Fermionic Hamiltonian Construction for Matrix Product State Architectures

A Guided Research by Fiona Fröhler supervised by Prof. Dr. Christian Mendl

October 2023

1 Introduction

Efficiently simulating molecules in quantum chemistry is a challenging task that requires finding numerical techniques capable of dealing with strongly correlated and highly entangled electrons. Among many methods, the Density Matrix Renormalization Group (DMRG) algorithm has proven to be useful for simulating strongly correlated quantum systems, including those in theoretical chemistry. The foundation underlying the DMRG is the class of wavefunctions it optimizes, i.e., Matrix Product States (MPS). MPS offer a concise representation of entanglement in one-dimensional systems, making the DMRG very efficient for chain-like molecules. [8]

In this guided research paper, a partitioning method for a molecular Hamiltonian was implemented utilizing the concept of complementary operators. Additionally, this research also provides theoretical foundations for using Matrix Product Operators (MPOs) in Hamiltonian construction, offering both an initial implementation and a framework for future development. These approaches aim to significantly enhance the efficiency of expectation value computations, a critical and expensive component of the DMRG algorithm. Both methods were also extensively tested to ensure their equivalence to the original Hamiltonian. Our study draws inspiration from the work detailed in Nakatani and Chan's paper [8].

2 Background

2.1 Fermionic Operators

Fermionic operators play a fundamental role in quantum mechanics and quantum field theory. They are mathematical objects used to describe the behavior of fermions, which are particles with half-integer spin, such as electrons, protons, and neutrons. Fermionic operators are typically denoted by creation ($a_i^\dagger$) and annihilation (a_i) operators, where i represents a quantum state or mode. These operators satisfy a crucial property known as anticommutation:

$$\{a_i, a_j^\dagger\} = a_i a_j^\dagger + a_j^\dagger a_i = \delta_{ij}, \tag{1}$$

$$\{a_i, a_j\} = \{a_i^\dagger, a_j^\dagger\} = 0, \tag{2}$$

where $\{A, B\}$ denotes the anticommutator of operators A and B, and δ_{ij} is the Kronecker delta, which is equal to 1 if $i = j$ and 0 otherwise. The anticommuting behavior of fermionic operators is a consequence of the Pauli exclusion principle, which states that no two identical fermions can occupy the same quantum state simultaneously. [3]

2.2 Tensor Networks for Molecular Systems

This research is based on the work by Nakatani and Chan in Ref. [8] giving an efficient algorithm for partitioning a Hamiltonian for matrix product states (MPS) in the density matrix renormalization group (DMRG) algorithm for quantum chemistry. The paper then generalizes the algorithm to also work for tree tensor network states (TTNS). However, in this work, we focus on the algorithm given for MPS.

The DMRG algorithm in quantum chemistry is used for optimizing the energy of an MPS wavefunction $|\psi\rangle$ by minimizing the Lagrangian $\langle\psi|\hat{H}|\psi\rangle - \lambda(\langle\psi \mid \psi\rangle - 1)$ w.r.t. the tensors in the MPS. In the one-site DMRG sweep algorithm, this minimization is carried out w.r.t. a single tensor at a time. So, in step x of the DMRG sweep, the coefficient vectors of the MPS at the x-th site are optimized. A detailed explanation of the form of the MPS can be found in Ref. [8].

The most expensive operation in the sweep is performing $\hat{H}|\psi\rangle$ to solve the eigenvalue problem. This is due to the considered Hamiltonian being in second quantization form,

$$\hat{H} = \sum_{ij} t_{ij} \hat{a}_i^\dagger \hat{a}_j + \frac{1}{2} \sum_{ijk\ell} v_{ij\ell k} \hat{a}_i^\dagger \hat{a}_j^\dagger \hat{a}_k \hat{a}_\ell \tag{3}$$

and, hence, containing numerous terms, which makes calculations expensive. As $\hat{H}$ is a molecular Hamiltonian, both the kinetic term coefficient matrix T and the interaction term coefficient matrix V are hermitian and V is also assumed to be symmetric w.r.t. integration variable interchange in two-body coefficients, i.e., $v_{ij\ell k} = v_{jik\ell}$ for all $i, j, l, k \in [1..L]$, where L is the number of sites of the MPS.

There are two generic strategies to handle the large number of terms in the Hamiltonian: *Complementary Operators* and *Matrix Product Operators* (MPOs). The first approach, using complementary operators, has been employed in existing quantum chemistry DMRG implementations as mentioned in Ref. [8]. These operators serve as a powerful strategy for optimizing computational efficiency by maximizing the reuse of intermediate results. Within quantum chemical calculations, many terms involve shared elements, allowing for substantial data overlap and reuse. For instance, consider two terms, $\langle a_1^\dagger a_2^\dagger a_3 a_4 \rangle$ and $\langle a_1^\dagger a_2^\dagger a_5 a_6 \rangle$, both of which contain the same partial expectation value concerning $a_1^\dagger a_2^\dagger$. Complementary operators efficiently exploit this commonality by reusing and combining partial traces, thereby minimizing redundant computations.

The second approach uses the more recent concept of MPOs, which are an intuitive representation of operators in MPS algorithms. The paper states that using the MPO approach would be more costly than using complementary operators, as the sparsity in each tensor resulting from the Hamiltonian decomposition is not considered. However, incorporating the sparsity into the tensors would ruin the simplicity of the approach, which is the main advantage. Thus, they favor using complementary operators for MPS and TTNS computations with quantum chemistry Hamiltonians. However, in a later paper [1] by the same authors, an approach using MPOs to improve the DMRG algorithm is introduced.

In this research, both approaches are considered, starting with the one using complementary operators for partitioning the Hamiltonian. For that at site x in the DMRG sweep, the Hilbert space is partitioned into two subspaces: A, containing the left block of sites $1 \dots x$, and B containing the right block of sites $x+1 \dots L$. Thus, the Hamiltonian at each site can be partitioned as follows:

$$\hat{H} = \hat{H}_A + \hat{H}_{AB} + \hat{H}_B. \tag{4}$$

H_A and H_B denote the Hamiltonians that only contain operators acting on sites in A or, B respectively, whereas H_{AB} contains the interacting terms, i.e., the terms containing indices in both subspaces.

