Notes on MPO canonical forms and compression

Jan Reimers

March 2022

1 Background

These are technical implementation notes used to develop the `ITensorMPOCompression` module as part of the `ITensor`[2] library. It is well known[4, 1, 3, 9] that naive compression of Hamiltonian MPOs results in numerical instabilities. What happens is that the largest two of the singular values resulting from the SVD factorization of an MPO matrix tend to grow rapidly with lattice size. Indeed they diverge in the thermodynamic limit. The root cause of the problem can be traced to the combination of intensive and extensive degrees of freedom within Hamiltonian operators. The recent paper Local Matrix Product Operators:Canonical Form, Compression, and Control Theory[5] contains a number of important insights for the handling of finite and infinite lattice MPOs. They show that the intensive degrees of freedom can be isolated from the extensive ones. If one only compresses the intensive portions of the Hamiltonian then the divergent singular values are removed from the problem. The Parker et. al. paper[5] contains a number of proofs and detailed derivations. The purpose of this document is not to repeat those important insights, but to simply restate the algorithms, and add any extra detail required for coding MPO compression into a generally useful `ITensor` module.

This document assumes the reader is familiar with concepts presented in first six chapters of legendary Schollwöck review article[6].

1.1 ITensor considerations

The indexing methodology used in the `ITensor` libraries (C++ and Julia) relieves the user from worrying about index order and as a result even factor ordering for matrix multiplication. For example accessing a tensor element `W[i=>3,j=>4]` is identical to `W[j=>4,i=>3]`. So in effect transposing an order 2 `ITensor` is a no-op. It then follows that defining an order 2 `ITensor` as being either upper or lower triangular is also meaningless. We can say it is triangular but not upper or lower.

In order to proceed we need some sort of convention for index ordering. A typical operator-valued matrix, $\hat{W}$, that appears on site $n = 1$ in a 1D MPO lattice will have four indexes with tags that look like:

1. A left link index: *Link,l=0*

2. A right link index: *Link,l=1*

3. An upward pointing site index: *Site,SpinHalf,n=1*

tag text	Interpret ion	Comments
Link, l=[site number]	internal link to neighbouring site	For site n the left link has tag l=\$$(n-1)$, and the right link is l=\$$n$
Site,[site type]*,n*=[site number]	physical/site index for a site	the site type is not interpreted
Link,qx	internal link between Q and R for QR decomposition	Same for RQ, QL and LQ decompositions
Link,u	internal link between U and s for SVD decomposition	
Link,v	internal link between s and V for SVD decomposition	

Table 1: `ITensor` tag conventions assumed in the MPO compression code

4. A primed version of the site index.

1&2 are virtual or internal indices, and 3&4 are the physical indices. When writing code we will think of $\hat{W}$ as an order 2 operator valued matrix (OVM) where the *l=0* index is the row index, and the *l=1* index is for columns. With this definition we can define $\hat{W}$ as upper or lower triangular.

Ideally the `ITensor` index abstractions will allow us to make the vast majority of the code independent of upper/lower and left/right canonical forms. However initially this may not be the case because we are adopting a pragmatic strategy of simply getting something working with a large array of unit tests. After that we can start refactoring to simplify the code while constantly checking the unit test to make sure nothing gets broken in the process. The same argument applies to run time optimization.

1.2 Index tag conventions

The ordering of indices inside an `ITensor` object is arbitrary, and therefore of no use in identifying indices. The code relies solely on tags and tag conventions for identifying and matching indices. The conventions in table 1 are used extensively in the code: `ITensor` follows these conventions when creating site sets and MPOs. If you are creating your own site types or generating your own MPOs then please keep these conventions in mind.

1.3 Definition of regular form

For a matrix operator of internal dimension $D_w \times D'_w = (\chi+2)\times(\chi'+2)$ the paper defines upper regular form as

$$\hat{W}_{upper} = \begin{bmatrix} \hat{\mathbb{I}} & \hat{\boldsymbol{c}} & \hat{d} \\ 0 & \hat{\boldsymbol{A}} & \hat{\boldsymbol{b}} \\ 0 & 0 & \hat{\mathbb{I}} \end{bmatrix} \tag{1}$$

where $\hat{\boldsymbol{c}}$ and $\hat{\boldsymbol{b}}$ are operator valued vectors of length χ and χ'. Parker et. al. do not explicitly state that $\hat{\boldsymbol{A}}$ must be triangular. Hand generated Hamiltonian MPOs are always either upper or lower triangular, but as we shall see SVD compression will not guarantee preservation of the triangular form, but it will preserve regular form.

All algorithms in module require and check for regular form. This is not just an algorithmic requirement. Regular form means that the Hamiltonian operator is local and extensive (energy scales with systems size N), which is required for any physical system.

