Notes on MPO canonical forms and compression

Jan Reimers

March 2023

1 Background

These are technical implementation notes used to develop the `ITensorMPOCompression` module as part of the `ITensor[2]` library. It is well known[5, 1, 4, 10] 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[6] 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[6] 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[7].

1.1 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'. $\hat{\boldsymbol{A}}$ does not need to be triangular. Hand generated Hamiltonian MPOs are generally 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.

1.2 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

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

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

The works directly with $\boldsymbol{l}\hat{W}^1$ and $\hat{W}^N\boldsymbol{r}$ which are column and row vectors.

2 Finite lattice MPOs

Before compression we must bring the MPO into canonical form.

2.1 V-block canonical form

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

2.1.1 V-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)

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

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 triangular (as long as $\hat{\boldsymbol{A}}$ was 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 $\hat{Q}$ 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 $\hat{Q}^\dagger\hat{Q} = dI$ where d is the dimension of the local Hilbert space. So Q and R must be re-scaled accordingly.
- 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 code the remove rows.

That is a summary V-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

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

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 1 and all four V-blocks are in table 2. The detailed steps for block respecting QX decomposition are written out in table 3. All of the R_+/L_+ matrices are written out in table 4, because it is easy (for me) to mix up where the t's go. And

	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 1: 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 2: V-blocks for all combinations of Upper/Lower triangular MPOs and Left/Right canonical forms.

finally we write all the orthogonality conditions that the canonical form satisfies in table 5 where the operator orthogonality condition is defined as (watch the canceling 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

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

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

	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 3: 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 4: 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 5: Orthogonality conditions for all combinations of Upper/Lower triangular MPOs and Left/Right canonical forms.

2.2 Ac-block respecting canonical form

An alternative to the V-block methods described above has been proposed by Loïc Herviou [3]. The idea is focus only the $\hat{\boldsymbol{A}}$ and $\hat{\boldsymbol{c}}$ block of the regular form. This has a number of advantages that will be enumerated below, with a trade-off in the form of slightly increased complexity in the algorithm. In particular we must first gauge transform the MPO prior to performing Ac-block respecting orthogonalization step. It turns out that we will need to do this gauge fix for compression of iMPOs anyway, so the trade-off becomes inconsequential.

2.2.1 Gauge transforms

The general gauge transform for a finite lattice looks like

$$\hat{W}'^{n}L^{n}=L^{n-1}\hat{W}^{n} \tag{7}$$

where $\hat{W}'^{n}$ is the transformed operator. By definition a gauge transform is not supposed affect expectation values of the Hamiltonian. For lower regular form the general form of L that maintains regular form is

$$L=\begin{bmatrix}1 & 0 & 0\\ t & \mathsf{L} & 0\\ r & s & 1\end{bmatrix}.$$

In particular we are interested in gauge transforms the are able to make the $\hat{\boldsymbol{b}}$ and $\hat{\boldsymbol{c}}$ blocks in

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

orthogonal to $\hat{\mathbb{I}}$. For this purpose L looks like

$$L=\begin{bmatrix}1 & \mathbf{0} & 0\\ \boldsymbol{s} & \mathbb{I} & \mathbf{0}\\ 0 & \boldsymbol{t} & 1\end{bmatrix},\quad L^{-1}=\begin{bmatrix}1 & \mathbf{0} & 0\\ -\boldsymbol{s} & \mathbb{I} & \mathbf{0}\\ 0 & -\boldsymbol{t} & 1\end{bmatrix}.$$