The paper then continues by partitioning the interacting Hamiltonian to obtain complementary operators. However, testing this exact partitioning yielded wrong results, possibly due to an index mistake. Hence, this work uses a slightly different partitioning that is explained in the following. The proof for this can be found in the appendix.

The updated interacting part of the Hamiltonian is then given by

$$\begin{aligned}\hat{H}_{AB} = &\sum_{i\in A, j\in B} (\hat{a}_i^{A\dagger} \hat{S}_i^B + \hat{a}_j^{B\dagger} \hat{S}_j^A) \\ &+ \sum_{ij\in A} (\hat{a}_i^{A\dagger} \hat{a}_j^{A\dagger} \hat{P}_{ij}^B + \hat{a}_i^{A\dagger} \hat{a}_j^{A} \hat{Q}_{ij}^B) + \text{adjoint},\end{aligned} \tag{5}$$

with the updated *Complementary Operators* $\hat{P}_{ij}^B, \hat{Q}_{ij}^B$ and $\hat{S}_i^B$ being

$$\hat{P}_{ij}^B = \frac{1}{2} \sum_{k\ell\in B} v_{ij\ell k} \hat{a}_k \hat{a}_\ell, \tag{6}$$

$$\hat{Q}_{ij}^B = \frac{1}{2} \sum_{k\ell\in B} (v_{ikj\ell} - v_{ik\ell j}) \hat{a}_k^\dagger \hat{a}_\ell, \tag{7}$$

$$\hat{S}_i^B = \frac{1}{2} \sum_{j\in B} t_{ij} \hat{a}_j + \sum_{jk\ell\in B} \frac{1}{2} (v_{ij\ell k} - v_{ji\ell k}) \hat{a}_j^\dagger \hat{a}_k \hat{a}_\ell. \tag{8}$$

3 Contribution of this work

The contribution of this work can be categorized into two significant parts: the first involves the partitioning of the Hamiltonian for MPS using complementary operators, as outlined in Ref. [8], while the second entails obtaining the expectation value via MPO construction. It should be noted that the latter is currently a work in progress.

The code for this research is available in the repository referenced in [2], which consists of three essential files. In the implementation, different data types were employed from the libraries *qib* [7], *pytenet* [5], and *fermitensor* [6]. The `block_partitioning` file is dedicated to the partitioning of the Hamiltonian using complementary operators and yields a `FieldOperator` from the *qib* library. The `block_partitioning_MPO` file facilitates the MPO construction for a simplified Hamiltonian, and additionally provides a framework for creating the MPO for the entire Hamiltonian through the utilization of the `MPO` class from pytenet. Furthermore, the `random_construction` file contains methods for generating random instances of classes necessary for various tests, which are also incorporated in the repository.

3.1 Approach 1: Complementary Operators

This work presents the implementation of the construction of the partitioned Hamiltonian as a `FieldOperator` data type from the *qib* library [7], following the methodology described in the referenced paper [8]. Notably, the implementation includes slight modifications to the partitioning strategy, as described in Sec. 2. The code for this construction is provided within the `block_partitioning` file, accompanied by pseudocode, as depicted in Alg. 1 - 6. In the pseudocode, `CreateOp`(i) and `AnnihilOp`(i) represent the methods creating the fermionic creation and annihilation operators acting on site i in the form of `FieldOperators`. The methods are omitted for conciseness, as they are trivial.

The primary method for obtaining the partitioned Hamiltonian is described in Alg. 1. It requires two arguments: a `MolecularHamiltonian` object from the *qib* library and an integer, denoted as x, indicating the site at which the Hamiltonian should be partitioned. Subsequently, the regions A and B are defined based on the value of x which are then used to create the components of the Hamiltonian. This is done with the two functions in Alg. 2 and 3 passing the coefficient matrices of H, `tkin` the kinetic terms and `vint` the interacting terms, as arguments.

To get H_A and H_B Alg. 2 is employed to extract terms of the Hamiltonian exclusively involving operators acting on indices provided in the argument `sites`,

Algorithm 1 Construct Partitioned Hamiltonian as Field Operator

```
1: procedure ConstructPartitionedHamiltonianFO($H$ : MolecularHamiltonian, $x$ : int)
2:     Input:
3:         $H$ - Molecular Hamiltonian
4:         $x$ - Site where the Hamiltonian is partitioned
5:     Initialize:
6:         $L \leftarrow$ number of lattice sites of H
7:         regionA $\leftarrow$ Range from 0 to $x$
8:         regionB $\leftarrow$ Range from $x$ to $L$
9:     Construct Hamiltonians:
10:        $H_A \leftarrow$ ConstructPartOfHamiltonian(regionA, $H$.tkin, $H$.vint)
11:        $H_B \leftarrow$ ConstructPartOfHamiltonian(regionB, $H$.tkin, $H$.vint)
12:        $H_{AB} \leftarrow$ ConstructInteractingHamiltonian(regionA, regionB, $H$.tkin, $H$.vint)
13:    return $H_A + H_{AB} + H_B$
14: end procedure
```

Algorithm 2 Construct Part of Hamiltonian

```
1: procedure ConstructPartOfHamiltonian(sites, tkin, vint)
2:     compl $\leftarrow$ List of indices not in sites
3:     for $i, j, k, l$ in range($L$) do
4:         if $i \in$ compl or $j \in$ compl then
5:             tkin[$i, j$] $\leftarrow 0$
6:             vint[$i, j, k, l$] $\leftarrow 0$
7:         end if
8:         if $k \in$ compl or $l \in$ compl then
9:             vint[$i, j, k, l$] $\leftarrow 0$
10:        end if
11:    end for
12:    return MolecularHamiltonian(tkin, vint)
13: end procedure
```

