The Journal of Open Source Software

NLSE: A Python package to solve the nonlinear Schrödinger equation

Tangui Aladjidi [1¶], **Clara Piekarski** [1], **and Quentin Glorieux** [1]

1 Laboratoire Kastler Brossel, Sorbonne University, CNRS, ENS-PSL University, Collège de France; 4 Place Jussieu, 75005 Paris, France ¶ Corresponding author

DOI: 10.21105/joss.06607

Software

- Review
- Repository
- Archive

Editor: Rocco Meli

Reviewers:

- @Abinashbunty
- @obliviateandsurrender

Submitted: 19 March 2024
Published: 22 July 2024

Summary

The non-linear Schrödinger equation (NLSE) is a general non-linear equation allowing to model the propagation of light in non-linear media. This equation is mathematically isomorphic to the Gross-Pitaevskii equation (GPE) (Pitaevskij & Stringari, 2016) describing the evolution of cold atomic ensembles. Recently, the growing field of quantum fluids of light (Carusotto & Ciuti, 2013) has proven a fruitful testbed for several fundamental quantum and classical phenomena such as superfluidity (Michel et al., 2018) or turbulence (Baker-Rasooli et al., 2023). Providing a flexible, modern and performant framework to solve these equations is crucial to model realistic experimental scenarios.

Statement of need

`Numpy` (Harris et al., 2020) is the default package for array operations in Python, which is the language of choice for most physicists. However, the performance of `Numpy` for large arrays and Fourier transforms quickly bottlenecks homemade implementations of popular solvers like the split-step spectral scheme.

Over the years, there have been several packages striving to provide performant split-step solvers for NLSE type equations. Here are a few examples:

- `FourierGPE.jl` for 1D to 3D Gross-Pitavskii equations in the context of cold atoms in Julia.
- `GPUE` (Schloss & O'Riordan, 2018) for 1D to 3D Gross-Pitaevskii equations accelerated on GPU, in C++ (currently unmaintained).
- `py-fmas` for 1D NLSE in optical fibers, with a split-step method (currently unmaintained).

With our project, we bring similar performance to C++ and Julia implementations, while striving for accessibility and maintainability by using Python. Using an easy to extend object-oriented classes, users can readily input experimental parameters to quickly model real setups.

Functionality

`NLSE` harnesses the power of pseudo-spectral schemes to solve efficiently the following general type of equation:

$$i\partial_t\psi = -\frac{1}{2m}\nabla^2\psi + V\psi + g|\psi|^2\psi.$$

To take advantage of the computing power of modern Graphics Processing Units (GPUs) for Fast Fourier Transforms (FFTs), the main workhorse of this code is the `Cupy` (Okuta et al.,

2017) package that maps Numpy functionalities onto the GPU using NVIDIA's CUDA API. It also heavily uses just-in-time compilation using Numba (Lam et al., 2015) to optimize performance while having an easily maintainable Python codebase. Compared to naive Numpy-based CPU implementations, this package provides a 100 to 10000 times speedup for typical sizes Figure 2. While optimized for the use with GPU, NLSE also provides a performant CPU fallback layer.

The goal of this package is to provide a natural framework to model the propagation of light in non-linear media or the temporal evolution of Bose gases. It can also be used to model the propagation of light in general. It supports lossy, non-linear and non-local media.

It provides several classes to model 1D, 2D or 3D propagation, and leverages the array functionalities of Numpy like broadcasting to allow scans of physical parameters to most faithfully replicate experimental setups. The typical output of a simulation run is presented in Figure 1.

This code has been developed initially developed in Aladjidi (2023) and used as the main simulation tool for several publications like Glorieux et al. (2023) and Baker-Rasooli et al. (2023).