1.4 MPOs for open and periodic boundary conditions

Just like the MPS structure for open boundary conditions, MPOs are capped at each end with row and column vectors. For periodic boundary conditions, full operator matrices are used throughout since there really are no ends to the tensor chain. In the paper they write the Hamiltonian as follows:

$$\hat{H} = \boldsymbol{l}\hat{W}^1\hat{W}^2\hat{W}^3\cdots\hat{W}^{N-1}\hat{W}^N\boldsymbol{r} \tag{2}$$

where $\hat{W}^1$ and $\hat{W}^N$ are full matrices (OVMs) and for an upper triangular MPO

$$\boldsymbol{l} = \begin{bmatrix}1 & \mathbf{0} & 0\end{bmatrix}$$

$$\boldsymbol{r} = \begin{bmatrix}0 \\ \mathbf{0} \\ 1\end{bmatrix}$$

After compression is finished, the normal open boundary condition structure can be restored by contracting $\boldsymbol{l}\hat{W}^1$ and $\hat{W}^N\boldsymbol{r}$.

1.5 Other conventions

The paper consistently refers to QR decomposition. But for generality we will also need RQ, QL and LQ decompositions. We will refer to all of these as QX decomposition.

2 Finite lattice MPOs

Before compression we must bring the MPO into canonical form.

2.1 Finite lattice canonical form

Similarly to the MPS canonical form, this can be done by a modified QX decomposition.

2.1.1 Block Respecting QR decomposition

This section applies to upper regular form of $\hat{W}$ targeting the left canonical form. The first step is to isolate the so called left V-block of 1:

$$\hat{V}_L = \begin{bmatrix}\hat{\mathbb{I}} & \hat{\boldsymbol{c}} \\ 0 & \hat{\boldsymbol{A}}\end{bmatrix}.$$

In other words the top left corner of $\hat{W}$ in eq. 1. $\hat{V}_L$ has four indices ab are link indices and mn are the physical indices. We must reshape $\hat{V}$and do a QR decomposition:

$$\hat{V}_{ab} \rightarrow V_{ab}^{mn} \rightarrow V_{(mna)b} = \sum_q Q_{(mna)q}R_{qb}$$

and then reshape Q back into an operator valued matrix (OVM)

$$Q_{(mna)q} \rightarrow Q^{mn}_{aq} \rightarrow \hat{Q}_{aq}.$$

By the definition of how QR works, $\hat{Q}$ will have orthogonal columns so that $Q^{\dagger}Q = I$, and it also happens to be upper triangular (as long as $\hat{\boldsymbol{A}}$ was upper triangular) with the same form as $\hat{V}$

$$\hat{Q} = \begin{bmatrix} \hat{\mathbb{I}} & \hat{\boldsymbol{c}}' \\ 0 & \hat{\boldsymbol{A}}' \end{bmatrix}.$$

So it is then straightforward to stuff $\hat{Q}$ back into the upper left corner of $\hat{W}$ Yielding

$$\hat{W}' = \begin{bmatrix} \hat{\mathbb{I}} & \hat{\boldsymbol{c}}' & \hat{d} \\ 0 & \hat{\boldsymbol{A}}' & \hat{\boldsymbol{b}} \\ 0 & 0 & \hat{\mathbb{I}} \end{bmatrix} \tag{3}$$

The scalar R matrix (order 2) is also upper triangular and will have the following form

$$R = \begin{bmatrix} 1 & t \\ 0 & \mathsf{R} \end{bmatrix}$$

which must be expanded into

$$R_{+} = \begin{bmatrix} 1 & t & 0 \\ 0 & \mathsf{R} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

and then transferred to the next site over

$$\hat{W}^{l+1} \rightarrow R^{l}_{+} \hat{W}^{l+1}$$

Most of these steps are straightforward, but the reshaping in particular can be error prone. However `ITensor` takes care of that for us under the hood.

There are a number of details not mentioned above that the code must also handle

- There is gauge freedom between Q and R. We want the gauge where the diagonal of R is positive, which is already supported by `ITensor`.
- In addition we also want the gauge with 1.0 in the top left corner of R. By default the `lapack` routines return R with a $\sqrt{d}$ in corner (d is the dimension of the local Hilbert space). So Q and R must be re-scaled accordingly. As a result Q no longer satisfies $Q^{\dagger}Q = I$, instead after gauge fixing we end up with $Q^{\dagger}Q = dI$
- Often there will be so called zero pivots in $V_{(mna)b}$ which will show up as zero (pseudo zero $< 10^{-14}$ for `Float64`) rows in R. These rows and their corresponding columns in Q can be safely removed. This is called rank revealing QR decomposition. This is not built into `lapack` so we must add this feature to that QX routines in `ITensor`.

