Research Journal: Magic

Siddhartha Harmalkar

June 19, 2020

Contents

12 January 2020 4
Motivation from Ads/CFT . . . 4

13 January 2020 5
Properties of generalized pauli matrices and phase-space operators . . . 5
Current goals . . . 5
Computing relative entropy . . . 5

14 January 2020 6
Meeting with Chris . . . 6

17 January 2020 7
Density matrices and the bloch sphere . . . 7
Meeting with Chris . . . 7

20 January 2020 8
Generalized pauli matrices $T_{(a_1,a_2)}$ . . . 8
Phase space operators $A_{(a_1,a_2)}$ . . . 8
Generalization to n qubits . . . 8

21 January 2020 10
Mistakes made while writing Python implementation . . . 10
States with maximal sum negativity . . . 10
State to density matrix . . . 10
Meeting with Chris . . . 10
Questions about stabilizer states and density matrices . . . 10
An intuitive picture of density matrices . . . 11
Maximal sum negativity states . . . 11
Density matrices in higher dimensions . . . 11
MPS connection between pure density matrices and pure states . . . 12
More properties of phase space and pauli operators . . . 12

22 January 2020 13
Debugging phase space operator implementation . . . 13
Proving hermiticity of phase space operators . . . 13

23 January 2020 14
Generalizing phase space operator data structure to $n \neq 1$ . . . 14
Tensor product indexing . . . 14
Meeting with Chris . . . 14
Questions/Notes . . . 14
SVDs in phase space - what do they mean? . . . 15
Make sure to do SVDs locally . . . 15
Moving forwards . . . 15
Testing code: Inputs and invariances . . . 15
Constrained optimization . . . 15
Confusion about phase space operators in literature . . . 15

24 January 2020 16
Meeting with Chris . . . 16
Convex optimization . . . 16
What does gauge freedom imply for our optimization algorithm? . . . 16
Matrix factorization - well-known problem, not convex . . . 16

26 January 2020 17
Constrained optimization: Karush-Kuhn-Tucker conditions . . . 17

28 January 2020 18
Better resource for constrained optimization . . . 18
Meeting with Christopher . . . 18

29 January 2020 19
Mtg. Christopher . . . 19
Use numpy lin alg solver . . . 19
Using delta functions to write a matrix for the variables . . . 19
Progress towards understanding constrained optimization . . . 19

3 February 2020 20
Learning about magic . . . 20
Dual vs. primal . . . 20

7 February 2020 21
Finally.. a lucid resource for constrained optimization . . . 21

4 March 2020 22
Generating random mixed states . . . 22
Issue with distance from identity . . . 22

9 March 2020 23
Correct distance from identity . . . 23

16 March 2020 24
Meeting with Chris . . . 24

18 March 2020 25
Meeting with Chris . . . 25

20 March 2020 26
Meeting with Christopher . . . 26
Notes . . . 26
Thoughts . . . 26

1 April 2020 27
At long last: Bounds for the maniac . . . 27

8 April 2020 28
Computing the mana directly . . . 28

9 April 2020 29
Transitioning from $2 \to n$ sites . . . 29

17 April 2020 30
Meeting with Chris . . . 30

21 April 2020 31
Generalizing code from $2 \to n$ sites . . . 31
Meeting with Chris: Derivative might not work . . . 31

24 April 2020 32
Meeting with Chris: Resolving MPO construction issues . . . 32

6 May 2020 33
Equations for talk . . . 33

4 June 2020 34
Can't perform contractions with matrix multiplication . . . 34

11 June 2020 35
Using np.tensordot . . . 35
Meeting with Chris . . . 35

14 June 2020 **36**
Fixing SVD procedure . . . 36
Meeting with Chris . . . 36
Misunderstanding of `np.tensordot` . . . 36
Bond dimension should not grow until the end! . . . 36
Forgot to take complex conjugate . . . 37
Next steps . . . 37

June 19 2020 **38**
Meeting with Chris . . . 38
Next steps . . . 38

12 January 2020

Motivation from Ads/CFT

Described in Cui *et al.* (2019).

13 January 2020

Properties of generalized pauli matrices and phase-space operators

You need to know some stuff not menitoned in the paper (TODO: add citation for 1307.7171), or maybe only briefly metioned:

- $\frac{a_1 a_2}{2}$ means $2^{-1} a_1 a_2$, where $2^{-1} = \frac{d+1}{2}$, or something like that.. it's the field inverse of $2 \in \mathbb{Z}_d$
- $\mathrm{Tr}[T_{\boldsymbol{a}} T_{\boldsymbol{b}}^\dagger] = d^n \delta_{\boldsymbol{ab}}$ - this is the property which leads to $A_{\boldsymbol{u}}$ being an orthonormal basis.
- $T^\dagger_{(a_1,a_2)} = T_{(-a_1,-a_2)}$ - this one does clearly follow from the definition in the paper. *This is actually a really weird thing to write down.. -1 and -2 aren't technically in $\mathbb{Z}_3$ right? So why talk about them?*

Note that some of these properties depend on having odd prime dimension. Some of this is described in (TODO: Cite hudson's theorem for discrete qm systems)

Current goals

1. Code the shit up so that you can understand what the hell Christopher is talking about with his "local change of basis at each site"/"unitary at each site" shit. It seems like it rests on the fact that $A_{\tilde{\boldsymbol{u}}} = A_{(a_1,a_2)} \otimes \cdots \otimes A_{(u_1,u_2)}$.
2. Understand what the stabilizer states are for $S = 1$. Try to make a "projection operator" of density matrices onto stabilizer density matrices for $S = \frac{1}{2}$, then try to do the same for $S = 1$.
3. Think about the following: Say you could split up the sum over $\tilde{\boldsymbol{u}}$ in the computation of mana into a region of positive and negative components of the vector. Then how could you speed up the computation of the two terms left to calculate?

Computing relative entropy

If we can show via LU decomposition that all states with positive representation can be represented as MPOs then we can do something like DMRG to optimize distance between the stabilizer states represented as MPOs and a "magic" state. This rests on the fact that all positive coefficients $A^{\alpha\beta}$ can be written as $A^\alpha B^\beta$ for some positive A^α and B^β. Christopher thinks that this is probably true and can be shown via an LU decomposition (because it's related to gaussian elimination? I don't see how that's relevant but he mentioned it and I forgot to ask). So we could then compute the relative entropy using matrix product state methods.

14 January 2020

Meeting with Chris

- If the (pure?) stabilizer states satisfied $\mathrm{Tr}[S_i S_j] = C\delta_{ij}$, then we could easily find the coefficients of a general stabilizer state S via $p_i = \frac{1}{C}\mathrm{Tr}[S_i S]$. The stabilizer states for $S = \frac{1}{2}$ are the points at edges of the bloch sphere, which are (TODO: Understand this!) not just σ_x, σ_y, and σ_z but also $1 \pm \sigma_z$ etc. So even though $\mathrm{Tr}[\sigma_i \sigma_j] = C\delta_{ij}$, the other states aren't orthogonal. I need to understand more about this - what *is* a state of orthogonal states? Do those form a basis for this space? And is there any way I can extract coefficients even though I don't have an orthogonal basis? Can the pure states even be thought of as a "basis"? Chris said something along the lines of: All 2x2 density matrices can be expressed as $(1 + A_x\sigma_x + A_y\sigma_y + A_z\sigma_z)/2$ for (A_x, A_y, A_z) small enough since the eigenvalue associated is $\sqrt{A_x^2 + A_y^2 + A_z^2}$ and has to be kept small for normalization.

