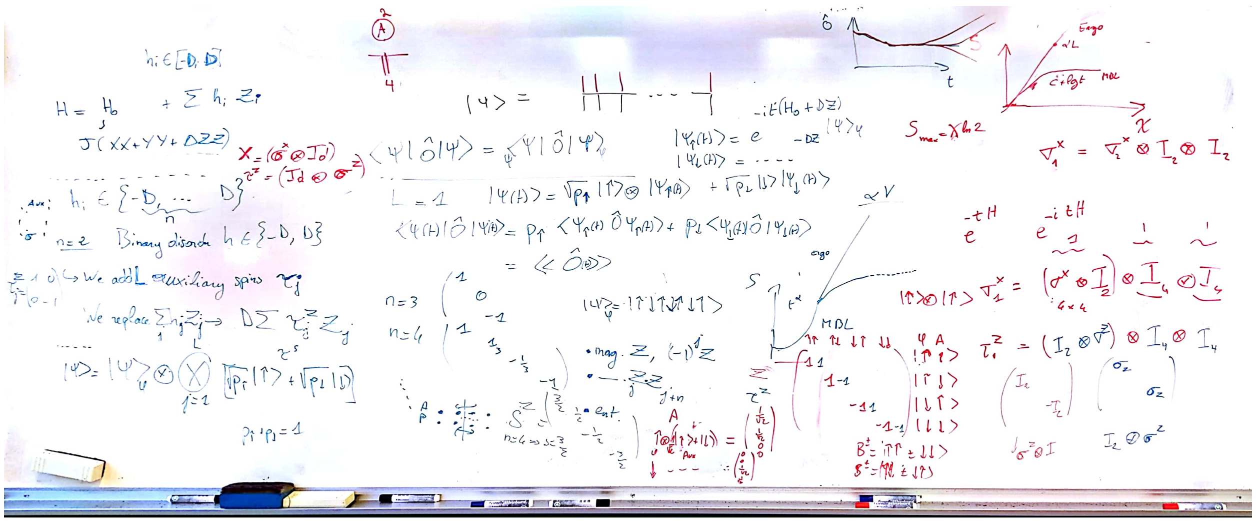
5. Consider now a time evolution of $|\Psi\rangle \to |\Psi(t)\rangle$. Because τ_i^Z is diagonal we can see that the time evolution is applied to up and down separately, so that

$|\Psi_\uparrow(t)\rangle = e^{-it(H_0 + DS^Z)}|\phi\rangle$

and

$|\Psi_\downarrow(t)\rangle = e^{-it(H_0 - DS^Z)}|\phi\rangle$.

6. Let's take an example with $L = 1$ particles. Then



$H = H_0 + \sum h_i Z_i$
$|\psi\rangle =$
Binary disorder $h \in \{-D, D\}$
We add L auxiliary spins τ_j
$\langle\psi|\hat{O}|\psi\rangle$
$L = 1$
$= \langle\langle \hat{O}(t) \rangle\rangle$
$S_{max} = X \ln 2$
$\sigma^x_1 = \sigma^x_2 \otimes I_2 \otimes I_2$
e^{-tH}
e^{-itH}
MBL
ergo

Figure 7: Fig

$$
|\Psi(t)\rangle = \sqrt{p_\uparrow}|\uparrow\rangle \otimes |\Psi_\uparrow(t)\rangle + \sqrt{p_\downarrow}|\downarrow\rangle \otimes |\Psi_\downarrow(t)\rangle
$$

An observable average is obtained simply by

$\langle\Psi(t)|\hat{O}|\Psi(t)\rangle = p_\uparrow\langle\Psi_\uparrow|\hat{O}|\Psi_\uparrow\rangle + p_\downarrow\langle\Psi_\downarrow|\hat{O}|\Psi_\downarrow\rangle = \langle\langle\hat{O}(t)\rangle\rangle$.

