TeMFpy: a Python library for converting fermionic mean-field states into tensor networks

Simon H. Hille*, Attila Szabó†

Department of Physics, University of Zürich, Zürich, Switzerland

* simon.hille@physik.uzh.ch , † attila.szabo@physik.uzh.ch

Abstract

We introduce TeMFpy, a Python library for converting fermionic mean-field states to finite or infinite matrix product state (MPS) form. TeMFpy includes new, efficient, and easy-to-understand algorithms for both Slater determinants and Pfaffian states. Together with Gutzwiller projection, these also allow the user to build variational wave functions for various strongly correlated electron systems, such as quantum spin liquids. We present all implemented algorithms in detail and describe how they can be accessed through TeMFpy, including full example workflows. TeMFpy is built on top of TeNPy and, therefore, integrates seamlessly with existing MPS-based algorithms.

Received Date
Accepted Date
Published Date

Contents

1 Introduction

Fermionic mean-field (or Gaussian [1]) states are a class of variational wave functions of central importance. All eigenstates of non-interacting fermionic Hamiltonians are Gaussian. As such, they form the starting point of such mean-field theories as Hartree–Fock theory and the BCS theory of superconductivity, which often provide an excellent account of strongly interacting quantum systems as well. Furthermore, "dressing" mean-field states with, e.g., Gutzwiller and Jastrow factors gives rise to even more powerful variational ansätze, which are widely deployed in variational quantum Monte Carlo (VMC) [2]. Through parton constructions and Gutzwiller projection, fermionic mean-field states also form the basis of the most widely used variational ansätze for quantum spin liquids [2–5].

Mean-field states are, however, a special class of wave function and cannot fully account for interaction-induced phenomena. A more general approach is offered by tensor networks, which can represent mean-field and interacting quantum states equally well, with an expressive power systematically improvable through the number of variational parameters. Matrix product states (MPS), designed for one-dimensional systems, are particularly useful, on account of such powerful ground-state optimisation algorithms as the density matrix renormalisation group (DMRG) [6] and variational uniform MPS (VUMPS) [7].

Given the individual success of these two approaches, it is desirable to combine the two. In particular, DMRG and VUMPS are typically initialised with a random MPS, which leaves them prone to converging to local minima, which potentially display qualitatively different physics to the true ground state. (Dressed) mean-field states (from, e.g., Hartree–Fock theory or VMC) are a much closer starting point, stabilising and reducing the computational cost of MPS optimisation [8, 9]. By initialising with different ansätze, we may also compare different mean-field theories as approximate descriptions of the physical system [10]. To use mean-field states as initial guesses, however, requires efficient algorithms to convert fermionic mean-field wave functions to an MPS form.

Furthermore, MPS representations give ready access to such quantities as the entanglement spectrum and entropy, which are difficult to obtain from VMC but often carry detailed information of exotic physics, such as topological order and the associated edge physics [11, 12]. This is a particularly appealing prospect for Gutzwiller-projected mean-field states, which are common ansatz wave functions for quantum spin liquids [2–5], fractional quantum Hall states [13–15], and other topological orders [16], as the Gutzwiller projection is a local oper-

ation, so it can easily be applied to real-space MPS representations of mean-field states.

As first noted in Ref. [17], Slater determinants can conveniently be converted to MPS form, since their Schmidt vectors are themselves Slater determinants [18–20] and so entries of the canonical MPS tensors are overlaps of Slater determinants, which can be computed efficiently. Since then, this idea has been expanded to Pfaffian (or Boguliubov mean-field) states [21] and successfully used to study Gutzwiller-projected mean-field states [22] and as starting points to DMRG [8–10].

In this paper, we introduce a new open-source Python library called **TeMFpy** for efficiently converting fermionic mean-field states to MPS form. TeMFpy provides both an easy-to-use interface for the key functionality of representing Slater determinants and Pfaffian states as either finite or infinite MPS and a number of well-defined building blocks for advanced users to build new algorithms. It is built on top of TeNPy [23], so that the resulting MPS representations can be used directly for further calculations with, e.g., DMRG. We hope that TeMFpy will open up the power of combining mean-field, VMC, and tensor-network approaches to a wide community of users.

1.1 Outline

The core functionality, constructing finite MPS representations of fermionic mean-field states, specified through either a (Nambu) mean-field Hamiltonian or a (Nambu) correlation matrix, is described in Sec. 2. In particular, Section 2.1 provides an overview of the interface and a guide to basic usage.

We introduce new algorithms for computing MPS tensor entries for Slater determinants and Pfaffian states in detail in Secs. 2.2 and 2.3, respectively. In both cases, we exploit the fact that the leading Schmidt vectors that we keep only differ in the occupation of a few "entangled" orbitals [24, 25], so their overlaps can be reduced to a determinant or a Pfaffian defined only on these orbitals. In a typical Schmidt decomposition, the χ leading Schmidt vectors can be captured through $n_{\text{ent}} = O(\log \chi)$ entangled orbitals, so a single MPS tensor entry can be computed in $O(n_{\text{ent}}^3) = O(\log^3 \chi)$, rather than $O(L^3)$, time, a dramatic speedup.

Section 3 discusses TeMFpy's functionality for computing infinite MPS, based on ideas outlined in Ref. [17]. We provide both a generic implementation able to convert any finite MPS into infinite ones, as well as specialised algorithms for mean-field states in Sec. 3.1, which bypass the need to construct finite MPS for large systems. Finally, we implement Gutzwiller projection schemes in Sec. 4.

1.2 Installing TeMFpy

TeMFpy is written in pure Python (version 0.1 requires at least Python 3.10) and is largely platform-agnostic.[1] It is available from PyPI using the command

```
pip install --upgrade temfpy
```

We will also make it available on CondaForge soon. The development version is available at https://github.com/temfpy/temfpy, where we also welcome issue reports, suggestions, and code contributions. A detailed API reference is available online at https://temfpy.github.io/temfpy.

[1] Installing the `pfapack` dependency (needed for handling Pfaffian states) for Windows is, however, not trivial, see here for details. You can also bypass this issue using the Windows Subsystem for Linux.

2 Converting mean-field states to MPS

Writing any quantum state as an MPS (in canonical form) relies on computing its Schmidt decomposition,

$$|\psi\rangle = \sum_\alpha \lambda_\alpha^{(i)} |L_\alpha^{(i)}\rangle \otimes_g |R_\alpha^{(i)}\rangle\,, \tag{1}$$

first, where $\lambda_\alpha^{(i)}$ are the Schmidt values across entanglement cut i, while $|L_\alpha^{(i)}\rangle$ and $|R_\alpha^{(i)}\rangle$ are the corresponding Schmidt vectors defined on the sites left and right to the same cut.[2] Now, the entries of the left-canonical MPS tensor for a given site are given by the overlaps of the left Schmidt vectors for the entanglement cuts on either side of it [6]:

$$A^{n_i}_{\alpha\beta} = \left(\langle n_i| \otimes_g \langle L_\alpha^{(i-1)}|\right) |L_\beta^{(i)}\rangle, \tag{2a}$$

while right-canonical tensors are given by overlaps of right Schmidt vectors:

$$B^{n_i}_{\beta\alpha} = \left(\langle R_\alpha^{(i)}| \otimes_g \langle n_i|\right) |R_\beta^{(i-1)}\rangle. \tag{2b}$$

For a generic state, Eq. (2) is impractical to use directly. However, the eigenstates of fermionic mean-field Hamiltonians are fermionic Gaussian states [1], i.e., Slater determinants or Pfaffian (i.e., Boguliubov mean-field) states. The Schmidt vectors of such states are in turn also Slater determinants or Pfaffian states and the (generalised) orbitals that define them can be obtained from diagonalising their single-particle correlation matrices [18–20]. The entries of the MPS tensor (2) thus reduce to overlaps of fermionic Gaussian states, which can be computed efficiently as determinants or Pfaffians of the overlaps of individual orbitals.

Furthermore, as the leading Schmidt vectors only differ in the occupation of a few of these (generalised) orbitals [24, 25], we can apply linear-algebra manipulations to these determinants and Pfaffians to reduce the computational complexity of computing each MPS tensor from $O(\chi^2 N^3)$ [17] to $O(N^3 + \chi^2 \log^3 \chi)$, where N is the size of the system and χ is the bond dimension of the tensor. Our algorithms thus differ significantly from those of Ref. [24,25], as we do not rely on introducing an array of virtual fermion degrees of freedom and effectively compress several intermediate steps in their approach. We describe our method in detail in Secs. 2.2.2 and 2.3.3.

2.1 How to use the TeMFpy implementation

TeMFpy implements the calculations outlined above for both Slater determinants (module `temfpy.slater`) and Pfaffian states (module `temfpy.pfaffian`). Both follow the same general structure and calling sequences:

1. `SchmidtModes.from_correlation_matrix()` computes (generalised) orbitals that build up the Schmidt vectors by diagonalising submatrices of the single-particle correlation matrix. The resulting orbital information is kept in `SchmidtModes` objects.

2. `SchmidtVectors.from_schmidt_modes()` finds the combinations of these orbitals that correspond to the largest Schmidt values.[3] These combinations and the corresponding

[2] In this paper, we always deal with Hilbert spaces of fermions, so the standard tensor product is replaced by the fermionic ($\mathbb{Z}_2$ graded) one, $\otimes_g$ [26]. This distinction will be particularly important when we come to Pfaffian states.

[3] Each Schmidt value $\lambda_\alpha^{(i)}$ is given by a product of eigenvalues from the previous diagonalisation, so finding the leading ones has the same complexity as finding the subsets of a set with the highest products. See Ref. [27] and Appendix A for details of how we solve this latter problem.

Schmidt values are stored in `SchmidtVectors` objects, which can be thought of as compressed representations of the leading Schmidt vectors on a given entanglement cut.

For convenience, we also provide `SchmidtVectors.from_correlation_matrix()` that combines the functionality of the two functions above.

3. `MPSTensorData.from_schmidt_vectors()` then takes two `SchmidtVectors` objects corresponding to the entanglement cuts on either side of a site, precomputes quantities related to orbital overlaps (see Secs. 2.2.2 and 2.3.3 for details), and stores them in an `MPSTensorData` object, which is effectively a compressed description of the MPS tensor on the given site.

   Finally, `MPSTensorData.to_npc_array()` converts this compressed format into an explicit MPS tensor.

In addition, the following convenience functions are provided:

- `correlation_matrix()` computes the ground-state correlation matrix of a given (tight-binding or Nambu) Hamiltonian.
- `C_to_MPS()` performs steps 1–3 for every site of the system to convert a correlation matrix directly into a TeNPy MPS object in mixed canonical form.
- `C_to_iMPS()` uses two correlation matrices of a gapped system that differ by one repeating unit cell to construct an infinite MPS representation of the ground state in the thermodynamic limit. This function and its usage is described in more detail in Sec. 3.1.

The basic usage of these functions (except `C_to_iMPS`) and objects is demonstrated in Listing 1.

2.2 Slater determinants

2.2.1 Schmidt decomposition

Given the correlation matrix $C_{ij} := \langle c_j^\dagger c_i \rangle$ for a Slater determinant, we start by diagonalizing the correlation matrix restricted to subsystem A:

$$C_{ij}^{AA} := \langle c_j^\dagger c_i \rangle_{i,j\in A} = U_A \Lambda_A U_A^\dagger. \tag{3}$$

The eigenvectors U_A define a new set of orbitals

$$d_{A,\alpha}^\dagger = \sum_{i\in A} (U_A)_{i,\alpha} c_i^\dagger. \tag{4}$$

The corresponding eigenvalues $\lambda_{A,\alpha}$ are between 0 and 1 (cf. Appendix B): If $\lambda_{A,\alpha} = 1$ (0), $d_{A,\alpha}^\dagger$ is an occupied (empty) orbital of the overall Slater determinant, contained entirely within subsystem A. Orbitals corresponding to eigenvalues $0 < \lambda_{A,\alpha} < 1$ are entangled with the rest of the system.