- I should read TODO, which is D. Gross' description of stabilizer states in odd prime dimension only and is therefore easier to get through (and more relevant for $S = 1 \to d = 3$) than TODO(hudsons thm).

- LU-decomposition didn't work - he found that there are matrices which factor into non-positive matrices by just looking at random ones in Julia. But I think we can just do $R^{\mu\nu}(\lambda_1, \lambda_2) = A^\mu_\alpha(\lambda_1) B^\nu_\alpha(\lambda_2)$ where λ_i are parameters such that the product of the matrices are positive, and do gradient descent in this space or something along those lines. Chris thinks that the λ_i's can just *be* the matrix elements with the constraint that they are positive - and that the alternative solution which makes the product positive (which is that matrix elements of A and B need to multiply to get a positive matrix element in R) is just a gauge transform of them all being positive so you probably don't need to worry about that possibillity.

17 January 2020

Density matrices and the bloch sphere

Apparently density matrices in the qutrit case have $\rho = \frac{1}{3}(1 - \sqrt{3}\boldsymbol{r} \cdot \boldsymbol{\lambda})$ with $|\boldsymbol{r}| \leq 1$. More information is here. (TODO cite that), and Chris sent you a reference for how the $S = \frac{1}{2}$ case turns out to be the points at the edges of the bloch sphere. (TODO loook at email "bravyi siddhartha")

Meeting with Chris

- The $S = \frac{1}{2}$ case is clear from thinking about stabilizer states in terms of clifford operators - the clifford operators are σ_i's and if you just hit the $|0\rangle$ state with one and rotate around with the others you'll get the other points.
- The gaussian-ness of the stabilizer states is related to the lemmas in Hudson's theorem paper which states that if ψ has support on two points then it must over all points and its modulus is constant - notice that the gaussian has constant modulus. But it's not clear where the phase structure comes from.
- There are some issues with the discrete hudson paper:
  - All of the lemmas are stated for $n = 1$ only, but stabilizer states look different in $n > 1$ and don't decompose into $n = 1$ states. Also, it seems like the "support" lemma (or the constant modulus lemma? not sure what he was referring to) might not hold
  - It doesn't seem to prove that the stabilizer states are gaussian, which is what it says it's going to prove. But it does seem to prove that the states reached by cliffords from the computational basis do have positive Wigner representation, which is good, maybe enough. It's not clear what the phase structure means/ where it comes from though.
  - Also it seems like there's a more basic definition of stabilizer states which actually has a notion of stabilizability in it. It's not clear how this is related but *this* is the framework in which he describes the connection to gaussian states so it must be involved in the overall picture somehow
- You can compute the mana efficiently if you can split up the summation - through gauge transformations or other means - because the remaining computation can be written as an overlap between two states since a sum over coefficients is just an overlap between the state in question and the state $\otimes^d |1,1,1\rangle$. He thinks about this as a local overlap with the state $|1,1,1\rangle$ at each point. I think that I should write code to take a state, compute the density matrix, represent in the $A_{\tilde{\boldsymbol{u}}}$ basis, and use an SVD to represent it as an MPS. I can then run the Norell and other maximally magic states through the code and see how it compares to doing the same to some stabilizer states. I can then look at the resulting matrices and try to find some patterns in terms of positive matrix elements of the matrix products.
- The optimization algorithm is a 1-site DMRG-like algorithm which would work as follows: Sweep through the MPS, and at each site try to find the tensor which best minimizes the distance to the stabilizer states while holding the rest of the matrices fixed.

20 January 2020

Generalized pauli matrices $T_{(a_1,a_2)}$

The qutrit pauli operators acting on a single site spanned by computational basis elements $\{|0\rangle, |1\rangle, |2\rangle\}$ are constructed from the shift and boost operators X and Z:

$$X|j\rangle = |j+1 \bmod 3\rangle \Rightarrow \langle i|X|j\rangle = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \tag{1}$$

$$Z|j\rangle = \omega^j |j\rangle \Rightarrow \langle i|Z|j\rangle = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \omega & 0 \\ 0 & 0 & \omega^2 \end{pmatrix}, \tag{2}$$

where $\omega = e^{2\pi i/3}$, as in the 3rd root of unity[1]. The generalized pauli matrices are then

$$T_{(a_1,a_2)} = \omega^{-2^{-1}a_1a_2} X^{a_1} Z^{a_2} = \omega^{-2a_1a_2} X^{a_1} Z^{a_2} \tag{3}$$

where we've used the fact that $2^{-1} = \frac{d+1}{2} = \frac{4}{2} = 2$ satisfies $2^{-1} \times 2 = 2 \times 2 = 4 = 1 \bmod 3$. The only T with non-zero trace is $T_{(0,0)} = 1$ which has $\operatorname{Tr} T_{(0,0)} = d$. All the other generalized paulis have zero trace because X is an elementary row operations which permutes all the rows from their original location, and no power of it (at least smaller than 2?) allows for any elements of it or it multiplied by a diagonal matrix like Z to lie on the diagonal - for example

$$T_{(1,2)} = \omega^{-4} XZ^2 = \omega^{-4} \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \omega^2 & 0 \\ 0 & 0 & \omega^4 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ \omega^{-4} & 0 & 0 \\ 0 & \omega^{-2} & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ \omega^2 & 0 & 0 \\ 0 & \omega & 0 \end{pmatrix}, \tag{4}$$

where we've used the fact tha $\omega^3 = 1 \Rightarrow \omega^{-3} = 1^{-1} = 1$. So $\omega{-}4 = \omega^{-1} = \omega^{-1} \times \omega^3 = \omega^2$ etc.

These operators are unitary because $T^\dagger_{(a,b)} = (\omega{-}2ab)^\dagger Z^\dagger X^\dagger = \omega^{2ab} Z^{-1} X^{-1} = T^{-1}_{(a,b)}$. TODO: Verify the last step - I think this involves using the fact that X is unitary which isn't obvious to me and that $\omega^\dagger = \omega^{-1}$ which also.. isn't clear I think.

Phase space operators $A_{(a_1,a_2)}$

The phase space operators are defined in terms of the generalized paulis like so:

$$A_{(a_1,a_2)} = \begin{cases} \frac{1}{d}\sum_{u_1,u_2} T_{(u_1,u_2)} & a_1 = a_2 = 0 \\ T_{(a_1,a_2)} A_{(0,0)} T^\dagger_{(a_1,a_2)} & \text{else} \end{cases} = T_{(a_1,a_2)} \left(\frac{1}{d} \sum_{u_1,u_2} T_{(u_1,u_2)} \right) T^\dagger_{(a_1,a_2)} \left(\text{since } T_{(0,0)} = 1\right) \tag{5}$$

Because the generalized paulis are unitary and $\operatorname{Tr}[A_{(0,0)}] = \frac{1}{d}(d + 0 + \cdots + 0) = 1$, all of the phase space operators have unit trace.