That is a summary block respecting QR decomposition. It can be written compactly as

$$\hat{QR}\left(\hat{W}\right) = \hat{Q}_{+} R_{+} = \hat{W}' R_{+}$$

where R_{+} is shown in eq. 4 and

$$\hat{Q}_{+} = \hat{W}' = \begin{bmatrix} \hat{\mathbb{I}} & \hat{\boldsymbol{c}}' & \hat{d} \\ 0 & \hat{\boldsymbol{A}}' & \hat{\boldsymbol{b}} \\ 0 & 0 & \hat{\mathbb{I}} \end{bmatrix} \tag{5}$$

	Upper	Lower
$\hat{W}$	$\begin{bmatrix} \hat{\mathbb{I}} & \hat{\boldsymbol{c}}_U & \hat{d}_U \\ 0 & \hat{\boldsymbol{A}}_U & \hat{\boldsymbol{b}}_U \\ 0 & 0 & \hat{\mathbb{I}} \end{bmatrix}$	$\begin{bmatrix} \hat{\mathbb{I}} & 0 & 0 \\ \hat{\boldsymbol{b}}_L & \hat{\boldsymbol{A}}_L & 0 \\ \hat{d}_L & \hat{\boldsymbol{c}}_L & \hat{\mathbb{I}} \end{bmatrix}$

Table 2: Upper and lower regular forms

V-Blocks	Upper	Lower
Left	$\hat{V}_L = \begin{bmatrix} \hat{\mathbb{I}} & \hat{\boldsymbol{c}}_U \\ 0 & \hat{\boldsymbol{A}}_U \end{bmatrix}$	$\hat{V}_L = \begin{bmatrix} \hat{\boldsymbol{A}}_L & 0 \\ \hat{\boldsymbol{c}}_L & \hat{\mathbb{I}} \end{bmatrix}$
Right	$\hat{V}_R = \begin{bmatrix} \hat{\boldsymbol{A}}_U & \hat{\boldsymbol{b}}_U \\ 0 & \hat{\mathbb{I}} \end{bmatrix}$	$\hat{V}_R = \begin{bmatrix} \hat{\mathbb{I}} & 0 \\ \hat{\boldsymbol{b}}_L & \hat{\boldsymbol{A}}_L \end{bmatrix}$

Table 3: V-blocks for all combinations of Upper/Lower triangular MPOs and Left/Right canonical forms.

2.1.2 Block Respecting QX decomposition

We can generalize all of this to include lower triangular MPOs and right canonical form. Upper and lower regular forms for $\hat{W}$ are shown in table 2 and all four V-blocks are in table 3. The detailed steps for block respecting QX decomposition are written out in table 4. All of the R_+/L_+ matrices are written out in table 5, because it is easy (for me) to mix up where the t's go. And finally we write all the orthogonality conditions that the canonical form satisfies in table 6 where the operator orthogonaly condition is defined as (watch the cancelling daggers!)

$$\sum_{a=0}^{\chi} \left\langle \hat{W}_{ab}^{\dagger}, \hat{W}_{ac} \right\rangle = \frac{\sum_{a=0}^{\chi} Tr\left[\hat{W}_{ab}\hat{W}_{ac}\right]}{Tr\left[\hat{\mathbb{I}}\right]} = \delta_{bc} \tag{6}$$

2.1.3 Canonical sweeps

To bring the whole MPO into left canonical form we simply start at site 1 and sweep to the right, resulting in the following sequence

$$\begin{aligned} \hat{H} &= \boldsymbol{l}\hat{W}^1\hat{W}^2\hat{W}^3\cdots\hat{W}^{N-1}\hat{W}^N\boldsymbol{r} \\ &= \hat{Q}R\left(\boldsymbol{l}\hat{W}^1\right)\hat{W}^2\hat{W}^3\cdots\hat{W}^{N-1}\hat{W}^N\boldsymbol{r} \\ &= \hat{Q}_+^1 R_+^1\hat{W}^2\hat{W}^3\cdots\hat{W}^{N-1}\hat{W}^N\boldsymbol{r} \\ &= \hat{W}_L^1 R_+^1\hat{W}^2\hat{W}^3\cdots\hat{W}^{L-1}\hat{W}^L\boldsymbol{r} \\ &= \hat{W}_L^1\left(R_+^1\hat{W}^2\right)\hat{W}^3\cdots\hat{W}^{N-1}\hat{W}^N\boldsymbol{r} \\ &= \hat{W}_L^1\hat{W}^{2\prime}\hat{W}^3\cdots\hat{W}^{N-1}\hat{W}^N\boldsymbol{r} \\ &= \hat{W}_L^1\hat{Q}R\left(\hat{W}^{2\prime}\right)\hat{W}^3\cdots\hat{W}^{N-1}\hat{W}^N\boldsymbol{r} \end{aligned}$$