Algorithm 3 Construct Interacting Hamiltonian

```
1: procedure ConstructInteractingHamiltonian(regionA, regionB, tkin, vint)
2:     Input:
3:         regionA - Set of indices in A
4:         regionB - Set of indices in B
5:         tkin - Kinetic energy term
6:         vint - Interaction energy term
7:     Construct Complementary Operators:
8:         S_i $\leftarrow$ sum[CreateOp($i$) $\cdot$ ConstructComplementaryS($i$, regionB, tkin, vint) | $i \in$ regionA]
9:         S_j $\leftarrow$ sum[CreateOp($j$) $\cdot$ ConstructComplementaryS($j$, regionA, tkin, vint) | $j \in$ regionB]
10:        P $\leftarrow$ sum[CreateOp($i$) $\cdot$ CreateOp($j$) $\cdot$ ConstructComplementaryP($i, j$, regionB, vint) | $i, j \in$ regionA]
11:        Q $\leftarrow$ sum[CreateOp($i$) $\cdot$ AnnihilOp($j$) $\cdot$ ConstructComplementaryQ($i, j$, regionB, vint) | $i, j \in$ regionA]
12:    HABhalf $\leftarrow$ S_i + S_j + P + Q
13:    return HABhalf + HABhalf.adjoint()
14: end procedure
```

Algorithm 4 Construct Complementary P-Operator according to Eq. 6

```
1: procedure ConstructComplementaryP($i, j$, sites, vint)
2:     $L \leftarrow$ number of lattice sites
3:     eb $\leftarrow$ zero array of length $L$
4:     for $k$ in sites do
5:         eb[$k$] $\leftarrow 1$
6:     end for
7:     return FieldOperator([FieldOperatorTerm([ANNIHIL, ANNIHIL],
               [$0.5 \cdot$ vint[$i, j, l, k$] $\cdot$ eb[$k$] $\cdot$ eb[$l$] for $k, l$ in range($L$)])
8: end procedure
```

Algorithm 5 Construct Complementary Q-Operator according to Eq. 7

```
1: procedure ConstructComplementaryQ($i, j$, sites, vint)
2:     $L \leftarrow$ number of lattice sites
3:     eb $\leftarrow$ zero array of length $L$
4:     for $k$ in sites do
5:         eb[$k$] $\leftarrow 1$
6:     end for
7:     return FieldOperator([FieldOperatorTerm([CREATE, ANNIHIL],
                 [$0.5 \cdot (\text{vint}[i,k,j,l] - \text{vint}[i,k,l,j]) \cdot \text{eb}[k] \cdot \text{eb}[l]$ for $k, l$ in range($L$)])
8: end procedure
```

Algorithm 6 Construct Complementary S-Operator according to Eq. 8

```
1: procedure ConstructComplementaryS($i$, sites, tkin, vint)
2:     $L \leftarrow$ number of lattice sites
3:     eb $\leftarrow$ zero array of length $L$
4:     for $k$ in sites do
5:         eb[$k$] $\leftarrow 1$
6:     end for
7:     skin $\leftarrow$ FieldOperatorTerm([ANNIHIL],
                 [$0.5 \cdot \text{tkin}[i,j] \cdot \text{eb}[j]$ for $j$ in range($L$)]
8:     sint $\leftarrow$ FieldOperatorTerm([CREATE, ANNIHIL, ANNIHIL],
                 [$0.5 \cdot (\text{vint}[i,j,l,k] - \text{vint}[j,i,l,k]) \cdot \text{eb}[j] \cdot \text{eb}[k] \cdot \text{eb}[l]$ for $j, k, l$ in range($L$)]
9:     return FieldOperator([skin, sint])
10: end procedure
```

while preserving the dimensions. This is achieved by setting the coefficients of the elements outside of `sites` to zero.

The interacting part of the Hamiltonian is then created using Alg. 3. The process begins by obtaining all the individual summands, using the remaining methods to create the complementary operators for the given indices and regions. Subsequently, the operators are multiplied with fermionic operators, represented by the `CreateOp(`i`)` and `AnnihilOp(`i`)` methods. Afterward, these components are summed up to form half of the interacting part of the Hamiltonian. In the final step, this half is added with its adjoint counterpart and returned.

The methods for constructing the complementary operators, detailed in Alg. 4-6, follow a consistent structure. Initially, a zero-array, denoted as `eb`, is generated, with a 1 at every index present in the specified region, `sites`. This array serves as a filter to select coefficients corresponding to the indices in `sites` while preserving the correct dimensions. These coefficients, along with the required fermionic operators, are then utilized to create the `FieldOperator` object, which is returned and applied within the previously mentioned methods to construct the complete Hamiltonian.

In the provided test, the partitioned Hamiltonian and the original `MolecularHamiltonian` are compared, utilizing their corresponding matrix representations to verify their equivalence. Extensive testing also first revealed that the Hamiltonian obtained from the initial partitioning was not identical to the reference object, prompting the modification in the partitioning approach. All tests with the current implementation yielded correct results. However, it was observed that the tests also passed when interchanging the indices of the fermionic operators, i.e., the arrays `eb`, in the complementary operators, which is not the intended behavior (or maybe it is). This is yet to be studied further.

3.2 Approach 2: MPOs

Once a functional partitioning of the Hamiltonian was established, the next logical step was to compute the expectation value $\langle\psi|\hat{H}|\psi\rangle$ - the primary goal of this partitioning method in the context of improving the DMRG algorithm. Multiple approaches were explored for calculating the expectation value, including the application of the partitioned Hamiltonian to the MPS $|\psi\rangle$ during the partitioning process to obtain $\hat{H}|\psi\rangle$. However, this approach encountered challenges due to the large number of terms, resulting in numerous additions of MPS and a consequent increase in virtual bond dimensions upon each addition. This significantly lowers computational efficiency due to the growth in both the size and sparsity of matrices, resulting in larger matrices with reduced information density. A general practice following additions is compressing the MPS to a lower bond dimension; nevertheless, this could lead to a loss of information [9]. Consequently, a strategic shift was made toward considering MPOs as a more effective approach.

Creating an MPO for the considered Hamiltonian turned out to be a non-trivial task. Thus, first a simplified version of the Hamiltonian was studied:

$$\hat{H} = \sum_{i \in A} \hat{a}_i^\dagger \hat{S}_i^B \tag{9}$$

with only one complementary operator,

$$\hat{S}_i^B = \frac{1}{2}\sum_{j\in B} t_{ij}\hat{a}_j. \tag{10}$$

Multiple strategies can be employed to derive an MPO from a given Hamiltonian. One effective approach involves employing *Finite State Machines* (FSM) as an intuitive visualization tool. In the following, we will explore the use of an FSM in the construction of the MPO for the simplified Hamiltonian introduced in Eq. 9.