This form of L affects the $\hat{\boldsymbol{b}}$, $\hat{\boldsymbol{c}}$ and $\hat{d}$ blocks of $\hat{W}$, but not the $\hat{\boldsymbol{A}}$ block. Doing the matrix algebra in eq. 7 we find

$$\begin{bmatrix}\hat{\mathbb{I}} & \mathbf{0} & 0\\ \hat{\boldsymbol{b}}' & \hat{A} & \mathbf{0}\\ \hat{d}' & \hat{\boldsymbol{c}}' & \hat{\mathbb{I}}\end{bmatrix}\begin{bmatrix}1 & \mathbf{0} & 0\\ \boldsymbol{s}_{n} & \mathbb{I} & \mathbf{0}\\ 0 & \boldsymbol{t}_{n} & 1\end{bmatrix}=\begin{bmatrix}1 & \mathbf{0} & 0\\ \boldsymbol{s}_{n-1} & \mathbb{I} & \mathbf{0}\\ 0 & \boldsymbol{t}_{n-1} & 1\end{bmatrix}\begin{bmatrix}\hat{\mathbb{I}} & \mathbf{0} & 0\\ \hat{\boldsymbol{b}} & \hat{A} & \mathbf{0}\\ \hat{d} & \hat{\boldsymbol{c}} & \hat{\mathbb{I}}\end{bmatrix}$$

$$\begin{bmatrix} \hat{\mathbb{I}} & \mathbf{0} & 0 \\ \hat{\boldsymbol{b}}' + \hat{A}\boldsymbol{s}_n & \hat{A} & \mathbf{0} \\ \hat{d}' + \hat{\boldsymbol{c}}'\boldsymbol{s}_n & \hat{\boldsymbol{c}}' + \hat{\mathbb{I}}\boldsymbol{t}_n & \hat{\mathbb{I}} \end{bmatrix} = \begin{bmatrix} \hat{\mathbb{I}} & \mathbf{0} & 0 \\ \hat{\boldsymbol{b}} + \boldsymbol{s}_{n-1}\hat{\mathbb{I}} & \hat{A} & \mathbf{0} \\ \hat{d} + \boldsymbol{t}_{n-1}\hat{\boldsymbol{b}} & \hat{\boldsymbol{c}} + \boldsymbol{t}_{n-1}\hat{A} & \hat{\mathbb{I}} \end{bmatrix}$$

from which we can read off the relations

$$\hat{\boldsymbol{b}}' = \hat{\boldsymbol{b}} + \boldsymbol{s}_{n-1}\hat{\mathbb{I}} - \hat{A}\boldsymbol{s}_n$$

$$\hat{\boldsymbol{c}}' = \hat{\boldsymbol{c}} - \hat{\mathbb{I}}\boldsymbol{t}_n + \boldsymbol{t}_{n-1}\hat{A}$$

$$\hat{d}' = \hat{d} + \boldsymbol{t}_{n-1}\hat{\boldsymbol{b}} - \hat{\boldsymbol{c}}'\boldsymbol{s}_n$$

2.2.2 Gauge fixing for Ac-block respecting decomposition

The orthogonality conditions are

$$\boldsymbol{b}_0' = \left\langle \hat{\boldsymbol{b}}', \hat{\mathbb{I}} \right\rangle = \frac{tr\left(\hat{\boldsymbol{b}}'\hat{\mathbb{I}}\right)}{d} = \boldsymbol{b}_0 + \boldsymbol{s}_{n-1} - \boldsymbol{s}_n A_0 = 0$$

$$\boldsymbol{c}_0' = \left\langle \hat{\boldsymbol{c}}', \hat{\mathbb{I}} \right\rangle = \frac{tr\left(\hat{\boldsymbol{c}}'\hat{\mathbb{I}}\right)}{d} = \boldsymbol{c}_0 - \boldsymbol{t}_n + \boldsymbol{t}_{n-1}A_0 = 0.$$

We need solve this system of recursion relations for all$\boldsymbol{s}_n$ and $\boldsymbol{t}_n$:

$$A_0\boldsymbol{s}_n = (\boldsymbol{b}_0 + \boldsymbol{s}_{n-1}) \tag{8}$$

$$\boldsymbol{t}_n = \boldsymbol{c}_0 + \boldsymbol{t}_{n-1}A_0 \tag{9}$$

There is no guarantee that A_0 is invertable. So a better strategy is to sweep left to right, calculating the $\boldsymbol{t}_n$'s and clearing out the $\boldsymbol{c}_0'$s and the sweep back right to left, calculating the $\boldsymbol{s}_n$'s and clearing out the $\boldsymbol{b}_0'$s.