TODO: Show they are orthonormal w.r.t frobenius norm

Generalization to n qubits

The n-qubit generalized paulis are (here I'm using a different labeling convention than in [TODO cite hudsons] which I find more convenient)

$$T_{(a_1,a_1')\oplus\cdots\oplus(a_n,a_n')} = \bigotimes_{i=1}^n T_{(a_i,a_i')} \tag{6}$$

and the n-qubit phase space operators are similarly

$$A_{(a_1,a_1')\oplus\cdots\oplus(a_n,a_n')} = \bigotimes_{i=1}^n A_{(a_i,a_i')}. \tag{7}$$

[1] As in, the regular division by 3. They can't possibly mean the field inverse 3^{-1} because $3 \equiv 0 \pmod 3$ which has no inverse.. I think

This agrees with the definition given in (TODO cite hudsons) because

$$A_{(0,0)\oplus\cdots\oplus(0,0)} = \bigotimes_{i=1}^{n} A_{(0,0)} = \frac{1}{d^n} \bigotimes_{i=1}^{n} \sum_{u_1,u_2} T_{(u_1,u_2)} \tag{8}$$

and

$$T_{(a_1,a_1')\oplus\cdots\oplus(a_n,a_n')} A_{(0,0)\oplus\cdots\oplus(0,0)} T^{\dagger}_{(a_1,a_1')\oplus\cdots\oplus(a_n,a_n')} \tag{9}$$

21 January 2020

Mistakes made while writing Python implementation

- Used python's `A*B` operator, which multiplies arrays element-wise, instead of numpy's `np.dot(A,B)` operation to multiply matrices correctly
- Did the previous mistake again immediately afterwards!

States with maximal sum negativity

(This is for $N = 1$, but generalizes as you would expect to larger N:) Given a state ρ, let

$$N(\rho) \equiv \{(a,b) \in \mathbb{Z}_d \times \mathbb{Z}_d | \operatorname{Tr}[\rho A_{(a,b)}] < 0\}. \tag{10}$$

The the sum negativity of ρ is

$$\operatorname{sn}(\rho) \equiv - \sum_{(a,b)\in N(\rho)} \operatorname{Tr}[\rho A_{(a,b)}] \tag{11}$$

Because the trace is linear and ρ is not a function of a or b, the sum negativity satisfies

$$\operatorname{sn}(\rho) = -\operatorname{Tr}\left[\sum_{(a,b)\in N(\rho)} \rho A_{(a,b)}\right] = -\operatorname{Tr}\left[\rho \sum_{(a,b)\in N(\rho)} A_{(a,b)}\right] \tag{12}$$

TODO: Understand the rest of the argument on page 14 of discrete Hudson's.
The strange state $|S\rangle$ has density matrix elements

$$\frac{1}{2}\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \end{pmatrix} \tag{13}$$

State to density matrix

The density matrix of a pure state $|\psi\rangle$ has matrix elements $\rho_{ij} = \langle i|\psi\rangle \langle\psi|j\rangle = \langle i|\psi\rangle \langle j|\psi\rangle^*$. So given a vector $V_{i1} = \langle i|\psi\rangle$, we can construct the density matrix via the outer product $P = VV^\dagger$, because:

$$P_{ij} = \left(VV^\dagger\right)_{ij} = \sum_k V_{ik} V^\dagger_{kj} = V_{i1} V^*_{j1} = \langle i|\psi\rangle \langle j|\psi\rangle^* = \rho_{ij} \tag{14}$$

Numpy doesn't seem to have a way of letting you multiply two vectors as if they were matrices - it always just does a scalar dot product no matter which order you give them in to `np.dot` or `np.matmult`, which apparently is the preferred way of multiplying matrices. So instead I used `np.outer` to do this.

Meeting with Chris

Questions about stabilizer states and density matrices

1. What is the difference between ITensors for C++ vs. for Julia? Was it originally written in C++ and wrapped to be used in Julia? There's a python module which somehow uses the C++ iTensor... how does this all work? Is the C++ reference a good place for up-to-date answers about how objects work in Julia as well? If not, is there a good reference for the Julia version Why do you use Julia instead of C++ or Python? (TODO Forgot to ask)
2. Its not clear to me why states with maximal sum negativity have to be eigenstates of $\sum_{(a,b)\in S} A_{(a,b)}$ for some S, nor how that would make it easier to perform an exhaustive search - which seems doable to begin with without any kind of restriction. For $d = 3, N = 1$, a density matrix has 9 complex parameters, and one real restriction given by $\operatorname{Tr}[\rho] = 1$, and some more restrictions due to positivity and hermiticity. So the space isn't that huge and one can just look in it. (see below)
3. Why are there only $\binom{9}{1}$ strange states? Aren't there infinite density matrices satisying the property that one of the wigner representation elements is $-1/3$? (TODO Forgot to ask)

4. Why are the $A_{(a,b)}$ matrices orthonormal? (It's because of the commutation between the T operators resulting in an ω^{ua-vb} term, which Gross sometimes refers to as a "symplectic form" due to some extra structure it has. That combined with the fact that $\sum_j \omega^j = 0$ gives orthonormality. Also note that the A matrices are hermitian - TODO check that this is true analytically and in your code)

5. How large is the space of $N \times N$ density matrices? It should be smaller than $\mathbb{C}^{N^2}$ due to positivity, hermiticity, and normalization. But we have N^2 phase space operators, so they span a larger space, assuming that we're using complex coefficients... this might not be a problem for finding coefficients but it might lead to some issues down the road when we're doing things like optimization in this larger space.

    - Look at $N = 2$: If you think of it as a vector space over $\mathbb{C}$ then the basis elements are $1, \sigma_x, \sigma_y, \sigma_z$ which are hermitian. If you think about it as a vector space over the reals, then the basis elements also contain $i1, i\sigma_x, i\sigma_y, i\sigma_z$ which are anti-hermitian. So you can impose the requirement that ρ is hermitian by making the coefficients on the latter 4 vanish if you think of it as being over the reals, or by making all the coefficients real if you think of it as being over the complex numbers. The fact that $\mathrm{Tr}\,\rho = 1$ then imposes the condition that the coefficient on the identity matrix be $\frac{1}{2}$, since $\mathrm{Tr}[\sigma_i] = 0$ (TODO not sure why the coefficient wouldn't be equal to 1 then..). And the fact that the eigenvalues are positive imposes that the coefficients need to lie within the bloch sphere (TODO understand this))
    - He proposes doing an optimization in the larger space to just get an upper bound on the relative entropy/mana etc. and then check the mana numerically to verify that it's low. That might be good enough.
    - You might be able to find a connection between the space of positive-eigenvalue operators as parametrized by the Gell-Mann matrices, but it probably won't generalize to higher dimensions due to the reasons described in the section below

An intuitive picture of density matrices

The coefficients in a density matrix tell you the expectation values of the state/ensemble of the operators related to the coefficients. For example, the state $\frac{1}{2}(1 + \sigma_z)$ has an expectation value of $\frac{1}{2}$ in the identity and spin-z operators.. not clear what to make of the identity part (TODO), but another example is the state $\rho = \sigma_z$ which is spin-up I think.. TODO.

Maximal sum negativity states

The fact that the states with maximal sum negativity are eigenstates of the operators described above might follow from the following argument:

1. First of all, ρ probably *has* to represent a pure state in order to have maximal sum negativity because the elements in the wigner representation need to add up to 1, so if all the elements are positive then they all have absolute value less than 1 but if some elements are negative then you can have some with absolute value greater than 1 so if you look at the sum of the wigner representation elements squared, it's going to be larger for states with more negative elements and this is just the second Renni entropy which is a measure of how pure a state is

2. Given that ρ represents a pure state, what you're trying to do is maximize the Rayleigh quotient (see here) by maximizing the sum negativity, which is the same as maximizing the "operator norm" (as opposed to the frobenius norm) - though I don't see how that's relevant - and in a discrete space is done by using the eigenstate with maximum eigenvalue. You can also see this by just eigen-decomposing the phase space operator and seeing that you must need the maximal eigenstate.

