QuantNBody: a Python package for quantum chemistry and physics to build and manipulate many-body operators and wave functions.

Saad Yalouz[*1], **Martin Rafael Gullin**[1], **and Sajanthan Sekaran**[1]

1 Laboratoire de Chimie Quantique, Institut de Chimie, CNRS/Université de Strasbourg, 4 rue Blaise Pascal, 67000 Strasbourg, France

DOI: 10.21105/joss.04759

Software

- Review
- Repository
- Archive

Editor: Jarvist Moore Frost

Reviewers:

- @wcwitt
- @erikkjellgren

Submitted: 05 August 2022
Published: 15 December 2022

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

The 'QuantNBody' package is a Python toolkit for quantum chemists/physicists interested in methods development to study quantum many-body problems ranging from electronic structure to condensed matter theory. It provides a quick and easy way to build matrix representations of bosonic and fermionic quantum many-body operators (*e.g.* Hamiltonians, spin or excitation operators) and get access to quantities/objects of interest (*e.g.* energies, reduced density matrices, many-body wave functions). The code includes various native functions and it is flexible enough to help users in building their own numerical tools to implement new methods.

Statement of need

The manipulation of many-body operators in the language of second quantization is a crucial step for accessing the properties of model or *ab initio* systems in quantum chemistry and physics. From a numerical point of view, this requires developing codes that can build matrix representations of quantum operators in a given quantum many-body basis (*e.g.* Hamiltonians, spin or excitation operators). In a great majority of cases, this aspect is kept as a ''blackbox'' in codes to spare the users from cumbersome numerical parts and to facilitate the use of already implemented methods. Nevertheless, this type of implementation can appear as a real obstacle for researchers in need of reliable numerical tools to quickly develop and test new methodologies based on second quantization algebra.

The 'QuantNBody' code was designed to answer this problem: it is a theoretician-friendly package which provides an efficient and simple-to-use numerical toolkit to help create and manipulate operators and wavefunctions related to quantum many-body systems (with either fermionic or bosonic particles). It allows straightforward implementations of second quantization into numerical objects (*i.e.* matrices) in order to facilitate the conversion of analytical developments into compact and easy to write numerical Python codes.

The user-friendly philosophy of QuantNBody regarding second quantization encoding can be compared (to some extent) to the FermiLib module originally developped for OpenFermion (McClean et al., 2020) (a quantum computing oriented package). In practice, the advantage of QuantNBody lies in its particle number conserving encoding which differs from what is done in OpenFermion (where the full Fockspace is encoded). As a result, QuantNBody allows the realization of larger scale calculations (up to 8 electrons in 8 spatial molecular orbitals in a few seconds). This new toolkit opens new perspectives for many-body simulations in Python and then completes the panel of already existing theoretician-friendly packages dedicated to many-body systems such as Quimb (Gray, 2018), Quspin (Weinberg & Bukov, 2017, 2019),

[*] yalouzsaad@gmail.com

QuTIP (J. Robert Johansson et al., 2012; J. R. Johansson et al., 2013), Yao (Luo et al., 2020) and OpenFermion (McClean et al., 2020).

Framework of the package

The QuantNbody package employs the SciPy Package (Virtanen et al., 2020) for the creation of sparse matrices representation of many-body operators. The Numba package (Lam et al., 2015) is also used to accelerate calculation when possible. The framework of the package lies in two fundamental ingredients. The first one is the creation of a reference many-body vector basis (based on a total number of quantum particles and modes/orbitals to fill) in which second quantization operators can be represented. The second ingredient consists in creating a general tool that can help build any particle-number conserving many-body operator which is: the single-body hopping operators $a_p^\dagger a_q$. Once these two ingredients have been created, the user can employ various pre-built functions in order to (i) construct different types of many-body operators (*e.g.* Hamiltonians, spin and excitation operators), (ii) manipulate/visualize quantum many-body states. All the native functions have been thought to facilitate calculations for new users and young students. Beyond this, the QuantNBody package has been also designed to provide flexibility to experimented users to develop their own tools to implement/test their own methods.

