Notes for DMRG++
A 2-site DMRG implementation on C++

David Aceituno

Contents

In these notes I try to collect some definitions, details, tricks and resources that are useful when implementing the DMRG algorithm. In particular those that I would have liked to seen explained somewhere in more detail during the development of my program. In other words, I write in these notes what I've learned after having been stuck on details for longer than I care to admit, with the hope of not having to get stuck again if I revisit the steps later.

I loosely follows the steps in these articles

> Schollwoeck, U. (2010). The density-matrix renormalization group in the age of matrix product states. Annals of Physics, 326(1), 96–192.

> Kjäll, J. A., Zaletel, M. P., Mong, R. S. K., Bardarson, J. H. & Pollmann, F. Phase diagram of the anisotropic spin-2 XXZ model: Infinite-system density matrix renormalization group study. *Phys. Rev. B* **87,** 235106 (2013).

but I begin by establishing some definitions and notation specific to my implementation.

Notation

<div style="border:1px solid; padding:10px;">

Tensor Diagram Notation

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

Operation	Index notation	Diagram notation	Resulting rank
Scalar product	$\sum_i a_i b_i = a^{\mathsf{T}} b$	$\boxed{a} - \boxed{b}$	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	$\sum_{m,n} A^{mn}_{ij} (B^{mn}_{kl})^{\dagger}$	$\boxed{A}$ $\boxed{B^{\dagger}}$	4

</div>

Latin letters are reserved for physical indices, and greek indices for bond indices.

The order of indices in all tensors follows doesn't follow a rule other than what's most convenient for computations:

$$\Lambda_{\alpha}\beta = \ \alpha - \boxed{\Lambda} - \beta \ = \ 0 - \boxed{\Lambda} - 1 \tag{1}$$

$$\Gamma^i_{\alpha\beta} = \ \begin{array}{c} \alpha - \boxed{\Gamma} - \beta \\ | \\ i \end{array} \ = \ \begin{array}{c} 1 - \boxed{\Gamma} - 2 \\ | \\ 0 \end{array} \tag{2}$$

$$\tag{3}$$

The tensor $\Theta^{ij}_{\alpha\beta}$, resulting from the merger of two sites, is subject to a Schmidt (SVD) decomposition, and therefore its indices are most conveniently ordered like

$$\theta^{ij}_{\alpha\beta} = \ \begin{array}{c} \alpha \ \boxed{\Theta} \ \beta \\ | \ | \\ i \ \ j \end{array} \ = \ \begin{array}{c} 1 \ \boxed{\Theta} \ 3 \\ | \ | \\ 0 \ \ 2 \end{array} \tag{4}$$

Finally we shall see MPO-type tensors, such as the Hamiltonian, with the following index order

$$\alpha - \boxed{H} - \beta \;\; \overset{i}{\underset{j}{\vert\;\;\vert}} \;\; = \;\; 0 - \boxed{H} - 1 \;. \overset{2}{\underset{3}{\vert\;\;\vert}} \tag{5}$$

In C++ these are objects of type `Eigen::Tensor<Scalar,rank>;`, where `Scalar` is typically the same as `double` if the MPS is real, or `std::complex<double>` if it is complex.

Model Hamiltonian: The Quantum Ising model

We study the 1D Ising model with a transverse field, given by the following Hamiltonian

$$H = \frac{1}{2} \sum_i \left[-J\sigma_i^z \sigma_{i+1}^z - g\sigma_i^x \right].$$

Naturally the transverse field can be put on the z direction inststead:

$$H = \frac{1}{2} \sum_i \left[-J\sigma_i^x \sigma_{i+1}^x - g\sigma_i^z \right].$$

We can split the Hamiltonian into odd and even parts by introducing

$$h_i = -J\sigma_i^z \sigma_{i+1}^z - \frac{g}{2}(\sigma_i^x + \sigma_{i+1}^x)$$
$$h_j = -J\sigma_j^z \sigma_{j+1}^z - \frac{g}{2}(\sigma_j^x + \sigma_{j+1}^x)$$

such that the total Hamiltonian becomes

$$H = \sum_{i \text{ even}} h_i + \sum_{j \text{ odd}} h_j,$$

For two sites this can be written out explicitly

```
from numpy import *
from sympy import *
# pauli spin

I  = mat([[1, 0],[ 0, 1]])
sx = mat([[0, 1],[ 1, 0]])
sy = mat([[0, -1j],[1j, 0]])
sz = mat([[1, 0],[0, -1]])

SX1 = kron(sx,I); SX2 = kron(I,sx)
SZ1 = kron(sz,I); SZ2 = kron(I,sz)
J,g = symbols("J g")
h_evn = -J * SZ1*SZ2  - 0.5*g * (SX1 + SX2)
h_odd = -J * SZ2*SZ1  - 0.5*g * (SX2 + SX1)
print("ZZ X:")
pprint(Matrix(h_evn + h_odd))
h_evn = -J * SX1*SX2  - 0.5*g * (SZ1 + SZ2)
```

3

```
h_odd = -J * SX2*SX1  - 0.5*g * (SZ2 + SZ1)
print("XX Z:")
pprint(Matrix(h_evn + h_odd))

## ZZ X:
##   -2 J   -1.0 g  -1.0 g      0
##
## -1.0 g    2 J      0      -1.0 g
##
## -1.0 g    0       2 J     -1.0 g
##
##    0     -1.0 g  -1.0 g    -2 J
## XX Z:
## -2.0 g    0       0       -2 J
##
##    0      0      -2 J       0
##
##    0     -2 J     0         0
##
## -2 J      0       0       2.0 g
```

