The Journal of Open Source Software

`lattice-symmetries`: A package for working with quantum many-body bases

Tom Westerhout[1]

1 Institute for Molecules and Materials, Radboud University

DOI: 10.21105/joss.03537

Software

- Review
- Repository
- Archive

Editor: Jed Brown

Reviewers:

- @jeffhammond
- @joselado

Submitted: 26 May 2021
Published: 26 July 2021

Summary

Exact diagonalization (ED) is one of the most reliable and established numerical methods of quantum many-body theory. It is precise, unbiased, and general enough to be applicable to a huge variety of problems in condensed matter physics. Mathematically, ED is a linear algebra problem involving a matrix called Hamiltonian. For a system of spin-$1/2$ particles, the size of this matrix scales exponentially (as $\mathcal{O}(2^N)$) with the number of particles N.

Very fast scaling of memory requirements with system size is the main computational challenge of the method. There are a few techniques allowing one to lower the amount of storage used by the Hamiltonian. For example, one can store only the non-zero elements of the Hamiltonian. This is beneficial when the Hamiltonian is sparse which is usually the case in condensed matter physics. One can even take it one step further and avoid storing the matrix altogether by instead computing matrix elements on the fly.

A complementary approach to reduce memory requirements is to make use of system symmetries. For example, many relevant Hamiltonians possess $U(1)$ symmetry which permits one to perform calculations assuming that the number of particles (or number of spins pointing upwards), is fixed. Another example would be translational invariance of the underlying lattice.

Although the algorithms for dealing with lattice symmetries are well known (Sandvik et al., 2010), implementing them remains a non-trivial task. Here we present `lattice-symmetries`, a package providing high-quality and high-performance realization of these algorithms. Instead of writing their own optimized implementation for every system of interest, a domain expert provides system-specific details (such as the number of particles or momentum quantum number) to `lattice-symmetries` and it will automatically construct a reduced Hamiltonian. Dimension of the new Hamiltonian can be multiple orders of magnitude smaller than of the original one.

Furthermore, in `lattice-symmetries` the Hamiltonian itself is never stored. Instead, its matrix elements are computed on the fly which reduces the memory requirements even more. Care is taken to keep the implementation generic such that different physical systems can be studied, but without sacrificing performance as we will show in the next section.

All in all, `lattice-symmetries` serves as a foundation for building state-of-the-art ED and VMC (Variational Monte Carlo) applications. For example, `SpinED` (Westerhout, 2020) is an easy-to-use application for exact diagonalization which is built on top of `lattice-symmetries` and can handle clusters of up to $\mathcal{O}(42)$ spins on a single node.

Statement of need

Exact diagonalization is an old and well-established method and many packages have been written for it. However, we find that for some reason most state-of-the-art implementations

(Läuchli et al., 2019; Wietek & Läuchli, 2018) are closed-source. There are but three notable open-source projects which natively support spin systems[1] : HΦ (Kawamura et al., 2017), SPINPACK (Schulenburg, 2017), and QuSpin (Weinberg & Bukov, 2017).

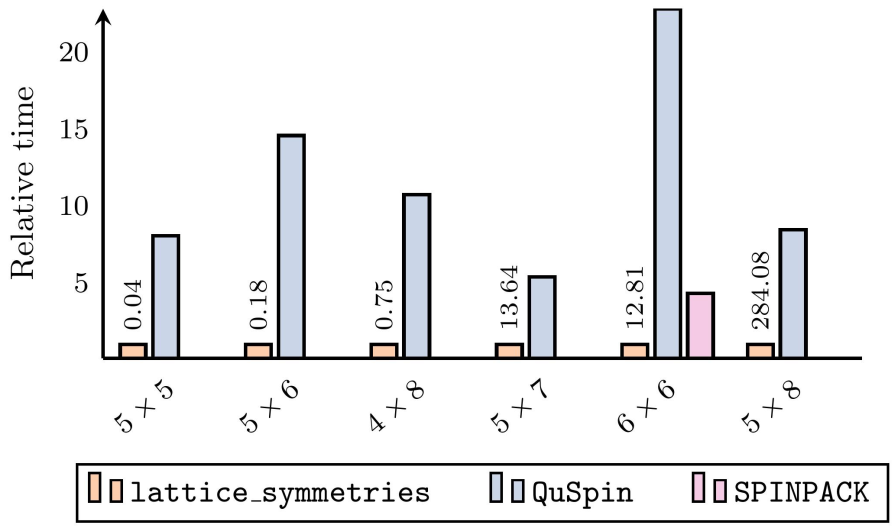

Figure 1: Performance of matrix-vector products in QuSpin, SPINPACK, and lattice-symmetries. For Heisenberg Hamiltonian on square lattices of different sizes, we measure the time it takes to do a single matrix-vector product. Timings for lattice-symmetries are normalized to 1 to show relative speedup compared to QuSpin, but for reference absolute times in seconds are listed as well. Depending on the system speedups over QuSpin vary between 5 and 22 times, but in all cases lattice-symmetries is significantly faster.

HΦ implements a variety of Hamiltonians, works at both zero and finite temperatures, and supports multi-node computations. However, there are a few points in which lattice-symmetries improves upon HΦ. Firstly, HΦ does not support arbitrary lattice symmetries. Secondly, it uses a custom input file format making it not user-friendly. Finally, since HΦ is an executable, it cannot be used to develop new algorithms.

SPINPACK is another popular solution for diagonalization of spin Hamiltonians. SPINPACK does support user-defined symmetries as opposed to HΦ, but its interface is even less user-friendly. Defining a lattice, Hamiltonian, and symmetries requires writing non-trivial amounts of C code. Finally, SPINPACK is slower than lattice-symmetries as illustrated in Figure 1.