A quick illustration

As an illustration of the various types of system one can treat with the QuantNBody package, we describe here a few native many-body Hamiltonians which have already been implemented for fermionic and bosonic problems.

Fermionic systems

Different fermionic Hamiltonians can be straightforwardly built with the QuantNBody package. For example, a pre-built function of the package allows a quick implementation of the quantum chemistry *ab initio* molecular electronic structure Hamiltonian

$$\hat{H} = \sum_{p,q} h_{pq} \sum_{\sigma}^{\uparrow,\downarrow} a^\dagger_{p,\sigma} a_{q,\sigma} + \frac{1}{2} \sum_{p,q,r,s} g_{pqrs} \sum_{\sigma,\tau}^{\uparrow,\downarrow} a^\dagger_{p,\sigma} a^\dagger_{r,\tau} a_{s,\tau} a_{q,\sigma}, \tag{1}$$

where $a^\dagger_{i,\sigma}$ ($a_{i,\sigma}$) are the fermionic creation (annihilation) operators in the orbital i with spin σ. In a similar way, another native function has been also created to numerically build the Fermi-Hubbard Hamiltonian from condensed matter theory

$$\hat{H}_{FH} = -t \sum_{i,j} \sum_{\sigma}^{\uparrow,\downarrow} a^\dagger_{i,\sigma} a_{j,\sigma} + U \sum_{i} a^\dagger_{i,\uparrow} a_{i,\uparrow} a^\dagger_{i,\downarrow} a_{i,\downarrow}, \tag{2}$$

In practice, native functions are already implemented in the QuantNBody package to build both types of fermionic Hamiltonians. This implementation is based on the use of the already built single-body hopping operators mentioned earlier to generate the one- and two-body fermionic excitation operators (present in Eq. 2 and Eq. 1). Nevertheless, the one-/two-body integrals (*i.e.* h_{pq}, g_{pqrs} and t and U) have to be defined by the user. They can be set as pure parameters or obtained from external chemistry Python packages like PySCF (Sun et al., 2020) or Psi4 (Parrish et al., 2017). As an illustration, we show in Fig. 1 results one can produce with the package for both fermionic Hamiltonians. We focus here on the calculation of the ground state (noted $|\Psi_0\rangle$) in a H_2 molecule and a Fermi-Hubbard dimer, and evaluate several associated properties (energy and many-body basis decomposition).