Now lets discuss some details of implementation. Consider $L = 3$ spins, and recall that a spin at position 1 in a chain of $L = 3$ spins is written as

$\sigma^x_{L,i=1} = \sigma^x \otimes I_2 \otimes I_2$,

where σ^x is simply a Pauli matrix for spin-$\frac{1}{2}$ and I_2 is a (2×2) identity matrix. If we introduce an auxilliary spin we have to remember that the local dimension grows. Let d be the physical dimension, and n the dimension of the auxiliary spin, then the local dimension is $d \times n$. Then, if $d = 2$ as before, introducing $n = 2$ auxiliary means that a spin at position 1 on a chain of $L = 3$ spins is written as:

$\sigma^x_{L,i=1} = (\sigma^x \otimes I_2) \otimes I_4 \otimes I_4$.

An auxiliary spin at the same position, on the other hand, can be written as

$\tau^z_{L,i=1} = (I_2 \otimes \sigma^z) \otimes I_4 \otimes I_4$.

To see that this is true we have to remember the base we're working in.

$$
\begin{pmatrix}
|\phi & Aux\rangle \\
\hline
|\uparrow & \uparrow\rangle \\
|\uparrow & \downarrow\rangle \\
|\downarrow & \uparrow\rangle \\
|\downarrow & \downarrow\rangle
\end{pmatrix}
$$

which means that in this base

$$
S^Z = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & -1 & \\ & & & -1 \end{pmatrix} = \begin{pmatrix} I_2 & \\ & -I_2 \end{pmatrix} = \sigma^z \otimes I_2,
$$

whereas

$$\tau^Z = \begin{pmatrix} 1 & & & \\ & -1 & & \\ & & 1 & \\ & & & -1 \end{pmatrix} = \begin{pmatrix} \sigma^z & \\ & \sigma^z \end{pmatrix} = I_2 \otimes \sigma^z.$$

In MPS language this all means that the "physical" dimension of our MPS-tensors Γ will have dimension $d \times n$.

Finally we discussed the expected physics, those are the plots in the picture of the white board above. We want to study the MBL phase first with $n = 2, 3, 4$ and 5. We want to do this with real TEBD. If we are in the MBL-phase, and start with an initial configuration with high energy, the time evolution should stabilize at some highly excited, "middle of the spectrum" -type state. We expect that the entanglement entropy S_E increases logarithmically w.r.t. χ_{max} when in MBL, and linearly when in the Ergodic phase. As a function of time we expect something similar, i.e. saturation of S_E in MBL and volume-law increase in Ergodic phase.

- Discussed studyplan matters with Jens. Many fields have to be filled in by the main supervisor but perhaps we should wait. Also, when exactly is the next studyplan due? Ask Pär?

Week 2 On Vacation (Thailand!)

Week 3 On Vacation (Thailand!)

Week 4 On Vacation (Thailand!)

Current short-term goals

- [ ] Compute the variance using both the mom. gen. function method and the slow method of keeping track of larger environments.
- [ ] Read simulation parameters from file instead of hard coding like now.
- [ ] Get fDMRG to write down tensors to disk instead? Perhaps above some threshold size.
- [ ] Transition to $A -B $ tensors instead of $\Gamma - \Lambda$
- [ ] Measure finite entanglement scaling. Reproduce plots in

  > Pollmann, Frank, Subroto Mukerjee, Ari M. Turner, and Joel E. Moore. 2009. "Theory of Finite-Entanglement Scaling at One-Dimensional Quantum Critical Points." *Physical Review Letters* 102 (25): 1–6. doi:10.1103/PhysRevLett.102.255701.

- [ ] Try measuring other observables
- [ ] Learn about speed-ups using symmetry.
- [ ] Start working on the mobility edge project suggested by Kjäll.
- [ ] Start working on the TEBD MBL simulation suggested by Loic (ask Jens first).
- [x] Compare a couple of eigenvalue solvers. Specifically, try using Lanczos algorithm.
- [x] Set up a Github repo for fDMRG and merge with notebook.
- [x] Finish the iTEBD algorithm (June week 2)
- [x] Compare performance and precision using RedSVD, and check how it scales with larger χ (June week 2)
- [x] Schollwöck and Pollmann both insist that TEBD using Vidal canonical form $\Gamma\Lambda$...involves inverses that inevitably leads to numerical headaches. To avoid this, implement Hastings workaround.

- [x] Try another Hamiltonian
- [x] Finish replicating idmrg.py.
- [x] Implement Finite DMRG
- [x] Restructure the code for clarity, make it maintainable and expandable.