In DMRG++ it is obtained by calling `H()` in the following code:

C++ code

```cpp
std::vector<Eigen::MatrixXcd> gen_manybody_spin(const Eigen::Matrix2cd &s, int sites) {
    std::vector<Eigen::MatrixXcd> S;
    Eigen::MatrixXcd tmp;
    S.clear();
    for (int i = 0; i < sites; i++) {
        tmp = i == 0 ? s : I();
        for (int j = 1; j < sites; j++) {
            tmp = Eigen::kroneckerProduct(tmp, i == j ? s : I()).eval();
        }
        S.emplace_back(tmp);
    }
    return S;
}

Eigen::MatrixXcd h(int sites, int position) {
    int i = mod(position, sites);
    int j = mod(position + 1, sites);
    SX = gen_manybody_spin(sx(), sites);
    SY = gen_manybody_spin(sy(), sites);
    SZ = gen_manybody_spin(sz(), sites);
    return (-J * SX[i] * SX[j] - 0.5 * g * (SZ[i] + SZ[j]));
  }

Eigen::MatrixXcd H(int sites) {
    Eigen::MatrixXcd hi = Eigen::MatrixXcd::Zero((long) pow(2, sites), (long) pow(2, sites));
    for (int position = 0; position < sites; position++) {
        hi += h(sites, position);
    }
    return hi;
  }
```

Here the `sx(), sy(), sz()` matrices are simply the 2×2 Pauli matrices in `Eigen` matrix type, and `h(sites,position)` generates the even/odd terms in the Hamiltonian as defined above.

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

$$|\psi\rangle = \sum_{\sigma_1...\sigma_n} c_{\sigma_1...\sigma_n} |\sigma_1...\sigma_n\rangle, \tag{6}$$

with complex coefficients, (or amplitudes) $c_{\sigma_1...\sigma_n}$, which together can be thought of as a rank-n tensor. For a given configuration $|\sigma_1...\sigma_n\rangle$, a coefficient can be expressed as the product of matrices of the form $A^{\sigma_i} = (A^i)^{\sigma_i}_{\alpha_{i-1}, \alpha_i}$, such that $c_{\sigma_1 \sigma_2...\sigma_n} = A^{\sigma_1}_{1,\alpha_1} A^{\sigma_2}_{\alpha_1, \alpha_2}...A^{\sigma_n}_{\alpha_{n-1}, \alpha_n}$, where each index a_i runs from $i = 1$ to $i = \chi_i$. Each $(A^i)^{\sigma_i}_{\alpha_{i-1}, \alpha_i}$ can therefore be interpreted as a rank-3 tensor, i.e. one $\chi_{i-1} \times \chi_i$ matrix for each possible value of σ_i.

Omitting the α's, the full state is therefore equivalently expressed as

$$|\psi\rangle = \sum_{\sigma_1...\sigma_n} A^{\sigma_1}...A^{\sigma_n} |\sigma_1...\sigma_n\rangle, \tag{7}$$

In this reformulation we have effectively *unfolded* a rank-n tensor to a series of rank-3 tensors. In tensor diagram notation the unfolding reads

From this picture it is clear why χ_i is called the *bond dimensions*, the dimension of of the contraction between A's, which encodes the level of entanglement with the subsystems to the left or right of a thought bipartition at i.

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

$$|\psi\rangle = \sum_{k=1}^{\chi} \lambda_k |\phi\rangle_L \otimes |\phi\rangle_R. \tag{8}$$

connected by singular values λ_k describing the *entanglement spectrum* between the two subsystems. This is also called the *Schmidt spectrum*. In diagrammatic notation
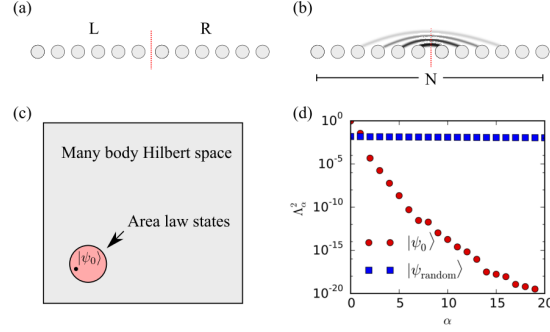
Figure 1: (a): Bipartition of a 1D system into two half chains. (b): Significant quantum fluctuations in gapped ground states occur only on short length scales. (c): 1D area law states make up a very small fraction of the many-body Hilbert space but contain all gapped ground states. (d): Comparison of the entanglement spectrum of the ground state of the transverse field Ising model ($g = 1.5$) and a random state for a system consisting of $N = 16$ spins.

Figure 1: From "Efficient Numerical Simulations Using Matrix-Product States" by Frank Pollmann (2015).

The singular values $\Lambda = \text{diag}[\lambda_1, ..., \lambda_\chi]$, directly give us the *entanglement entropy*

$$S_E = -\sum_{k=1}^{\chi} \lambda_k^2 \ln \lambda_k^2. \tag{9}$$

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

Properties of the entanglement spectrum

The success of the DMRG algorithm for 1D systems relies on the observation that the singular values decay fast if the system is gapped, exponentially in fact, and can be discarded if they are below some threshold. See the figure from Pollmann's article.

MPS in Vidal's $\Gamma\Lambda$ notation.