	Upper	Lower
Left	1. Extract $\hat{V}^{(i)} = \begin{bmatrix} \hat{\mathbb{I}} & \hat{\boldsymbol{c}}_U \\ 0 & \hat{\boldsymbol{A}}_U \end{bmatrix}$ 2. Reshape $\hat{V}_{ab} \rightarrow V_{ab}^{mn} \rightarrow V_{(mna)b}$ 3. QR decomp. $V_{(mna)b} = \sum_{k=0}^{\chi} Q_{(mna)k} R_{kb}$ 4. Reshape $Q_{(mna)k} \rightarrow \hat{Q}_{ak}$ 5. Transfer $\hat{W}^{(i+1)} \rightarrow R\hat{W}^{(i+1)}$	1. Extract $\hat{V}^{(i)} = \begin{bmatrix} \hat{\boldsymbol{A}}_L & 0 \\ \hat{\boldsymbol{c}}_L & \hat{\mathbb{I}} \end{bmatrix}$ 2. Reshape $\hat{V}_{ab} \rightarrow V_{ab}^{mn} \rightarrow V_{(mna)b}$ 3. QL decomp. $V_{(mna)b} = \sum_{k=1}^{\chi+1} Q_{(mna)k} L_{kb}$ 4. Reshape $Q_{(mna)k} \rightarrow \hat{Q}_{ak}$ 5. Transfer $\hat{W}^{(i+1)} \rightarrow L\hat{W}^{(i+1)}$
Right	1. Extract $\hat{V}^{(i)} = \begin{bmatrix} \hat{\boldsymbol{A}}_U & \hat{\boldsymbol{b}}_U \\ 0 & \hat{\mathbb{I}} \end{bmatrix}$ 2. Reshape $\hat{V}_{ab} \rightarrow V_{ab}^{mn} \rightarrow V_{a(mnb)}$ 3. RQ decomp. $V_{a(mnb)} = \sum_{k=1}^{\chi+1} R_{ak} Q_{k(mnb)}$ 4. Reshape $Q_{k(mnb)} \rightarrow \hat{Q}_{kb}$ 5. Transfer $\hat{W}^{(i-1)} \rightarrow \hat{W}^{(i-1)} R$	1. Extract $\hat{V}^{(i)} = \begin{bmatrix} \hat{\mathbb{I}} & 0 \\ \hat{\boldsymbol{b}}_L & \hat{\boldsymbol{A}}_L \end{bmatrix}$ 2. Reshape $\hat{V}_{ab} \rightarrow V_{ab}^{mn} \rightarrow V_{a(mnb)}$ 3. LQ decomp. $V_{a(mnb)} = \sum_{k=0}^{\chi} L_{ak} Q_{k(mnb)}$ 4. Reshape $Q_{k(mnb)} \rightarrow \hat{Q}_{kb}$ 5. Transfer $\hat{W}^{(i-1)} \rightarrow \hat{W}^{(i-1)} L$

Table 4: Left/Right canonical form algorithms for site i for upper/lower regular forms. R matrices are upper triangular and L matrices are lower triangular.

R_+/L_+	Upper	Lower
Left	$R_+ = \begin{bmatrix} 1 & t & 0 \\ 0 & \mathsf{R} & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$L_+ = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathsf{L} & 0 \\ 0 & t & 1 \end{bmatrix}$
Right	$R_+ = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathsf{R} & t \\ 0 & 0 & 1 \end{bmatrix}$	$L_+ = \begin{bmatrix} 1 & 0 & 0 \\ t & \mathsf{L} & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Table 5: Scalar R_+/L_+ matrices for all combinations of Upper/Lower triangular MPOs and Left/Right canonical forms.

Definition of Canonical	Upper	Lower
Left	Orth. Columns: $\sum_{a=0}^{\chi}\left\langle \hat{W}_{ab},\hat{W}_{ac}\right\rangle=\delta_{bc}$	Orth. Columns: $\sum_{a=1}^{\chi+1}\left\langle \hat{W}_{ab},\hat{W}_{ac}\right\rangle=\delta_{bc}$
Right	Orth. Rows: $\sum_{a=1}^{\chi+1}\left\langle \hat{W}_{ba},\hat{W}_{ca}\right\rangle=\delta_{bc}$	Orth. Rows: $\sum_{a=0}^{\chi}\left\langle \hat{W}_{ba},\hat{W}_{ca}\right\rangle=\delta_{bc}$

Table 6: Orthogonality conditions for all combinations of Upper/Lower triangular MPOs and Left/Right canonical forms.

$$\vdots$$

$$=\hat{W}_L^1\hat{W}_L^2\hat{W}_L^3\cdots\hat{W}_L^{N-1}\hat{W}_L^N$$

Of course we just sweep backwards using RQ or LQ decomposition to get a right canonical form.

2.2 Finite MPO compression

2.2.1 QR version in detail

In this example we will assume the MPO is right canonical, upper triangular, which means we are again doing a sweep from left to right. We start by doing a block respecting QR decomposition on site number 1:

$$\hat{W}^1\rightarrow\hat{Q}_+^1R_+^1$$

where $\hat{Q}_+$ is defined in eq. 5 and R_+ is shown in table 5. We now further decompose the R_+ matrix as follows

$$R_+=\begin{bmatrix}1 & t & 0\\ 0 & \mathsf{R} & 0\\ 0 & 0 & 1\end{bmatrix}=MR'=\begin{bmatrix}1 & 0 & 0\\ 0 & \mathsf{M} & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & t & 0\\ 0 & \mathbb{I} & 0\\ 0 & 0 & 1\end{bmatrix} \tag{7}$$

and do an SVD decomposition on the dimension $\chi\times\chi'$ sub block M. In fact the new symbol M is a little confusing because it is the same thing as the R block inside R_+. However we are trying to follow notation used in ref. [5] which also switches from R to M at this point. Also M is left/right neutral so to speak. We should notice that matrices in eq. 7 appear to be square but in general they will not be square. There are two cases to consider:

1. $\chi\geq\chi'$ in which case M (and M) will also be $(\chi+2)\times(\chi'+2)$ and R' will square $(\chi'+2)\times(\chi'+2)$
2. $\chi<\chi'$ in which case M will now be square $(\chi+2)\times(\chi+2)$ and R' will be rectangular $(\chi+2)\times(\chi'+2)$

Case 1: This case is straightforward because R' will have precisely the structure shown in eq. 7. All it does is make sure we don't lose the t block. Next we carry out an SVD decomposition on M,$\mathsf{M}=\mathsf{UsV}^\dagger$, or equivalently

$$M=\begin{bmatrix}1&0&0\\0&\mathsf{M}&0\\0&0&1\end{bmatrix}=\begin{bmatrix}1&0&0\\0&\mathsf{UsV}^\dagger&0\\0&0&1\end{bmatrix}.$$

This can now be compressed $\mathsf{UsV}^\dagger\sim\tilde{\mathsf{U}}\tilde{\mathsf{s}}\tilde{\mathsf{V}}^\dagger$ by removing the small singular values below some threshold, giving

$$M\sim\begin{bmatrix}1&0&0\\0&\tilde{\mathsf{U}}\tilde{\mathsf{s}}\tilde{\mathsf{V}}^\dagger&0\\0&0&1\end{bmatrix}=\begin{bmatrix}1&0&0\\0&\tilde{\mathsf{U}}&0\\0&0&1\end{bmatrix}\begin{bmatrix}1&0&0\\0&\tilde{\mathsf{s}}\tilde{\mathsf{V}}^\dagger&0\\0&0&1\end{bmatrix}=UT.$$

Where the singular values are absorbed into T. Putting it all together we get

$$\hat{W}=\hat{Q}_+R_+=\hat{Q}_+MR'\sim\hat{Q}_+UTR'$$

At this point we update the $\hat{V}$ block

$$\hat{Q}\begin{bmatrix}1&0\\0&\tilde{\mathsf{U}}\end{bmatrix}\to\hat{V}$$

or equivalently

$$\hat{Q}_+U\to\hat{W}.$$

And we update the neighbouring site

$$\hat{W}^{n+1}\to TR'\hat{W}^{n+1}$$

Since $\hat{Q}_+$ and U are both left canonical so is their product.

This process is conceptually straightforward but it is tricky to code correctly because we are dealing with many different matrix dimensions $\chi\times\chi$, $\chi\times\chi'$, $(\chi+1)\times(\chi+1)$, $(\chi+2)\times(\chi+2)\ldots$ with similar symbols. Fortunately we can easily test that our code maintains gauge invariance by calculating the energy expectation for some random wave function before and after compression, using a very small SVD cutoff, $\epsilon_{SVD}=10^{-14}$. Whether or not the resulting $\hat{W}=\hat{Q}_+U$ is still upper triangular is a bit murky because the SVD algorithm in Lapack has some freedom in laying out U when there are degenerate singular values. In general we should assume that any upper triangular structure we start with will be lost after compression. But the upper regular form will be maintained. In short it is only the $\hat{\boldsymbol{A}}$ block that loses its "triangularity".

Case 2: This is discussed in A. Right now the code just bails out and does not try to compress.

2.3 General MPO compression

Here we show the MPO compression algorithm for all combinations of upper/lower ⊗ left/right regular forms in tables 7 and 8.

2.4 Emperically determined strategy

Based on testing with spin system Hamiltonians the following strategy appears to be the most effective. The surprise here is that orthogonalization with rank reduction is much more effective and efficient than SVD compression. In most cases 3 sweeps of orthogonalization with rank reduction will yield about 98% of the possible reduction of a spin Hamiltonian. If you really want to grab the last ~2% then do 1 or 2 truncation/compression sweeps. Of course we need to build experience with other types of Hamiltonians to see if these insights hold more generally.