Consider now a chain split into left and right halves by an entanglement cut. By diagonalizing the correlation matrices restricted to these subsystems, we obtain two sets of operators $d_L^\dagger$ and $d_R^\dagger$. When transforming the correlation matrix into the combined basis of these, the entangled orbitals on either side form 2×2 blocks of the form

$$\begin{pmatrix} \langle d_{L,\alpha}^\dagger d_{L,\alpha}\rangle & \langle d_{R,\alpha}^\dagger d_{L,\alpha}\rangle \\ \langle d_{L,\alpha}^\dagger d_{R,\alpha}\rangle & \langle d_{R,\alpha}^\dagger d_{R,\alpha}\rangle \end{pmatrix}_{\alpha\in E} = \begin{pmatrix} \lambda_\alpha & \sqrt{\lambda_\alpha(1-\lambda_\alpha)} \\ \sqrt{\lambda_\alpha(1-\lambda_\alpha)} & 1-\lambda_\alpha \end{pmatrix}, \tag{5}$$

```python
import numpy as np
from temfpy import slater

# Hamiltonian, in this case a 10-site tight-binding chain
H = -(np.eye(10, k=1) + np.eye(10, k=-1))

# Ground-state correlation matrix
C = slater.correlation_matrix(H)

# Control parameters for MPS truncation
trunc_par = {
    "chi_max": 100, # maximum bond dimension
    "svd_min": 1e-6, # lowest Schmidt value kept (default: 1e-6)
    "degeneracy_tol": 1e-12, # largest relative difference where Schmidt values are considered degenerate (default: 1e-12)
}

# Orbitals that make up the Schmidt vectors (cut between sites 4 and 5)
modes = slater.SchmidtModes.from_correlation_matrix(C, 5, trunc_par)

# Schmidt vectors in terms of these orbitals
schmidt = slater.SchmidtVectors.from_correlation_matrix(C, 5, trunc_par)
schmidt = slater.SchmidtVectors.from_schmidt_modes(modes, trunc_par)

# MPS tensor for site 5
# first the Schmidt vectors for the cut to the right
schmidt_R = slater.SchmidtVectors.from_correlation_matrix(C, 6, trunc_par)
# left canonical
A = slater.MPSTensorData.from_schmidt_vectors(schmidt, schmidt_R, "left")
A = A.to_npc_array() # explicit MPS tensor as a TeNPy array
# right canonical
B = slater.MPSTensorData.from_schmidt_vectors(schmidt_R, schmidt, "right")
B = B.to_npc_array() # explicit MPS tensor as a TeNPy array

# Build the full MPS
mps = slater.C_to_MPS(C, trunc_par)
```

Listing 1: Basic usage of the `temfpy.slater` module. The calling sequence of the corresponding functions in `temfpy.pfaffian` is identical, except that one must specify the full Nambu Hamiltonian/correlation matrix in the input (cf. Sec. 2.3.1).

see Appendix B. Each of these blocks yields one occupied orbital in the overall Slater determinant, with creation operator

$$b_\alpha^\dagger = \sqrt{\lambda_\alpha} d_{L,\alpha}^\dagger + \sqrt{1-\lambda_\alpha} d_{R,\alpha}^\dagger. \tag{6}$$

Therefore, the Slater determinant can be written in terms of the occupied orbitals of the left (O_L) and right (O_R) subsystems and the entangled orbitals (E) shared between the two subsystems:

$$|\psi\rangle = \prod_{\mu\in O_L} d_{L,\mu}^\dagger \prod_{\alpha\in E} \left(\sqrt{\lambda_\alpha} d_{L,\alpha}^\dagger + \sqrt{1-\lambda_\alpha} d_{R,\alpha}^\dagger\right) \prod_{\nu\in O_R} d_{R,\nu}^\dagger |0\rangle\,. \tag{7}$$

Multiplying (7) out shows that the Schmidt values are products with a term of either $\sqrt{\lambda_\alpha}$ or $\sqrt{1-\lambda_\alpha}$ for each $\alpha \in E$. The corresponding Schmidt vectors are themselves Slater determinants, with each entangled orbital occupied in either the left (if the corresponding factor

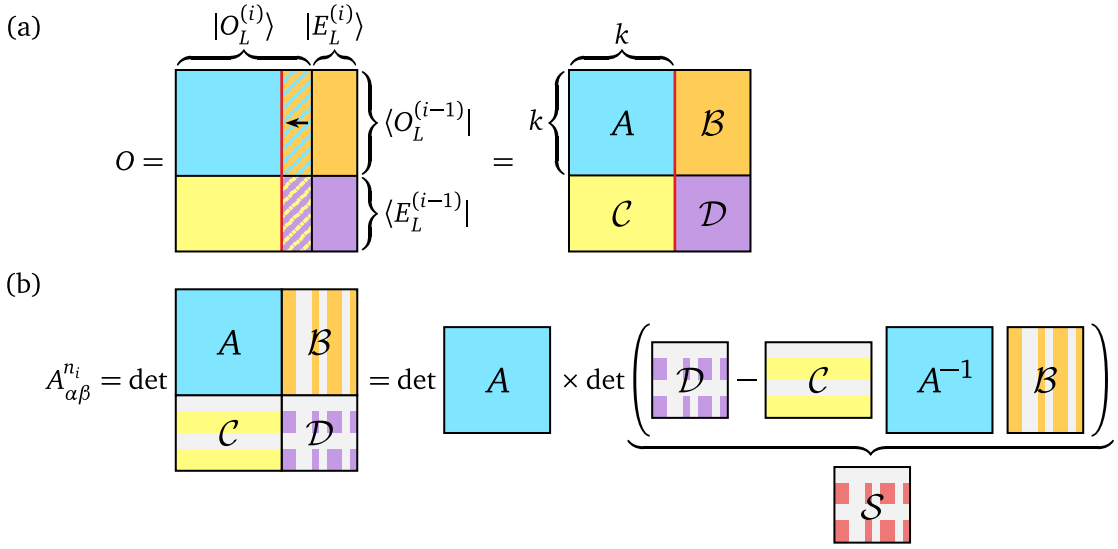

Figure 1: **(a)** In order to make the submatrix A of O (11) that contains the overlaps of occupied orbitals, we may need to reclassify some occupied orbitals as entangled. The size k of A is the smaller of the number of occupied orbitals in the bra and the ket state. **(b)** The matrix A appears in all overlap matrices (8) that determine the elements of the left-canonical tensor $A^{n_i}_{\alpha\beta}$. The remaining blocks are subsets of the rows/columns of the submatrices $\mathcal{B}, \mathcal{C}$, and $\mathcal{D}$ (gray lines indicate the rows/columns left out), depending on the entangled orbitals in the respective bra (rows) and ket (columns) Schmidt vectors. To efficiently compute the tensor entries, we precompute $\det(A)$ and the matrix $\mathcal{S}$ (10) and select the appropriate rows and columns of $\mathcal{S}$ for each tensor element.

in the Schmidt value is $\sqrt{\lambda_\alpha}$) or the right Schmidt vector (if the corresponding factor in the Schmidt value is $\sqrt{1-\lambda_\alpha}$).

2.2.2 Fast overlap computation

Evaluating (2) thus requires us to compute the overlaps of Slater determinants. (The on-site degree of freedom can be included in the "bra" Slater determinant as an additional site-localised orbital [17]). It is well known that such overlaps are given by the determinant of the matrix of overlaps of the individual orbitals; evaluating these determinants independently would yield the whole MPS tensor in $O(\chi^2 N^3)$ time [17], where χ is the bond dimension of the MPS.

TeMFpy improves on this scaling by noting that the overlaps of the "occupied" orbitals in (7) appear in all of these determinants, i.e., the determinants can be written in the following block structure:[4]

$$A^{n_i}_{\alpha\beta} = \begin{vmatrix} \langle O_L^{(i-1)}|O_L^{(i)}\rangle & \langle O_L^{(i-1)}|E_{L,\beta}^{(i)}\rangle \\ \langle E_{L,\alpha}^{(i-1)}|O_L^{(i)}\rangle & \langle E_{L,\alpha}^{(i-1)}|E_{L,\beta}^{(i)}\rangle \end{vmatrix} =: \begin{vmatrix} A & B \\ C & D \end{vmatrix}, \tag{8}$$

where $|O_L\rangle$ and $|E_\alpha\rangle$ stand for the wave functions of the occupied orbitals and the entangled orbitals $d_L^\dagger$ contained in the Schmidt vector $|L_\alpha\rangle$, respectively. [We may reclassify a few occupied orbitals as entangled to make the block A square, see Fig. 1(a).] The submatrix A appears

[4] For the sake of simplicity, we follow the case of left Schmidt vectors in what follows. Right Schmidt vectors can be dealt with analogously.

identically in each determinant, which motivates evaluating (8) using the identity[5]

$$\begin{vmatrix} A & B \\ C & D \end{vmatrix} = \det(A)\det(D - CA^{-1}B). \tag{9}$$

As illustrated in Fig. 1(b), each entry $(D-CA^{-1}B)_{ab}$ depends only on the two entangled orbitals $d_{L,a}^{(i-1)}$ and $(d_{L,b}^{(i)})^\dagger$. Therefore, they can be precomputed for every pair of entangled orbitals as the single matrix

$$\mathcal{S} = \mathcal{D} - \mathcal{C}A^{-1}\mathcal{B} \tag{10}$$

$$O := \begin{pmatrix} \langle O_L^{(i-1)}|O_L^{(i)}\rangle & \langle O_L^{(i-1)}|E_L^{(i)}\rangle \\ \langle E_L^{(i-1)}|O_L^{(i)}\rangle & \langle E_L^{(i-1)}|E_L^{(i)}\rangle \end{pmatrix} := \begin{pmatrix} A & \mathcal{B} \\ \mathcal{C} & \mathcal{D} \end{pmatrix}, \tag{11}$$

where $|E_L^{(i-1)}\rangle$ and $|E_L^{(i)}\rangle$ stand for all the entangled orbitals across the respective entanglement cuts. This precomputation takes $O(N^3)$ time, after which each entry of the MPS tensor requires computing a determinant of size $O(\log\chi)$, resulting in an overall time cost of $O(N^3+\chi^2\log^3\chi)$. This matches the asymptotic complexity of [24, 25], but by forgoing the construction of intermediate fermionic orbitals, we achieve equivalent results in the fewest possible steps, thus likely improving on wall-clock time.

2.3 Pfaffian (Boguliubov mean-field) states

2.3.1 How to specify Hamiltonians and correlation matrices

From the user's point of view, the main complication of the Pfaffian case compared to Slater determinants is the appearance of anomalous correlations $\langle cc\rangle$, $\langle c^\dagger c^\dagger\rangle$, which need to be specified together with the number-conserving ones. There are a number of conventions one can choose to specify the overall Nambu Hamiltonians and correlation matrices, of which TeMFpy supports two:

Complex-fermion basis. The "basis" of single-particle operators are the creation and annihilation operators $c_i^\dagger$ and c_i, listed as the row vector $(c_i^\dagger, c_i)$. The Nambu Hamiltonian matrix H is interpreted as $\hat{H} = (c_i^\dagger, c_i)\, H\, (c_j^\dagger, c_j)^\dagger$, i.e., the matrix is expected to contain the coefficients of

$$\begin{pmatrix} c_i^\dagger c_j & c_i^\dagger c_j^\dagger \\ c_i c_j & c_i c_j^\dagger \end{pmatrix}$$

in this layout, while the correlation matrix is given by [28]

$$C = \mathbb{1} - \left\langle (c_i^\dagger, c_i)^\dagger\, (c_j^\dagger, c_j)\right\rangle = \begin{pmatrix} \langle c_j^\dagger c_i\rangle & \langle c_j c_i\rangle \\ \langle c_j^\dagger c_i^\dagger\rangle & \langle c_j c_i^\dagger\rangle \end{pmatrix}. \tag{12}$$

In particular, TeMFpy expects that the matrices consist of 2×2 blocks containing the Hamiltonian terms or correlators for a given pair of sites, i.e., that `C[2*i : 2*i+2, 2*j : 2*j+2]` contain the correlators (12) for sites i and j.

[5] For right Schmidt vectors, the entangled orbitals come first, so we use $|\cdot| = \det(D)\det(A-BD^{-1}C)$ instead.

Majorana basis. Instead of the complex-fermion operators $c, c^\dagger$, we may also introduce the Majorana operators

$$\gamma_{2n} = (c_n^\dagger + c_n)/\sqrt{2}, \qquad \gamma_{2n+1} = i(c_n^\dagger - c_n)/\sqrt{2} \tag{13}$$

to define Nambu Hamiltonians and correlation matrices. Note that the normalisation of (13) differs from the usual one by a factor of $\sqrt{2}$: This makes the transformation between the two bases unitary, so the eigenvalues of the correlation matrix [and thus formulas like (16)] remain unchanged, allowing us to use largely the same code in both cases.

Naturally, entries H_{ij} of the Hamiltonian matrix are expected to be the coefficients of $\gamma_i\gamma_j$, while the correlation matrix is given by $C_{ij} = \mathbb{1} - \langle\gamma_i\gamma_j\rangle = \langle\gamma_j\gamma_i\rangle$.

Specifying the basis in TeMFpy. `SchmidtModes.from_correlation_matrix()`, `SchmidtVectors.from_correlation_matrix()`, `C_to_MPS()`, and `C_to_iMPS()` must know which of the two bases the correlation matrix is specified in. This must be set with the keyword argument `basis`, which may be either `"M"` (Majorana basis) or `"C"` (complex-fermion basis).