2.2.3 Ac-block respecting QR decomposition

We start with the transformed $\hat{W}'$ that satisfies the orthogonality conditions

$$\hat{W}' = \begin{bmatrix} \hat{\mathbb{I}} & \mathbf{0} & 0 \\ \hat{\boldsymbol{b}}' & \hat{\boldsymbol{A}} & \mathbf{0} \\ \hat{d}' & \hat{\boldsymbol{c}}' & \hat{\mathbb{I}} \end{bmatrix}, \quad \boldsymbol{b}_{\mathbf{0}}' = \left\langle \hat{\mathbb{I}}, \hat{\boldsymbol{b}}' \right\rangle = 0, \quad \boldsymbol{c}_{\mathbf{0}}' = \left\langle \hat{\mathbb{I}}, \hat{\boldsymbol{c}}' \right\rangle = 0.$$

Now instead of QL decomposing the whole V-block $V = \begin{bmatrix} \hat{\boldsymbol{A}} & \mathbf{0} \\ \hat{\boldsymbol{c}}' & \hat{\mathbb{I}} \end{bmatrix}$, we only decompose the left columns of V, and we can us QR instead of QL decomposition to make the Independence of triangular form explicit. Hence

$$\begin{bmatrix} \hat{\boldsymbol{A}} \\ \hat{\boldsymbol{c}}' \end{bmatrix} = \hat{Q}\mathsf{R} = \begin{bmatrix} \hat{\boldsymbol{A}}'' \\ \hat{\boldsymbol{c}}'' \end{bmatrix} \mathsf{R}$$

where the bottom row of $\hat{Q}$ has been isolated as $\hat{\boldsymbol{c}}''$. We can now write this as a factoring of V

$$V = \begin{bmatrix} \hat{\boldsymbol{A}} & \mathbf{0} \\ \hat{\boldsymbol{c}}' & \hat{\mathbb{I}} \end{bmatrix} = V = \begin{bmatrix} \hat{\boldsymbol{A}}'' & \mathbf{0} \\ \hat{\boldsymbol{c}}'' & \hat{\mathbb{I}} \end{bmatrix} \begin{bmatrix} \mathsf{R} & 0 \\ 0 & 1 \end{bmatrix}.$$

Now because

$$\left\langle \hat{\mathbb{I}}, \hat{\boldsymbol{c}}' \right\rangle = 0 = \left\langle \hat{\mathbb{I}}, \hat{\boldsymbol{c}}''\mathsf{R} \right\rangle = \left\langle \hat{\mathbb{I}}, \hat{\boldsymbol{c}}'' \right\rangle \mathsf{R}$$

and $R \neq 0$ we conclude that

$$\left\langle \hat{\mathbb{I}}, \hat{\boldsymbol{c}}'' \right\rangle = 0$$

and

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

has orthogonal columns, as desired. The full factorization of $\hat{W}'$ now looks like

$$\hat{W}' = \begin{bmatrix} \hat{\mathbb{I}} & \mathbf{0} & 0 \\ \hat{\boldsymbol{b}} & \hat{\boldsymbol{A}} & \mathbf{0} \\ \hat{d}' & \hat{\boldsymbol{c}}' & \hat{\mathbb{I}} \end{bmatrix} = \begin{bmatrix} \hat{\mathbb{I}} & \mathbf{0} & 0 \\ \hat{\boldsymbol{b}}' & \hat{\boldsymbol{A}}'' & \mathbf{0} \\ \hat{d}' & \hat{\boldsymbol{c}}'' & \hat{\mathbb{I}} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathsf{R} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

When using this method we don't care about the shape of R, which has the following advantages:

1. We can use QR or QL decomposition, which ever is most easily available (usually QR) and convenient.

2. We can use column pivoting QR decomposition: $Ap = QR$, where p is a column permutation matrix. This is the most numerically stable method for handling QR decomposition of singular matrices. Rearranging: $A = QRp^{-1}$ so we see that R gets scrambled by p^{-1} and is non-triangular, which precludes using column pivoting QR decomposition when using the V-block method

3. For efficiency reasons the `ITensor` library does its own sorting and combining of symmetry adapted blocks when using block-sparse tensors. As a result there is no guarantee then tensors will appear triangular after QR decomposition. This is a non issue for Ac-block method, but can cause serious problems when using the V-block method.

2.3 Finite MPO compression

2.3.1 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:

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

where $\hat{Q}_+$ is defined in eq. 5 and R_+ is shown in table 4. We now further decompose the R_+ matrix as follows

$$R_+ = \begin{bmatrix} 1 & t & 0 \\ 0 & \mathsf{R} & 0 \\ 0 & 0 & 1 \end{bmatrix} = MR' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathsf{M} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & t & 0 \\ 0 & \mathbb{I} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{10}$$

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. [6] 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. 10 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. 10. All it does is make sure we don't lose the t block. Next we carry out an SVD decomposition on M,$\mathsf{M} = \mathsf{U}\mathsf{s}\mathsf{V}^\dagger$, or equivalently

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

This can now be compressed $\mathsf{U}\mathsf{s}\mathsf{V}^\dagger \sim \tilde{\mathsf{U}}\tilde{\mathsf{s}}\tilde{\mathsf{V}}^\dagger$ 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

$$\hat{W} = \hat{Q}_+ R_+ = \hat{Q}_+ M R' \sim \hat{Q}_+ U T R'$$

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

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

or equivalently

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

And we update the neighbouring site

$$\hat{W}^{n+1} \rightarrow T R' \hat{W}^{n+1}$$

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, $\epsilon_{SVD} = 10^{-14}$.

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

2.4 General MPO compression

Here we show the MPO compression algorithm for all combinations of upper/lower $\otimes$ left/right regular forms in tables 6 and 7.

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	$U = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \tilde{\mathsf{U}}\tilde{\mathsf{s}} & 0 \\ 0 & 0 & 1 \end{bmatrix}, V = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \tilde{\mathsf{V}}^\dagger & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$U = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \tilde{\mathsf{U}}\tilde{\mathsf{s}} & 0 \\ 0 & 0 & 1 \end{bmatrix}, V = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \tilde{\mathsf{V}}^\dagger & 0 \\ 0 & 0 & 1 \end{bmatrix}$
	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 6: 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 7: Explicit structure of all four $R = MR'$ type factorizations.

3 Infinite lattice MPO compression

In order establish the notation 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 shown 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 Gauge fixing for Ac-block respecting decomposition

Prior to Ac-block respecting decomposition we again need the gauge fix the MPO using the same recusion relations as above (eq. 8 and eq. 9) with two additional contraints

$$s_0 = s_N, \quad t_0 = t_N$$

where N is the size of the repeating unit cell. Because of these constraints we need to solve two systems of equations covering the whole unit cell.

For $N = 1$ we have

$$A_0 \boldsymbol{s}_1 = (\boldsymbol{b}_0 + \boldsymbol{s}_1) \tag{11}$$

$$\boldsymbol{t}_1 = \boldsymbol{c}_0 + \boldsymbol{t}_1 A_0 \tag{12}$$

or

$$(A_0 - \mathbb{I})\, \boldsymbol{s}_1 = \boldsymbol{b}_0$$

$$\boldsymbol{t}_1 (\mathbb{I} - A_0) = \boldsymbol{c}_0$$

which can be solved for $\boldsymbol{s}_1$ and $\boldsymbol{t}_1$. For local Hamiltonians $A_0 - \mathbb{I}$ is guaranteed[6] to be non singular.