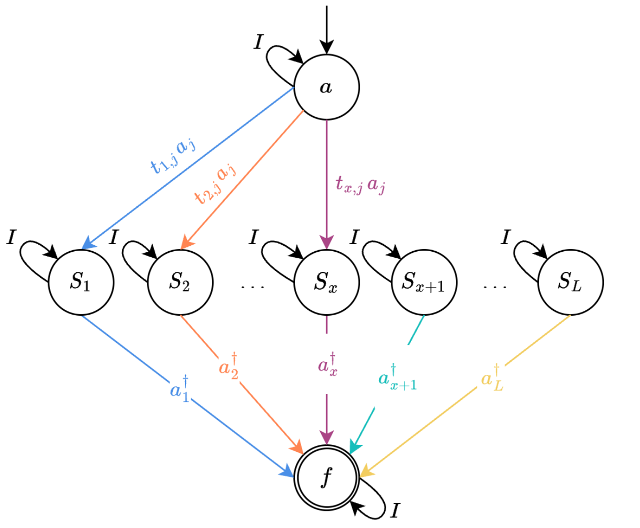

Figure 1: The finite state machine for the MPO creation of the simplified Hamiltonian shown in Eq. 9. The initial state is a and the final state is f.

Figure 1 presents the FSM for the tensors comprising the MPO. Each node within this FSM represents a unique state, and collectively, these FSMs correspond to the individual tensors at site j, denoted as A_j. A valid path through the machine is then a series of consecutive transitions, that originates at state a and ends at state f. It's worth noting that a single FSM with an index j suffices for all tensors, given their uniform structure for $j \in B$. However, tensors associated with indices outside the region B essentially represent identity operators and can be considered trivial in this case. The matrices corresponding to the state machines have dimensions equivalent to the number of states, and describe all possible transitions within them. It is important to note that the first and last matrices, each having one dummy leg, are essentially vectors. The first one at site $j = 1$ takes the form of a row vector, which is identical to the last row of the complete matrix. Conversely, the last matrix at site $j = L - 1$ is a column vector, mirroring the first column of the full matrix.

In this case, each state S_i within the FSM corresponds to a complementary operator $\hat{S}_i^B$. As the index i resides in region A and the regions were previously defined as $A = [1..x]$ and $B = [x+1..L]$, i ranges from 1 to x. However, all states until S_L are retained in the FSM to provide flexibility for adjusting the value of x. Furthermore, when extending this approach to the complete Hamiltonian, it necessitates the inclusion of all potential complementary operators, while also preserving dimensions for future combination of the summands. Since these additional states do not relate to the present simplification, they remain unreachable from the initial state a, rendering them inconsequential to the machine's behavior. Consequently, transitions from these states to the final state are depicted in the diagram but are omitted from the code, as they are superfluous, yielding an equivalent representation. Revisiting the first and last MPO tensors, it becomes apparent that they are effectively represented by the lower and upper halves, respectively, of the same FSM.

The transition table, illustrated in Fig. 2, serves as a valuable tool for visualizing all potential paths within the FSM. Each horizontal line within this table corresponds to a state within the FSM, while each vertical dotted gray line corresponds to a tensor A_j at site j. This table is read from right to left, starting in the upper right corner with the initial state a and concluding in the lower left corner with the final state f. Potential paths are represented by consecutive transitions from a to f. For the sake of clarity, most of the transitions are omitted in Fig. 2. Instead, the table adopts a more focused approach, providing each transition separately on the left side of the diagram to ensure a precise and unambiguous representation. This compact representation is possible because the FSMs are uniform, and every tensor A_j exhibits identical transitions when j resides in region B. These transitions, as shown at A_L and A_{L-1}, repeat at all sites, showing the possible transitions in the respective FSM from one tensor to the next one. Additionally, the horizontal lines should have identity transitions everywhere, representing the possibility of staying in the same state. Tensors with $j < x$ only consist of identity operations and have therefore been omitted in the diagram.

The next step would then be extending the Hamiltonian to

$$\hat{H} = \sum_{i\in A}\hat{a}_i^\dagger \hat{S}_i^B + \sum_{j\in B}\hat{a}_j^\dagger \hat{S}_j^A \tag{11}$$

with the complementary operator given in Eq. 10. It is worth noting that the exact implementation of this extension is beyond the scope of this paper. Nevertheless, in this case, the FSM should maintain its existing states while introducing additional transitions to reach the states S_j with $j \in B$, mirroring the structure of the existing transitions. Furthermore, all matrices will possess a uniform structure, and none of them will be identity operators, thus warranting an extension of the transition table to the left to accommodate these modifications.

The simplified MPO given in Eq. 9 according to the shown diagrams has been successfully implemented in the `block_partitioning_MPO` file. Additionally, placeholders for the full MPO, utilizing the `MPO` class of the *pytenet* library [5], have been included for future implementation. To test the methods creating the

