`pyED` documentation (v1.0 - stable release)

Jamin Kidd

Tulane University

E-mail: `jkidd@tulane.edu`

Contents

1 Introduction

pyED is a simple toolkit for numerically diagonalizing an interacting spinful fermion Hamiltonian on a small lattice. It makes use of the implicitly restarted Lanczos algorithm (Chapter 4 of Ref. [1]) to obtain a low-energy subspace of the full eigenspectrum. It also contains analysis tools for computing observables at arbitrary temperature.

2 Theoretical background

2.1 Fermion operators

The model is written in terms of fermionic raising/lowering operators satisfying the anti-commutation relations

$$\left\{c_{i\alpha}^{(d_1)}, c_{j\beta}^{(d_2)}\right\} = \left(1 - \delta_{d_1 d_2}\right) \delta_{ij} \delta_{\alpha\beta}, \tag{2.1}$$

where

- Latin letters (i.e., i and j) label Wannier orbitals spanning a lattice with L sites,
- Greek letters (i.e., α and β) label spin ($\uparrow$ or $\downarrow$), and
- the superscript (i.e., d_1 and d_2) is used to distinguish raising ($d = +1$) and lowering ($d = -1$) operators. Explicitly,

$$\begin{cases} c_{i\alpha}^{-1} = c_{i\alpha} \\ c_{i\alpha}^{+1} = c_{i\alpha}^{\dagger} \end{cases} \tag{2.2}$$

The occupation number is given by

$$n_{i\alpha} = c_{i\alpha}^{+1} c_{i\alpha}^{-1} \tag{2.3}$$

The vacuum state, denoted $|0\rangle$, has the property that for any i and α,

$$c_{i\alpha}^{-1} |0\rangle = 0, \tag{2.4}$$

i.e., the null state.

2.2 Basis set

The full Hilbert space possesses a complete, orthonormal basis consisting of states that generically have the form

$$|n\rangle = \left(\prod_{j=1}^{N_\uparrow} c_{i_j\uparrow}^{+1}\right) \left(\prod_{j=1}^{N_\downarrow} c_{k_j\downarrow}^{+1}\right) |0\rangle \tag{2.5}$$

The state is normally ordered such that $c_{1\alpha}^{+1}$ always appears to the left of $c_{2\alpha}^{+1}$, etc. To assign each basis state a unique integer label, *pyED* uses the hashing method of Lin [2], which we summarize in Appendix **??**.

2.3 Matrix representation

Any operator $\hat{O}$ on the Hilbert space can be written as a linear combination of fermion "strings" with complex coefficients, i.e.,

$$\hat{O} = \sum_l \lambda_l \hat{O}^{(l)} \tag{2.6}$$

where $\lambda_l \in \mathbb{C}$. A generic string $\hat{O}^{(l)}$ appearing in the sum is a product of M fermion operators,

$$\hat{O}^{(l)} = \prod_{j=1}^{M} c_{i_j \alpha_j}^{(d_j)} \tag{2.7}$$

which may or may not be normally ordered. To encode the operator numerically, *pyED* uses a simple matrix representation, i.e.,

$$O_{mn} = \sum_l \lambda_l \langle m|\hat{O}^{(l)}|n\rangle \tag{2.8}$$

The idea is to directly compute the state $\hat{O}^{(l)}|n\rangle$ by applying Eqns. 2.1 and 2.4 as necessary. In Fock representation (Eqn. A.2),

$$\hat{O}^{(l)}|n\rangle = c_{i_1\alpha_1}^{(d_1)} \; \ldots \; c_{i_{M-1}\alpha_{M-1}}^{(d_{M-1})} \left(c_{i_M\alpha_M}^{(d_M)} \,|\, \ldots \; n_{i_M\alpha_M} \; \ldots \rangle \right) \tag{2.9}$$

The first step is to consider the term inside the parentheses. There are apparently four cases, summarized in the following table:

	$n_{i_M\alpha_M} = 1$	$n_{i_M\alpha_M} = 0$
$d_M = -1$	$e^{i\phi_M} \,|\, \ldots \; 0 \; \ldots \rangle$	$\ldots \; c_{i_M\alpha_M}^{-1} \, |0\rangle = 0$
$d_M = +1$	$\ldots \; \left(c_{i_M\alpha_M}^{+1}\right)^2 \; \ldots \; |0\rangle = 0$	$e^{i\phi_M} \,|\, \ldots \; 1 \; \ldots \rangle$

For the two non-zero cases highlighted in green, Eqn. 2.1 must be applied such that the resulting state is normally ordered. Hence, a phase factor $e^{i\phi_M} = \pm 1$ is generally acquired, depending on the parity of the exchange (even $= +1$, odd $= -1$). This procedure is then iterated over each operator, from right to left (i.e., from the $j = M - 1$ term to the $j = 1$ term). There are two possible outcomes:

1. If the state became null at *any* step, then $\hat{O}^{(l)}|n\rangle = 0$.

2. If the state survives at each step, then the iteration yields

$$\hat{O}^{(l)}|n\rangle = e^{i\Phi_n(l)} \, |f_n(l)\rangle \tag{2.10}$$

where

$$\Phi_n(l) = \sum_{j=1}^{M} \phi_j \tag{2.11}$$

and

$$|f_n(l)\rangle = |\, \ldots \; n_{i_j\alpha_j} + d_j \; \ldots \rangle \tag{2.12}$$

The set of all $|n\rangle$ that fall under the second category generally depends on the form of the operator. Denoting this set as $\mathcal{N}_l$, the matrix element for the fermion string is

$$O_{mn}^{(l)} = \begin{cases} e^{i\Phi_n(l)} & \text{if } n \in \mathcal{N}_l \text{ and } m = f_n(l) \\ 0 & \text{otherwise} \end{cases} \tag{2.13}$$

which immediately yields O_{mn} when combined with Eqn. 2.8.

2.4 Hamiltonian

The Hamiltonian is defined as

$$\hat{H} = \sum_{m=1}^{N_0} \hat{H}_0^{(m)} + \sum_{m=1}^{N_{\text{int}}} \hat{H}_{\text{int}}^{(m)}, \tag{2.14}$$

where the individual terms are of the following form:

- $\hat{H}_0^{(m)}$ is a *free* Hamiltonian, i.e., quadratic in fermion operators:

$$\hat{H}_0^{(m)} = \sum_{i,j} \sum_{\alpha,\beta} T_{ij}^{\alpha\beta} c_{i\alpha}^{(d_1)} c_{j\beta}^{(d_2)} \tag{2.15}$$

- $\hat{H}_{\text{int}}^{(m)}$ is an *interaction* Hamiltonian, i.e., quartic in fermion operators:

$$\hat{H}_{\text{int}}^{(m)} = \sum_{i,j,k,l} \sum_{\alpha,\beta,\gamma,\sigma} V_{ijkl}^{\alpha\beta\gamma\sigma} c_{i\alpha}^{(d_1)} c_{j\beta}^{(d_2)} c_{k\gamma}^{(d_3)} c_{l\sigma}^{(d_4)} \tag{2.16}$$

It is useful to consider the symmetries of the model. We first define

$$\hat{N}_\sigma = \sum_i n_{i\sigma} \tag{2.17}$$

for spin σ ($= \uparrow$ or $\downarrow$), which satisfies the eigenvalue equation $\hat{N}_\sigma |n\rangle = N_\sigma |n\rangle$ by Eqn. 2.5. We then define the total particle number,

$$N = N_\uparrow + N_\downarrow, \tag{2.18}$$

and total spin,

$$S_z = N_\uparrow - N_\downarrow. \tag{2.19}$$

Since each site can be at most doubly occupied, the eigenvalues of N range from 0 to $2L$, while the eigenvalues of S_z range from $-L$ to L. For fixed N, flipping one spin from $\downarrow$ to $\uparrow$ increases S_z by 2. Hence, the allowed values of S_z are