Canonical form

Matrix Product Operators (MPO)

Hamiltonian as an MPO

In MPO form the Hamiltonian of the quantum Ising model with a transverse field reads

$$H_{\text{MPO}} = \begin{pmatrix} I & 0 & 0 \\ \sigma^z & 0 & 0 \\ -g\sigma^x & -J\sigma^z & I \end{pmatrix}.$$

where each element is a 2×2 -matrix. To transform into an MPO note that this matrix is essentially an outer (3×3) matrix containing inner (2×2) matrices, or equivalently a $(2 \times 3) \times (2 \times 3)$ matrix. The goal is to obtain an MPO with dimensions $(3, 3, 2, 2)$, depicted as

$$
\begin{array}{c}
2_{dim=2} \\
| \\
0_{dim=3} -\boxed{H}- 1_{dim=3} \\
| \\
3_{dim=2}
\end{array}
\quad = \quad
\begin{array}{c}
a_1 \\
| \\
b_1 -\boxed{H}- b_2 \\
| \\
a_2
\end{array} ,
\tag{10}
$$

where the left figure indicates the numbering of the indices and also their dimension.

Returning to H_{MPO}, remember that it should follow column-major notation in C++, such that the left (2×3) column is counted first. Calling these indices $(a_1 \times b_1)$, we note that a_1 should "tick" before b_1. Likewise for the (2×3) rows, denoted $(a_2 \times b_2)$, note that a_2 ticks before b_2. Therefore to get a rank 4 MPO we should simply reshape into indices $(a_1, b_1, a_2, b_2) = (2, 3, 2, 3)$.

Next we should reorder the indices to get the right diagram above. To do this, we do a transpose, or shuffle like $(1, 3, 0, 2)$ to get the order of indices $(b_1, b_2, a_1, a_2) = (3, 3, 2, 2)$. Then the first index b_1 selects the outer matrix row, and b_2 the outer matrix column, while a_1 selects inner row, and a_2 inner column.

Environments

Expectation values using MPO's

The energy expectation value $\langle \psi | H | \psi \rangle$ computed as

$$
\langle E \rangle =
\begin{array}{c}
\boxed{\Lambda^B} - \boxed{\Gamma^A} - \boxed{\Lambda^A} - \boxed{\Gamma^B} - \boxed{\Lambda^B} \\
\boxed{L} \qquad \boxed{H} \qquad \boxed{H} \qquad \boxed{R} \\
\boxed{\Lambda^B} - \boxed{\Gamma^{A\dagger}} - \boxed{\Lambda^A} - \boxed{\Gamma^{B\dagger}} - \boxed{\Lambda^B}
\end{array}
\tag{11}
$$

The 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. $LA \rightarrow L$ and $BR \rightarrow R$ 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.

7
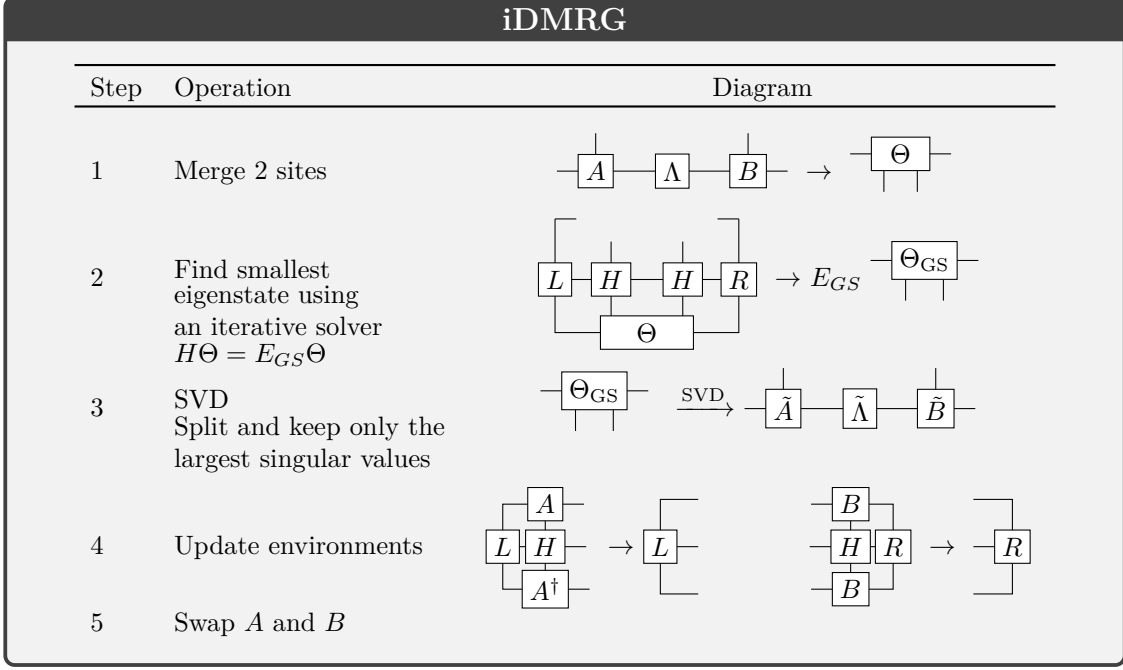
Step 1. Merge two sites

In two-site DMRG, the first step merges the two sites around the bond at i into one single tensor

$$\Theta^{\sigma_i,\sigma_{i+1}}_{\alpha_{i-1},\alpha_{i+1}} = \sum_{\alpha_i} A^{\sigma_i}_{\alpha_{i-1},\alpha_i} \Lambda_{\alpha_i,\alpha_i} B^{\sigma_{i+1}}_{\alpha_i,\alpha_{i+1}} = \sum_{\alpha_i} \Lambda_{\alpha_{i-1},\alpha_{i-1}} \Gamma^{\sigma_i}_{\alpha_{i-1},\alpha_1} \Lambda_{\alpha_i,\alpha_i} \Gamma^{\sigma_{i+1}}_{\alpha_i,\alpha_{i+1}} \Lambda_{\alpha_{i+1},\alpha_{i+1}}$$

Step 2. Finding the smallest eigenstate

Step 3. Schmidt decomposition using SVD

Step 4. Update environments

MPO with substracted expectation value

Computing the expecation value using $H = H_{\mathrm{MPO}}$ as defined earlier yields the total energy of the entire chain, and one has to divide by the chain-length L to get the per-site energy density. As L grows this becomes problematic, for instance when calculating the variance $\mathrm{Var} H = \langle \Delta H \rangle = \langle (H - \langle H \rangle)^2 \rangle = \langle H^2 \rangle - E^2$, as this would imply loss of precision when subtracting two very large numbers. According to C. Hubig, the loss of precision incurred is approximately $\log_{10}(E^2)$ digits.