`correlation_matrix()` can return the correlation matrix in a different basis to the Hamiltonian. This is controlled by the argument `basis`, which can be `"M->C"` to compute the complex-fermion correlation matrix of a Majorana Hamiltonian, or vice versa with `"C->M"`. One can also specify `"M->M"` or `"C->C"`, which does not change the basis but checks Nambu symmetry in the appropriate basis. (If `basis` is not specified, the correlation matrix in the input basis is computed, with no checks.) In addition, helper functions are provided for converting matrices and eigenvectors between the two bases and for checking Nambu symmetry.

Internally, `SchmidtModes.from_correlation_matrix()` uses the Majorana basis to diagonalise the correlation matrix, as this makes it easier to ensure Nambu symmetric eigenvectors, see appendix B.2. From there on, however, we use the complex-fermion basis, as quantities such as the parity of Boguliubov vacua or the overlaps (22) are more naturally written in those terms.

2.3.2 Schmidt decomposition

Schmidt vectors. Given the anomalous correlations $\langle cc\rangle, \langle c^\dagger c^\dagger\rangle$ in Pfaffian states, we need to diagonalise the full Nambu correlation matrix (rather than just the number-conserving correlators $\langle c^\dagger c\rangle$) restricted to subsystem A:

$$C^{AA} := \begin{pmatrix} \langle c_j^\dagger c_i\rangle & \langle c_j c_i\rangle \\ \langle c_j^\dagger c_i^\dagger\rangle & \langle c_j c_i^\dagger\rangle \end{pmatrix}_{i,j\in A} = N_A \Lambda N_A^\dagger. \tag{14}$$

By definition, C^{AA} satisfies the Nambu symmetry $C^{AA} = \mathbb{1} - \tau^x (C^{AA})^* \tau^x$, whence its eigenvalues are of the form $\lambda_1, \ldots, \lambda_n, 1-\lambda_1, \ldots, 1-\lambda_n$ with $0 \le \lambda_\alpha \le 1/2$, while the corresponding eigenvectors can be chosen in the form

$$N_A = \begin{pmatrix} U & V^* \\ V & U^* \end{pmatrix}. \tag{15}$$

Now, the reduced density matrix of subsystem A is a mixed fermionic Gaussian mixed state [18, 21], which can be written in the form

$$\rho_A = \prod_{\alpha=1}^{n} \left[\lambda_\alpha \gamma_\alpha^\dagger \gamma_\alpha + (1-\lambda_\alpha)\gamma_\alpha \gamma_\alpha^\dagger\right] \qquad (\gamma_\alpha^\dagger, \gamma_\alpha) = (c_i^\dagger, c_i)\begin{pmatrix} U & V^* \\ V & U^* \end{pmatrix}. \tag{16}$$

Multiplying this expression out shows that the Schmidt values are products with a term of either $\sqrt{\lambda_\alpha}$ or $\sqrt{1-\lambda_\alpha}$ for each $1 \leq \alpha \leq n$. The corresponding Schmidt vectors are all Pfaffian states that are annihilated by either γ_α (if the corresponding factor in the Schmidt value is $\sqrt{1-\lambda_\alpha}$) or $\gamma_\alpha^\dagger$ (if the corresponding factor in the Schmidt value is $\sqrt{\lambda_\alpha}$). Since we have defined $\lambda_\alpha \leq 1/2$, the leading Schmidt state $|\mathrm{vac}_A\rangle$ is annihilated by every γ_α. To fix the relative signs of the other Schmidt vectors, we will write them explicitly as $\gamma_\alpha^\dagger \ldots \gamma_\beta^\dagger |\mathrm{vac}_A\rangle$.

Consistency relations of left and right Schmidt vectors. Similar to the Slater-determinant case, the entangled eigenvectors (that is, those corresponding to $\lambda \neq 0$) of C^{LL} and C^{RR} are singular vectors of the off-diagonal block of the Nambu correlation matrix, C^{LR}, see appendix B. However, the singular vectors (B.3) obtained from the SVD of C^{LR} would violate the Nambu symmetry (15), which requires a sign difference between $\langle \gamma^\dagger \gamma \rangle$ and $\langle \gamma\gamma^\dagger \rangle$ correlators even after diagonalisation. Therefore, the closest we can get to an SVD of C^{LR} while respecting Nambu symmetry is

$$C^{LR} = N_{L,\mathrm{ent}} \begin{pmatrix} 0 & \mathrm{diag}\left(\sqrt{\lambda_\alpha(1-\lambda_\alpha)}\right) \\ -\mathrm{diag}\left(\sqrt{\lambda_\alpha(1-\lambda_\alpha)}\right) & 0 \end{pmatrix} N_{R,\mathrm{ent}}^\dagger, \tag{17}$$

where the subscript "ent" indicates the entangled eigenvectors corresponding to $\lambda \neq 0$. The "singular values" appear in the off-diagonal blocks because the eigenvalues corresponding to a matching pair of left and right singular vectors are λ_α and $1-\lambda_\alpha$. This also implies that the nonzero λ_α are the same on both sides of the entanglement cut.

In the basis of the $\gamma, \gamma^\dagger$ operators as defined by (16, 17), the entangled part of the correlation matrix thus consists of 2×2 blocks of the form

$$\begin{pmatrix} \langle \gamma_{L,\alpha}^\dagger \gamma_{L,\alpha} \rangle & \langle \gamma_{R,\alpha} \gamma_{L,\alpha} \rangle \\ \langle \gamma_{L,\alpha}^\dagger \gamma_{R,\alpha}^\dagger \rangle & \langle \gamma_{R,\alpha} \gamma_{R,\alpha}^\dagger \rangle \end{pmatrix} = \begin{pmatrix} \lambda_\alpha & \sqrt{\lambda_\alpha(1-\lambda_\alpha)} \\ \sqrt{\lambda(1-\lambda_\alpha)} & 1-\lambda_\alpha \end{pmatrix}$$

$$\begin{pmatrix} \langle \gamma_{L,\alpha} \gamma_{L,\alpha}^\dagger \rangle & \langle \gamma_{R,\alpha}^\dagger \gamma_{L,\alpha}^\dagger \rangle \\ \langle \gamma_{L,\alpha} \gamma_{R,\alpha} \rangle & \langle \gamma_{R,\alpha}^\dagger \gamma_{R,\alpha} \rangle \end{pmatrix} = \begin{pmatrix} 1-\lambda_\alpha & -\sqrt{\lambda_\alpha(1-\lambda_\alpha)} \\ -\sqrt{\lambda(1-\lambda_\alpha)} & \lambda_\alpha \end{pmatrix},$$

which implies that $\langle \Gamma_\alpha^\dagger \Gamma_\alpha \rangle = \langle \tilde{\Gamma}_\alpha^\dagger \tilde{\Gamma}_\alpha \rangle = 0$ for

$$\Gamma_\alpha = \sqrt{1-\lambda_\alpha}\, \gamma_{L,\alpha} - \sqrt{\lambda_\alpha}\, \gamma_{R,\alpha}^\dagger \qquad \tilde{\Gamma}_\alpha = \sqrt{\lambda_\alpha}\, \gamma_{L,\alpha}^\dagger + \sqrt{1-\lambda_\alpha}\, \gamma_{R,\alpha},$$

that is, that the overall wave function is annihilated by Γ_α and $\tilde{\Gamma}_\alpha$. This requirement is satisfied by

$$|\psi\rangle = \prod_{\lambda_\alpha \neq 0} \left(\sqrt{1-\lambda_\alpha} + \sqrt{\lambda_\alpha}\, \gamma_{L,\alpha}^\dagger \gamma_{R,\alpha}^\dagger \right) |\mathrm{vac}_L\rangle \otimes_g |\mathrm{vac}_R\rangle\,, \tag{18}$$

which is consistent with (16) and fixes all sign ambiguity in the Schmidt decomposition, allowing us to obtain MPS in mixed canonical form.

2.3.3 Fast overlap computation

Eq. (2) gives the MPS tensor entries as overlaps of Pfaffian states; the physical site can be included on the "bra" side through an additional Boguliubov operator γ, set to either c_i or $c_i^\dagger$ so as to ensure that the vacuum states on the two sides have the same parity (cf. Appendix C.2). Therefore, we need an efficient way of evaluating overlaps of the form

$$O = \langle \mathrm{vac}_A |\, a_{\alpha_k} \ldots a_{\alpha_1} b_{\beta_1}^\dagger \ldots b_{\beta_m}^\dagger \,|\mathrm{vac}_B\rangle\,, \tag{19}$$

where the a_α and b_β are two complete sets of Boguliubov operators and $|\mathrm{vac}_A\rangle$ and $|\mathrm{vac}_B\rangle$ are the Pfaffian states annihilated by them. Let the two sets of operators be related to one another by

$$(b^\dagger, b) = (a^\dagger, a)\begin{pmatrix} U & V^* \\ V & U^* \end{pmatrix} \tag{20}$$

and assume that U is not singular: this means that $|\mathrm{vac}_{A,B}\rangle$ have the same parity and their overlap is nonzero. Then we have [28, 29]

$$|\mathrm{vac}_B\rangle = \frac{1}{|\det U|}\exp\left(\frac{1}{2}M_{\alpha\beta}a_\alpha^\dagger a_\beta^\dagger\right)|\mathrm{vac}_A\rangle\,, \tag{21}$$

where $M = (VU^{-1})^*$ is antisymmetric; the normalisation follows from the Onishi formula [30]. We can then show (Appendix D) that the overlap (19) is given by

$$O = \frac{1}{|\det U|}\,\mathrm{pf}\begin{pmatrix} N^{BB} & N^{BA} \\ -(N^{BA})^{\mathrm{T}} & N^{AA} \end{pmatrix}, \tag{22a}$$

where

$$N_{ij}^{AA} = M_{\alpha_i\alpha_j}, \qquad N_{ij}^{BA} = \left[(U^*)^{-1}\right]_{\beta_{m+1-i}\alpha_j}, \qquad N_{ij}^{BB} = \left[(U^*)^{-1}V\right]_{\beta_{m+1-i}\beta_{m+1-j}}. \tag{22b}$$

That is, each overlap is given by a Pfaffian the size of which is determined by the number of Boguliubov excitations in the defining Schmidt vectors, which is capped by the $O(\log\chi)$ number of active generalised orbitals. Therefore, computing a full MPS tensor takes $O(N^3+\chi^2\log^3\chi)$ time, just like the Slater-determinant case. In terms of asymptotic complexity, this matches the approach of Ref. [21]; however, instead of a full Bloch–Messiah decomposition, the most complex operations involved here are matrix inversions, which does reduce the CPU time cost of the algorithm. TeMFpy uses the `pfapack` library [31] to compute the Pfaffians in (22).

3 Infinite MPS for gapped systems

The ground states of gapped, translation-invariant systems can efficiently be represented using matrix product states with finite bond dimensions even in the thermodynamic limit [7]. Due to translation invariance, the tensors of such an *infinite* MPS repeat with the periodicity of the Hamiltonian, so the whole infinite system can be captured by MPS tensors for a single unit cell of length k.

Naïvely, we could obtain such an iMPS representation by computing the ground state of a sufficiently large finite system and extracting k consecutive tensors $A_n, \ldots, A_{n+k-1}$ near the middle. However, we have to account for the gauge redundancy of MPS: on any given entanglement cut, redefining the MPS tensors as $A'_i = A_i X$, $A'_{i+1} = X^{-1}A_{i+1}$ leaves the wave function invariant [6]. Even in the limit of very long systems, these gauge choices on entanglement cuts n and $n+k$ will generically be different, so contracting them directly would change the wave function, see Fig. 2(a).