$$S_z = |N - L| - L,\ |N - L| - L + 2,\ \ldots,\ L - |N - L|. \tag{2.20}$$

Similarly, for fixed S_z, the allowed particle numbers are

$$N = |S_z|,\ |S_z| + 2,\ \ldots,\ 2L - |S_z|. \tag{2.21}$$

One can easily verify the above expressions by writing down the possible spin configurations for the case $L = 3$. The Hamiltonian may be symmetric under one, both, or neither of the two operators $\hat{N}$ and $\hat{S}_z$. The condition for particle number symmetry is that each of the $N_0 + N_{\text{int}}$ terms in Eqn. 2.14 satisfies $\sum_i d_i = 0$. For example, terms of the form

$$c_{i\alpha}^{+1} c_{j\beta}^{+1} \tag{2.22}$$

do not conserve N. Similarly, S_z symmetry requires that there are no "spin flip" terms, e.g.,

$$c_{i\uparrow}^{+1} c_{j\downarrow}^{-1} \tag{2.23}$$

2.5 Diagonalization

In any *pyED* calculation, the primary goal is to solve the eigenvalue problem

$$\hat{H} \left|\Psi_k\right\rangle = E_k \left|\Psi_k\right\rangle \tag{2.24}$$

where $\left|\Psi_k\right\rangle$ is a many-body eigenstate. Expressing this numerically is very simple using the basis described in §2.2. Given this basis, we have

$$\left|\Psi_k\right\rangle = \sum_n \alpha_n^{(k)} \left|n\right\rangle \tag{2.25}$$

Applying $\left\langle m\right|$ to both sides of Eqn. 2.24 yields

$$\sum_n H_{mn} \alpha_n^{(k)} = E_k \alpha_m^{(k)} \tag{2.26}$$

which is just a matrix eigenvalue equation $H\vec{\alpha} = E\vec{\alpha}$. Typically, the matrix H_{mn} is very sparse, meaning that most of its entries are zero. Hence, it benefits greatly from using the Lanczos algorithm. This algorithm will introduce a small error that is typically negligible at low temperatures. Nevertheless, full diagonalization is also available in *pyED* through `numpy.linalg.eig`. Generally, one should test how the memory usage of a full diagonalization job scales with system size L before running the target calculation.

2.6 Observables

When computing observables at finite temperature $T = 1/\beta$, the Boltzmann factors $e^{-\beta E_k}$ are generally too large to store as a float. Therefore, instead of computing the partition function, we compute the quantity

$$Z_0 = Z e^{\beta E_0} = \sum_k e^{-\beta(E_k - E_0)} \tag{2.27}$$

where E_0 is the ground state energy. The average value of an operator $\hat{O}$ is then

$$\langle\hat{O}\rangle = \frac{1}{Z_0} \mathrm{Tr}\left[e^{-\beta\left(\hat{H}-E_0\right)}\hat{O}\right] \tag{2.28}$$

$$= \frac{1}{Z_0} \sum_k e^{-\beta(E_k - E_0)} \langle\Psi_k|\hat{O}|\Psi_k\rangle \tag{2.29}$$

$$= \frac{1}{Z_0} \sum_k \sum_{m,n} e^{-\beta(E_k - E_0)} \bar{\alpha}_m^{(k)} \alpha_n^{(k)} O_{mn} \tag{2.30}$$

This expression simplifies at $T = 0$. Denoting the ground state degeneracy as g, we have

$$\lim_{\beta \to \infty} Z_0 = g \tag{2.31}$$

and hence

$$\lim_{\beta \to \infty} \langle \hat{O} \rangle = \frac{1}{g} \sum_{k \ < \ g} \sum_{m,n} \bar{\alpha}_m^{(k)} \alpha_n^{(k)} O_{mn} \tag{2.32}$$

The above expressions can be used to compute energies, order parameters, correlation functions, etc.

3 Using the code

3.1 Installation

Navigate to your preferred installation folder. Then, run the following:

```
git clone https://github.com/jkidd1/pyED.git
cd pyED
bash install.bash
```

To check the installation, you can run

```
bash tests/run_tests.bash
```

3.2 Minimal run

The main input file required by pyED is called `params.yaml`. It is a YAML file, meaning it has the following basic structure:

```
parameter_1: value_1
parameter_2: value_2
```

pyED has several built-in models and lattices, described in §??