To remedy this one can use *reduced* Hamiltonian where we subtract the current per-site energy $\langle e \rangle$ from the lower left corner of the MPO. Considering the k-th DMRG-step, we have $\tilde{H}_{\mathrm{MPO}}(k) = H_{\mathrm{MPO}} - \langle e \rangle_k I_{2\times 2}$, where $\langle e \rangle_k I_{2\times 2}$ is a diagonal matrix subtracted from the lower left corner. For instance, for the quantum Ising model the reduced Hamiltonian is

$$\tilde{H}_{\mathrm{MPO}}(k) = \begin{pmatrix} I & 0 & 0 \\ \sigma^z & 0 & 0 \\ -g\sigma^x - \langle e \rangle_k I & -J\sigma^z & I \end{pmatrix}.$$

Using this reduced version we obtain *reduced* environments $\tilde{L}$ and $\tilde{R}$

$$\text{(diagram)} \tag{12}$$

The per-site energy $\langle e \rangle_k$ has to be computed at each step using the original, non-reduced H_{MPO}, together with the reduced environments:

$$2\langle e \rangle_k = \text{(diagram)}. \tag{13}$$

Practical considerations

1) We set $\langle e \rangle_0 = 0$, which implies $\tilde{H}_{\mathrm{MPO}}(0) = H_{\mathrm{MPO}}(0)$, $\tilde{L}(0) = L(0)$ and $\tilde{R}(0) = R(0)$.

2) At the k-th step $L(k)$ and $R(k)$ contain k sites each, so the chain length is $L = 2k$.

3) At the k-th step we have subtracted $\sum_{i=0}^{k} \langle e \rangle_i \approx k \langle e \rangle_k$ from $\tilde{L}(k)$ and $\tilde{R}(k)$.

4) The following relations can be used as a sanity check:

$$0 = \text{(diagram)} = \text{(diagram)}. \tag{14}$$

Error measures

Truncation error

The truncation error is perhaps the simplest measure of convergence, obtained at almost no cost from during the Schmidt decomposition. Recall that the singular values $\Lambda = \mathrm{diag}[\lambda_1, ..., \lambda_\chi]$ (sorted in decreasing order) gives us the *entanglement entropy*

$$S_E = -\sum_{k=1}^{\chi_{\max}} \lambda_k^2 \ln \lambda_k^2. \tag{15}$$

Where we have enforced an upper limit on the bond dimension $\chi = \chi_{\max}$, to keep the size of the MPS tractable. Enforcing this upper limit will essentially discard entanglement information from the MPS. The *truncation error* $\epsilon(\chi_{\max})$ is then defined as the discarded weight

9

$$\epsilon(\chi_{\max}) = \sum_{k > \chi_{\max}} \lambda_k = 1 - \sum_{k=1}^{\chi_{\max}} \lambda_k. \tag{16}$$

Energy Variance

The energy variance $\mathrm{Var}(H) = \langle (H - \langle H \rangle)^2 \rangle = \langle H^2 \rangle - E^2$ signals how close a given MPS is to an eigenstate of the Hamiltonian, independently of implementation details, and as such it is a more robust alternative to the truncation error. The total variance is extensive, growing linearly with L, so in practice we want to measure the per-site variance $\mathrm{Var}(H)/L$, which should converge as $L \to \infty$.

There are four ways to compute the energy variance of an MPS, described in the following articles

1. Using a double-layer of MPO's
2. Using the moments calculated from the characteristic function of the transfer matrix
3. Using the local 2-site Hamiltonians
4. Using an approximation, the 2-site variance

Variance from a double-layer of MPO's

An easy but costly way of obtaining the variance is by using a double layer of Hamiltonian MPO's to get the squared expectation value $\langle H^2 \rangle$.

$$\langle H^2 \rangle = \tag{17}$$

This method is costly because it requires us to keep track of a second set of environments L_2 and R_2, which need to be grown concurrently with L and R.

Because each term in $\langle H^2 \rangle - \langle E \rangle^2$ grows quadratically with L, the subtraction will quickly suffer from loss of precision, as discussed earlier. For this reason it is more convenient to work with the reduced quantities, and as a bonus we can obtain the variance directly from the definition $\mathrm{Var}(H) = \langle (H - \langle H \rangle)^2 \rangle = \langle \tilde{H}^2 \rangle$.

$$\mathrm{Var}(H) = \tag{18}$$

Variance from characteristic functions

The method below will follow the steps given in

West, C. G., Garcia-Saez, A., & Wei, T. C. (2015). Efficient evaluation of high-order moments and cumulants in tensor network states. Physical Review B - Condensed Matter and Materials Physics, 92(11), 1–20.

This is perhaps the simplest method to implement and performs very well for generic observables. Most details are clear in the article except for just a few pitfalls that had me stuck for a while, so I'll just sumarize those here.