Canonical form	Operation	Upper	Lower
Left	QX	$\hat{V} = \hat{Q}R$	$\hat{V} = \hat{Q}L$
	Decompose R_+/L_+	$\begin{bmatrix} 1 & t & 0 \\ 0 & \mathsf{R} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathsf{R} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & t & 0 \\ 0 & \mathbb{I} & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathsf{L} & 0 \\ 0 & t & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathsf{M} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathbb{I} & 0 \\ 0 & t & 1 \end{bmatrix}$
	U/V	$U = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \tilde{\mathsf{U}} & 0 \\ 0 & 0 & 1 \end{bmatrix}, V = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \tilde{\mathsf{s}}\tilde{\mathsf{V}}^\dagger & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$U = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \tilde{\mathsf{U}} & 0 \\ 0 & 0 & 1 \end{bmatrix}, V = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \tilde{\mathsf{s}}\tilde{\mathsf{V}}^\dagger & 0 \\ 0 & 0 & 1 \end{bmatrix}$
	Final V-block	$\hat{Q}U \to \hat{V}$	$\hat{Q}U \to \hat{V}$
	Transfer	$\hat{W}^{(i+1)} \to VR'\hat{W}^{(i+1)}$	$\hat{W}^{(i+1)} \to VL'\hat{W}^{(i+1)}$
Right	QR	$\hat{V} = R\hat{Q}$	$\hat{V} = L\hat{Q}$
		$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathsf{R} & t \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathbb{I} & t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathsf{R} & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ t & \mathsf{L} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ t & \mathbb{I} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathsf{L} & 0 \\ 0 & 0 & 1 \end{bmatrix}$
	U/V	$U = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \tilde{\mathsf{U}}\tilde{s} & 0 \\ 0 & 0 & 1 \end{bmatrix}, V = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \tilde{\mathsf{V}}^\dagger & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$U = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \tilde{\mathsf{U}}\tilde{s} & 0 \\ 0 & 0 & 1 \end{bmatrix}, V = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \tilde{\mathsf{V}}^\dagger & 0 \\ 0 & 0 & 1 \end{bmatrix}$
	Final V-block	$V\hat{Q} \to \hat{V}$	$V\hat{Q} \to \hat{V}$
	Transfer	$\hat{W}^{(i-1)} \to \hat{W}^{(i-1)}R'U$	$\hat{W}^{(i-1)} \to \hat{W}^{(i-1)}L'U$

Table 7: MPO compression for all combinations of upper/lower triangular and left and right orthogonality.

	Upper	Lower
Left	$\begin{bmatrix} 1 & t_2 & t_1 & 0 \\ 0 & \mathsf{R}_2 & \mathsf{M} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathsf{M} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & t_2 & t_1 & 0 \\ 0 & \mathsf{R'}_2 & \mathbb{I} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \mathsf{M} & \mathsf{L}_2 & 0 \\ 0 & t_1 & t_2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathsf{M} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \mathbb{I} & \mathsf{L'}_2 & 0 \\ 0 & t_1 & t_2 & 1 \end{bmatrix}$
Right	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathsf{M} & t_1 \\ 0 & \mathsf{R}_2 & t_2 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathbb{I} & t_1 \\ 0 & \mathsf{R'}_2 & t_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathsf{M} & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ t_2 & \mathsf{L}_2 & 0 \\ t_1 & \mathsf{M} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ t_2 & \mathsf{L'}_2 & 0 \\ t_1 & \mathbb{I} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathsf{M} & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Table 8: Explicit structure of all four $R = MR'$ type factorizations.

3 Infinite lattice MPO compression

A large portion of the literature on infinite MPS and MPO methods tends to focus on models with a one site unit cell. Attention to models or ground states with ,multiple sites per unit tends to be an afterthought. This makes sense as one can explain the essence of a new algorithm with less notational baggage if one repeating site is assumed. However for a software library we must keep everything general and not assume a one site, or uniform MPS or MPO. And even if the MPO is perfectly uniform that does not by any means suggest that the ground state MPS is uniform. Antiferromagnetic models are an obvious example. In order establish the notation for this we write the iMPO Hamiltonian as

$$\hat{H} = \cdots \hat{W}^N \left[\hat{W}^1 \hat{W}^2 \cdots \hat{W}^N\right] \hat{W}^1 \cdots$$

where we have show the repeating unit cell, with N sites, inside the square brackets. We will use superscripts on $\hat{W}$'s to indicate site number inside the unit cell, and subscripts to indicate iteration number.

3.1 QR Orthogonalization

There are three algorithms for this, going from simplest and least reliable to the most complex and most reliable. In this section we describe the iterated QX orthogonalization scheme. This idea was first introduced for MPS networks[8] and then described for 1-site MPOs in [5]. The extension to two site MPOs is described in [7].

We begin by creating a collection of gauge transforms and set them equal to the identity:

$$G_L^1 = G_L^2 \cdots = G_L^N = \mathbb{I}$$

Next we do a block respecting, rank revealing QX decomposition on each site

$$\hat{QX}\left(\hat{W}_0^n\right) \rightarrow \hat{V}_0^n = \hat{Q}_0^n R_0^n$$

and now $\hat{H}$ looks like

$$\hat{H} = \cdots R_0^N \left[\hat{Q}_0^1 R_0^1 \hat{Q}_0^2 R_0^2 \cdots \hat{Q}_0^N R_0^N\right] \hat{Q}_0^1 \cdots$$

But we can demarcate the unit cell any place we want, moving the square brackets on tensor to the left we have

$$= \cdots \left[R_0^N \hat{Q}_0^1 R_0^1 \hat{Q}_0^2 R_0^2 \cdots R_0^{(N-1)} \hat{Q}_0^N\right] \cdots$$

which defines a new set $\hat{W}$'s with iteration index 1

$$\hat{H} = \cdots\cdots \left[\hat{W}_1^1 \hat{W}_1^2 \cdots \hat{W}_1^N\right] \cdots$$

where

$$\hat{W}_1^n = R_0^{(n-1)} \hat{Q}_0^n.$$

In addition we must update the accumulated gauge transforms

$$G_L^n \rightarrow R_0^n G_L^n$$

At this point we can go back the block QX step.