A Implementation notes

A.1 Lattice convention

The 2D lattice is assumed to be a finite tiling of a unit cell with side lengths L_x and L_y in the x and y directions, respectively. Each cell has N_{sub} sublattices, meaning that the total number of sites is $L = N_{\text{sub}} \times L_x \times L_y$. To label the Wannier orbitals, a multi-index convention must be chosen. *pyED* uses the following convention:

$$\{\mathbf{R}, s\} \to j = N_{\text{sub}} \left(L_x R_y + R_x \right) + s, \tag{A.1}$$

where $\mathbf{R} = (R_y, R_x)$ is the cell coordinate and s denotes the sublattice index. R_y can be viewed as a layer index, with each layer consisting of $N_{\text{sub}} \times L_x$ sites. To remain consistent with Python, each index always starts at zero. Table 1 highlights the basic structure of the mapping, while Fig. 1 provides an illustration for the case of the 2×2 zigzag honeycomb lattice.

j	0	1	$\dots$	$N_{\text{sub}}-1$	N_{sub}	$\dots$	$N_{\text{sub}} \times L_x - 1$	$N_{\text{sub}} \times L_x$	$\dots$	$L-1$
R_y	0	0	$\dots$	0	0	$\dots$	0	1	$\dots$	$L_y - 1$
R_x	0	0	$\dots$	0	1	$\dots$	$L_x - 1$	0	$\dots$	$L_x - 1$
s	0	1	$\dots$	$N_{\text{sub}}-1$	0	$\dots$	$N_{\text{sub}}-1$	0	$\dots$	$N_{\text{sub}}-1$

Table 1. Site labelling convention used by *pyED* following Eqn. A.1 (the top row is the multi-index j). The colors indicate different partitions of the lattice: red contains only one unit cell (i.e., fixed $\mathbf{R} = \mathbf{0}$), blue contains only one layer (i.e., fixed $R_y = 0$), and yellow-green can generally contain multiple layers (i.e., $R_y = 1, \dots, L_y - 1$).

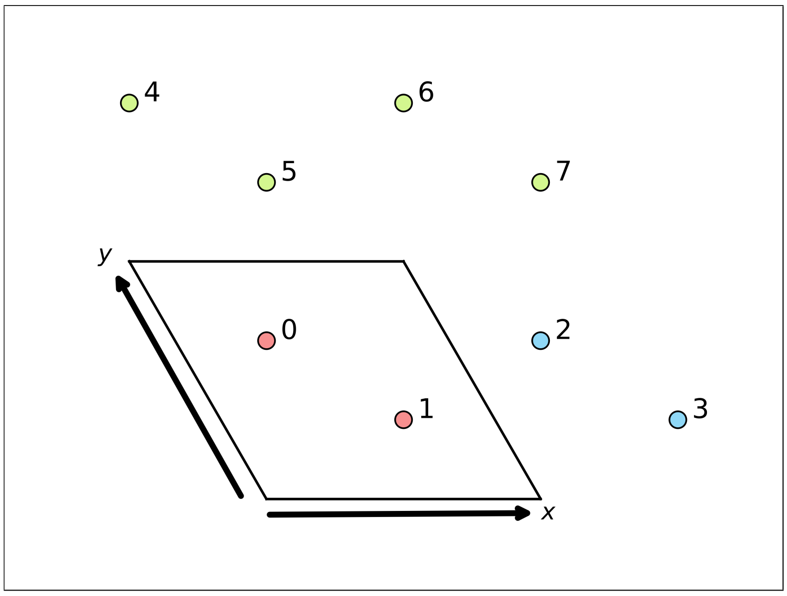

Figure 1. Illustration of the 2×2 zigzag honeycomb lattice with $N_{\text{sub}} = 2$. Sites are labelled from 0 to 7 according to the convention of Eqn. A.1. The color of each site is chosen to match Table 1.