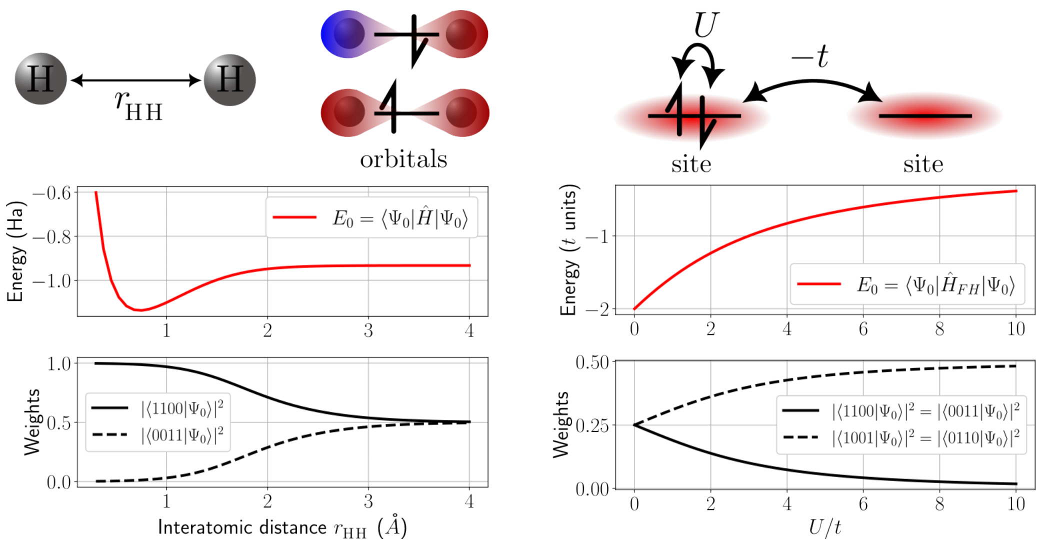

Figure 1: H_2 molecule and Fermi-Hubbard dimer. **Left column:** ground state energy and decomposition in the many-body basis for the dissociation of the H_2 molecule in a minimal basis (STO-3G) using integrals from Psi4 (Parrish et al., 2017). **Right column:** similar properties for the Fermi-Hubbard dimer as a function of U/t (2 electrons on 2 sites and $t = 1$).

Bosonic systems

Bosonic Hamiltonians can also be numerically built with the QuantNBody package. For example, a pre-built function allows a quick implementation of the Bose-Hubbard Hamiltonian

$$\hat{H}_{BH} = -t\sum_{i,j} a_i^\dagger a_j + U\sum_i a_i^\dagger a_i (a_i^\dagger a_i - 1), \tag{3}$$

where $a_i^\dagger$ (a_i) are now the bosonic creation (annihilation) operators of the local site i. Here again, the native function implementing the Bose-Hubbard Hamiltonian in the QuantNBody package manages on its own the building of all the one- and two-body bosonic operators *via* the already built single-particle hopping operators. The one-/two-body integrals (i.e. t and U) have to be defined by the user. As an illustration, we present in Fig. 2 the ground state properties one can obtain using the native function that implements the Bose-Hubbard Hamiltonian (two bosons on two sites).

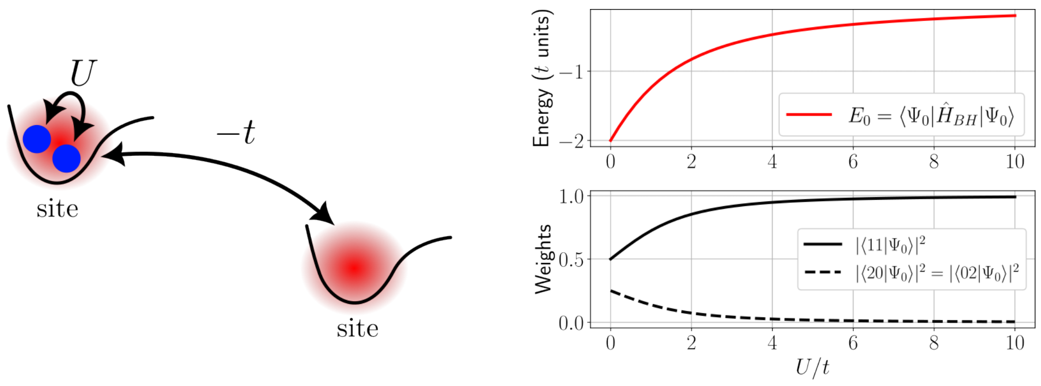

Figure 2: Bose-Hubbard dimer with two bosons. **Left column:** illustration of the system. **Right column:** ground state energy and its decomposition in the many-body basis for the Bose-Hubbard dimer as a function of U/t (with $t = 1$).

Related projects

The QuantNBody package is being currently used in the "Laboratoire de Chimie Quantique de Strasbourg" in several projects dedicated to strongly correlated systems. These projects including the study of spin properties in metal-ligand molecular systems (Roseiro et al., 2022), the development of embedding methods for large fermionic systems (Yalouz et al., 2022) and the development of orbital optimisation techniques for fermionic excited states (Yalouz & Robert, 2022) to cite but a few. As future developments, we plan to extend the capacities of the package to the treatment of hybrid systems mixing both fermions and bosons degrees of freedom (*e.g.* for polaritonic chemistry or exciton-phonon systems).