As the iterations proceed the R (or L) matrices will usually converge towards the identity matrix. In addition the rank revealing algorithm will be constantly reducing the dimensions of the W and R (or L) matrices. Convergence is achieved when $\left\|\tilde{R}_i - \mathbb{I}\right\| < \varepsilon$ and there are no more dimension changes. Finally $\hat{W}_\infty = \hat{Q}_\infty$ which we can label as $\hat{W}_L$ if we are doing a left orthogonalization sweep. The full gauge transformation is accumulated in the set of G_L^n matrices. The gauge transforms relate $\hat{W}_L$ and $\hat{W}_0$ as follows

$$\hat{W}_L G_L = G_L \hat{W}_0$$

3.1.1 ITensor indexing

In order to get all of the matrix products in the previous section to execute correctly, we need to be very careful about how we organize all the "Link" indices of the tensors. To begin with the link indices need to obey periodic boundary conditions

$$\hat{H} = \cdots \hat{W}^N \left[\underline{l=N}\,\hat{W}^1\,\underline{l=1}\,\hat{W}^2\,\underline{l=2} \cdots \underline{l=N-1}\,\hat{W}^N \right] \underline{1=N}\,\hat{W}^1 \cdots$$

QX decomposition for left orthogonal form looks like

$$\hat{QX}\left(\underline{l=n-1}\,\hat{W}_0^n\,\underline{l=n} \right) = \underline{l=n-1}\,\hat{Q}_0^n\,\underline{qx}\,R_0^n\,\underline{l=n}$$

and we need to do some tag replacements before the next step:

$$\underline{l=n-1}\,\hat{Q}_0^n\,\underline{qx} \longrightarrow \underline{l=n}\,\hat{Q}_0^n\,\underline{(l=n)'}$$

$$\underline{qx}\,R_0^n\,\underline{l=n} \longrightarrow \underline{(l=n)'}\,R_0^n\,\underline{l=n}$$

Now the $\hat{W}_1^n$ update will look like

$$\hat{W}_1^n = \underline{(l=n-1)'}\,R_0^{n-1}\,\underline{l=n}\,\hat{Q}_0^n\,\underline{(l=n)'}.$$

As for the gauge transforms they need to begin as

$$\underline{l=n}\,G_0^n\,\underline{l=n-1}$$

and then

$$G_1^n = \underline{(l=n)'}\,R_0^n\,\underline{l=n}\,G_0^n\,\underline{l=n-1}$$

After these steps the primes can be removed to get ready for the next iteration.

3.2 SVD Compression

To prepare an iMPO for compression we first left orthogonalize and the right orthogonalize. The gauge transform returned by the last step relates the left and right forms through a daisy chain of relations

$$G^{(n-1)}\hat{W}_R^n = \hat{W}_L^n G^n \quad n = 1 \cdots N$$

with modular, $mod(N)$, arithmetic assumed for n indices. This full gauge transform G is the input for SVD compression. We can now imagine a mixed canonical form Hamiltonian created by introducing G^N at left infinity onto a right orthogonal Hamiltonian, and sweeping towards to right:

$$\begin{aligned}\hat{H} &= G^N \left[\hat{W}_R^1 \hat{W}_R^2 \cdots \hat{W}_R^N\right] \cdots \\ &= \left[\hat{W}_L^1 G^1 \hat{W}_R^2 \cdots \hat{W}_R^N\right] \cdots \\ &= \left[\hat{W}_L^1 \hat{W}_L^2 G^2 \cdots \hat{W}_R^N\right] \cdots \\ &\vdots \\ &= \cdots \hat{W}_L^N\right] \left[\hat{W}_L^1 G^1 \hat{W}_R^2 \cdots \hat{W}_R^N\right] \left[\hat{W}_R^1 \cdots\end{aligned}$$

The gauge transform should have the form below which can be decomposed in a block respecting manner (in order avoid those nasty diverging singular values), and then compressed

$$G^n = \begin{bmatrix}1 & 0 & 0\\ 0 & \mathsf{G}^n & 0 \\ 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\\ 0 & \mathsf{U}^n\mathsf{s}^n\mathsf{V}^n & 0 \\ 0 & 0 & 1\end{bmatrix} \approx \begin{bmatrix}1 & 0 & 0\\ 0 & \tilde{\mathsf{U}}^n\tilde{\mathsf{s}}^n\tilde{\mathsf{V}}^n & 0 \\ 0 & 0 & 1\end{bmatrix} = \tilde{U}^n \tilde{s}^n \tilde{V}^n$$

$$\hat{H}_{trunc} = \cdots \hat{W}_L^N\right] \left[\hat{W}_L^1 \tilde{U}^1 \tilde{s}^1 \tilde{V}^1 \hat{W}_R^2 \cdots \hat{W}_R^N\right] \left[\hat{W}_R^1 \cdots$$