This is important because the set of sets of strings is discrete, and the set of eigenstates of sums of phase space operators on the strings in those sets is also discrete, so you can actually do an exhaustive search! Whereas if you did a search on density matrix elements you would have to introduce a grid spacing and you'd never get an exact proof.

Note that if you were to actually go about doing this search, you should be able to use symmetries of the phase space operators to reduce the space in which you need to search.

Density matrices in higher dimensions

Density matrices in higher dimensions can't always be written as a tensor product of density matrices over the N sites. For example $\frac{1}{2}(1 + ZZ)$ can't.. (TODO why not?)

MPS connection between pure density matrices and pure states

If you write a state as an MPS and construct its density matrix (just draw it out and you'll see this!), you'll see that if one of the bond dimensions is not equal to 1 then the bond dimension of the density matrix can't be equal to 1 either which means that if a state can't be factorized into a tensor product then its density matrix can't be factored either.

TODO: Does this argument work going the other way?

More properties of phase space and pauli operators

The "boost" and "shift" operators are unitary: $X^\dagger X = Z^\dagger Z = 1$. This makes the pauli operators unitary as well, since $\omega^x 1$ is also a unitary operator for any $x \in \mathbb{R}$ and a product of unitary operators is also a unitary operator and each of the pauli operators is a product of an $\omega^x 1$ operator and some number of X and Z operators.

The $A_{(0,0)}$ operator is hermitian because.. refer to this maybe... I can't figure it out.

22 January 2020

Debugging phase space operator implementation

As always, trying to do the simplest possible thing led me to the answer. I decided to just compute the matrix elements of the phase space operators and compare directly with what I was seeing in the code to find my error, since they weren't coming out to be hermitian as they should (which I also couldn't verify analytically, and had been spending most of my time thinking about why the proof wasn't working out). As soon as I started to compute matrix elements of the generalized pauli operators, I realized that I had implemented them as ωXZ instead of ωZX. That was the fix I needed, and it also made me realize what was going wrong in my proof - I needed to use commutation relations of X and Z!

Proving hermiticity of phase space operators

Using the fact that $T^\dagger_{(a,b)} = T_{(-a,-b)}$, how can I show that $A_{(0,0)} = \frac{1}{d}\sum_{(u,v)} T_{(u,v)}$ is hermitian? It seems like the hermitian conjugate should just rearrange terms in the sum, but I don't see how that works out unless we can equate $-a$ with $d-a$ since its an element of $\mathbb{Z}_d$. Technically the two are equal modulo d, but.. I don't see how that makes $T_{(-a,-b)}$ equal to $T_{(d-a,d-b)}$. Unless $X^d = Z^d = 1$ and a similar property for ω.. maybe that's it.

23 January 2020

Generalizing phase space operator data structure to $n \neq 1$

I currently have the phase space operators for $N = 1$, $A_{(a,b)}$, stored in a $d \times d$ array of numpy matrices of size $d \times d$. The phase space operators for general n can be constructed out of these via (8) and (9). The way I'm storing them now doesn't generalize well to larger n, because I'd have to create a $\underbrace{(d \times d) \times (d \times d) \cdots \times (d \times d)}_{n}$ size array of numpy matrices of size $d^n \times d^n$, which I'm not sure how to even construct in python - since it would be something along the lines of `np.zeros((d,d,...,d,d,d**n,d**n))`.

Instead, I could index the $A_{(a_1,a_1')\oplus\cdots\oplus(a_n,a_n')}$ object using a tensor-product indexing scheme. Unpacking the indexing might be a mess though.. I should get a sense of what I'm going to use them for first in order to come up with the simplest indexing scheme:

My goal is to take a state, decompose it into its coefficients in the $A_{\vec{u}}$ operator basis, and use SVD to represent it as a matrix product state. This means that the data structure I use to represent these coefficients must be local so that I can take SVDs at the right point easily. So it seems better to store the phase space operators as numpy arrays of dimension $d^n \times d^n$ in a larger numpy array of size $n \times (d \times d)$. In other words, I should store them in a numpy array with shape (n, d, d, d^n, d^n), but a numpy array `A=np.zeros((n,d,d,d**n,d**n))` only has $nd^2(d^n)^2$ elements, which falls far short of the $(d^2)^n(d^n)^2 = d^{4n}$ I need to index a matrix element of a phase space operator.

Tensor product indexing

Consider two matrices A and B of size $a \times b$ and $c \times d$. Their tensor product $A \otimes B$ has dimensions $(ac) \times (bd)$, which can be seen from looking at the matrix elements at the four corners of the tensor product:

$$A \otimes B = \begin{pmatrix} A_{11}B & \cdots & A_{1b}B \\ \vdots & \ddots & \vdots \\ A_{a1}B & \cdots & A_{ab}B \end{pmatrix} = \begin{pmatrix} A_{11}B_{11} & \cdots & A_{1b}B_{1d} \\ \vdots & \ddots & \vdots \\ A_{a1}B_{c1} & \cdots & A_{ab}B_{cd} \end{pmatrix} \tag{15}$$

The structure of the tensor product can also be written like so:

$$(A \times B)_{ij} = A_{f_1(i),f_2(j)}B_{g_1(i),g_2(j)} \tag{16}$$

or equivalently:

$$(A \times B)_{f(x,y),g(w,z)} = A_{xy}B_{wz} \tag{17}$$

TODO: Figure out all four relevant maps - unpacking and packing functions for indexing schemes starting at 0 and starting at 1.

Meeting with Chris

Questions/Notes

- Apparently the anisotropic Heisenberg model is also integrable? So it shouldn't thermalize either right? (TODO: Didn't ask)
- I can only test my MPS code for $n > 1$ because I don't really have an MPS for $n = 1$. But I don't know what maximally magic states look like for $n \neq 1$. What states should I use to learn about how the MPS representation of states (bond dimension etc.) varies with the mana? (See if you can find $n = 2$ states which you can compute the exact relative entropy of. For example, in the qubit $d = 2$ case you can draw the stabilizer convex hull and make a state which has no expectation value in Z so it lies in the XY plane and just draw it and look at it. For $n = 3$ it's unclear how to visualize it)
- Any tips on indexing the phase space operators? Is there an easier way to index the phase space operators using Julia? (Use numpy's reshape method and then transpose / permutation methods to move indices around to get local structure)
- He uses Julia because he likes having a notebook interface. One point he made which I think is valuable is that with a notebook, you can make a plot by running a snippet of code and then change the code slightly in another part of the notebook and re-run it and have both plots there right after their respective codes, showing what each did. This helps a lot during the phase of reserach when you're playing around with things to try to find something that works, and it's much more tedious and practically impossible to do the way I currently plot things - by writing to files and using gnuplot to produce an image.

SVDs in phase space - what do they mean?

When we do an SVD on a density matrix in the physical basis, we get back $\rho_{ij} = \sum_\alpha U_{i\alpha} s_\alpha U^\dagger_{\alpha j}$, where U contains the eigenvectors of ρ (TODO: verify this, using the fact that $\rho^\dagger = \rho$). This is a statistical mixture of states: $\rho = \sum_\alpha s_\alpha |u_\alpha\rangle \langle u_\alpha|$ describes a classical ensemble of states $|u_\alpha\rangle$ with probabilities s_α. But what does an SVD in phase space mean? Chris thinks it might show some kind of entanglement structure between "position" and "momentum".. I have to think about this some more.