To construct valid iMPS tensors, we need to fix this gauge freedom. We follow the approach outlined in Ref. [17], which effectively replaces the Schmidt vectors $|L^{(L,n)}\rangle$ and $|R^{(L,n+k)}\rangle$, which form the environment of the extracted unit cell, with those of a system one unit cell shorter, $|L^{(L-k,n)}\rangle$ and $|R^{(L-k,n)}\rangle$: The latter are defined on the same entanglement cut, so their gauge degrees of freedom are necessarily consistent. The unitary rotations between these bases are given by

$$(C_0)_{\alpha\beta} = \langle L_\alpha^{(L-k,n)}|L_\beta^{(L,n)}\rangle\,, \qquad (D_0)_{\alpha\beta} = \langle R_\beta^{(L-k,n)}|R_\alpha^{(L,n+k)}\rangle\,. \tag{23}$$

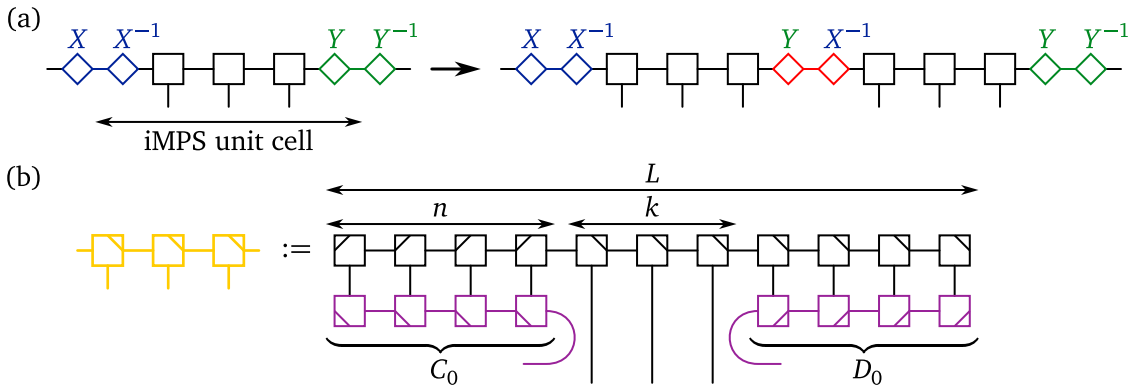

Figure 2: **(a)** One unit cell's worth of MPS tensors extracted from a finite MPS cannot be used directly as an iMPS unit cell due to the mismatched gauge choices X and Y on its two ends. Upon repeating such a naïve unit cell, these unequal gauge tensors do not cancel out but alter the wave function by inserting spurious unitaries YX^{-1} (red) on the virtual legs connecting unit cells. **(b)** Suitable iMPS unit cells (yellow) can be constructed by contracting the first and last tensors of a unit cell extracted from the longer MPS (black) with unitary matrices C_0, D_0 obtained as the overlaps of the left and right environments $|L^{(L,n)}\rangle$ and $|R^{(L,n+k)}\rangle$ with their counterparts $\langle L^{(L-k,n)}|$ and $\langle R^{(L-k,n)}|$ in a shorter chain (purple). This transforms the Schmidt bases on both ends of the unit cell to that of the shorter chain at entanglement cut n (free purple legs), thus making them consistent upon repeating. To obtain normalised Schmidt vectors in the environment, both chains must be in mixed canonical form (indicated by diagonal slashes in each tensor block).

As illustrated in Fig. 2(b), replacing the first and last tensors in the unit cell, A_n and A_{n+k-1}, with $C_0 A_n$ and $A_{n+k-1} D_0$ fixes the gauge on both ends of the unit cell to that of entanglement cut n of the shorter chain. As such, the unit cell can now be repeated indefinitely without deforming the wave function, i.e., it is a valid iMPS unit cell. [For finite bond dimensions, the matrices C_0, D_0 defined in (23) are only approximately unitary: we deal with this issue in Appendix E.]

The algorithm outlined above is implemented in the module `temfpy.iMPS` for generic MPS, see Listing 2 for an example workflow.

3.1 Implementation for mean-field states

For Slater determinants and Pfaffian states, however, it is not necessary to construct the Schmidt states surrounding the iMPS unit cell as MPS, as their overlaps (23) can efficiently be computed the same way as MPS tensor entries, cf. Secs. 2.2.2 and 2.3.3: Such overlaps can also be computed using the classes `slater.MPSTensorData` and `pfaffian.MPSTensorData`.

`slater.C_to_iMPS()` and `pfaffian.C_to_iMPS()` thus compute the MPS tensors of the longer chain for a single unit cell of the iMPS *only*, as well as the left and right Schmidt vectors for the reference entanglement cut n of the shorter chain. The unitary rotation C on the left end of the iMPS unit cell is applied the same way as explained above and in Appendix E for a generic MPS. By contrast, as we compute right canonical iMPS tensors, the unitary rotation at the right end can be combined into the construction of the last tensor by using the Schmidt

```python
import numpy as np
from temfpy import slater, iMPS

def H(L, t1=-1, t2=-1.5):
    M = t1 * np.ones(L - 1)
    M[1::2] = t2
    M = np.diag(M, 1)
    return M + M.T

# Control parameters
trunc_par = {"chi_max": 100}  # cf. Listing 1
L_short = 128
cell = 2  # cf. periodicity of H

# Shorter chain
C_short, _ = slater.correlation_matrix(H(L_short))
mps_short = slater.C_to_MPS(C_short, trunc_par)

# Longer chain
C_long, _ = slater.correlation_matrix(H(L_short + cell))
mps_long = slater.C_to_MPS(C_long, trunc_par)

# Build the iMPS
iMPS, error = iMPS.MPS_to_iMPS(
    mps_short, mps_long, sites_per_cell=cell, cut=L_short // 2
)
```

```
iMPSError(
    left_unitary=1.04496936e-05,
    left_schmidt=9.77293322e-08,
    right_unitary=1.04564911e-05,
    right_schmidt=5.58724587e-07
)
```

Listing 2: Basic usage of the `temfpy.iMPS` module for a dimerised tight-binding Hamiltonian with a unit cell of 2 sites. The arguments `sites_per_cell` (k in Fig. 2) and `cut` (n) are mandatory and can be provided as positional arguments.

vectors of the shorter chain [17]:

$$\begin{aligned}\tilde{B}^{n_k}_{\alpha\beta} = B^{n_k}_{\alpha\gamma}(D_0)_{\gamma\beta} &= \langle R^{(L-k,n)}_\beta | R^{(L,n+k)}_\gamma \rangle \times \left(\langle R^{(L,n+k)}_\gamma | \otimes_g \langle n_k | \right) | R^{(L,n+k-1)}_\alpha \rangle \\ &= \left(\langle R^{(L-k,n)}_\beta | \otimes_g \langle n_k | \right) | R^{(L,n+k-1)}_\alpha \rangle, \end{aligned} \tag{24}$$

(summation over Greek Schmidt-vector indices implied) because the Schmidt vectors $|R^{(L,n+k)}\rangle$ form a complete orthonormal basis. As such, these routines do not return any estimate of the conversion error introduced on the right end of the iMPS unit cell.

An example workflow with these routines is shown in Listing 3. As only a fraction of all MPS tensors needs to be computed, this approach is much faster than that in Listing 2. Furthermore, it avoids the truncation errors introduced by converting the environments of the iMPS unit cell to MPS form (cf. the error estimates in Listings 2 and 3).

```python
import numpy as np
from temfpy import slater

def H(L, t1=-1, t2=-1.5):
    M = t1 * np.ones(L - 1)
    M[1::2] = t2
    M = np.diag(M, 1)
    return M + M.T

# Control parameters
trunc_par = {"chi_max": 100}  # cf. Listing 1
L_short = 128
cell = 2  # cf. periodicity of H

# Correlation matrices
C_short, _ = slater.correlation_matrix(H(L_short))
C_long, _ = slater.correlation_matrix(H(L_short + cell))

# Build the iMPS
iMPS, error = slater.C_to_iMPS(
    C_short, C_long, trunc_par, sites_per_cell=cell, cut=L_short // 2
)
print(error)
```

```
iMPSError(
    left_unitary=8.81564585e-08,
    left_schmidt=9.79503138e-14
)
```

Listing 3: Usage of the function `slater.C_to_iMPS()` for a dimerised tight-binding Hamiltonian with a unit cell of 2 sites. The arguments `sites_per_cell` (k in Fig. 2) and `cut` (n) are mandatory and can be provided as positional arguments. The calling sequence of `pfaffian.C_to_iMPS()` is identical, except that one must specify the full Nambu Hamiltonian/correlation matrix in the input (cf. Sec. 2.3.1).

4 Gutzwiller projection

In addition to studying them directly or using them as starting points for DMRG on fermionic systems, mean-field wave functions are also useful building blocks for studying such exotic systems as quantum spin liquids or fractional quantum Hall states [2–5, 13–16]. These are often described by parton constructions, where the physical degrees of freedom are split up into a number of fractionalised quasiparticles, which capture the physics of the strongly entangled phase already at mean-field level. This construction, however, enlarges the Hilbert space on each lattice site: To recover a wave function of the original system, the additional degrees of freedom must be removed by acting with a *Gutzwiller projection* operator on every site. This is quite a natural operation for matrix product states, as the Gutzwiller projector can be applied directly to the physical leg(s) associated with each site, resulting in a (generally no longer canonical) MPS with unchanged virtual space.

There are many Gutzwiller projection schemes, appropriate for different parton constructions. As an illustration of the general principle, we implemented Gutzwiller projection for quantum spin liquids in spin-1/2 systems, which are often described using mean-field theories

of Abrikosov fermions:

$$\vec{S}_i = \frac{1}{2} f^\dagger_{i\alpha} \vec{\sigma}_{\alpha\beta} f_{i\beta}, \qquad (f^\dagger_{i\alpha} f_{i\alpha} = 1) \tag{25}$$

where $\vec{\sigma}$ are the Pauli matrices, Greek indices (summation implied) stand for the Abrikosov-fermion species $\uparrow, \downarrow$. The on-site Hilbert space of the Abrikosov fermions is four-dimensional, which is reduced to the spin-1/2 Hilbert space by imposing single occupancy on each site. Spin liquids in Heisenberg models, however, preserve spin-rotation symmetry, which implies that the Abrikosov-fermion mean-field theories that describe them may only contain singlet hopping and pairing terms [3, 32, 33]. Similar to singlet superconductivity [2], these terms can all be treated as hopping if we perform a particle–hole transformation on one spin species:

$$c_{i,\uparrow} := f_{i,\uparrow}, \qquad c_{i,\downarrow} := f^\dagger_{i,\downarrow}. \tag{26}$$

In terms of the c operators, the valid spin Hilbert space corresponds to empty (spin down) and fully occupied (spin up) sites: this also means that the fermion occupation number can be used as a proxy for the total S^z spin component, if it is conserved. We also note that the full gauge freedom of spin liquids is most naturally expressed in terms of mixing the c particles [3, 34].

The Gutzwiller projection from the Hilbert space of c fermions to that of spins is implemented in `temfpy.gutzwiller.abrikosov_ph()`. The input MPS is expected to have $2L$ spinless fermion sites: sites $2i$ and $2i+1$ represent modes $c_{i\uparrow}$ and $c_{i\downarrow}$, respectively. The function groups these pairs of sites, projects out the unphysical (single-occupation) sector of each on-site Hilbert space, and reinterprets the remaining two-dimensional Hilbert space as a spin-1/2 degree of freedom.

We have not tried to build a catalogue of all useful Gutzwiller projection schemes with this first release. However, we welcome and look forward to users who deploy TeMFpy to other strongly correlated systems enriching the library by contributing their Gutzwiller projection schemes.

5 Conclusion

We have presented TeMFpy, a Python library for converting fermionic mean-field states to MPS form. The library incorporates functionality for both Slater determinants and Pfaffian states, as well as finite and infinite MPS representations. We have aimed to make the key functionality readily accessible to users without understanding the details of the algorithm. Nevertheless, the library is fully modular, allowing users to easily prototype new applications of our algorithms. While the algorithms we use are not entirely new [17, 21, 24, 25], we have devised and incorporated several improvements that make the calculations both more efficient and numerically stable. Equally importantly, an easy-to-use open-source implementation of these algorithms, based around an existing tensor-network library [23], opens up its potential to a wide range of potential users.

In the future, we will improve and expand the capabilities of TeMFpy, while maintaining a clean, user-friendly, and backwards-compatible interface. A particularly interesting prospect is incorporating the approach of [24] adapted to bosonic Gaussian states, where our linear-algebra manipulations fail, but by introducing virtual boson degrees of freedom, the problem might still be reduced to computing permanents of manageable size. Furthermore, we will expand the Gutzwiller projection facility in `temfpy.gutzwiller` to a wider range of parton constructions. In particular, we plan to design a general interface that allows users to prototype the projection schemes relevant to their work without a detailed understanding of TeNPy. In addition to TeNPy, we would like to support generating MPS in formats suitable for other

tensor-network libraries, such as ITensor [35]. Last but not least, we will also expand the worked examples published with the codebase in order to illustrate more workflows and to help users get started with similar projects.

In all of this, we are looking forward to the support of the community of users, which we hope will build up around TeMFpy, as it will be deployed for studying (Gutzwiller-projected) mean-field states, seeding DMRG/VUMPS, and other applications we do not yet anticipate.

Acknowledgements

We thank Hao Chen, Johanna Ockenfels, and Hong-Hao Tu for helpful discussions.

Funding information S. H. H. and A. Sz. were supported by Ambizione grant No. 215979 by the Swiss National Science Foundation.