A.2 Lin hashing

In the Fock representation, $|n\rangle$ is expressed as a binary string of occupation numbers:

$$|n_{1\uparrow} \dots n_{L\uparrow}\, n_{1\downarrow} \dots n_{L\downarrow}\rangle \tag{A.2}$$

with each $n_{i\alpha}$ either 0 or 1. For a given $N_\uparrow$ and $N_\downarrow$, the corresponding domain includes all possible permutations of $N = N_\uparrow + N_\downarrow$ copies of 1 and $2L - N$ copies of 0, subject to the constraint that the first L (last L) digits of the sequence have $N_\uparrow$ ($N_\downarrow$) copies of 1. With this structure in mind, it is convenient to define $S_z = N_\uparrow - N_\downarrow$, a quantity that may or may not be conserved by the Hamiltonian (see §2.4). The full (labelled) basis set can then be generated using the following simple algorithm:

- If S_z is conserved:
    1. From S_z and N, compute $N_\uparrow$ and $N_\downarrow$
    2. From $N_\uparrow$ and $N_\downarrow$, programmatically obtain the list of all valid permutations
    3. Sort the list by binary values (e.g., $1010 > 0110 > 0101$)
    4. Finally, label each string according its position in the list
- If S_z is not conserved:
    1. Repeat the above steps 1–3 for each $S_z = -N,\ -N+2,\ \dots,\ N-2,\ N$ (combining the lists each time)
    2. Optionally, re-sort the large list (e.g., $0101 > 0100$)[1]
    3. As before, assign labels based on list index

[1]This step removes the block structure of the Hamiltonian. In *pyED*, it is turned off by default.

B params.yaml

References

[1] R.B. Lehoucq, D.C. Sorensen and C. Yang, *ARPACK Users' Guide* (Society for Industrial and Applied Mathematics, 1998) .

[2] H.Q. Lin, *Exact diagonalization of quantum-spin models*, *Phys. Rev. B* **42** (1990) 6561.

Parameter	**Type**	**Description**
	Run	
solver	`str`	The method used to diagonalize the Hamiltonian. Options are *full* or *Lanczos*: • *full*: convert sparse matrix to a NumPy array and call `np.linalg.eig`. This has poor scaling, so it is not recommended unless the total number of sites $L \leq 6$. • *Lanczos*: use the sparse matrix algorithm from `scipy.sparse.linalg.eigsh`.
num_ eigs	`int`	In the Lanczos solver, only the *num_ eigs* lowest-energy eigenstates will be found. If set to `null`, then the number of eigenstates is automatically set to half the dimension of the Hilbert space.
	Lattice	
lattice_ type	`str`	Name of either (1) a built-in type or (2) the Python script containing the neighbor function(s). *Do not include* `.py` *at the end!*
LX	`int`	Number of unit cells in the x-direction.
LY	`int`	Number of unit cells in the y-direction.
	Model	
model_ type	`str`	Name of either (1) a built-in type or (2) the Python script containing the couplings. *Do not include* `.py` *at the end!*
(couplings)	`float`	The values for all coupling parameters present in the model, e.g., t and U for `nnHubbard`.
	Conserved quantities	
N	`int`	Number of fermions, i.e., $N_\uparrow + N_\downarrow$. Options are the following: • If the model conserves N, set to one of $$\{0,\ \ldots,\ 2L\}$$ where $L = L_X \cdot L_Y \cdot N_{\text{orb}}$ (for half-filling, use $N = L$). • If the model does not conserve N, set to `null`. The Hamiltonian will include matrix elements that couple different sectors of N. Be mindful of system size scaling!
Sz	`int`	Net spin, i.e., $N_\uparrow - N_\downarrow$. Options are the following: • If the model conserves S_z, set to one of $$\{-N,\ -N+2,\ \ldots,\ N-2,\ N\}.$$ • If the model does not conserve S_z, set to `null`. The Hamiltonian will include matrix elements that couple different sectors of S_z. Be mindful of system size scaling!

Boundary conditions		
openX	`bool`	If `True`, remove all bonds between cell 0 and cell $L_X - 1$.
openY	`bool`	If `True`, remove all bonds between cell 0 and cell $L_Y - 1$.