Make sure to do SVDs locally

When you do your SVDs, cut along each dimension of the physical space, not the d^{2n} phase space. Group position and momenta together into an index before taking the SVD between sites.

Moving forwards

There's two different paths that we are currently taking, which are more distinct than they might seem:

1. Developing the general framework for matrix product operators (in a slightly different order than what you are doing: $|S\rangle \to \rho \to \sum_{i_n, j_n} \prod_k M^{(k) i_k j_k} (\otimes_\ell |i_\ell\rangle \langle j_\ell|) \to \sum_{a_n, b_n} \prod_k W^{(k) a_k b_k} (\otimes_\ell A_{(a_\ell, b_\ell)})$, where the second step requires numpy's reshape and transposition/permutation operators, and the last step can be implemented via a *local* basis change/unitary operation. From there, we can implement things like expectation values and mana etc. in the phase space basis.

2. Computing an upper bound of the relative entropy of a state via optimization for $n = 2$: $|S\rangle \to \rho \to (C_\alpha^{(1) a_1, b_1}, C_\alpha^{(2) a_2, b_2}$ with $C_\alpha^{(i) a_i, b_i} \geq 0$ optimized so that $\tilde{\rho}(C) = C_\alpha^{(1) a_1, b_1} C_\alpha^{(2) a_2, b_2} A_{(a_1, b_1)} \otimes A_{(a_2, b_2)}$ has smallest possible frobenius norm distance ρ - we can later change the cost function to be something else).

His recommendation is to go down the second path first, which does not involve creating a framework for MPOs with $n > 2$, and will give me an idea of what my framework is building towards anyway. Once I do that, I should use numpy's reshape and permutation tools to build the framework with the goal of "marrying" the two paths together in mind. The final result will be the general computation of mana and bounds on the relative entropy of magic for general n.

Testing code: Inputs and invariances

I was trying to think of possible test inputs with known outputs to give to my code, but Chris said he tends to think in terms of invariances which the code should have instead of specific inputs. For example, if the density matrix input into my MPO code is normalized, the wigner coefficients that are outputted from the code should add to 1. Also, the 2nd renni entropy (sum of matrix elements of the density matrix squared) should be invariant - not sure what he meant by this.

Constrained optimization

Check out this book, specifically chapter 5, for the methods you learned in CMSC644. Also see Hector Bravo's lecture notes here.

Confusion about phase space operators in literature

It seems like one-parameter phase space operators are frequently used instead of the two-parameter ones I've been using. I'm trying to find a proof in the literature of their hermiticity but I can't seem to find the ones introduced in the 1307.7171 anywhere else.

24 January 2020

Meeting with Chris

Notes: Our goal is to minimize the norm of $AB - \rho$ by sweeping through coefficients and performing the optimization on each one with the rest held fixed. What I should do now is figure out constrained optimization, write out exactly what the procedure is, and send it to him as a note. And then code it up. Also - you should send a note with exact computation of qubit and qutrit relative entropy of magic if you can figure that out. And a note on what SVDs in phase space mean.

Convex optimization

The matrix elements of ρ can be positive or negative, so the norm of $AB - \rho$ is not a convex function. A convex function is one which satisfies

$$\alpha f(x_1) + (1-\alpha) f(x_2) \geq f(\alpha x_1 + (1-\alpha) x_2) \tag{18}$$

for all x_1 and x_2. A function which never saturates that bound is called strictly convex. For example, x^2 is strictly convex and x is convex.

Optimization of a convex function on a convex set (I forgot to ask what a convex set is, TODO figure this out) is easy because there is gauranteed to be only one local minima/maxima, which is the global minima/maxima. TODO This doesn't make sense to me because we should have to consider the endpoints/boundaries of the set whether it's "convex" or not.. I need to think about this.

Apparently, a function in multiple variables is convex if its Hessian matrix is positive definite.

What does gauge freedom imply for our optimization algorithm?

Any "optimal" set of matrices $\{M^{(1)}, M^{(2)}\}$ found by our algorithm will be equivalent to $\{M^{(1)}U, U^{-1}M^{(2)}\}$. What

Matrix factorization - well-known problem, not convex

26 January 2020

Constrained optimization: Karush-Kuhn-Tucker conditions

These notes this online note, which defines the KKT conditions, contains examples and a simpler way to deal with non-negativity conditions, is exactly what I'm looking for. TODO: Understand the "geometric" and "economic" interpretations of equality and inequality constraint solutions by reading this notes' previous sections and the paragraph in the section linked above.

The main theorem is the following: To find the max value of $f(x)$ under the restrictions $g_i(x) = b_i$ and $h_i(x) \leq d_i$, solve the following system of equations, where ∇ seems to mean a derivative with respect to all components of x only, not λ_i or μ_i:

$$\begin{cases} \nabla f(x) - \sum_i \lambda_i \nabla g_i(x) - \sum_j \mu_j \nabla h_j(x) = 0 \\ g_i(x) = b_i \\ h_j(x) \leq d_j \\ \mu_j(d_j - h_j(x)) = 0 \\ \mu_j \geq 0 \end{cases} \tag{19}$$

Solving these equations gives a set of $\{x^*, \lambda^*, \mu^*\}$ candidates, of which the optimal x^* is the one with the largest value of $f(x^*)$. If $f(x)$ is concave (or if you're minimizing it and $f(x)$ is convex), $g(x)$ is convex, and $h_j(x)$ are linear - *which seems to be true in our case!* then any x^* is an optimum - which must mean that $f(x^*)$ is the same for all x^*.

28 January 2020

Better resource for constrained optimization

This resources seems way better than the previous one - the KKT equations are written more succintly for a quadratic, which is what we care about, and in general its a much more thorough discussion.

Meeting with Christopher

1. If you move in only one direction at a time, you might end up no longer in a local minimum of the variable you last moved in, once you move in the direction of the next variable. So instead we should move a whole tensor at once by recognizing it as a quadratic optimization problem with a matrix representing the couplings, and moving in the directions of eigenvectors of that matrix! Since those variables don't couple with each other, this is the right thing to do. The issue is that diagonalization will take a while.

2. What I need to do now is figure out how to write our cost function in the form of a matrix I need to diagonalize, and the constraints in a way that can be implemented into the algorithm, either by a series of steps or a lagrangian style method.

29 January 2020

Mtg. Christopher

Use numpy lin alg solver

instead of choosing between methods of inverting matrices yourself, just use numpy's lin alg solver to solve the linear equations associated with finding the stationary point of the quadratic!

Using delta functions to write a matrix for the variables

Progress towards understanding constrained optimization

3 February 2020

Learning about magic

I'm reading through **?**, which states in the appendix that the relative entropy is an "efficiently computed" convex maximization problem, but cites a paper to back up this claim which is confusing to read. So instead I'm just reading through **?** itself to learn more background about magic and the resource theory of stabilizer computation. I have the following unanswered questions:

- In what sense of convexity are stabilizer states the convex mixture of eigenstates of the pauli operators? Is it obvious that states which are not eigenstates of equal-modulus-weight mixtures of eigenstates cannot be formed as convex mixtures of eigenstates? How general is that statement? And where does the gaussanity of the phases of the coefficients come from? Is that also related to convexity?