We could also have just as easily chosen the orthogonality center to be between sites 2 and 3.

$$\hat{H}_{trunc} = \cdots \left[\cdots \hat{W}_L^N\right] \left[\hat{W}_L^1 \hat{W}_L^2 \tilde{U}^2 \tilde{s}^2 \tilde{V}^2 \hat{W}_R^3 \cdots \hat{W}_R^N\right] \left[\hat{W}_R^1 \cdots\right] \cdots$$

and so on for each bond in the unit cell. Now we can define

$$\hat{Q}^n = \tilde{U}^{(n-1)\dagger} \hat{W}_L^n \tilde{U}^n$$

$$\hat{P}^n = \tilde{V}^{(n-1)\dagger} \hat{W}_R^n \tilde{V}^n$$

And in terms of these new tensors we have

$$\hat{H} = \cdots \left[\hat{Q}^1 \tilde{s}^1 \hat{P}^2 \cdots \hat{P}^N\right] \cdots = \cdots \left[\hat{Q}^1 \hat{Q}^2 \tilde{s}^2 \cdots \hat{P}^N\right] \cdots$$

where we can chose to put the orthogonality center on any bond in the unit cell by using the gauge relation

$$\tilde{s}^{(n-1)} \hat{P}^n = \hat{Q}^n \tilde{s}^n$$

4 Bibliography

References

[1] Garnet Kin-Lic Chan, Anna Keselman, Naoki Nakatani, Zhendong Li, and Steven R. White. Matrix product operators, matrix product states, and ab initio density matrix renormalization group algorithms. *The Journal of Chemical Physics*, 145(1):014102, jul 2016.

[2] Matthew Fishman, Steven R. White, and E. Miles Stoudenmire. The itensor software library for tensor network calculations, 2020.

[3] C. Hubig, I. P. McCulloch, and U. SchollwÃ¶ck. Generic construction of efficient matrix product operators. *Physical Review B*, 95(3), jan 2017.

[4] L. Michel and I. P. McCulloch. Schur forms of matrix product operators in the infinite limit.

[5] Daniel E. Parker, Xiangyu Cao, and Michael P. Zaletel. Local matrix product operators: Canonical form, compression, and control theory. *Phys. Rev. B 102, 035147 (2020)*, 2019.

[6] Ulrich Schollwoeck. The density-matrix renormalization group in the age of matrix product states. *Annals of Physics 326(1)*, 2010.

[7] Tomohiro Soejima, Daniel E. Parker, Nick Bultinck, Johannes Hauschild, and Michael P. Zaletel. Efficient simulation of moiré materials using the density matrix renormalization group. *Physical Review B*, 102(20), nov 2020.

[8] Laurens Vanderstraeten, Jutho Haegeman, and Frank Verstraete. Tangent-space methods for uniform matrix product states. *SciPost Physics Lecture Notes*, jan 2019.

[9] Michael P. Zaletel, Roger S. K. Mong, Christoph Karrasch, Joel E. Moore, and Frank Pollmann. Time-evolving a matrix product state with long-ranged interactions. *Physical Review B*, 91(16), apr 2015.

A Appendix:

Disclaimer: This issue is not discussed in ref. [5] so I am flying by the seat of my pants here. This case requires some extra work in order to construct the R' matrix. We rewrite eq. 8 as follows

$$R = \begin{bmatrix} 1 & t_2 & t_1 & 0 \\ 0 & \mathsf{R}_2 & \mathsf{M} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = MR' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathsf{M} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & t_2 & t_1 & 0 \\ 0 & \mathsf{R'}_2 & \mathbb{I} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

where M (and M) must be square. We now must solve the linear system

$$\mathsf{R}_2 = \mathsf{M}\mathsf{R'}_2 \tag{9}$$

for $\mathsf{R'}_2$. This requires care because in general M will be close to or even fully singular. Indeed that is the whole point of compressing. So standard linear solvers and/or matrix inversions are going to fail spectacularly! Again SVD comes to the rescue:

$$\mathsf{R'}_2 \approx \tilde{\mathsf{M}}^{-1}\mathsf{R}_2 = \tilde{V}\frac{1}{\tilde{\mathsf{s}}}\tilde{\mathsf{U}}^\dagger \tag{10}$$

This is often referred to as the pseudo inverse. Warning: don't just do $R' \approx \tilde{M}^{-1}R$ because then you will get a very lousy approximation of the unit matrix in the right side of R'.

Notice in eq. 8 that we chose to push M over to the right side instead of positioning it on the left. This is so that we capture as much weight (non zero elements) as possible in M and as little as possible in R_2 (and therefore R'_2). This gives us more to compress in M and should improve the numerical stability when solving eq. 10. I should also note that at the moment this is the weakest part of the whole MPO compression code. When ϵ_{SVD} becomes large, then eq. 9 is poorly satisfied, introducing large gauge errors and ultimately degrading energy expectation values. Perhaps some iterative techniques will be required to improve the accuracy of eq. 9.