A Finding the highest Schmidt values

The Schmidt values implied by the Schmidt decompositions (7) and (18) are both of the form

$$\lambda(P) = \prod_{i\in P} a_i \prod_{j\in\overline{P}} b_j \tag{A.1}$$

for some $a_1,\dots,a_n$ and $b_1,\dots,b_n$, where n is the number of entangled orbitals, P is a subset of $\{1,2,\dots,n\}$, and $\overline{P} = \{1,2,\dots,n\}\backslash P$. Therefore, finding the most significant χ Schmidt vectors is equivalent to finding the subsets $P_1,\dots,P_\chi$ for which $\lambda(P)$ is the highest. Defining $M_i = \max(a_i,b_i)$ and $m_i = \min(a_i,b_i)$, $\lambda(P)$ for any subset P can also be written as

$$\lambda(P) = \tilde{\lambda}(P') = \prod_{i\in P'} m_i \prod_{j\in\overline{P}'} M_j = \prod_{i\in P'} \frac{m_i}{M_i} \prod_{j=1}^{n} M_j \tag{A.2}$$

$$\log\tilde{\lambda}(P') = \text{const.} - \sum_{i\in P'} \underbrace{\log\left(\frac{M_i}{m_i}\right)}_{r_i} \tag{A.3}$$

for another subset $P'\subseteq\{1,\dots,n\}$. That is, finding the highest Schmidt values is equivalent to finding the subsets of $\{r_1,\dots,r_n\}$ with the lowest sum. Note that all $r_i \geq 0$ by construction.

In the following we show that Algorithm 1, given in [27] without proof, indeed generates every subset in order of increasing sum. We first note that every tuple (P,t,ℓ) pushed on the heap satisfies $t = \sum_{i\in P} r_i$ and $\ell = \max(P)$; it is easy to see that lines 4, 10 and 11 obey these relations.

All subsets are obtained. We can view the subsets $P_i\cup\{\ell+1\}$ and $P_i\cup\{\ell+1\}\backslash\{\ell\}$ pushed to the heap on lines 10 and 11 as children of P_i that are located on level $(\ell+1)$ of an n-level binary tree. That is, sets with highest entry K can only appear on level K of this binary tree. To see that every nonempty[6] subset of $\{1,\dots,n\}$ does appear in the tree, we observe that $\{k\}$ is always added to the subset at level k and it may only be removed at level $k+1$. Therefore, subset P can only appear on level $K=\max(P)$ of the binary tree, on the node obtained by taking line 10 at levels $\{k+1 : k\in P, k<K\}$ and line 11 on all other levels. As such, the 2^n-1 nonempty subsets of $\{1,\dots,n\}$ are in one-to-one correspondence with the nodes of the binary tree, i.e., every subset is generated precisely once.

[6] The empty subset corresponds to the lowest possible sum. We deal with it separately on line 2.

Algorithm 1 Subsets of a set of nonnegative numbers in order of increasing sum.

Input: $[r_1, \ldots, r_n]$, an array of nonnegative numbers, sorted in increasing order.
function LOWESTSUM($[r_1, \ldots, r_n]$)
  $S = [\{\}]$ ▷ The very lowest sum must be handled specially
  initialize min-heap H ▷ Ordered by subset sum
  H.push($\{1\}, r_1, 1$) ▷ Second lowest sum
  for $i = 2, \ldots, 2^n$ **do**
    $(P_i, t_i, \ell) \leftarrow$ H.min
    H.pop(P_i, t_i, ℓ)
    append P_i to S
    if $\ell < n$ **then**
      H.push($P_i \cup \{\ell+1\}, t_i + r_{\ell+1}, \ell+1$) ▷ Descendants contain $\{\ell\}$
      H.push($P_i \cup \{\ell+1\} \setminus \{\ell\}, t_i + r_{\ell+1} - r_\ell, \ell+1$) ▷ Descendants don't contain $\{\ell\}$
    end if
  end for
  return S
end function

Subsets are returned in order of increasing sum. Since every subset of $\{1, \ldots, n\}$ is returned at some point, this is equivalent to showing that the sum t_i returned in each step is no less than the sum t_{i-1} in the previous step. $0 = t_1 \leq t_2 = r_1$ holds by definition. For $i > 2$, we have two cases to consider: If P_i was already contained in the heap when P_{i-1} was popped, then $t_{i-1} \leq t_i$ by the heap property. Otherwise, it is one of the children of P_{i-1} in the binary tree introduced above, pushed on the heap in step $i-1$. By assumption, $r_\ell \geq 0$ and $r_\ell - r_{\ell-1} \geq 0$, so both children of P_{i-1} have a sum no less than t_{i-1}. Therefore, $t_i \geq t_{i-1}$ again.

B Mutual diagonalisation of C^{LL}, C^{LR}, and C^{RR}

The correlation matrices of both Slater determinants and Pfaffian states must satisfy $C^2 = C$ [28]: Intuitively, all eigenvalues of C must be either 1 (corresponding to filled orbitals or Boguliubov operators that annihilate the Pfaffian) or 0 (corresponding to empty orbitals or Boguliubov excitations). This also implies that the eigenvalues of the diagonal submatrices of C are between 0 and 1.

Upon introducing an entanglement cut that splits the system into a left (L) and a right (R) half, it is natural to split the correlation matrix into blocks:

$$C = \begin{pmatrix} C^{LL} & C^{LR} \\ (C^{LR})^\dagger & C^{RR} \end{pmatrix} \tag{B.1}$$

(note that C is Hermitian by definition), for which $C^2 = C$ implies

$$(C^{LL})^2 + C^{LR}(C^{LR})^\dagger = C^{LL} \tag{B.2a}$$
$$C^{LL}C^{LR} + C^{LR}C^{RR} = C^{LR} \tag{B.2b}$$
$$(C^{LR})^\dagger C^{LR} + (C^{RR})^2 = C^{RR}; \tag{B.2c}$$

as C is Hermitian, the $(C^{LR})^\dagger$ block does not yield an independent constraint.

Let u be an eigenvector of C^{LL} with eigenvalue λ. Then by (B.2a), u is also an eigenvector of $C^{LR}(C^{LR})^\dagger$ with eigenvalue $\lambda(1-\lambda)$, i.e., it is a left singular vector of C^{LR} with singular value

$\sigma = \sqrt{\lambda(1-\lambda)}$. Likewise, (B.2c) implies that every eigenvector of C^{RR} is a right singular vector of C^{LR}.

Let now $0 < \lambda < 1$, i.e., let u define an entangled orbital. Then the normalised right singular vector corresponding to u is

$$v = \frac{(C^{LR})^\dagger u}{\sigma} \qquad\qquad u = \frac{C^{LR} v}{\sigma}. \tag{B.3}$$

Now, multiplying (B.2b)† from the right by u/σ yields

$$\begin{aligned} \frac{(C^{LR})^\dagger C^{LL} u}{\sigma} + C^{RR} v &= v \\ C^{RR} v &= (1-\lambda) v. \end{aligned} \tag{B.4}$$

That is, the eigenvalues of C^{LL} and C^{RR} corresponding to matching left and right singular vectors of C^{LR} sum to 1.

Finally, noting that the eigenbases of C^{LL} and C^{RR} provide a full complement of left and right singular vectors of C^{LR} and that $\sigma \neq 0$ only for the entangled modes $0 < \lambda < 1$, we have the following singular-value decomposition of C^{LR}:

$$C^{LR} = \sum_{\text{ent}} \sqrt{\lambda(1-\lambda)}\, u_\lambda\, v_{1-\lambda}^\dagger, \tag{B.5}$$

where the subscripts indicate the eigenvalues corresponding to each eigenvector, and u and v are related by (B.3).

B.1 Implementation for Slater determinants

In case both left and right Schmidt vectors are needed for a single entanglement cut, the phase relation (B.3) between the entangled eigenvectors of C_{LL} and C_{RR} must be fixed in addition to diagonalising them.

Suppose that the columns of the matrix $\tilde{U}$ contain the eigenvectors of C^{LL} with eigenvalue λ, while $\tilde{V}$ contains the eigenvectors of C^{RR} with eigenvalue $1-\lambda$. Then (B.2) implies

$$\tilde{C}^{LR} := \tilde{U}^\dagger C^{LR} \tilde{V} = \sigma(1-\sigma) W, \tag{B.6}$$

where W is an arbitrary unitary matrix. We fix this unitary degree of freedom by an additional SVD within the degenerate block:

$$\tilde{C}^{LR} = P \Sigma Q^\dagger \implies \qquad U := \tilde{U} P \qquad\qquad V := \tilde{V} Q. \tag{B.7}$$

U and V are still eigenvectors of C^{LL} and C^{RR} with respective eigenvalues λ and $1-\lambda$, but in addition, they are also matching singular vectors of C^{LR}. Doing this for all degenerate blocks, we obtain all corresponding pairs of entangled Schmidt modes. To make the best use of vectorisation in NumPy, we group the degenerate subspaces by multiplicity, so that all matrices $U, V, \tilde{C}^{LR}, \ldots$ of the same size may be handled together.

B.2 Implementation for Pfaffian states

Degenerate eigenspaces of C^{LL} and C^{RR} are handled largely the same way for Pfaffian states. Maintaining the Nambu symmetry (15), however, requires two modifications:

- Generically, the numerical diagonalisation of C^{LL} and C^{RR} does not return eigenvectors that respect (15). Therefore, we perform the gauge fixing (B.7) only for eigenspaces $\tilde{U} := N_{L,\text{ent},\lambda}$ and $\tilde{V} := N_{R,\text{ent},1-\lambda}$ with $\lambda < 1/2$. Then, we enforce (15) by replacing $N_{L,\lambda>1/2}$ and $N_{R,\lambda<1/2}$ with the Nambu counterparts of the transformed $N_{L,\lambda<1/2}$ and $N_{R,\lambda>1/2}$. (This latter is also done for the non-entangled modes.)

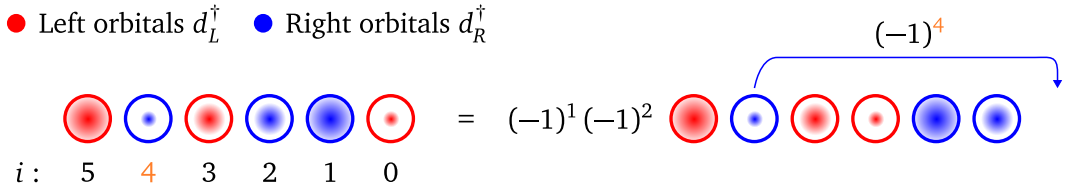

Figure 3: Separating left and right entangled operators $d_L^\dagger, d_R^\dagger$ in one term of the Schmidt decomposition (7). Starting point is the entangled orbitals sorted in decreasing order of λ (the magnitude of λ or $(1-\lambda)$ is indicated by the sizes of the inner red or blue blobs, respectively). The right entangled operators are then swapped one by one to the right end, starting from the rightmost. The resulting fermionic anticommutation sign for each right entangled orbital is given by $(-1)^i$, where i denotes the initial position in the list of entangled orbitals, counted from the right.

- For $\lambda = 1/2$, the Nambu-partner eigenvectors are degenerate. In this case, it is convenient to work in the Majorana basis, where these pairs of eigenvectors are simply complex conjugates of one another: This implies that their real and imaginary parts are also eigenvectors and orthogonal to one another.

  That is, the $\lambda = 1/2$ eigenspace (of dimension d) has a complete real basis. Numerically, we obtain this by an SVD of the concatenated real and imaginary parts of the generically complex eigenvectors: To numerical accuracy, d of the singular values is 1, corresponding to the real eigenvectors r_α; the remaining d singular values are 0.

  In the Majorana basis, C^{LR} is purely imaginary, so $r_L^\dagger \,\mathrm{Im}(C^{LR}) r_R$ has a purely real SVD, which can be used to transform r_L, r_R so that $r_L^\dagger C^{LR} r_R = i/2 \times \mathbb{1}_d$. From these, finally, orthonormal and Nambu-symmetric eigenvectors of C^{LL} and C^{LR} satisfying (17) can be obtained as

$$n_{L,\alpha} = \frac{r_{L,\alpha} + i r_{L,\alpha+d/2}}{\sqrt{2}}, \qquad \overline{n_{L,\alpha}} = \frac{r_{L,\alpha} - i r_{L,\alpha+d/2}}{\sqrt{2}},$$
$$n_{R,\alpha} = \frac{i r_{R,\alpha} + r_{R,\alpha+d/2}}{\sqrt{2}}, \qquad \overline{n_{R,\alpha}} = \frac{-i r_{R,\alpha} + r_{R,\alpha+d/2}}{\sqrt{2}}. \qquad (0 \le \alpha < d/2)$$

C Handling fermion anticommutation

C.1 Slater determinants

Expanding (7). For a proper Schmidt decomposition we need to multiply (7) out, taking care of the fermionic anticommutation signs. A typical term takes the form

$$|S\rangle = \prod_{\mu\in O_L} d_{L,\mu}^\dagger \left(\dots d_{R,i+1}^\dagger d_{L,i}^\dagger d_{R,i-1}^\dagger \dots\right) \prod_{\nu\in O_R} d_{R,\nu}^\dagger |\mathrm{vac}\rangle \qquad (C.1)$$

$$= \mathrm{sgn}(\pi) \left[\prod_{\mu\in O_L} d_{L,\mu}^\dagger \left(\dots d_{L,i}^\dagger \dots\right)|\mathrm{vac}_L\rangle\right] \otimes_g \left[\left(\dots d_{R,i-1}^\dagger d_{R,i+1}^\dagger \dots\right) \prod_{\nu\in O_R} d_{R,\nu}^\dagger |\mathrm{vac}_R\rangle\right], \qquad (C.2)$$

where π is the permutation that permutes all entangled operators $d_R^\dagger$ through the entangled operators $d_L^\dagger$ and reverses their order.