For $N = 2$

$$A_0^1 \boldsymbol{s}_1 = \left(\boldsymbol{b}_0^1 + \boldsymbol{s}_2\right)$$

$$A_0^2 \boldsymbol{s}_2 = \left(\boldsymbol{b}_0^2 + \boldsymbol{s}_1\right)$$

$$\boldsymbol{t}_1 = \boldsymbol{c}_0^1 + \boldsymbol{t}_2 A_0^1$$

$$\boldsymbol{t}_2 = \boldsymbol{c}_0^2 + \boldsymbol{t}_1 A_0^2$$

or

$$\begin{bmatrix} A_0^1 & 0 \\ 0 & A_0^2 \end{bmatrix} \begin{bmatrix} \boldsymbol{s}_1 \\ \boldsymbol{s}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_0^1 \\ \boldsymbol{b}_0^2 \end{bmatrix} + \begin{bmatrix} 0 & \mathbb{I} \\ \mathbb{I} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{s}_1 \\ \boldsymbol{s}_2 \end{bmatrix}$$

$$\begin{bmatrix} \mathbb{I} & 0 \\ 0 & \mathbb{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{t}_1 \\ \boldsymbol{t}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{c}_0^1 \\ \boldsymbol{c}_0^2 \end{bmatrix} + \begin{bmatrix} 0 & A_0^1 \\ A_0^2 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{t}_1 \\ \boldsymbol{t}_2 \end{bmatrix}$$

or

$$\begin{bmatrix} A_0^1 & -\mathbb{I} \\ -\mathbb{I} & A_0^2 \end{bmatrix} \begin{bmatrix} \boldsymbol{s}_1 \\ \boldsymbol{s}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_0^1 \\ \boldsymbol{b}_0^2 \end{bmatrix}$$

$$\begin{bmatrix} \mathbb{I} & -A_0^1 \\ A_0^2 & \mathbb{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{t}_1 \\ \boldsymbol{t}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{c}_0^1 \\ \boldsymbol{c}_0^2 \end{bmatrix}$$

For general N:

$$A_0^1 \boldsymbol{s}_1 = \left(\boldsymbol{b}_0^1 + \boldsymbol{s}_N\right)$$

$$A_0^2\boldsymbol{s}_2 = \left(\boldsymbol{b}_0^2 + \boldsymbol{s}_1\right)$$
$$A_0^3\boldsymbol{s}_3 = \left(\boldsymbol{b}_0^3 + \boldsymbol{s}_2\right)$$
$$\vdots$$
$$A_0^N\boldsymbol{s}_N = \left(\boldsymbol{b}_0^N + \boldsymbol{s}_{N-1}\right)$$
$$\boldsymbol{t}_1 = \boldsymbol{c}_0^1 + \boldsymbol{t}_N A_0^1$$
$$\boldsymbol{t}_2 = \boldsymbol{c}_0^2 + \boldsymbol{t}_1 A_0^2$$
$$\boldsymbol{t}_3 = \boldsymbol{c}_0^3 + \boldsymbol{t}_2 A_0^3$$
$$\vdots$$
$$\boldsymbol{t}_N = \boldsymbol{c}_0^N + \boldsymbol{t}_{N-1} A_0^N$$

or

$$\begin{bmatrix} A_0^1 & 0 & 0 & \cdots & 0 \\ 0 & A_0^2 & 0 & \cdots & 0 \\ 0 & 0 & A_0^3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & A_0^N \end{bmatrix} \begin{bmatrix} \boldsymbol{s}_1 \\ \boldsymbol{s}_2 \\ \boldsymbol{s}_3 \\ \vdots \\ \boldsymbol{s}_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_0^1 \\ \boldsymbol{b}_0^2 \\ \boldsymbol{b}_0^3 \\ \vdots \\ \boldsymbol{b}_0^N \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & \mathbb{I} \\ \mathbb{I} & 0 & 0 & \cdots & 0 & 0 \\ 0 & \mathbb{I} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \mathbb{I} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{s}_1 \\ \boldsymbol{s}_2 \\ \boldsymbol{s}_3 \\ \vdots \\ \boldsymbol{s}_N \end{bmatrix}$$