QuSpin is much closer in functionality to lattice-symmetries. It is a high-level Python package which natively supports (but is not limited to) spin systems, can employ user-defined lattice symmetries, and can also perform matrix-free calculations (where matrix elements are computed on the fly). However, QuSpin mostly focuses on ease of use and functionality rather than performance. In lattice-symmetries we follow UNIX philosophy (Salus, 1994) and try to "do one thing but do it well." Even though lattice-symmetries uses essentially the same algorithms as QuSpin, careful implementation allows us to achieve an order of magnitude speedup as shown in Figure 1.

lattice-symmetries is a library implemented in C++ and C. It provides two interfaces:

- Low-level C interface which can be used to implement ED and VMC applications with focus on performance.

[1]There are a few projects targeting fermionic systems and the Hubbard model in particular. Although it is possible to transform a spin Hamiltonian into a fermionic one, it is impractical for large-scale simulations since lattice symmetries are lost in the new Hamiltonian.

- A higher-level Python wrapper which allows to easily test and prototype algorithms.

We make the library easily installable via Conda package manager.

The general workflow is as follows: the user starts by defining a few symmetry generators (ls_symmetry/Symmetry in C/Python) from which lattice-symmetries automatically constructs the symmetry group (ls_group/Group in C/Python). The user then proceeds to constructing the Hilbert space basis (ls_spin_basis/SpinBasis in C/Python). For some applications functionality provided by SpinBasis may be sufficient, but typically the user will construct one (or multiple) quantum mechanical operators (ls_operator/Operator in C/Python) corresponding to the Hamiltonian and various observables. Operator can be efficiently applied to vectors in the Hilbert space (i.e. wavefunctions). Also, in cases when the Hilbert space dimension is so big that the wavefunction cannot be written down explicitly (as a list of coefficients), Operator can be applied to individual spin configurations to implement Monte Carlo local estimators.

As an example of what can be done with lattice-symmetries, we implemented a standalone application for exact diagonalization studies of spin-1/2 systems: SpinED. By combining lattice-symmetries with PRIMME eigensolver (Stathopoulos & McCombs, 2010), it allows one to treat systems of up to $\mathcal{O}(42)$ sites on a single node. SpinED is distributed as a statically-linked executable — one can download one file and immediately get started with physics. All in all, it makes large-scale ED more approachable for non-experts.

Finally, we would like to note that lattice-symmetries and SpinED have already been used in a number of research projects (Astrakhantsev et al., 2021; Bagrov et al., 2020; Westerhout et al., 2020), and we feel that they could benefit many more. For example, lattice-symmetries is currently even being used to simulate quantum circuits.

Acknowledgements

The author thanks Nikita Astrakhantsev for useful discussions and testing of (most) new features. The assistance of Askar Iliasov and Andrey Bagrov with with group-theoretic aspects of this work is appreciated. This work was supported by European Research Council via Synergy Grant 854843 – FASTCORR.

References

Astrakhantsev, N., Westerhout, T., Tiwari, A., Choo, K., Chen, A., Fischer, M. H., Carleo, G., & Neupert, T. (2021). Broken-symmetry ground states of the Heisenberg model on the pyrochlore lattice. *arXiv Preprint arXiv:2101.08787*.

Bagrov, A. A., Iliasov, A. A., & Westerhout, T. (2020). Kinetic samplers for neural quantum states. *arXiv Preprint arXiv:2011.02986*.

Kawamura, M., Yoshimi, K., Misawa, T., Yamaji, Y., Todo, S., & Kawashima, N. (2017). Quantum lattice model solver HΦ. *Comput. Phys. Commun.*, *217*, 180–192. https://doi.org/10.1016/j.cpc.2017.04.006

Läuchli, A. M., Sudan, J., & Moessner, R. (2019). $S = \frac{1}{2}$ kagome Heisenberg antiferromagnet revisited. *Phys. Rev. B*, *100*(15, 15), 155142. https://doi.org/10.1103/physrevb.100.155142

Salus, P. H. (1994). *A quarter century of UNIX*. ACM Press/Addison-Wesley Publishing Co.

Sandvik, A. W., Avella, A., & Mancini, F. (2010). Computational studies of quantum spin systems. *AIP Conference Proceedings*, *1297*, 135–338. https://doi.org/10.1063/1.3518900

Schulenburg, J. (2017). SPINPACK. *Magdeburg University*. https://www-e.ovgu.de/jschulen/spin/

Stathopoulos, A., & McCombs, J. R. (2010). Primme: Preconditioned iterative multimethod eigensolver—methods and software description. *ACM Trans. Math. Softw.*, *37*(2), 1–30. https://doi.org/10.1145/1731022.1731031

Weinberg, P., & Bukov, M. (2017). QuSpin: A python package for dynamics and exact diagonalisation of quantum many body systems part i: Spin chains. *SciPost Phys.*, *2*(1, 1), 003. https://doi.org/10.21468/scipostphys.2.1.003

Westerhout, T. (2020). SpinED. In *GitHub repository*. https://github.com/twesterhout/spin-ed; GitHub.

Westerhout, T., Astrakhantsev, N., Tikhonov, K. S., Katsnelson, M. I., & Bagrov, A. A. (2020). Generalization properties of neural network approximations to frustrated magnet ground states. *Nat Commun*, *11*(1), 1–8. https://doi.org/10.1038/s41467-020-15402-w

Wietek, A., & Läuchli, A. M. (2018). Sublattice coding algorithm and distributed memory parallelization for large-scale exact diagonalizations of quantum many-body systems. *Phys. Rev. E*, *98*(3, 3), 033309. https://doi.org/10.1103/physreve.98.033309