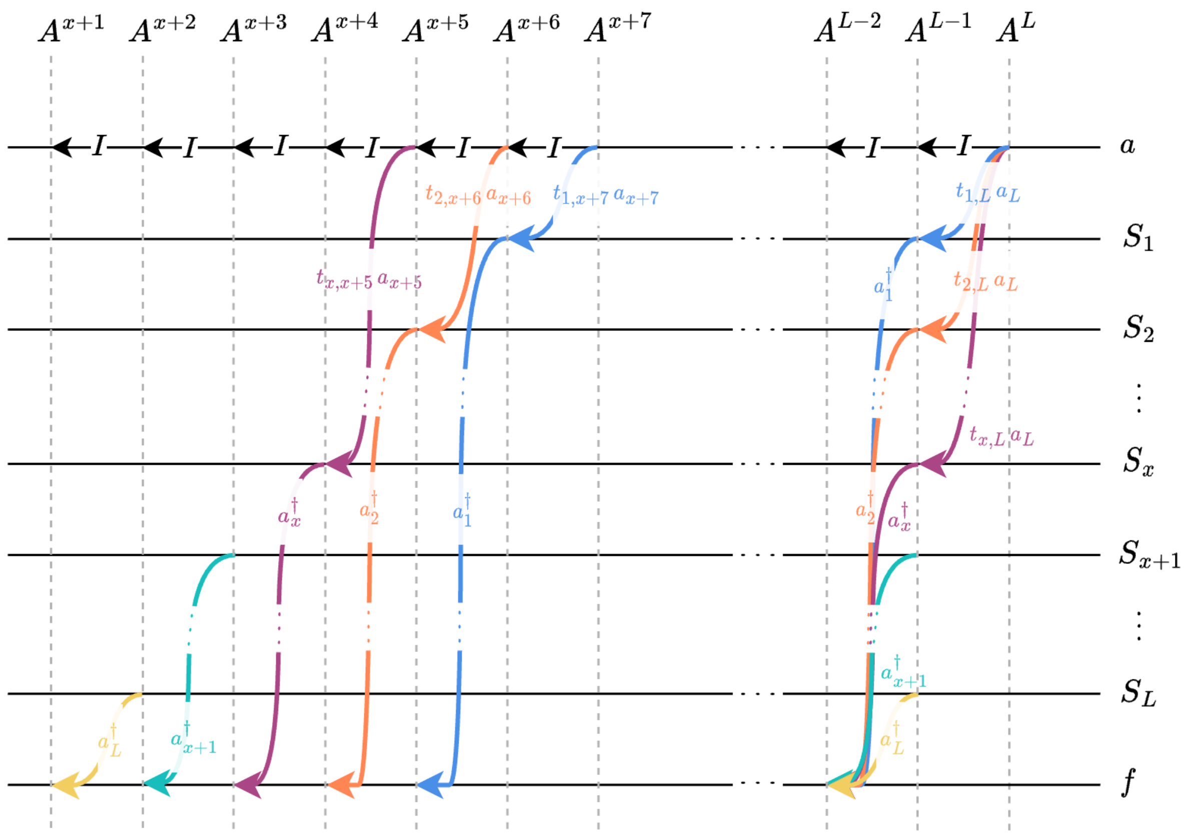

Figure 2: The transition table of the FSM in Fig. 1 corresponding to the MPO of the simplified Hamiltonian, shown in Eq. 9. For the sake of clarity, most transitions are omitted as they repeat at every site $j \in B$.

simplified MPO the `MPS` class of *pytenet* was used for the MPO-MPS contraction process to derive the expectation value, aligning with the intended approach. Furthermore, the repository contains tests designed for computing the expectation value using the partitioned Hamiltonian, which currently employ matrix and vector representations; however, these should be updated with the appropriate methods in subsequent work.

4 Conclusion and Future Work

This work focused on the implementation of a molecular Hamiltonian partitioning technique utilizing complementary operators to aid in solving the eigenvalue problem within the Density Matrix Renormalization Group (DMRG) algorithm for Matrix Product States (MPS). Despite slight modifications to the initial approach [8], comprehensive testing reaffirmed the equivalence of the partitioned Hamiltonian to its pre-partitioned counterpart. The research also presented theoretical foundations for employing Matrix Product Operators (MPOs) in the Hamiltonian construction, offering an initial implementation and a framework for future development. It's worth noting that, as the MPO construction remains a work in progress, a key consideration moving forward will be devising a methodology for the integration of kinetic terms into the MPO structure.

Furthermore, there is still significant room for further optimization and exploration in this research. One interesting concept involves drawing inspiration from the methodology outlined in Ref. [1]. This approach suggests rewriting the Hamiltonian as a sum of sub-Hamiltonians, denoted as $\hat{H}^m$, where each individual term is independently represented by an MPO. This concept builds upon the existing partitioning of the Hamiltonian into two distinct segments and takes it a step further by dividing it through the fixation of a specific index in each sub-Hamiltonian while summing over it. Additionally, another approach for optimization under consideration is the incorporation of quantum number conservation into the algorithm, leveraging the inherent sparsity structure of the Hamiltonian for improved efficiency and accuracy.

As in Ref. [8], an efficient Tree Tensor Network States (TTNS) algorithm was formulated, generalizing the DMRG algorithm for MPS, the natural next step for this research could follow the same path. Tensor Network States (TNS) are mathematical generalizations of MPS that can code more general entanglement networks. The structure of a TNS wavefunction is directly analogous to an MPS wavefunction, but with a tensor instead of a rotation matrix. Tree Tensor Network States (TTNS) are a special class of TNS where the tensors are connected as a tree, leading to many simplifying mathematical properties. The TTNS algorithm significantly reduces computational costs when compared to MPS calculations by utilizing a half-renormalization transformation. This transformation effectively maps the multi-block DMRG algorithm to a conventional and efficient two-block DMRG algorithm. In practice, TTNS calculations have demonstrated enhanced efficiency compared to MPS calculations, particularly in cases involving tree-like molecular structures. This efficiency stems from the fact that TTNS generally require a substantially smaller number of renormalized states to achieve the same level of accuracy as MPS. Nevertheless, whether this reduction translates into substantial computational savings depends on the specific characteristics of the system under consideration [8]. As a result, the precise advantages of TTNS over MPS remain a topic of ongoing exploration and investigation [4].

Acknowledgements

Special thanks to my professor and supervisor Prof. Dr. Christian Mendl for his invaluable explanations and feedback. I would also like to thank Cristian Emiliano Godinez Ramirez, Emily Haworth and Kathrin Kühne for their patient proofreading and the emotional and nutritional support.

Appendix: Proof of the Hamiltonian Partitioning

The Hamiltonian considered in this work is second-quantized and, thus, contains numerous terms.

$$\hat{H} = \sum_{ij} t_{ij} \hat{a}_i^\dagger \hat{a}_j + \frac{1}{2} \sum_{ijk\ell} v_{ij\ell k} \hat{a}_i^\dagger \hat{a}_j^\dagger \hat{a}_k \hat{a}_\ell \tag{12}$$

Both the coefficient matrices T and V are hermitian and V is also assumed to be symmetric w.r.t. integration variable interchange in two-body coefficients, i.e., $v_{ij\ell k} = v_{jik\ell}$ for all $i, j, l, k \in [1..L]$, where L is the number of sites of the MPS. To handle this large number of terms, existing quantum chemistry DMRG implementations partition the Hamiltonian into complementary operators to reuse as many terms as possible. As mentioned in Sec. 2 this work uses a slightly different partitioning from Ref. [8]. The proof that this partitioning is valid follows.