Anyway, this doesn't seem to be productive.. I should figure out the convex optimization shit.

Dual vs. primal

On primal vs. dual problems: It's not clear why the dual problem is easier - it's also a constrained optimization problem because of the constraints $\lambda_i \geq 0$. However, according to the "Convex Optimization" book, the dual problem is concave even when the primal problem isn't convex, for reasons I don't understand.

7 February 2020

Finally.. a lucid resource for constrained optimization

Found a good set of notes here - chapter 2 introduces the KKT conditions and chapter 3 discusses how to solve them for quadratics.

4 March 2020

Generating random mixed states

I'm generating a random mixed state of N qu*d*its via the Scott algorithm: Generate M pure states and M numbers uniformly distributed on $[0, 1]$. Then mix the states with those probabilities.

Issue with distance from identity

One would expect the resulting density matrix to approach the identity as $M \to \text{inf}$, in fact even for moderately large M. But when I plot the average frobenius norm of the difference between the identity and a density matrix generated in this fashion with $N = 2, d = 3$, I find that it doesn't decrease much for large M. It starts at 2.82 for $M = 1$ and converges pretty quickly to $\frac{8}{3}$ after about $M = 10^5$ - it stays converged even till 10^7 (the largest I could do on my laptop). I also find that the error in this quantity is much smaller for $M = 1$ (it's machine precision there) than it is for $M = 10$ (it's on the order of 10^{-6} there). After $M = 10$ it does start to get lower, but what gives? That makes no sense.. for $M = 1$ I should definitely get a huge error.. I'm taking 20 samples for all of these measurements, for reference.

9 March 2020

Correct distance from identity

The identity isn't a density matrix! But the identity divided by the dimension of the hilbert space is. Computing the distance to that gave me something which goes to 0.

16 March 2020

Meeting with Chris

- Your hypothesis is that doing one sweep is enough for 2 sites, because all the degrees of freedom can be moved into the first site via a gauge transformation. Chris thinks that a test of this would be to feed your code random product states of the form $\rho = \rho_1 \otimes \rho_2$, and classical/quantum mechanical mixtures of these states - for the former you do $\rho = \frac{1}{2}(\rho_1 + \rho_2)$, and for the latter you do $\rho = \rho_{(|1\rangle+|2\rangle/\sqrt{2})}$ I think... actually I'm not sure what he meant by this. Maybe he meant to do either $\rho_1 \otimes \rho_2$ or the quantum superposition state.
- An upper bound for the mana is easily found - use Jenson's inequality to get $\frac{n\log(3)}{2} - S_2(\rho)$ for qutrits, where S_2 is the second renni entropy, or something like that.. and S_2 is always between 0 and $n\log(3)$. And the bound on pure states is when $S_2 = 0$ because that is always true for pure states. Note that this bound is not always saturated - for 1 qutrit you can see that the strange and norrell states are explicitly a little bit below it.

18 March 2020

Meeting with Chris

- The plot of (distance-true distance) w.r.t half-sweeps for 2-site optimization exhibits some weird features which we should understand:
  - Goes up at certain points - the problem seems to not be convex (the function is convex, but the space isn't!)
  - It plateaus for a while before dropping considerably - is there some stochasticity?
- The space isn't convex - if you add two states the resulting MPO will have the sum of the bond dimensions of the states (TODO understand this). If you had allowed for negative and/or complex values instead of restricting to positive values, then the space wouldn't be as bad because it would be gauge-equivalent to the space of MPOs of $d \times d$ operators, which has an SVD of max rank d (TODO understand this)
- What you should do next:
  1. Start using version control - commit this plot to git so you have it
  2. Give it your own seed so that you can reproduce the results - you've lost this result by not doing that
  3. Plot the second Renni entanglement entropy between sites as a function of half sweeps by doing an SVD on the matrix with physical indices and plotting the sum of the squares of singular values
  4. Write up notes explaining what you did and send them to him and Brian
  5. Play around with 2-sites to understand what's going on before continuing to N sites.
     - Try different optimizers
     - Use different types of states (product states, quantum mixtures)
     - Try different kinds of initial conditions
     - I think I should also try to do global optimization in the larger space and see if that works better, but I'm not sure how to formulate that.. which is related to the nonconvexity and other weird properties of the space noted above
     - Try running the same code for longer and see if it ever does better - note that it goes back up?

20 March 2020

Meeting with Christopher

Notes

1. On the convexity of the space of positive-valued MPOs with fixed bond dimension: If you relax the positivity constraint, the space is *not* convex, in the sense that adding MPOs will, in general (see below), increase the bond dimension and take us out of the space of fixed bond dimension. It seems unlikely that adding in the positivity constraint would make the space convex - it probably just makes it more complicated.

   - Adding MPOs does not always increase the bond dimension - consider MPOs of $\sum_{i=1}^{D} r_i \left|\psi_i\right\rangle \left\langle\psi_i\right|$ and $\sum_{i=1}^{D} s_i \left|\phi_i\right\rangle \left\langle\phi_i\right|$. The sum of these MPOs represents a state $\sum_{i=1}^{D'} \left|\eta_i\right\rangle \left\langle\eta_i\right|$ If $\{\left|\psi_i\right\rangle, \left|\phi_i\right\rangle\}$ is a linearly independent set, then $D' = 2D$. But if it isn't, then $D' < 2D$, and will equal D if $\text{Span}\{\left|\psi_i\right\rangle\} = \text{Span}\{\left|\phi_i\right\rangle\}$. In general, we don't know what D' will be.
   - Note that the MPO of the sum of two states is given by the direct product of the MPOs of the states: Let the MPOs be $v_\alpha A^{u_1}_{\alpha\beta} A^{u_2}_{\beta\gamma} w_\gamma$ and $v'_\alpha (A')^{u_1}_{\alpha\beta} (A')^{u_2}_{\beta\gamma} w'_\gamma$, where v and w are some boundary conditions. Then an MPO representing the sum will satisfy:

$$\tilde{v}_\alpha \tilde{A}^{u_1}_{\alpha\beta} \tilde{A}^{u_2}_{\beta\gamma} \tilde{v}_\gamma = v_\alpha A^{u_1}_{\alpha\beta} A^{u_2}_{\beta\gamma} v_\gamma + v'_\alpha (A')^{u_1}_{\alpha\beta} (A')^{u_2}_{\beta\gamma} w'_\gamma, \tag{20}$$

     which is clearly accomplished by $\tilde{v} = v \oplus v'$, $\tilde{A} = A \oplus A'$, and $\tilde{w} = w \oplus w$. This MPO will always have bond dimension which is the sum of the bond dimensions of A and A'. But if the states in the A and A' are linearly dependent, doing an SVD on $A \oplus A'$ will result in some zero singular values, and will reduce the actual bond dimension of the summed MPO if we represent it in that form.

2. What you've been measuring is the "operator space entanglement". Which is different from the state's entanglement entropy and other measures of entanglement. It is the thing that Chris wants you to measure, though.

3. Your von-neumman entropy has to be wrong because it violates the bound $2 \log 3$ for a state of 2 qutrits... figure out why that is a bound and fix your code. Your Renyi entropy is probably wrong too.