$$\begin{bmatrix} \mathbb{I} & 0 & 0 & \cdots & 0 \\ 0 & \mathbb{I} & 0 & \cdots & 0 \\ 0 & 0 & \mathbb{I} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \mathbb{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{t}_1 \\ \boldsymbol{t}_2 \\ \boldsymbol{t}_3 \\ \vdots \\ \boldsymbol{t}_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{c}_0^1 \\ \boldsymbol{c}_0^2 \\ \boldsymbol{c}_0^3 \\ \vdots \\ \boldsymbol{c}_0^N \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & A_0^1 \\ A_0^2 & 0 & 0 & \cdots & 0 & 0 \\ 0 & A_0^3 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A_0^N & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{t}_1 \\ \boldsymbol{t}_2 \\ \boldsymbol{t}_3 \\ \vdots \\ \boldsymbol{t}_N \end{bmatrix}$$

or

$$\begin{bmatrix} A_0^1 & 0 & 0 & \cdots & 0 & -\mathbb{I} \\ -\mathbb{I} & A_0^2 & 0 & \cdots & 0 & 0 \\ 0 & -\mathbb{I} & A_0^3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -\mathbb{I} & A_0^N \end{bmatrix} \begin{bmatrix} \boldsymbol{s}_1 \\ \boldsymbol{s}_2 \\ \boldsymbol{s}_3 \\ \vdots \\ \boldsymbol{s}_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_0^1 \\ \boldsymbol{b}_0^2 \\ \boldsymbol{b}_0^3 \\ \vdots \\ \boldsymbol{b}_0^N \end{bmatrix}$$

$$\begin{bmatrix} \mathbb{I} & 0 & 0 & \cdots & 0 & -A_0^1 \\ -A_0^2 & \mathbb{I} & 0 & \cdots & 0 & 0 \\ 0 & -A_0^3 & \mathbb{I} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -A_0^N & \mathbb{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{t}_1 \\ \boldsymbol{t}_2 \\ \boldsymbol{t}_3 \\ \vdots \\ \boldsymbol{t}_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{c}_0^1 \\ \boldsymbol{c}_0^2 \\ \boldsymbol{c}_0^3 \\ \vdots \\ \boldsymbol{c}_0^N \end{bmatrix}$$

Obviously we should use a sparse system solver for this problem.

3.2 QR Orthogonalization

In this section we describe the iterated QR orthogonalization scheme. This idea was first introduced for MPS networks[9] and then described for 1-site MPOs in [6]. The extension to two site MPOs is described in [8].

3.2.1 Left Canonical (Right Sweep)

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

$$G_{L0}^{1} = G_{L0}^{2} \cdots = G_{L0}^{N} = \mathbb{I}$$

Next we do a V-block or Ac-block respecting, rank revealing QR decomposition on each site

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

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 one 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

$$G_{L1}^n = R_0^n G_{L0}^n$$

At this point we can go back to the block QR step.

As the iterations proceed the R 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 matrices. Convergence is achieved when $\left\|\tilde{R}_i - \mathbb{I}\right\| < \varepsilon$ and there are no more rank/dimension changes. Finally $\hat{W}_{L\infty} = \hat{Q}_\infty$ which we can label as $\hat{W}_L$ if we are doing a left orthogonalization sweep. The left gauge transformation is accumulated in the set of $G_{L\infty}^n$ matrices. The gauge transforms relate $\hat{W}_{L\infty}$ and $\hat{W}_0$ are as follows

$$G_L^{(n-1)} \hat{W}_0^n = \hat{W}_L^n G_L^n \tag{13}$$

3.2.2 Right Canonical (left sweep)

When we sweep the other direction

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

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

$$= \cdots R_0^1 \left[\hat{W}_1^1 \hat{W}_1^2 \cdots \hat{W}_1^N\right] \hat{Q}_0^1 \cdots$$