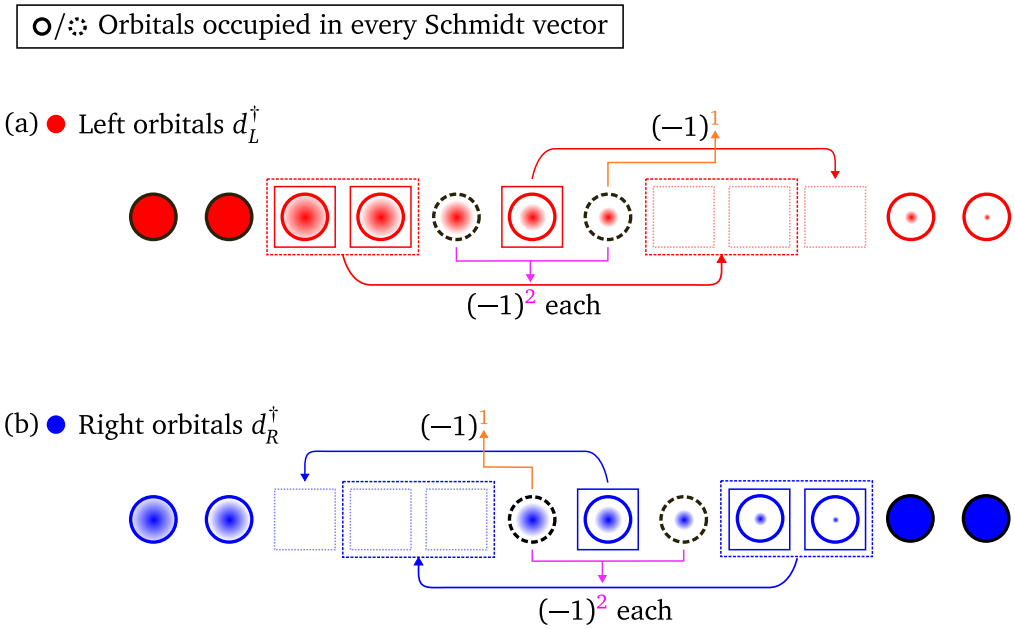

Figure 4: To include entangled orbitals that are in fact occupied in every Schmidt vector (dashed black border and partially filled) in the submatrix A in (9), all entangled operators $d_L^\dagger$ **(a)** / $d_R^\dagger$ **(b)** with mixed occupation (red/blue border) have to be moved past them. Starting from the entangled orbital with the smallest/largest eigenvalue (indicated by the size of the inner blob), they are moved one by one to the right/left, without changing their relative order. Therefore, each entangled orbital picks up a fermionic anticommutation sign determined by the number of reclassified orbitals to its right/left.

One can build π by moving the right entangled orbitals one by one to the right. Due to their reversed order, the number of swaps needed for each orbital depends only on its initial position i in the list of entangled orbitals, where the indexing starts from 0 on the right, see Fig. 3. To account for the resulting anticommutation signs, we multiply each entangled orbital operator $d_{R,i}^\dagger$ by $(-1)^i$.

Reclassification of entangled orbitals. If the bond dimension is limited to χ, some entangled orbitals may in fact be occupied in every Schmidt vector we keep. To be able to include them in the submatrix A in (9), these need to be commuted through the remaining entangled orbitals, introducing an anticommutation sign. It is useful to view this as commuting all remaining entangled orbitals one by one through the newly classified ones. The anticommutation sign incurred in each of these steps depends only on the number of occupied orbital operators it needs to be commuted through, see Fig. 4: We account for these signs by multiplying them to the corresponding orbital eigenvectors.

C.2 Pfaffian states

Expanding (18). For a proper Schmidt decomposition, we need to multiply (18) out, taking care of the fermionic anticommutation signs. A typical term takes the form

$$|S\rangle = \gamma_{L,\alpha_1}^\dagger \gamma_{R,\alpha_1}^\dagger \dots \gamma_{L,\alpha_k}^\dagger \gamma_{R,\alpha_k}^\dagger |\mathrm{vac}_L\rangle \otimes_g |\mathrm{vac}_R\rangle = \gamma_{L,\alpha_1}^\dagger \dots \gamma_{L,\alpha_k}^\dagger \gamma_{R,\alpha_k}^\dagger \dots \gamma_{R,\alpha_1}^\dagger |\mathrm{vac}_L\rangle \otimes_g |\mathrm{vac}_R\rangle\,, \tag{C.3}$$

which differs from (1) in that the operators and states of the left- and right-half Hilbert spaces are not separated from one another. That is, we need to commute the k $\gamma_R^\dagger$ operators across $|\mathrm{vac}_L\rangle$. If the latter is parity odd, each of these steps incur a negative sign [26], so we end up with

$$|S\rangle = (-1)^{k\pi_L}\left(\gamma^\dagger_{L,\alpha_1}\dots\gamma^\dagger_{L,\alpha_k}|\mathrm{vac}_L\rangle\right)\otimes_g\left(\gamma^\dagger_{R,\alpha_k}\dots\gamma_{R,\alpha_1}|\mathrm{vac}_R\rangle\right), \tag{C.4}$$

where $\pi_L = 0, 1$ is the parity of $|\mathrm{vac}_L\rangle$, computed using the Bloch–Messiah decomposition [36]. To account for this additional fermionic sign in the TeMFpy implementation, we

- list the $\gamma^\dagger$ operators in the definition of left and right Schmidt states in opposite order (namely, in the code, we list them in increasing order of λ_α on the left and in decreasing order on the right);
- and flip the sign of all $\gamma_R, \gamma_R^\dagger$ operators if the Boguliubov vacuum $|\mathrm{vac}_L\rangle$ is parity odd.

MPS tensor entries as Pfaffian overlaps. The overlaps to evaluate for (2) are not exactly of the form (19), but rather

$$A^{n_i}_{\alpha\beta} = \langle 0_i|\, c_i^{n_i}\otimes_g \langle\mathrm{vac}_L^{(i-1)}|\,\gamma^{(i-1)}_{L,\alpha_k}\dots\gamma^{(i-1)}_{L,\alpha_1}\otimes_g \gamma^{(i)\dagger}_{L,\beta_1}\dots\gamma^{(i)\dagger}_{L,\beta_m}|\mathrm{vac}_L^{(i)}\rangle \tag{C.5a}$$

$$B^{n_i}_{\beta\alpha} = \langle\mathrm{vac}_R^{(i)}|\,\gamma^{(i)}_{R,\alpha_k}\dots\gamma^{(i)}_{R,\alpha_1}\otimes_g \langle 0_i|\, c_i^{n_i}\otimes_g \gamma^{(i-1)\dagger}_{R,\beta_1}\dots\gamma^{(i-1)\dagger}_{R,\beta_m}|\mathrm{vac}_R^{(i-1)}\rangle\,. \tag{C.5b}$$

As before, we need to commute $\langle\mathrm{vac}_L^{(i-1)}|$ or $\langle 0_i|$ through a number of Boguliubov operators:

$$A^{n_i}_{\alpha\beta} = (-1)^{n_i\pi_L^{(i-1)}}\langle 0_i|\otimes_g\langle\mathrm{vac}_L^{(i-1)}|\,c_i^{n_i}\gamma^{(i-1)}_{L,\alpha_k}\dots\gamma^{(i-1)}_{L,\alpha_1}\gamma^{(i)\dagger}_{L,\beta_1}\dots\gamma^{(i)\dagger}_{L,\beta_m}|\mathrm{vac}_L^{(i)}\rangle \tag{C.6a}$$

$$B^{n_i}_{\beta\alpha} = \langle\mathrm{vac}_R^{(i)}|\otimes_g\langle 0_i|\,\gamma^{(i)}_{R,\alpha_k}\dots\gamma^{(i)}_{R,\alpha_1}c_i^{n_i}\gamma^{(i-1)\dagger}_{R,\beta_1}\dots\gamma^{(i-1)\dagger}_{R,\beta_m}|\mathrm{vac}_R^{(i-1)}\rangle\,, \tag{C.6b}$$

where $\pi_L^{(i-1)}$ is the parity of $|\mathrm{vac}_L^{(i-1)}\rangle$; Eq. (C.6b) incurs no extra sign as $|0_i\rangle$ is parity even. To account for the additional sign in (C.6a), we use $-c_i$ rather than c_i as additional Boguliubov annihilation operator if $|\mathrm{vac}_L^{(i-1)}\rangle$ is parity odd.

Parity matching for Pfaffian overlaps. The Pfaffian overlap formula (22) is only valid if the vacua $|\mathrm{vac}_A\rangle$ and $|\mathrm{vac}_B\rangle$ have the same fermion parity, which is not necessarily the case for Schmidt states on different entanglement cuts. We remedy this problem by particle-hole transforming one Boguliubov operator a_0 on the bra side: $a_0 \leftrightarrow a_0^\dagger$, $\langle\mathrm{vac}'_A| := \langle\mathrm{vac}_A|\,a_0$. For numerical stability, we choose the most entangled (i.e., λ closest to $1/2$, or the on-site mode for MPS tensor entries) Boguliubov mode: This defines $|\mathrm{vac}'_A\rangle$ as the highest-weight bra Schmidt state of the same parity as the leading ket Schmidt state, which are expected to have a high overlap in our use cases. Furthermore, across all Schmidt vectors, this mode is expected to be excited the most often, so flipping it leads to the smallest possible increase in the size of the Pfaffians that need to be computed.

For left Schmidt vectors, the Boguliubov mode operators are listed in order of increasing λ, so the most entangled mode a_0 appears closest to $\langle\mathrm{vac}_A|$ in the string of operators (19). Upon the particle-hole transformation, bra Schmidt states with or without an a_0 Boguliubov excitation transform as

$$\langle\mathrm{vac}_A|\,a_0 a_{\alpha_k}\dots a_{\alpha_1} = \langle\mathrm{vac}'_A|\,a_{\alpha_k}\dots a_{\alpha_1} \tag{C.7a}$$

$$\langle\mathrm{vac}_A|\,a_{\alpha_k}\dots a_{\alpha_1} = \langle\mathrm{vac}_A|\,a_0 a_0^\dagger a_{\alpha_k}\dots a_{\alpha_1} = \langle\mathrm{vac}'_A|\,a'_0 a_{\alpha_k}\dots a_{\alpha_1}, \tag{C.7b}$$

without generating any additional fermionic sign.

For right Schmidt vectors, the Boguliubov mode operators are listed in order of decreasing λ, so the most entangled mode a_0 appears at the end of the string of a_α in (19). Upon the particle-hole transformation, bra Schmidt states with or without an a_0 Boguliubov excitation transform as

$$\langle \mathrm{vac}_A | a_{\alpha_k} \dots a_{\alpha_1} a_0 = (-1)^k \langle \mathrm{vac}_A | a_0 a_{\alpha_k} \dots a_{\alpha_1} = (-1)^k \langle \mathrm{vac}'_A | a_{\alpha_k} \dots a_{\alpha_1} \tag{C.8a}$$

$$\langle \mathrm{vac}_A | a_{\alpha_k} \dots a_{\alpha_1} = \langle \mathrm{vac}_A | a_{\alpha_k} \dots a_{\alpha_1} a_0 a_0^\dagger = (-1)^k \langle \mathrm{vac}'_A | a_{\alpha_k} \dots a_{\alpha_1} a'_0. \tag{C.8b}$$

To account for this additional fermionic sign, we flip the sign of every bra Schmidt mode operator other than a_0.