Proof. First, the Hilbert space is partitioned into two subspaces, A and B, that are used to partition the Hamiltonian accordingly:

$$\hat{H} = \hat{H}_A + \hat{H}_{AB} + \hat{H}_B. \tag{13}$$

If all the indices in a term are in A then the term is considered in $\hat{H}_A$, analogously for B and $\hat{H}_B$.

However, if some are in A and some in B then the term is in $\hat{H}_{AB}$. Here, two different cases for the kinetic terms ($i \in A, j \in B$ and vice versa) and three different cases for the interaction terms are considered. The first case is one index being in A and the rest in B. In the second case, two indices are in A and two in B, and in the third case A contains three of the indices. Hence, the interacting part of the Hamiltonian would consist of these terms:

$$\hat{H}_{AB} = \sum_{i\in A, j\in B} t_{ij} \hat{a}_i^\dagger \hat{a}_j + t_{ji} \hat{a}_j^\dagger \hat{a}_i$$

Case 1 - one index in A and three in B:

$$+ \frac{1}{2} \Big(\sum_{i\in A, jk\ell\in B} v_{ij\ell k} \hat{a}_i^\dagger \hat{a}_j^\dagger \hat{a}_k \hat{a}_\ell + \sum_{j\in A, ik\ell\in B} \cdots$$
$$+ \sum_{\ell\in A, ijk\in B} \cdots + \sum_{k\in A, ij\ell\in B} \cdots \Big)$$

Case 2 - two indices in A and two in B:

$$+ \frac{1}{2} \Big(\sum_{ij\in A, k\ell\in B} v_{ij\ell k} \hat{a}_i^\dagger \hat{a}_j^\dagger \hat{a}_k \hat{a}_\ell + \sum_{ik\in A, j\ell\in B} \cdots$$
$$+ \sum_{i\ell\in A, jk\in B} \cdots + \sum_{jk\in A, i\ell\in B} \cdots$$
$$+ \sum_{j\ell\in A, ik\in B} \cdots + \sum_{k\ell\in A, ij\in B} \cdots \Big)$$

Case 3 - three indices in A and one in B:

$$+ \frac{1}{2} \Big(\sum_{jk\ell\in A, i\in B} v_{ij\ell k} \hat{a}_i^\dagger \hat{a}_j^\dagger \hat{a}_k \hat{a}_\ell + \sum_{ik\ell\in A, j\in B} \cdots$$
$$+ \sum_{ijk\in A, \ell\in B} \cdots + \sum_{ij\ell\in A, k\in B} \cdots \Big) \tag{14}$$

As the terms are repetitive, $v_{ij\ell k} \hat{a}_i^\dagger \hat{a}_j^\dagger \hat{a}_k \hat{a}_\ell$ is often replaced by $\ldots$ to increase readability.

The terms are then rearranged to obtain the complementary operators. By splitting some sums in half and reordering the terms, we get these two columns of sums:

$$\begin{aligned}
\hat{H}_{AB} = & \frac{1}{2} \sum_{i\in A, j\in B} t_{ij} \hat{a}_i^\dagger \hat{a}_j + t_{ji} \hat{a}_j^\dagger \hat{a}_i \\
& + \frac{1}{2} \sum_{i\in A, j\in B} t_{ij} \hat{a}_i^\dagger \hat{a}_j + t_{ji} \hat{a}_j^\dagger \hat{a}_i \\
& + \frac{1}{2} \sum_{i\in A, jk\ell\in B} v_{ij\ell k} \hat{a}_i^\dagger \hat{a}_j^\dagger \hat{a}_k \hat{a}_\ell + \frac{1}{2} \sum_{\ell\in A, ijk\in B} \cdots \\
& + \frac{1}{2} \sum_{j\in A, ik\ell\in B} \cdots + \frac{1}{2} \sum_{k\in A, ij\ell\in B} \cdots \\
& + \frac{1}{2} \sum_{ij\in A, k\ell\in B} \cdots + \frac{1}{2} \sum_{k\ell\in A, ij\in B} \cdots \\
& + \frac{1}{4} \sum_{i\ell\in A, jk\in B} \cdots + \frac{1}{4} \sum_{i\ell\in A, jk\in B} \cdots \\
& + \frac{1}{4} \sum_{jk\in A, i\ell\in B} \cdots + \frac{1}{4} \sum_{jk\in A, i\ell\in B} \cdots \\
& + \frac{1}{2} \sum_{ik\in A, j\ell\in B} \cdots + \frac{1}{2} \sum_{j\ell\in A, ik\in B} \cdots \\
& + \frac{1}{2} \sum_{jk\ell\in A, i\in B} \cdots + \frac{1}{2} \sum_{ijk\in A, \ell\in B} \cdots \\
& + \frac{1}{2} \sum_{ik\ell\in A, j\in B} \cdots + \frac{1}{2} \sum_{ij\ell\in A, k\in B} \cdots
\end{aligned} \tag{15}$$

The indices of the sums on the right of each line are renamed to make it more evident that they are the adjoint of their respective counterparts on the left side.