where

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

$$G_{R1}^{n-1} = G_{R0}^{n-1} R_0^n$$

$$G_R^n \hat{W}_R^n = \hat{W}_0^n G_R^{(n+1)} \tag{14}$$

3.2.3 Full Gauge Transform

If we left multiply eq. 14 with G_L^{n-1} from eq. 13 we get

$$G_L^{(n-1)} G_R^n \hat{W}_R^n = G_L^{(n-1)} \hat{W}_0^n G_R^{(n+1)} = \hat{W}_L^n G_L^n G_R^{(n+1)}$$

Now we can define the full gauge transform as

$$G^n = G_L^n G_R^{(n+1)}$$

and

$$G^{(n-1)} \hat{W}_R^n = \hat{W}_L^n G^n \; n = 1 \cdots N_{cell} \tag{15}$$

3.3 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 in eq. 15 with modular, $mod\,(N)$, arithmetic assumed for n indices. This full gauge transform G is the input for SVD compression.

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

$$G^n = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathsf{G^n} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathsf{U^n s^n V^n} & 0 \\ 0 & 0 & 1 \end{bmatrix} \approx \begin{bmatrix} 1 & 0 & 0 \\ 0 & \tilde{\mathsf{U}}^{\mathsf{n}} \tilde{\mathsf{s}}^{\mathsf{n}} \tilde{\mathsf{V}}^{\mathsf{n}} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \tilde{U}^n \tilde{s}^n \tilde{V}^n \tag{16}$$

where as usual

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

and similarly for $\tilde{s}^n$ and $\tilde{V}^n$. To understand how this applies to a system Hamiltonian it is actually better to look at the gauge relations. Combining eq. 15 with 16 we have

$$\tilde{U}^{n-1} \tilde{s}^{n-1} \tilde{V}^{n-1} \hat{W}_R^n = \hat{W}_L^n \tilde{U}^n \tilde{s}^n \tilde{V}^n \; n = 1 \cdots N$$

which can rearranged as

$$\tilde{s}^{n-1} \tilde{V}^{n-1} \hat{W}_R^n \tilde{V}^{\dagger n} = \tilde{U}^{\dagger n-1} \hat{W}_L^n \tilde{U}^n \tilde{s}^n \; n = 1 \cdots N$$

$$\tilde{s}^{n-1} \hat{W}_{R'}^n = \hat{W}_{L'}^n \tilde{s}^n \; n = 1 \cdots N_{cell} \tag{17}$$

where

$$\hat{W}_{R'}^n = \tilde{V}^{n-1} \hat{W}_R^n \tilde{V}^{\dagger n}, \quad \hat{W}_{L'}^n = \tilde{U}^{\dagger n-1} \hat{W}_L^n \tilde{U}^n \tag{18}$$

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] Loic Herviou. Some brief notes on my branch of itensors. *Private Communication*, 2023.

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

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

[6] 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.

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

[8] 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.

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

[10] 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. [6] 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. 19 as follows

$$R = \begin{bmatrix} 1 & t_2 & t_1 & 0 \\ 0 & \mathsf{R}_2 & \mathsf{M} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = MR' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathsf{M} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & t_2 & t_1 & 0 \\ 0 & \mathsf{R'}_2 & \mathbb{I} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{19}$$

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

$$\mathsf{R}_2 = \mathsf{M}\mathsf{R'}_2 \tag{20}$$

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:

$$\mathsf{R'}_2 \approx \tilde{\mathsf{M}}^{-1}\mathsf{R}_2 = \tilde{V}\frac{1}{\tilde{\mathsf{s}}}\tilde{\mathsf{U}}^{\dagger}\mathsf{R}_2 \tag{21}$$

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. 19 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. 21. 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. 20 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. 20.