D Proof of the Pfaffian overlap formula

In the following, we use the Boguliubov operators a_α, b_β and vacua $|\mathrm{vac}_{A,B}\rangle$ defined in Eqs. (20, 21). We start with a simpler result than (22):

Lemma 1. Let γ_i $(1 \le i \le n)$ be either a_α or $a_\alpha^\dagger$ for some α. Then

$$O := |\det U| \times \langle \mathrm{vac}_A | \gamma_n \dots \gamma_2 \gamma_1 | \mathrm{vac}_B \rangle = \mathrm{pf}\, N, \tag{D.1}$$

where the antisymmetric $n \times n$ matrix N is defined by

$$N_{i>j} = \begin{cases} 0 & \gamma_i = a_\alpha^\dagger \text{ for some } \alpha \\ -\delta_{\alpha_i \alpha_j} & \gamma_i = a_{\alpha_i}, \gamma_j = a_{\alpha_j}^\dagger \\ M_{\alpha_i \alpha_j} & \gamma_i = a_{\alpha_i}, \gamma_j = a_{\alpha_j}. \end{cases} \tag{D.2}$$

Proof. By induction on the number of $a^\dagger$ operators among the γ_i. If all $\gamma_i = a_{\alpha_i}$ for all i,

$$O = \langle \mathrm{vac}_A | a_{\alpha_n} \dots a_{\alpha_1} \exp\left(\frac{1}{2} M_{\alpha\beta} a_\alpha^\dagger a_\beta^\dagger \right) | \mathrm{vac}_A \rangle = \mathrm{pf}[M_{\alpha_i \alpha_j}]_{ij}$$

is well known. Now, let there be $k > 0$ $a^\dagger$ among the γ_i and let the leftmost of these be $\gamma_j = a_{\alpha_j}^\dagger$. Let $|S\rangle = |\det U| \times \gamma_{j-1} \dots \gamma_1 |\mathrm{vac}_B\rangle$ for brevity. Let us commute $a_{\alpha_j}^\dagger$ through to the left:

$$\begin{aligned} O &= \langle \mathrm{vac}_A | a_{\alpha_n} \dots a_{\alpha_{j+1}} a_{\alpha_j}^\dagger | S \rangle \\ &= \delta_{\alpha_{j+1} \alpha_j} \langle \mathrm{vac}_A | a_{\alpha_n} \dots a_{\alpha_{j+2}} | S \rangle - \langle \mathrm{vac}_A | a_{\alpha_n} \dots a_{\alpha_{j+2}} a_{\alpha_j}^\dagger a_{\alpha_{j+1}} | S \rangle = \dots \\ &= \sum_{i=j+1}^{n} (-1)^{i+j+1} \delta_{\alpha_i \alpha_j} \langle \mathrm{vac}_A | a_{\alpha_n} \dots a_{\alpha_{i+1}} a_{\alpha_{i-1}} \dots a_{\alpha_{j+1}} | S \rangle , \end{aligned}$$

since $\langle \mathrm{vac}_A | a_{\alpha_j}^\dagger = 0$. The expectation values on the last line contain $(k-1)$ $a^\dagger$ operators, so by the induction step, they are given by the Pfaffian of $[N_{\setminus ij}]$, i.e., the matrix N with columns and rows i and j removed. By definition, $\delta_{\alpha_i \alpha_j} = N_{ji}$ for $j < i$ (N is antisymmetric), and $N_{ji} = 0$ for $j \ge i$. Therefore,

$$O = \sum_{i=j+1}^{n} (-1)^{i+j+1} N_{ji}\, \mathrm{pf}[N_{\setminus ij}],$$

which is precisely the expansion of $\mathrm{pf}\, N$ by its jth row/column. □

In the case of interest, the operators in O are not simply a or $a^\dagger$, but rather their linear combinations. However, the result is still a single Pfaffian:

Lemma 2. Let $\gamma_i = a_\alpha C_{\alpha i} + a_\alpha^\dagger D_{\alpha i}$ for $1 \le i \le n$. Then,

$$O := \langle \mathrm{vac}_A | \gamma_n \dots \gamma_2 \gamma_1 | \mathrm{vac}_B \rangle = \frac{\mathrm{pf}\, N}{|\det U|}, \tag{D.3}$$

where the antisymmetric $n \times n$ matrix N is defined by

$$N_{i>j} = C_{\cdot i}^{\mathrm{T}} (M C_{\cdot j} - D_{\cdot j}). \tag{D.4}$$

Proof. We consider the intermediate case where γ_i for all $i > k$ are either a_{α_i} or $a_{\alpha_i}^\dagger$, i.e., there is precisely one nonzero element of 1 in $C_{\cdot i}$ and $D_{\cdot i}$ together, and proceed by induction on k. $k = 0$ is precisely Lemma 1.

For $0 < k \le L$, we use that the overlap (D.3) is linear in γ_k, so we have

$$O = \sum_\alpha C_{\alpha k} \langle \mathrm{vac}_A | \gamma_n \dots \gamma_{k+1} a_\alpha \gamma_{k-1} \dots \gamma_1 | \mathrm{vac}_B \rangle + D_{\alpha k} \langle \mathrm{vac}_A | \gamma_n \dots \gamma_{k+1} a_\alpha^\dagger \gamma_{k-1} \dots \gamma_1 | \mathrm{vac}_B \rangle .$$

By the induction step, these are all Pfaffians that only differ in their kth row/column. Since the Pfaffian too is linear in its rows/columns, O is also a Pfaffian, whose kth row/column is the linear combination of these, with coefficients $C_{\cdot k}, D_{\cdot k}$. The claim follows from the fact that (D.4) is also linear in $C_{\cdot k}, D_{\cdot k}$. □

Eq. (22) is a special case of Lemma 2 with

$$\gamma_i = \begin{cases} b_{\beta_{m+1-i}}^\dagger = a_\alpha U_{\alpha \beta_{m+1-i}} + a_\alpha^\dagger V_{\alpha \beta_{m+1-i}} & 1 \le i \le m \\ a_{\alpha_{i-m}} & m < i \le m+k, \end{cases}$$

for which the overlap can be written as

$$|\det U| \times \langle \mathrm{vac}_A | a_{\alpha_k} \dots a_{\alpha_1} b_{\beta_1}^\dagger \dots b_{\beta_m}^\dagger | \mathrm{vac}_B \rangle = \mathrm{pf} \begin{pmatrix} N^{BB} & N^{BA} \\ N^{AB} & N^{AA} \end{pmatrix}, \tag{D.5}$$

where

$$N_{ij}^{AA} = M_{\alpha_i \alpha_j} \tag{D.6a}$$
$$N_{ij}^{AB} = (M V_{\cdot \beta_{m+1-j}} - U_{\cdot \beta_{m+1-j}})_{\alpha_i} \tag{D.6b}$$
$$N_{ij}^{BB} = V_{\cdot \beta_{m+1-i}}^{\mathrm{T}} (M V_{\cdot \beta_{m+1-j}} - U_{\cdot \beta_{m+1-j}}) \tag{D.6c}$$

and $N^{BA} = -(N^{AB})^{\mathrm{T}}$. Note that (D.4) applies as written to all entries of N^{AB} as all $b^\dagger$ operators precede all a operators. We can also directly obtain the lower triangle $i > j$ of N^{BB}, but (D.6c) turns out to be antisymmetric: $V^{\mathrm{T}} M V$ is antisymmetric because M is, while $V^{\mathrm{T}} U + U^{\mathrm{T}} V = 0$ follows from the unitarity of the Boguliubov transformation matrix. We can simplify (D.6b) and (D.6c) further by noting that M is antisymmetric:

$$M = (V U^{-1})^* = -(V U^{-1})^\dagger = -(U^\dagger)^{-1} V^\dagger$$
$$M V - U = -(U^\dagger)^{-1} V^\dagger V - U = -(U^\dagger)^{-1} \left(\mathbb{1} - U^\dagger U \right) - U = \left[U - (U^\dagger)^{-1} \right] - U = -(U^\dagger)^{-1},$$

so we have

$$N_{ij}^{AB} = -\left[(U^\dagger)^{-1} \right]_{\alpha_i \beta_{m+1-j}} \tag{D.7a}$$
$$N_{ij}^{BB} = -\left[V^{\mathrm{T}} (U^\dagger)^{-1} \right]_{\beta_{m+1-i} \beta_{m+1-j}} = \left[(U^*)^{-1} V \right]_{\beta_{m+1-i} \beta_{m+1-j}}; \tag{D.7b}$$

the last equality follows because N^{BB} is antisymmetric.

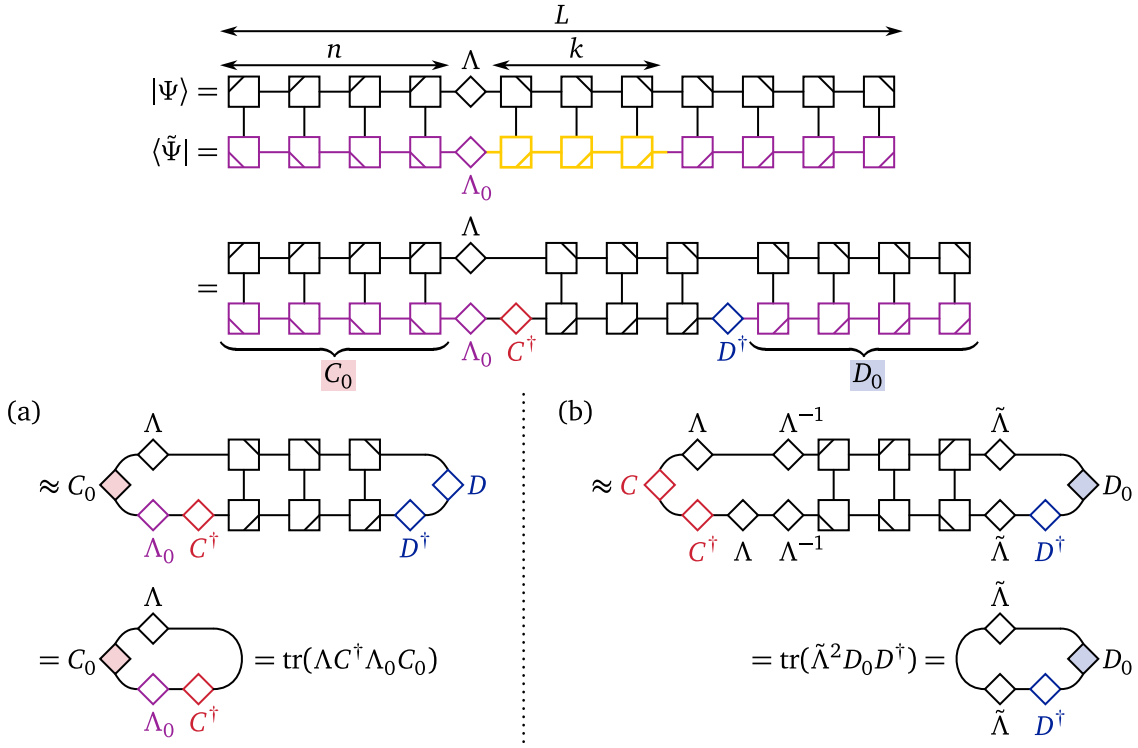

Figure 5: Overlap of the wave function $|\Psi\rangle$ on the longer chain of length L and its reconstruction $|\tilde{\Psi}\rangle$ from MPS tensors of a shorter chain of length $L-k$ and a single iMPS unit cell of length k. Diagonal slashes indicate whether each tensor is left- or right-canonical. **(a, b)** Estimates of the overlap assuming that the reconstruction of the environment to the right **(a)** or to the left **(b)** of the iMPS unit cell is perfect.

E Practical implementation of the iMPS conversion

For finite bond dimensions, the Schmidt vectors $|L_\alpha^{(L,n)}\rangle$ and $|L_\alpha^{(L-k,n)}\rangle$ (or $|R_\beta^{(L,n+k)}\rangle$ and $|R_\beta^{(L-k,n)}\rangle$) do not span the whole Hilbert space, so, generically, the overlap matrices (23) are not unitary. This means that instead of rotating the tensors at the ends of the iMPS unit cell using C_0, D_0 directly, we have to find appropriate unitary rotations C and D.

Furthermore, for finite systems, the Schmidt values Λ and Λ_0 are not exactly equal for two different finite system sizes, adding an additional source of mismatch between the wave function $|\Psi\rangle$ on the longer chain, and its reconstruction $|\tilde{\Psi}\rangle$ using the wave function on the shorter chain and the iMPS unit cell. In the following, we will find the unitary matrices C and D that maximise the overlap $\langle\tilde{\Psi}|\Psi\rangle$ (Fig. 5).

For the sake of concreteness, we assume from here on that the iMPS unit cell is right canonical (this choice is built into TeMFpy as well). We will also assume that errors introduced by the iMPS construction on the left and right ends of the unit cell are independent, that is, when looking for the optimal C, we assume that the optimal unitary D is D_0 and vice versa.