For a given operator M, the characteristic function $G(a) = \langle e^{iaM} \rangle$ contains all of the nonlocal information about the higher order moments $\langle M^n \rangle$. This is clear from a Taylor expansion about $a = 0$:

$$G(a) = 1 + ia\langle M \rangle + \frac{(ia)^2}{2}\langle M^2 \rangle + ... \tag{19}$$

and subsequently we can obtain every moment $\langle M^n \rangle$ from the n-th derivative of $G(a)$.

A useful quantity is $\log G(a)$, the derivatives of which gives us the n-th cumulants κ_n directly. This can be seen from the same Taylor expansion as above

$$\log G(a) \approx \log(1 + ia\langle M \rangle + \frac{(ia)^2}{2}\langle M^2 \rangle + ...) \tag{20}$$

together with the expansion $\log(1 + x) \approx x - \frac{1}{2}x^2 + ...$ for small x, which gives us

$$\log G(a) = ia\langle M \rangle + \frac{(ia)^2}{2}\langle M^2 \rangle - \frac{(ia)^2}{2}\langle M \rangle^2 + O(a^3) \tag{21}$$

$$= ia\kappa_1 + \frac{(ia)^2}{2}\kappa_2 ... \tag{22}$$

If M is the sum of single-site operators (such as the total spin) the computation $\langle e^{iaM} \rangle$ is straightforward. Let's here consider an infinite chain, and a 2-site DMRG algorithm as usual (see the article for the finite and 1-site cases). Define a transfer matrix with the operator applied

$$\tag{23}$$

Then in the infinite limit it is shown that

$$G_\infty(a) = \lim_{L \to \infty} \langle e^{iaM} \rangle^{l/L} = \left(\frac{\lambda(a)}{\lambda(0)} \right)^{1/l}, \tag{24}$$

where

- l is the number of sites in the transfer matrix, (here $l = 2$),
- $\lambda(a)$ is the largest eigenvalue of the transfer matrix, and
- $\lambda(0)$ is the largest eigenvalue for the case $a = 0$, which is used if the MPS isn't normalized already.

If on the other hand M is a sum of multi-site operator (such as the Ising Hamiltonian), the operator can to be approximated by a Suzuki-Trotter decomposition which produces layers of two-body terms. For instance, the second-order in a Suzuki-Trotter approximation results in three unitary gates

$$e^{iaH} = e^{iaH_{\text{even}} + iaH_{\text{odd}}} \approx e^{\frac{ia}{2}H_{\text{even}}} e^{iaH_{\text{odd}}} e^{\frac{ia}{2}H_{\text{even}}}. \tag{25}$$

The resulting transfer matrix is obtained by applying these layers in exactly the same way one does a single ´TEBD´-step, i.e. one evolves the MPS layer by layer,

1. Merge AB into Θ^{AB}
2. Contract (or "time evolve") Θ^{AB} with the gate $e^{(\cdots)}$ to obtain an updated $\tilde{\Theta}^{AB}$.
3. Split by SVD $\tilde{\Theta}^{AB}$ down to $\tilde{A}$ and $\tilde{B}$. If this iteration used the last gate, end here.
4. Do the AB swap by setting $\tilde{A} \to B$ and $\tilde{B} \to A$.
5. Start again from 1, using the next gate.

When all the gates have been applied, the transfer matrix can be constructed by simply overlapping the evolved MPS with the original, i.e.,

$$\tag{26}$$

The energy variance per site for an infinite system can now be obtained from the second cumulant, which is simply the second derivative of $\log G(a)$:

$$\lim_{L \to \infty} \text{Var}(H)/L = \lim_{L \to \infty} \frac{\kappa_2}{L} = \frac{\log G_\infty(a) + \log G_\infty(-a) - 2\log G_\infty(0)}{a^2} + O(a^2) \tag{27}$$

However, note that $G(-a) = G(a)^*$ and $G(0) = 1$, so the relation above simplifies to

$$\text{Var}(H)/L = \frac{\log |G_\infty(a)|^2}{a^2} + O(a^2) \tag{28}$$

Incidentally, the energy per site itself is also obtained easily through the first derivative

$$\langle e \rangle = \frac{\log G_\infty(a) - \log G_\infty(-a)}{2a} + O(a^2) \tag{29}$$

NOTE

Two important remarks are in order here

- The SVD split in step 3 should not truncate too much. This is a high-precision calculation, and rather than keeping a set number of singular values, prefer keeping all the singular values above some small threshold, like 10^{-14}.
- The choice of a can be tricky. Too small means some of the calculations suffer from underflow of double precision. Too big gives significant errors. Empirically, choosing a around $10^{-2} - 10^{-4}$ has worked best for me.

Variance from local Hamiltonian

The method below will follow the steps given in

Vanderstraeten, L. (n.d.). Tangent space methods for matrix product states

and the appendix in

Zauner-Stauber, V., Vanderstraeten, L., Fishman, M. T., Verstraete, F., & Haegeman, J. (2017). Variational optimization algorithms for uniform matrix product states, 1–32

where you can also read more.