Acknowledgements

Saad Yalouz acknowledges support from the Interdisciplinary Thematic Institute ITI-CSC *via* the IdEx Unistra (ANR-10-IDEX-0002) within the program Investissement d'Avenir.

References

Gray, J. (2018). Quimb: A python library for quantum information and many-body calculations. *Journal of Open Source Software*, *3*(29), 819. https://doi.org/10.21105/joss.00819

Johansson, J. Robert, Nation, P. D., & Nori, F. (2012). QuTiP: An open-source python framework for the dynamics of open quantum systems. *Computer Physics Communications*, *183*(8), 1760–1772. https://doi.org/10.1016/j.cpc.2012.02.021

Johansson, J. R., Nation, P. D., & Nori, F. (2013). QuTiP 2: A python framework for the dynamics of open quantum systems. *Computer Physics Communications*, *184*(4), 1234–1240. https://doi.org/10.1016/j.cpc.2012.11.019

Lam, S. K., Pitrou, A., & Seibert, S. (2015). Numba: A llvm-based python jit compiler. *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, 1–6.

Luo, X.-Z., Liu, J.-G., Zhang, P., & Wang, L. (2020). Yao. Jl: Extensible, efficient framework for quantum algorithm design. *Quantum*, *4*, 341. https://doi.org/10.22331/q-2020-10-11-341

McClean, J. R., Rubin, N. C., Sung, K. J., Kivlichan, I. D., Bonet-Monroig, X., Cao, Y., Dai, C., Fried, E. S., Gidney, C., Gimby, B., & others. (2020). OpenFermion: The electronic structure package for quantum computers. *Quantum Science and Technology*, *5*(3), 034014. https://doi.org/10.1088/2058-9565/ab8ebc

Parrish, R. M., Burns, L. A., Smith, D. G., Simmonett, A. C., DePrince III, A. E., Hohenstein, E. G., Bozkaya, U., Sokolov, A. Y., Di Remigio, R., Richard, R. M., & others. (2017). Psi4 1.1: An open-source electronic structure program emphasizing automation, advanced libraries, and interoperability. *Journal of Chemical Theory and Computation*, *13*(7), 3185–3197. https://doi.org/10.1021/acs.jctc.7b00174

Roseiro, P., Brook, D. J., Amor, N. B., Robert, V., & Yalouz, S. (2022). Excited state spinmerism in high-field fe (II)-verdazyl molecular complex: Versatile local spins for quantum information. *arXiv Preprint arXiv:2210.02325*. https://doi.org/10.48550/arXiv.2210.02325

Sun, Q., Zhang, X., Banerjee, S., Bao, P., Barbry, M., Blunt, N. S., Bogdanov, N. A., Booth, G. H., Chen, J., Cui, Z.-H., & others. (2020). Recent developments in the PySCF program package. *The Journal of Chemical Physics*, *153*(2), 024109. https://doi.org/10.1063/5.0006074

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2

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

Weinberg, P., & Bukov, M. (2019). QuSpin: A python package for dynamics and exact diagonalisation of quantum many body systems. Part II: Bosons, fermions and higher spins. *SciPost Physics*, *7*(2), 020. https://doi.org/10.21468/SciPostPhys.7.2.020

Yalouz, S., & Robert, V. (2022). Orthogonally constrained orbital optimization: Assessing changes of optimal orbitals for orthogonal multi-reference states. *arXiv Preprint arXiv:2211.08329*. https://doi.org/10.48550/arXiv.2211.08329

Yalouz, S., Sekaran, S., & Saubanère, M. (2022). Quantum embedding of multi-orbital fragments using the block-householder-transformation. *arXiv Preprint arXiv:2209.10302*. https://doi.org/10.48550/arXiv.2209.10302