4. The lower bound of $M_2(\rho) = \sum_{u \in N(p)} W_\rho(u)^2$ is *not* really the lower bound in a fixed bond dimension calculation. With the best optimizer you could find ("L-BFGS-B"), you could get a 20-bond dim. MPO to within 10^{-7} of the random MPO you were looking at, and it seemed to asymptote there. This could be because of training, but it could also be because of fixed bond dimension!

Thoughts

- I'm still very confused about what the different types of entropies there are and what they represent. I should make a note about entropies and try to sort it all out. For example, I'm confused about how I would measure the entanglement entropy between sites. Oh..... I would construct the reduced density matrix... duh. I should do that...
- I'm not convinced that the space isn't convex... something just feels off about the above argument. I feel like we should be thinking about our space as the parameter space of values in fixed dimensional matrices, not the space of MPOs which represent states and all that garbage.. I have to think about this a bit more.
  - An interesting insight into the convexity argument above and what the positivty argument might do to it: If we allow our coefficients to be negative or complex then we could have $D' < D$ because some linearly independent states could cancel out. But if all coefficients are positive then that's not the case anymore, and D' must be greater than or equal to D.
  - It would be interesting to try to figure out how the coefficients of $\tilde{A}$ are related to A and A' in the case where $D' = D$.
- It should be easy to prove that the space isn't convex in the sense that Christopher means, though - just find two positive-valued MPOs of fixed bond dimension which add up to make a positive-valued operator of larger bond dimension. (My first thought to construct such a guy was $\left|0\right\rangle \left\langle 0\right|$ and $\left|1\right\rangle \left\langle 1\right|$, but that doesn't work... it needs to be positive valued in the phase space operator basis. I'm not sure how to construct a simple example of MPOs in that basis. I need to think about that more, it'll give me some intuition for Wigner functions)
- I should try to find the correct analytic expression for a lower bound of the Frobenius norm distance with fixed bond dimension.. don't have high hopes but I might learn something from trying.

1 April 2020

At long last: Bounds for the maniac

I've been trying to work this out furiously for the past two days, after having it in the back of my head for the past week, after thinking about how I might figure it out for the past month. I was so frustrated that I decided to write out a note to send to Chris about how I just can't get it to work. But as I was writing the note out, I realized that I hadn't been using the right definition of the second Reny entropy. As soon as I plugged that in.. it all just worked:

The second Renyi entropy of a state ρ is

$$S_2(\rho) = -\ln \mathrm{tr}[\rho^2] = -\ln \mathrm{tr}\left[\left(\sum_u W_u(\rho) A_u\right)^2\right] = -\ln\left(d^n \sum_u W_u(\rho)^2\right) \tag{21}$$

$$= -n \ln d - \ln \sum_u W_u(\rho)^2. \tag{22}$$

Jensen's inequality for $f(x) = x^2$ gives us:

$$\left(\sum_u |W_u(\rho)|\right)^2 \leq \sum_u W_u(\rho)^2 \tag{23}$$

Since $\ln x$ is an increasing function, this means that we can find a bound on the mana

$$\mathcal{M}(\rho) = \ln \sum_u |W_u(\rho)| \tag{24}$$

like so:

$$n \ln d - S_2(\rho) = \ln \sum_u W_u(\rho)^2 \geq \ln\left(\mathcal{M}(\rho)^2\right) = 2 \ln \mathcal{M}(\rho), \tag{25}$$

which gives:

$$\mathcal{M}(\rho) \leq \frac{1}{2}\left(n \ln d - S_2(\rho)\right) \leq \frac{1}{2} n \ln d \tag{26}$$

Where in the last step we used the fact that $S_2(\rho) > 0$: $S_2 = -\ln(\sum_i p_i^2) > 0 \leftrightarrow \ln(\sum_i p_i^2) < 0 \leftrightarrow \sum_i p_i^2 < 1$, which is true because $0 < p_i, \sum_i p_i = 1 \rightarrow p_i < 1, \mathrm{so} \sum_i p_i^2 < \sum_i p_i = 1$

8 April 2020

Computing the mana directly

(From meeting with Chis:) Is there a way to construct an MPO σ such that $\mathcal{M}(\rho) = \langle\rho|\sigma\rangle$? Chris pointed out that it might be possible to do this *using the result of the optimization*, because the constraints that the optimization makes active should be related to the set of strings $N(\rho) = \{u : W_u(\rho) < 0\}$. What if, for example, you made σ an MPO with 1s in the place of constraints and 0s everywhere else? Would that work?

9 April 2020

Transitioning from $2 \to n$ sites

Chris says that the major portion of doing this is a permutation of indices in a high/low/high/low order which comes from wanting to group indices corresponding to the same physical sites together, even though they are initially separate from each other. So I'll have to use `np.permutation` for this... it'll be a bit more complicated than I initially thought.

17 April 2020

Meeting with Chris

I forgot to ask the questions I had witten down... gotta be more organized. Some notes on what I should do next:

1. When grouping matrices from the SVD to form an MPO, group them like so: $USV \Rightarrow M_1 = U, M_2 = SV$. This ensures that all of the matrices are unitary and pushes the dirty shit in S to the very end, where it becomes a scalar and encodes $\mathrm{Tr}\,\rho^2$! Then just keep that value around and use it in normalization calculations etc.

2. Try to average $\mathcal{N}$ *analytically* over pure states using a Gaussian integral which naturaly pops up when you average something over normally distributed coefficients in $\mathbb{C}^n$. This involves analytically changing to the phase space basis, which can be done because the phase space operators have the structure $A_{a_1 a_2} = \sum_{u_1 u_2} \omega^{u_1 a_2 - u_2 a_1} X^{u_1} Z^{u_2}$ or something along those lines.. it's a tedious calculation but seems like it can be done.

3. There's actually no such thing as `np.permute` - he was thinking about Julia. It seems like `np.moveaxes` and `np.transpose` do the same thing - use either one

4. A good check of your code is expectation values - compute $\hat{Z}$ on each site in ρ and in the MPO. You should get the same thing

21 April 2020

Generalizing code from $2 \to n$ sites

I'm not sure if numpy reshape does things the right way.. I'm not even sure how to verify it. TODO do this!

Meeting with Chris: Derivative might not work

1. He thought about the whole derivative of MPO -¿ direct mana computation thing for an hour and couldn't figure it out. It might not work.

24 April 2020

Meeting with Chris: Resolving MPO construction issues

1. You need a tensor at each site which can be thought of as a list of matrices, one for each value of the physical index. Each of these matrices has virtual legs given by the result of the SVD - so you don't want to sum over that extra index like you've been doing! You want to actually keep that index around.

2. Make sure to store a *tensor* at each site, not a list of matrices. You do not want to be doing a python for loop in your contraction.