We can compute the variance by calculating the series for the squared Hamiltonian explicitly.

$$\langle H^2 \rangle = \left\langle \left(\sum_{i=0}^{L} h_i \right)^2 \right\rangle = \left\langle \left(\sum_{i \text{ even}}^{L} h_i + \sum_{j \text{ odd}}^{L} h_j \right)^2 \right\rangle \tag{30}$$

$$= \left\langle \left(\sum_{i \text{ even}}^{L} h_i \right) \left(\sum_{i' \text{ even}}^{L} h_{i'} \right) \right\rangle \qquad \mathbf{I} \tag{31}$$

$$+ \left\langle \left(\sum_{j \text{ odd}}^{L} h_j \right) \left(\sum_{j' \text{ odd}}^{L} h_{j'} \right) \right\rangle \qquad \mathbf{II} \tag{32}$$

$$+ \left\langle \left(\sum_{i \text{ even}}^{L} h_i \right) \left(\sum_{j' \text{ odd}}^{L} h_{j'} \right) \right\rangle \qquad \mathbf{III} \tag{33}$$

$$+ \left\langle \left(\sum_{j \text{ odd}}^{L} h_i \right) \left(\sum_{i' \text{ even}}^{L} h_{i'} \right) \right\rangle \qquad \mathbf{IV} \tag{34}$$

Below, each term **I-IV** can be further decomposed into

$$\mathbf{I} = \langle \sum_{i=i'} h_i h_{i'} \rangle + \langle \sum_{i-i' \geq 2} h_i h_{i'} \rangle + \langle \sum_{i'-i \geq 2} h_i h_{i'} \rangle \qquad = \mathbf{I}_a + \mathbf{I}_b + \mathbf{I}_c \tag{35}$$

$$\mathbf{II} = \langle \sum_{j=j'} h_j h_{j'} \rangle + \langle \sum_{j-j' \geq 2} h_j h_{j'} \rangle + \langle \sum_{j'-j \geq 2} h_i h_{i'} \rangle \qquad = \mathbf{II}_a + \mathbf{II}_b + \mathbf{II}_c \tag{36}$$

$$\mathbf{III} = \langle \sum_{i+1=j'} h_i h_{j'} \rangle + \langle \sum_{i=j'+1} h_i h_{j'} \rangle + \langle \sum_{i-j' \geq 3} h_i h_{j'} \rangle + \langle \sum_{j'-i \geq 3} h_i h_{j'} \rangle \qquad = \mathbf{III}_a + \mathbf{III}_b + \mathbf{III}_c + \mathbf{III}_d \tag{37}$$

$$\mathbf{IV} = \langle \sum_{j+1=i'} h_j h_{i'} \rangle + \langle \sum_{j=i'+1} h_j h_{i'} \rangle + \langle \sum_{j-i' \geq 3} h_j h_{i'} \rangle + \langle \sum_{i'-j \geq 3} h_j h_{i'} \rangle \qquad = \mathbf{IV}_a + \mathbf{IV}_b + \mathbf{IV}_c + \mathbf{IV}_d \tag{38}$$

To translate all these terms into tensor contractions, we first need to introduce some useful notation. Let T_e and T_o be even and odd transfer matrices, defined as

$$T_e = \tag{39}$$

$$T_o = \tag{40}$$

Let also l_e, r_e, l_o and r_o be the left and right dominant eigenvectors for the transfer matrices, defined through the following eigenvalue equations:

$$
\begin{array}{c}
\boxed{l_e}\;\boxed{\Lambda^B}-\boxed{\Gamma^A}-\boxed{\Lambda^A}-\boxed{\Gamma^B} \\
\boxed{\Lambda^B}-\boxed{\Gamma^{A\dagger}}-\boxed{\Lambda^A}-\boxed{\Gamma^{B\dagger}}
\end{array}
= \eta_e \boxed{l_e}
\tag{41}
$$

$$
\begin{array}{c}
\boxed{\Lambda^B}-\boxed{\Gamma^A}-\boxed{\Lambda^A}-\boxed{\Gamma^B}\;\boxed{r_e} \\
\boxed{\Lambda^B}-\boxed{\Gamma^{A\dagger}}-\boxed{\Lambda^A}-\boxed{\Gamma^{B\dagger}}
\end{array}
= \eta_e \; \boxed{r_e}
\tag{42}
$$

$$
\begin{array}{c}
\boxed{l_o}\;\boxed{\Lambda^A}-\boxed{\Gamma^B}-\boxed{\Lambda^B}-\boxed{\Gamma^A} \\
\boxed{\Lambda^A}-\boxed{\Gamma^{B\dagger}}-\boxed{\Lambda^B}-\boxed{\Gamma^{A\dagger}}
\end{array}
= \eta_o \boxed{l_o}
\tag{43}
$$

$$
\begin{array}{c}
\boxed{\Lambda^A}-\boxed{\Gamma^B}-\boxed{\Lambda^B}-\boxed{\Gamma^A}\;\boxed{r_o} \\
\boxed{\Lambda^A}-\boxed{\Gamma^{B\dagger}}-\boxed{\Lambda^B}-\boxed{\Gamma^{A\dagger}}
\end{array}
= \eta_o \; \boxed{r_o}.
\tag{44}
$$

The transfer matrices can be rescaled so that their largest eigenvalue becomes one, by redefining $\Phi_e \to \Phi_e/\sqrt{\eta_e}$ and $\Phi_o \to \Phi_o/\sqrt{\eta_o}$, or equivalently $T_e \to T_e/\eta_e$ and $T_o \to T_o/\eta_o$. Furthermore the left and right eigenvectors should be normalized such that

$$
\boxed{l_e}\qquad\boxed{r_e} = 1
\tag{45}
$$

$$
\boxed{l_o}\qquad\boxed{r_o} = 1.
\tag{46}
$$

With these definitions we can now identify all the terms with the following tensor contractions. *Remember to use rescaled MPS (divided by η) everywhere!*