$$
\begin{aligned}
\hat{H}_{AB} = &\\
&\frac{1}{2}\sum_{i\in A, j\in B} t_{ij}\hat{a}_i^\dagger\hat{a}_j + t_{ji}\hat{a}_j^\dagger\hat{a}_i + \frac{1}{2}\sum_{i\in A, j\in B} t_{ji}\hat{a}_j^\dagger\hat{a}_i + t_{ij}\hat{a}_i^\dagger\hat{a}_j\\
&+\frac{1}{2}\sum_{i\in A, jk\ell\in B} v_{ij\ell k}\hat{a}_i^\dagger\hat{a}_j^\dagger\hat{a}_k\hat{a}_\ell + \frac{1}{2}\sum_{i\in A, jk\ell\in B} v_{\ell kij}\hat{a}_\ell^\dagger\hat{a}_k^\dagger\hat{a}_j\hat{a}_i\\
&+\frac{1}{2}\sum_{j\in A, ik\ell\in B}\dots + \frac{1}{2}\sum_{j\in A, ik\ell\in B} v_{\ell kij}\hat{a}_\ell^\dagger\hat{a}_k^\dagger\hat{a}_j\hat{a}_i\\
&+\frac{1}{2}\sum_{ij\in A, k\ell\in B}\dots + \frac{1}{2}\sum_{ij\in A, k\ell\in B} v_{\ell kij}\hat{a}_\ell^\dagger\hat{a}_k^\dagger\hat{a}_j\hat{a}_i\\
&+\frac{1}{4}\sum_{i\ell\in A, jk\in B}\dots + \frac{1}{4}\sum_{i\ell\in A, jk\in B} v_{\ell kij}\hat{a}_\ell^\dagger\hat{a}_k^\dagger\hat{a}_j\hat{a}_i\\
&+\frac{1}{4}\sum_{jk\in A, i\ell\in B}\dots + \frac{1}{4}\sum_{jk\in A, i\ell\in B} v_{\ell kij}\hat{a}_\ell^\dagger\hat{a}_k^\dagger\hat{a}_j\hat{a}_i\\
&+\frac{1}{2}\sum_{ik\in A, j\ell\in B}\dots + \frac{1}{2}\sum_{ik\in A, j\ell\in B} v_{\ell kij}\hat{a}_\ell^\dagger\hat{a}_k^\dagger\hat{a}_j\hat{a}_i\\
&+\frac{1}{2}\sum_{jk\ell\in A, i\in B}\dots + \frac{1}{2}\sum_{jk\ell\in A, i\in B} v_{\ell kij}\hat{a}_\ell^\dagger\hat{a}_k^\dagger\hat{a}_j\hat{a}_i\\
&+\frac{1}{2}\sum_{ik\ell\in A, j\in B}\dots + \frac{1}{2}\sum_{ik\ell\in A, j\in B} v_{\ell kij}\hat{a}_\ell^\dagger\hat{a}_k^\dagger\hat{a}_j\hat{a}_i
\end{aligned}
\tag{16}
$$

In this next step, all the terms on the right are summarized as the adjoint of the rest, and the sums are reordered in preparation for further steps.

$$
\begin{aligned}
\hat{H}_{AB} = &\frac{1}{2}\sum_{i\in A, j\in B} t_{ij}\hat{a}_i^\dagger\hat{a}_j + t_{ji}\hat{a}_j^\dagger\hat{a}_i\\
&+\frac{1}{2}\sum_{i\in A, jk\ell\in B} v_{ij\ell k}\hat{a}_i^\dagger\hat{a}_j^\dagger\hat{a}_k\hat{a}_\ell + \frac{1}{2}\sum_{j\in A, ik\ell\in B}\dots\\
&+\frac{1}{2}\sum_{jk\ell\in A, i\in B}\dots + \frac{1}{2}\sum_{ik\ell\in A, j\in B}\dots\\
&+\frac{1}{2}\sum_{ij\in A, k\ell\in B}\dots + \frac{1}{4}\sum_{i\ell\in A, jk\in B}\dots\\
&+\frac{1}{2}\sum_{ik\in A, j\ell\in B}\dots + \frac{1}{4}\sum_{jk\in A, i\ell\in B}\dots\\
&+\text{adjoint}
\end{aligned}
\tag{17}
$$

By renaming the indices of the terms on the right side and switching operators, some minus signs are introduced because of the anti-commuting property of fermionic operators as shown in Eq. 1 and 2.

$$
\begin{aligned}
\hat{H}_{AB} = &\frac{1}{2}\sum_{i\in A, j\in B} t_{ij}\hat{a}_i^\dagger\hat{a}_j + t_{ji}\hat{a}_j^\dagger\hat{a}_i\\
&+\frac{1}{2}\sum_{i\in A, jk\ell\in B}\dots - \frac{1}{2}\sum_{i\in A, jk\ell\in B} v_{ji\ell k}\hat{a}_i^\dagger\hat{a}_j^\dagger\hat{a}_k\hat{a}_\ell\\
&+\frac{1}{2}\sum_{jk\ell\in A, i\in B}\dots - \frac{1}{2}\sum_{jk\ell\in A, i\in B} v_{ji\ell k}\hat{a}_i^\dagger\hat{a}_j^\dagger\hat{a}_k\hat{a}_\ell\\
&+\frac{1}{2}\sum_{ij\in A, k\ell\in B}\dots - \frac{1}{4}\sum_{ik\in A, j\ell\in B} v_{ijk\ell}\hat{a}_i^\dagger\hat{a}_j^\dagger\hat{a}_k\hat{a}_\ell\\
&+\frac{1}{2}\sum_{ik\in A, j\ell\in B}\dots - \frac{1}{4}\sum_{ik\in A, j\ell\in B} v_{ji\ell k}\hat{a}_i^\dagger\hat{a}_j^\dagger\hat{a}_k\hat{a}_\ell\\
&+\text{adjoint}
\end{aligned}
\tag{18}
$$

Naturally, we now contract the sums that run over the same indices and have the same factor, and additionally use the symmetric properties of V.

$$
\begin{aligned}
\hat{H}_{AB} = &\frac{1}{2}\sum_{i\in A, j\in B} t_{ij}\hat{a}_i^\dagger\hat{a}_j + t_{ji}\hat{a}_j^\dagger\hat{a}_i\\
&+\frac{1}{2}\sum_{i\in A, jk\ell\in B} (v_{ij\ell k} - v_{ji\ell k})\hat{a}_i^\dagger\hat{a}_j^\dagger\hat{a}_k\hat{a}_\ell\\
&+\frac{1}{2}\sum_{jk\ell\in A, i\in B} (v_{ij\ell k} - v_{ji\ell k})\hat{a}_i^\dagger\hat{a}_j^\dagger\hat{a}_k\hat{a}_\ell\\
&+\frac{1}{2}\sum_{ij\in A, k\ell\in B} v_{ij\ell k}\hat{a}_i^\dagger\hat{a}_j^\dagger\hat{a}_k\hat{a}_\ell\\
&+\frac{1}{2}\sum_{ik\in A, j\ell\in B} (v_{ij\ell k} - v_{ijk\ell})\hat{a}_i^\dagger\hat{a}_j^\dagger\hat{a}_k\hat{a}_\ell\\
&+\text{adjoint}
\end{aligned}
\tag{19}
$$