To find the optimal left gauge rotation matrix C, we can thus contract down all the right canonical tensors in the right environment and the iMPS unit cell [Fig. 5(a)]. That is, the overlap of the two wave functions is given by $\mathrm{tr}(\Lambda_0 C_0 \Lambda C^\dagger)$, which can be maximised using the following result:

Lemma. Let the singular value decomposition of a square matrix A be $A = U\Sigma V^\dagger$ and let all singular values be strictly positive. Then, the unitary matrix R for which $\mathrm{tr}(AR^\dagger)$ is maximal is

$R = UV^\dagger$.

Proof. Let R be a unitary matrix. Then $\mathrm{tr}(AR^\dagger) = \mathrm{tr}(U\Sigma V^\dagger R^\dagger) = \mathrm{tr}(\Sigma V^\dagger R^\dagger U) = \sum_i \sigma_i (V^\dagger R^\dagger U)_{ii}$. Since $V^\dagger R^\dagger U$ is unitary, its diagonal elements can be at most 1. Therefore, $\mathrm{tr}(AR^\dagger)$ is highest if all diagonal elements of $V^\dagger R^\dagger U$ are 1, i.e., if it is the identity matrix. Rearranging yields $R = UV^\dagger$. □

Finally, the smallest possible deviation between $|\Psi\rangle$ and $|\tilde{\Psi}\rangle$ is

$$\begin{aligned} \left\| |\Psi\rangle - |\tilde{\Psi}\rangle \right\|^2 &= \mathrm{tr}\,\Lambda^2 + \mathrm{tr}\,\Lambda_0^2 - \mathrm{tr}(C_0 \Lambda C^\dagger \Lambda_0) - \mathrm{tr}(\Lambda_0 C \Lambda C_0^\dagger) \\ &= \mathrm{tr}\left[(\Lambda_0 C - C_0 \Lambda) \times \mathrm{H.c.}\right] + \mathrm{tr}\left[\Lambda(\mathbb{1} - C_0^\dagger C_0)\Lambda\right], \end{aligned} \tag{E.1}$$

where we used that C is unitary. Both terms are nonnegative: the first is the Frobenius norm of $\Lambda_0 C - C_0 \Lambda$; as for the second, diagonal entries of $C_0^\dagger C_0$ are at most 1 (equal to 1 if C_0 is unitary). In fact, we may assign distinct meanings to the two error terms:

- The second term measures the deviation of the overlap matrix C_0 from being unitary, weighted with the corresponding Schmidt values. This error term can be reduced by keeping more Schmidt vectors, i.e., using higher bond dimensions.
- The first term, by contrast, measures the deviation of the Schmidt values Λ and Λ_0 and how accurately C maps between them. This error term can be reduced by using larger system sizes, where finite-size differences between $|L_\alpha^{(L,n)}\rangle$ and $|L_\alpha^{(L-k,n)}\rangle$ are smaller.

Likewise, to find the optimal right gauge rotation matrix D, we neglect the error introduced on the left hand side; in particular, we assume that $C = C_0$ and $\Lambda_0 C = C\Lambda$ [first line of Fig. 5(b)]. Now, in order to contract down the left environment and the iMPS unit cell, we have to bring the latter to *left* canonical form by inserting the Schmidt values Λ^{-1} and $\tilde{\Lambda}$ on the left and right ends of the unit cell, respectively. In all, the overlap $\langle\tilde{\Psi}|\Psi\rangle$ is reduced to $\mathrm{tr}(\tilde{\Lambda}^2 D_0 D^\dagger)$, where $\tilde{\Lambda}$ are the Schmidt values in the longer chain at the right end of the iMPS unit cell. The optimal D follows from the same Lemma as above. The lowest possible deviation is given by

$$\begin{aligned} \left\| |\Psi\rangle - |\tilde{\Psi}\rangle \right\|^2 &= \mathrm{tr}\,\tilde{\Lambda}^2 + \mathrm{tr}\,\tilde{\Lambda}^2 - \mathrm{tr}(\tilde{\Lambda} D_0 D^\dagger \tilde{\Lambda}) - \mathrm{tr}(\tilde{\Lambda} D D_0^\dagger \tilde{\Lambda}) \\ &= \mathrm{tr}\left[(\tilde{\Lambda} D - \tilde{\Lambda} D_0) \times \mathrm{H.c.}\right] + \mathrm{tr}\left[\tilde{\Lambda}(\mathbb{1} - D_0 D_0^\dagger)\tilde{\Lambda}\right]. \end{aligned} \tag{E.2}$$

`temfpy.iMPS.MPS_to_iMPS()` returns estimates of both error terms separately for both sides of the unit cell.

References

[1] S. Bravyi, *Lagrangian representation for fermionic linear optics*, Quant. Inf. Comput. **5**(3), 216 (2005), doi:10.26421/QIC5.3-3.

[2] F. Becca and S. Sorella, *Quantum Monte Carlo Approaches for Correlated Systems*, Cambridge University Press, Cambridge, ISBN 9781107129931, doi:10.1017/9781316417041 (2017).

[3] X.-G. Wen, *Quantum orders and symmetric spin liquids*, Phys. Rev. B **65**(16), 165113 (2002), doi:10.1103/PhysRevB.65.165113.

[4] X.-G. Wen, *Quantum Field Theory of Many-Body Systems: From the Origin of Sound to an Origin of Light and Electrons*, Oxford Graduate Texts. Oxford University Press, Oxford, ISBN 9780199227259, doi:10.1093/acprof:oso/9780199227259.001.0001 (2007).

[5] L. Savary and L. Balents, *Quantum spin liquids: a review*, Rep. Prog. Phys. **80**, 016502 (2017).

[6] U. Schollwöck, *The density-matrix renormalization group in the age of matrix product states*, Ann. Phys. **326**(1), 96 (2011), doi:https://doi.org/10.1016/j.aop.2010.09.012, January 2011 Special Issue.

[7] L. Vanderstraeten, J. Haegeman and F. Verstraete, *Tangent-space methods for uniform matrix product states*, SciPost Phys. Lect. Notes p. 7 (2019), doi:10.21468/SciPostPhysLectNotes.7.

[8] H.-K. Jin, H.-H. Tu and Y. Zhou, *Density matrix renormalization group boosted by Gutzwiller projected wave functions*, Phys. Rev. B **104**, L020409 (2021), doi:10.1103/PhysRevB.104.L020409.

[9] H.-K. Jin, R.-Y. Sun, H.-H. Tu and Y. Zhou, *A promising method for strongly correlated electrons in two dimensions: Gutzwiller-guided density matrix renormalization group*, AAPPS Bull. **35**(1), 16 (2025), doi:10.1007/s43673-025-00156-8.

[10] R.-Y. Sun, H.-K. Jin, H.-H. Tu and Y. Zhou, *Possible chiral spin liquid state in the $s = 1/2$ kagome Heisenberg model*, npj Quantum Mater. **9**(1), 16 (2024), doi:10.1038/s41535-024-00627-5.

[11] H. Li and F. D. M. Haldane, *Entanglement spectrum as a generalization of entanglement entropy: Identification of topological order in non-abelian fractional quantum hall effect states*, Phys. Rev. Lett. **101**, 010504 (2008), doi:10.1103/PhysRevLett.101.010504.

[12] X.-L. Qi, H. Katsura and A. W. W. Ludwig, *General relationship between the entanglement spectrum and the edge state spectrum of topological quantum states*, Phys. Rev. Lett. **108**, 196402 (2012), doi:10.1103/PhysRevLett.108.196402.

[13] J. K. Jain, *Incompressible quantum Hall states*, Phys. Rev. B **40**, 8079 (1989), doi:10.1103/PhysRevB.40.8079.

[14] Y.-M. Lu and Y. Ran, *Symmetry-protected fractional Chern insulators and fractional topological insulators*, Phys. Rev. B **85**, 165134 (2012), doi:10.1103/PhysRevB.85.165134.

[15] J. McGreevy, B. Swingle and K.-A. Tran, *Wave functions for fractional Chern insulators*, Phys. Rev. B **85**, 125105 (2012), doi:10.1103/PhysRevB.85.125105.

[16] T. Liu, Y.-H. Wu, H.-H. Tu and T. Xiang, *Bridging conformal field theory and parton approaches to $SU(n)_k$ chiral spin liquids*, Phys. Rev. B **111**, 205137 (2025), doi:10.1103/PhysRevB.111.205137.

[17] G. Petrica, B.-X. Zheng, G. K.-L. Chan and B. K. Clark, *Finite and infinite matrix product states for Gutzwiller projected mean-field wave functions*, Phys. Rev. B **103**, 125161 (2021), doi:10.1103/PhysRevB.103.125161.

[18] I. Peschel, *Calculation of reduced density matrices from correlation functions*, J. Phys. A **36**(14), L205 (2003), doi:10.1088/0305-4470/36/14/101.

[19] S.-A. Cheong and C. L. Henley, *Many-body density matrices for free fermions*, Phys. Rev. B **69**, 075111 (2004), doi:10.1103/PhysRevB.69.075111.

[20] I. Peschel, *Special review: Entanglement in solvable many-particle models*, Braz. J. Phys. **42**(3), 267 (2012), doi:10.1007/s13538-012-0074-1.

[21] H.-K. Jin, R.-Y. Sun, Y. Zhou and H.-H. Tu, *Matrix product states for Hartree-Fock-Bogoliubov wave functions*, Phys. Rev. B **105**, L081101 (2022), doi:10.1103/PhysRevB.105.L081101.

[22] Y.-H. Wu and H.-H. Tu, *Non-abelian topological order with* $\mathrm{SO}(5)_1$ *chiral edge states*, Phys. Rev. B **106**, 115129 (2022), doi:10.1103/PhysRevB.106.115129.

[23] J. Hauschild and F. Pollmann, *Efficient numerical simulations with tensor networks: Tensor Network Python (TeNPy)*, SciPost Phys. Lect. Notes p. 5 (2018), doi:10.21468/SciPostPhysLectNotes.5.

[24] T. Liu, Y.-H. Wu, H.-H. Tu and T. Xiang, *Efficient conversion from fermionic Gaussian states to matrix product states*, Quantum Sci. Technol. **10**(3), 035033 (2025), doi:10.1088/2058-9565/addae0.

[25] K. Li, Y.-B. Zhang and H. C. Po, *Constructive fermionic matrix product states for projected Fermi sea*, Phys. Rev. Res. **7**, 013182 (2025), doi:10.1103/PhysRevResearch.7.013182.

[26] N. Bultinck, D. J. Williamson, J. Haegeman and F. Verstraete, *Fermionic matrix product states and one-dimensional topological phases*, Phys. Rev. B **95**, 075108 (2017), doi:10.1103/PhysRevB.95.075108.

[27] StackOverflow user "btilly", *Answer to "How to generate k largest subset sums for a given array (contains positive and negative numbers)"*, StackOverflow, https://stackoverflow.com/a/72117947/27202449 (2022).

[28] J.-P. Blaizot and G. Ripka, *Quantum theory of finite systems*, The MIT Press (1986).

[29] L. M. Robledo, *Sign of the overlap of Hartree-Fock-Bogoliubov wave functions*, Phys. Rev. C **79**, 021302 (2009), doi:10.1103/PhysRevC.79.021302.

[30] L. M. Robledo, *Practical formulation of the extended Wick's theorem and the Onishi formula*, Phys. Rev. C **50**, 2874 (1994), doi:10.1103/PhysRevC.50.2874.

[31] M. Wimmer, *Algorithm 923: Efficient numerical computation of the Pfaffian for dense and banded skew-symmetric matrices*, ACM Trans. Math. Softw. (TOMS) **38**(4), 1 (2012), doi:10.1145/2331130.2331138, Python implementation: https://pfapack.readthedocs.io.

[32] G. Baskaran, Z. Zou and P. W. Anderson, *The resonating valence bond state and high-Tc superconductivity - A mean field theory*, Solid State Commun. **63**(11), 973 (1987), doi:10.1016/0038-1098(87)90642-9.

[33] G. Baskaran and P. W. Anderson, *Gauge theory of high-temperature superconductors and strongly correlated Fermi systems*, Phys. Rev. B **37**(1), 580 (1988), doi:10.1103/PhysRevB.37.580.

[34] I. Affleck, Z. Zou, T. Hsu and P. W. Anderson, *SU(2) gauge symmetry of the large-U limit of the Hubbard model*, Phys. Rev. B **38**(1), 745 (1988), doi:10.1103/PhysRevB.38.745.

[35] M. Fishman, S. R. White and E. M. Stoudenmire, *The ITensor Software Library for Tensor Network Calculations*, SciPost Phys. Codebases p. 4 (2022), doi:10.21468/SciPostPhysCodeb.4.

[36] C. Bloch and A. Messiah, *The canonical form of an antisymmetric tensor and its application to the theory of superconductivity*, Nucl. Phys. **39**, 95 (1962), doi:https://doi.org/10.1016/0029-5582(62)90377-2.