$$
\mathbf{I}_a = \frac{L}{2}\;\boxed{l_e}
\begin{array}{c}
\boxed{\Lambda^B}-\boxed{\Gamma^A}-\boxed{\Lambda^A}-\boxed{\Gamma^B} \\
\boxed{h_{\text{even}}} \\
\boxed{h_{\text{even}}} \\
\boxed{\Lambda^B}-\boxed{\Gamma^{A\dagger}}-\boxed{\Lambda^A}-\boxed{\Gamma^{B\dagger}}
\end{array}
\boxed{r_e}
\tag{47}
$$

$$
\mathbf{I}_b = \frac{L}{2}\;\boxed{l_e}
\begin{array}{c}
\boxed{\Lambda^B}-\boxed{\Gamma^A}-\boxed{\Lambda^A}-\boxed{\Gamma^B} \\
\boxed{h_{\text{even}}} \\
\boxed{\Lambda^B}-\boxed{\Gamma^{A\dagger}}-\boxed{\Lambda^A}-\boxed{\Gamma^{B\dagger}}
\end{array}
\boxed{\sum_{n=0}^{L/2-1}(T_e)^n}
\begin{array}{c}
\boxed{\Lambda^B}-\boxed{\Gamma^A}-\boxed{\Lambda^A}-\boxed{\Gamma^B} \\
\boxed{h_{\text{even}}} \\
\boxed{\Lambda^B}-\boxed{\Gamma^{A\dagger}}-\boxed{\Lambda^A}-\boxed{\Gamma^{B\dagger}}
\end{array}
\boxed{r_e}
\tag{48}
$$

$$
\mathbf{I}_c = \frac{L}{2}\;\boxed{l_o}
\begin{array}{c}
\boxed{\Lambda^A}-\boxed{\Gamma^B}-\boxed{\Lambda^B}-\boxed{\Gamma^A} \\
\boxed{h_{\text{odd}}} \\
\boxed{\Lambda^A}-\boxed{\Gamma^{B\dagger}}-\boxed{\Lambda^B}-\boxed{\Gamma^{A\dagger}}
\end{array}
\boxed{\sum_{n=0}^{L/2-1}(T_o)^n}
\begin{array}{c}
\boxed{\Lambda^A}-\boxed{\Gamma^B}-\boxed{\Lambda^B}-\boxed{\Gamma^A} \\
\boxed{h_{\text{odd}}} \\
\boxed{\Lambda^A}-\boxed{\Gamma^{B\dagger}}-\boxed{\Lambda^B}-\boxed{\Gamma^{A\dagger}}
\end{array}
\boxed{r_o}
\tag{49}
$$

14

$$\mathbf{II}_a = \frac{L}{2} \quad \boxed{l_o}\ \boxed{\Lambda^A}\ \boxed{\Gamma^B}\ \boxed{\Lambda^B}\ \boxed{\Gamma^A}\ \boxed{h_{\text{odd}}}\ \boxed{h_{\text{odd}}}\ \boxed{\Lambda^A}\ \boxed{\Gamma^{B\dagger}}\ \boxed{\Lambda^B}\ \boxed{\Gamma^{A\dagger}}\ \boxed{r_o} \tag{50}$$

$$\mathbf{II}_b = \frac{L}{2} \quad \boxed{l_o}\ \boxed{\Lambda^A}\ \boxed{\Gamma^B}\ \boxed{\Lambda^B}\ \boxed{\Gamma^A}\ \boxed{h_{\text{odd}}}\ \boxed{\Lambda^A}\ \boxed{\Gamma^{B\dagger}}\ \boxed{\Lambda^B}\ \boxed{\Gamma^{A\dagger}}\ \boxed{\textstyle\sum_{n=0}^{L/2-1}(T_o)^n}\ \boxed{\Lambda^A}\ \boxed{\Gamma^B}\ \boxed{\Lambda^B}\ \boxed{\Gamma^A}\ \boxed{h_{\text{odd}}}\ \boxed{\Lambda^A}\ \boxed{\Gamma^{B\dagger}}\ \boxed{\Lambda^B}\ \boxed{\Gamma^{A\dagger}}\ \boxed{r_o} \tag{51}$$

$$\mathbf{II}_c = \frac{L}{2} \quad \boxed{l_e}\ \boxed{\Lambda^B}\ \boxed{\Gamma^A}\ \boxed{\Lambda^A}\ \boxed{\Gamma^B}\ \boxed{h_{\text{even}}}\ \boxed{\Lambda^B}\ \boxed{\Gamma^{A\dagger}}\ \boxed{\Lambda^A}\ \boxed{\Gamma^{B\dagger}}\ \boxed{\textstyle\sum_{n=0}^{L/2-1}(T_e)^n}\ \boxed{\Lambda^B}\ \boxed{\Gamma^A}\ \boxed{\Lambda^A}\ \boxed{\Gamma^B}\ \boxed{h_{\text{even}}}\ \boxed{\Lambda^B}\ \boxed{\Gamma^{A\dagger}}\ \boxed{\Lambda^A}\ \boxed{\Gamma^{B\dagger}}\ \boxed{r_e} \tag{52}$$

$$\mathbf{III}_a = \frac{L}{2} \quad \boxed{l_e}\ \boxed{\Lambda^B}\ \boxed{\Gamma^A}\ \boxed{\Lambda^A}\ \boxed{\Gamma^B}\ \boxed{\Lambda^B}\ \boxed{\Gamma^A}\ \boxed{h_{\text{odd}}}\ \boxed{h_{\text{even}}}\ \boxed{\Lambda^B}\ \boxed{\Gamma^{A\dagger}}\ \boxed{\Lambda^A}\ \boxed{\Gamma^{B\dagger}}\ \boxed{\Lambda^B}\ \boxed{\Gamma^{A\dagger}}\ \boxed{r_o} \tag{53}$$