We now regroup the terms and rename some indices again to subsequently exclude the operators of each term that will be reused for efficiency to finally obtain the complementary operators.

$$
\begin{aligned}
\hat{H}_{AB} = &\frac{1}{2}\sum_{i\in A, j\in B} t_{ij}\hat{a}_i^\dagger\hat{a}_j\\
&+\frac{1}{2}\sum_{i\in A, jk\ell\in B} (v_{ij\ell k} - v_{ji\ell k})\hat{a}_i^\dagger\hat{a}_j^\dagger\hat{a}_k\hat{a}_\ell\\
&+\frac{1}{2}\sum_{i\in A, j\in B} t_{ji}\hat{a}_j^\dagger\hat{a}_i\\
&+\frac{1}{2}\sum_{ik\ell\in A, j\in B} (v_{ji\ell k} - v_{ij\ell k})\hat{a}_j^\dagger\hat{a}_i^\dagger\hat{a}_k\hat{a}_\ell\\
&+\frac{1}{2}\sum_{ij\in A, k\ell\in B} v_{ij\ell k}\hat{a}_i^\dagger\hat{a}_j^\dagger\hat{a}_k\hat{a}_\ell\\
&+\frac{1}{2}\sum_{ij\in A, k\ell\in B} (v_{ikj\ell} - v_{ik\ell j})\hat{a}_i^\dagger\hat{a}_j\hat{a}_k^\dagger\hat{a}_\ell\\
&+\text{adjoint}
\end{aligned}
\tag{20}
$$

$$\hat{H}_{AB} = \sum_{i\in A, j\in B}\Bigg[\hat{a}_i^\dagger\left(\frac{1}{2}\sum_{j\in B} t_{ij}\hat{a}_j + \frac{1}{2}\sum_{jk\ell\in B}(v_{ij\ell k} - v_{ji\ell k})\hat{a}_j^\dagger\hat{a}_k\hat{a}_\ell\right) + \hat{a}_j^\dagger\left(\frac{1}{2}\sum_{i\in A} t_{ji}\hat{a}_i + \frac{1}{2}\sum_{ik\ell\in A}(v_{ji\ell k} - v_{ij\ell k})\hat{a}_i^\dagger\hat{a}_k\hat{a}_\ell\right)\Bigg] + \sum_{ij\in A}\Bigg[\hat{a}_i^\dagger\hat{a}_j^\dagger\left(\frac{1}{2}\sum_{k\ell\in B} v_{ij\ell k}\hat{a}_k\hat{a}_\ell\right) + \hat{a}_i^\dagger\hat{a}_j\left(\frac{1}{2}\sum_{k\ell\in B}(v_{ikj\ell} - v_{ik\ell j})\hat{a}_k^\dagger\hat{a}_\ell\right)\Bigg] + \text{adjoint} \tag{21}$$

The resulting Hamiltonian would then be

$$\hat{H}_{AB} = \sum_{i\in A, j\in B}(\hat{a}_i^\dagger\hat{S}_i^B + \hat{a}_j^\dagger\hat{S}_j^A) + \sum_{ij\in A}(\hat{a}_i^\dagger\hat{a}_j^\dagger\hat{P}_{ij}^B + \hat{a}_i^\dagger\hat{a}_j\hat{Q}_{ij}^B) + \text{adjoint}, \tag{22}$$

with the updated complementary operators $\hat{P}_{ij}^B, \hat{Q}_{ij}^B$ and $\hat{S}_i^B$ being:

$$\hat{P}_{ij}^B = \frac{1}{2}\sum_{k\ell\in B} v_{ij\ell k}\hat{a}_k\hat{a}_\ell, \tag{23}$$

$$\hat{Q}_{ij}^B = \frac{1}{2}\sum_{k\ell\in B}(v_{ikj\ell} - v_{ik\ell j})\hat{a}_k^\dagger\hat{a}_\ell, \tag{24}$$

$$\hat{S}_i^B = \frac{1}{2}\sum_{j\in B} t_{ij}\hat{a}_j + \frac{1}{2}\sum_{jk\ell\in B}(v_{ij\ell k} - v_{ji\ell k})\hat{a}_j^\dagger\hat{a}_k\hat{a}_\ell, \tag{25}$$

□

References

[1] Chan, G.K.L., Keselman, A., Nakatani, N., Li, Z., White, S.R.: Matrix product operators, matrix product states, and ab initio density matrix renormalization group algorithms. The Journal of Chemical Physics **145**(1), 014102 (07 2016), https://doi.org/10.1063/1.4955108

[2] Fröhler, F.: hamiltrees: Fermionic hamiltonian construction for tree tensor network architectures. https://github.com/fiyoooo/hamiltrees/ (2023)

[3] Helgaker, T., Jørgensen, P., Olsen, J.: Molecular Electronic Structure Theory. John Wiley & Sons, LTD, Chichester (2000)

[4] Larsson, H.R.: Computing vibrational eigenstates with tree tensor network states (ttns). The Journal of chemical physics **151**(20) (2019)

[5] Mendl, C.B.: Pytenet: A concise python implementation of quantum tensor network algorithms. Journal of Open Source Software **3**(30), 948 (2018), https://doi.org/10.21105/joss.00948

[6] Mendl, C.B., Haworth, E.M.: fermitensor: Fermionic tensor networks. https://github.com/cmendl/fermionic_tensor_networks/ (2023)

[7] Mendl, C.B., Nibbi, M., Huang, K.: qib - quantum library. https://github.com/qc-tum/qib/ (2023)

[8] Nakatani, N., Chan, G.K.L.: Efficient tree tensor network states (TTNS) for quantum chemistry: Generalizations of the density matrix renormalization group algorithm. The Journal of Chemical Physics **138**(13), 134113 (04 2013), https://doi.org/10.1063/1.4798639

[9] Schollwöck, U.: The density-matrix renormalization group in the age of matrix product states. Annals of Physics **326**(1), 96–192 (jan 2011), https://doi.org/10.1016%2Fj.aop.2010.09.012