6 May 2020

Equations for talk

$\sum_{x \in S} |W(x)| \Rightarrow \sum_{x \in S} W(x)^2$

$$W_x(\rho) = \mathrm{Tr}[A_x \rho]$$

$$\mathcal{N}(\rho) = \sqrt{\sum_{x \in N(\rho)} W_\rho(x)^2} = \min_\sigma d(\rho, \sigma)$$

$$d(\rho_1, \rho_2) = \sqrt{\sum_x (W_x(\rho_1) - W_x(\rho_2))^2}$$

$$d(\rho, \sigma)^2 = \sum_x (W_x(\rho) - W_x(\sigma))^2 = \mathrm{Tr}[(\rho - \sigma)^2] = \mathrm{Tr}[\rho^2] - 2\,\mathrm{Tr}[\rho\sigma] + \mathrm{Tr}[\sigma^2]$$

$$W_{(x,y)}(\rho) = M^{(x)} M^{(y)}$$

$$N(\rho) = \{x \mid W_x(\rho) < 0\}$$

$$\sum_{x \in N(\rho)} |W_x(\rho)|$$

$$\mathcal{M}(\rho) = \ln\left(2 \sum_{x \in N(\rho)} |W_\rho(x)| + 1\right) \Rightarrow \mathcal{N}(\rho) = \sqrt{\sum_{x \in N(\rho)} W_\rho(x)^2}$$

4 June 2020

Can't perform contractions with matrix multiplication

We can do it for the first one:

$$M^{(1)}_{\beta\gamma} = \rho^{(1)}_{1u\beta}\sigma^{(1)}_{1u\gamma} \Rightarrow M^{(1)} = (\rho^{(1)}_1)^T(\sigma^{(1)}_1) \tag{27}$$

But what about the second?

$$M^{(2)}_{\alpha\sigma} = \rho^{(2)}_{\alpha u\beta}M^{(1)}_{\beta\gamma}\sigma^{(2)}_{\gamma u\sigma} = (\rho^{(2)}_\alpha)(M^{(1)})_{u\gamma}(\sigma^{(2)}_{..\sigma})_{\gamma u} \Rightarrow M^{(2)}_{\alpha\sigma} = (\rho^{(2)}_\alpha)(M^{(1)})(\sigma^{(2)}_{..\sigma})$$

Too weird... use `np.tensordot` instead!

11 June 2020

Using np.tensordot

According to the documentation, we can replace the previous equations with `M_1=np.tensordot(rho_1,sigma_1,axes=([1,1])` and `M_2=np.tensordot(np.tensordot(rho_2,M_1,axes=([2,0])),sigma_2,axes=([1,1],[2,0]))`

How do we deal with fixed boundary conditions?

Meeting with Chris

What to do next: Make a plot of cost function vs. sweep for 3 sites

14 June 2020

Fixing SVD procedure

I'm not getting the expected dimensions of my MPO matrices, so let's review what the procedure is: Given a density matrix / operator $\rho_{i,j}$ with indices $i, j \in \mathbb{Z}_d^n$ we do the following to get its matrix product representation $\{M^{(i)u}_{\alpha\beta}\}_{u\in\mathbb{Z}_d}$:

1. Reshape: $\rho_{i,j} = P_{(i_1\cdots i_n),(j_1\cdots j_n)}$ has dimensions $d^n \times d^n$
2. Permute: $P_{(i_1\cdots i_n),(j_1\cdots j_n)} = T_{(i_1j_1),\cdots,(i_nj_n)}$ has dimensions $d^2 \times \cdots \times d^2$ (n times)
3. Collect: $T_{(i_1j_1),(i_2j_2\cdots i_nj_n)}$ has dimensions $d^2 \times d^{2(n-1)}$
4. First site:
   (a) SVD: $T_{(i_1j_1),(i_2j_2\cdots i_nj_n)} = \sum_{\alpha\in\mathbb{Z}_{D_1}} U_{(i_1j_1),\alpha} S_{\alpha,\alpha} (V^\dagger)_{\alpha,(i_2j_2\cdots i_nj_n)}$, where $D_1 \in \{1,\cdots,d^2\}$ is determined by truncation
   (b) Save: $M^{(1)}_{\alpha,u,\beta} = \delta_{\alpha,1} U_{u,\beta}$ has dimensions $1 \times d^2 \times D_1$
   (c) Update: $T^{(1)}_{(\alpha i_2j_2),(i_3j_3\cdots i_nj_n)} = S_{\alpha,\alpha}(V^\dagger)_{\alpha,(i_2j_2\cdots i_nj_n)}$ has dimensions $D_1d^2 \times d^{2(n-2)}$
5. Inner sites: For $k \in \{2,\cdots,n-1\}$:
   (a) SVD: $T^{(k)}_{(\alpha i_kj_k),(i_{k+1}j_{k+1}\cdots i_nj_n)} = \sum_{\beta\in\mathbb{Z}_{D_k}} U_{(\alpha i_kj_k),\beta} S_{\beta,\beta} (V^\dagger)_{\beta,(i_{k+1}j_{k+1}\cdots i_nj_n)}$, where $D_k \in \{1,\cdots,D_{k-1}d^2\}$
   (b) Save: $M^{(k)}_{\alpha,u,\beta} = U_{(\alpha u),\beta}$ has dimensions $D_{k-1} \times d^2 \times D_k$
   (c) Update: $T_{(\alpha i_{k+1}j_{k+1}),(i_{k+2}j_{k+2}\cdots i_nj_n)} = S_{\alpha,\alpha}(V^\dagger)_{\alpha,(i_{k+1}j_{k+1}\cdots i_nj_n)}$ has dimensions $D_kd^2 \times d^{2(n-k-1)}$
6. Last site:
   (a) Save: $M^{(n)}_{\alpha,u,\beta} = T^{(n-1)}_{(\alpha,u),\beta}$ has dimensions $D_{n-1} \times d^2 \times 1$ (note that $T^{(n-1)}_{(\alpha i_nj_n),\beta}$ has dimensions $D_{n-1}d^2 \times 1$)
7. Return $\{M^{(k)}\}_{k\in\mathbb{Z}_n}$

This means that if my truncation scheme is to not do any truncation at all, and if I never get any 0 singular values (or just leave them in the matrices), then I should expect to see the following dimensions of my matrices and local tensor after performing each SVD for $d = 3, n = 5$. The density matrix has dimensions $3^5 \times 3^5$ and the environment tensor T starts out with dimensions 9×9^4. The dimensions of T and the matrix saved in the MPO are:

k	$\dim(M^{(k)})$	$\dim(T^{(k)})$	D_k
1	$1 \times 9 \times 9$	9×9^4	9
2	$9 \times 9 \times 9^2$	$9^2 \times 9^3$	9^2
3	$9^2 \times 9 \times 9^3$	$9^3 \times 9^2$	9^3
4	$9^3 \times 9 \times 9^4$	$9^4 \times 9$	9^4
5	$9^4 \times 9 \times 1$	$9^5 \times 1$	-

TODO Fix bond dim to only grow up till halfway

Meeting with Chris

Misunderstanding of `np.tensordot`

The way the `axes=([a,b],[c,d])` argument works is that it sums over $A_{ab}B_{cd}$, not $A_{ac}B_{bd}$. Also, you can't split this up into two calls of `tensordot`, because that would give you a result that scales quadratically in B_{ab}, not linearly.

Bond dimension should not grow until the end!

Your bond dimension is actually correct. It's not supposed to be $D_k = d^{2k}$, only until halfway through and then it's supposed to shrink back down. This is because you could have started doing SVDs from the right and moved leftwards! So your MPO code is correct.

Forgot to take complex conjugate

Element-wise complex conjugation (`.conj()`) of the second MPO is necessary

Next steps

There's probably something wrong in your contraction algorithm, because you're not getting a normalized result. Chris thinks that you haven't got your axes / dimensions matched up properly. Try inputting MPOs with different bond dimensions into your code to catch an error.

June 19 2020

Meeting with Chris

Next steps

Check that optimization for 2 sites agrees with old code.

Bibliography

S. X. Cui, P. Hayden, T. He, M. Headrick, B. Stoica, and M. Walter, Communications in Mathematical Physics (2019), 10.1007/s00220-019-03510-8.