$$\mathbf{III}_b = \frac{L}{2} \quad \boxed{l_o}\ \boxed{\Lambda^A}\ \boxed{\Gamma^B}\ \boxed{\Lambda^B}\ \boxed{\Gamma^A}\ \boxed{\Lambda^A}\ \boxed{\Gamma^B}\ \boxed{h_{\text{odd}}}\ \boxed{h_{\text{even}}}\ \boxed{\Lambda^A}\ \boxed{\Gamma^{B\dagger}}\ \boxed{\Lambda^B}\ \boxed{\Gamma^{A\dagger}}\ \boxed{\Lambda^A}\ \boxed{\Gamma^{B\dagger}}\ \boxed{r_e} \tag{54}$$

$$\mathbf{III}_c = \frac{L}{2} \quad \boxed{l_o}\ \boxed{\Lambda^A}\ \boxed{\Gamma^B}\ \boxed{\Lambda^B}\ \boxed{\Gamma^A}\ \boxed{h_{\text{odd}}}\ \boxed{\Lambda^A}\ \boxed{\Gamma^{B\dagger}}\ \boxed{\Lambda^B}\ \boxed{\Gamma^{A\dagger}}\ \boxed{\textstyle\sum_{n=0}^{L/2-1}(T_o)^n}\ \boxed{\Lambda^A}\ \boxed{\Gamma^B}\ \boxed{\Lambda^B}\ \boxed{\Gamma^A}\ \boxed{\Lambda^A}\ \boxed{\Gamma^B}\ \boxed{h_{\text{odd}}}\ \boxed{\Lambda^B}\ \boxed{\Gamma^{B\dagger}}\ \boxed{\Lambda^B}\ \boxed{\Gamma^{A\dagger}}\ \boxed{\Lambda^A}\ \boxed{\Gamma^{B\dagger}}\ \boxed{r_e} \tag{55}$$

$$\mathbf{III}_d = \frac{L}{2} \quad \boxed{l_e}\ \boxed{\Lambda^B}\ \boxed{\Gamma^A}\ \boxed{\Lambda^A}\ \boxed{\Gamma^B}\ \boxed{h_{\text{even}}}\ \boxed{\Lambda^B}\ \boxed{\Gamma^{A\dagger}}\ \boxed{\Lambda^A}\ \boxed{\Gamma^{B\dagger}}\ \boxed{\textstyle\sum_{n=0}^{L/2-1}(T_e)^n}\ \boxed{\Lambda^B}\ \boxed{\Gamma^A}\ \boxed{\Lambda^A}\ \boxed{\Gamma^B}\ \boxed{\Lambda^B}\ \boxed{\Gamma^A}\ \boxed{h_{\text{even}}}\ \boxed{\Lambda^A}\ \boxed{\Gamma^{A\dagger}}\ \boxed{\Lambda^A}\ \boxed{\Gamma^{B\dagger}}\ \boxed{\Lambda^B}\ \boxed{\Gamma^{A\dagger}}\ \boxed{r_o} \tag{56}$$

15

$$\mathbf{IV}_a = \frac{L}{2} \quad [\text{tensor network diagram}] \tag{57}$$

$$\mathbf{IV}_b = \frac{L}{2} \quad [\text{tensor network diagram}] \tag{58}$$

$$\mathbf{IV}_c = \frac{L}{2} \quad [\text{tensor network diagram}] \tag{59}$$

$$\mathbf{IV}_d = \frac{L}{2} \quad [\text{tensor network diagram}] \tag{60}$$

Note that series of the form $\sum_{n=0}^{L/2-1}(T)^n$ diverge as $L \to \infty$ due to having leading eigenvalue 1. However, the divergent contribution can safely be discarded, according to the references, as they correspond to a constant (albeit infinite) offset to the Hamiltonian.

To discard the divergent contribution we regularize the transfer matrix by subtracting the eigenvector corresponding to the leading eigenvalue, $\tilde{T} = T - |0\rangle\langle 0|$, or in diagrammatic notation

$$\tilde{T}_e = T_e - \boxed{r_e}\boxed{l_e} \tag{61}$$

$$\tilde{T}_o = T_o - \boxed{r_o}\boxed{l_o} \tag{62}$$

The contribution from powers of T^n decay fast with increasing n, so for large enough L the sum can be approximated by an infinite geometric series:

$$\sum_{n=0}^{L/2-1}(\tilde{T})^n \approx \sum_{n=0}^{\infty}(\tilde{T})^n = (1 - \tilde{T})^{-1}, \tag{63}$$

where the last inverse can be safely computed due to the regularization.

Finally we can collect the results from each term **I-IV** to obtain the $\langle H^2 \rangle$. Of course, since we are really after the per-site variance, we can leave out all the L prefactors to get instead $\langle (h_{\text{even}} + h_{\text{odd}})^2 \rangle$. Then $\text{Var}(H)/L = \langle (h_{\text{even}} + h_{\text{odd}})^2 \rangle - \langle e \rangle^2$

16

An additional simplification can be made if we work with reduced local Hamiltonians, where we've subtracted the current per-site energy from the diagonal: $\tilde{h}_{\text{even}} = h_{\text{even}} - \langle \Phi_{\text{even}} | h_{\text{even}} | \Phi_{\text{even}} \rangle$ and similarly for $\tilde{h}_{\text{even}}$. Then the same diagrams above will directly give us the per-site variance, i.e. $\text{Var}(H)/L = \langle (\tilde{h}_{\text{even}} + \tilde{h}_{\text{odd}})^2 \rangle$.