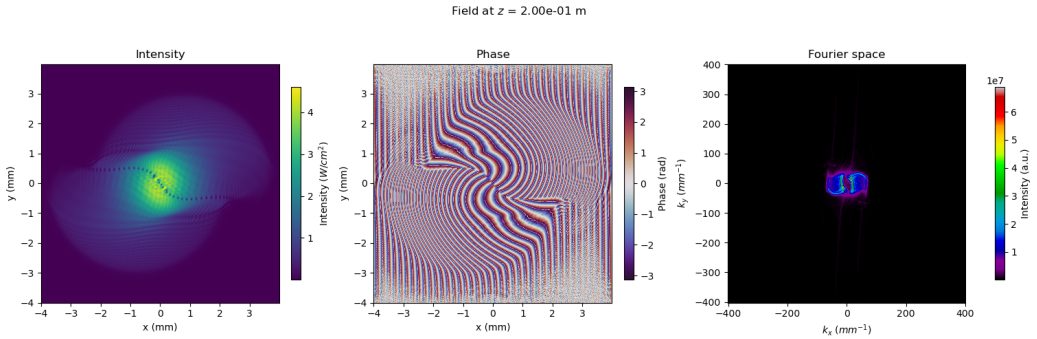

Figure 1: Example of an output of the solver. A shearing layer is observed nucleating vortices, that are attracted towards the center due to an attractive potential. The density and phase of the field are represented as well as the momentum distribution get a quick overview of the state of the field.

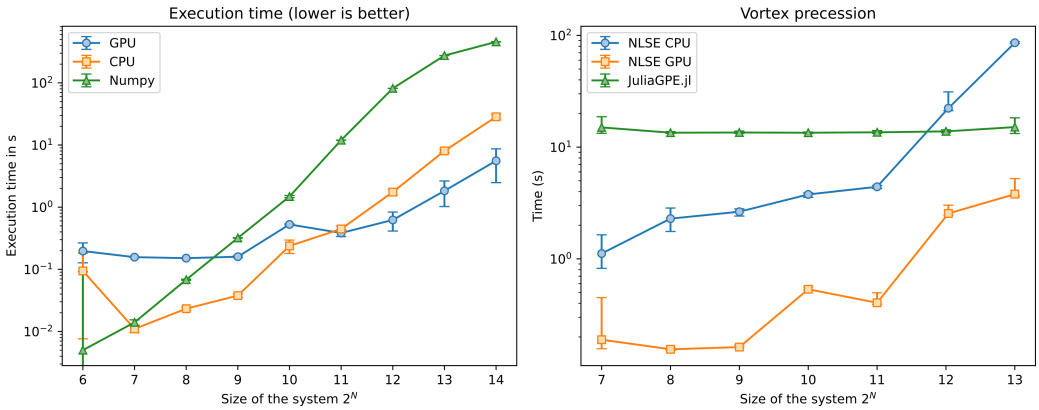

Figure 2: Left: CPU vs GPU vs Numpy benchmark for 1 cm of propagation (200 evolution steps).Right: Comparison versus the `JuliaGPE.jl` package on the study of vortex precession.

Reproducibility

The code used to generate the figures can be found in the `examples` folder of the repository with the `fig1_turbulence.py` and `fig2_benchmarks.py` scripts. Note that you will need a

Julia install to run the `JuliaGPE.jl` script.

Acknowledgements

We acknowledge contributions from Myrann Baker-Rasooli as our most faithful beta tester.

Authors contribution

TA wrote the original code and is the main maintainer, CP extended the functionalities to include coupled systems. QG supervised the project.

References

Aladjidi, T. (2023). *Full optical control of quantum fluids of light in hot atomic vapors* (Theses No. 2023SORUS406, Sorbonne Université). https://doi.org/10.5281/zenodo.12698001

Baker-Rasooli, M., Liu, W., Aladjidi, T., Bramati, A., & Glorieux, Q. (2023). Turbulent dynamics in a two-dimensional paraxial fluid of light. *Physical Review A*, *108*(6), 063512. https://doi.org/10.1103/PhysRevA.108.063512

Carusotto, I., & Ciuti, C. (2013). Quantum fluids of light. *Rev. Mod. Phys.*, *85*(1), 299–366. https://doi.org/10.1103/RevModPhys.85.299

Glorieux, Q., Aladjidi, T., Lett, P. D., & Kaiser, R. (2023). Hot atomic vapors for nonlinear and quantum optics. *New Journal of Physics*, *25*(5), 051201. https://doi.org/10.1088/1367-2630/acce5a

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

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. https://doi.org/10.1145/2833157.2833162

Michel, C., Boughdad, O., Albert, M., Larré, P.-É., & Bellec, M. (2018). Superfluid motion and drag-force cancellation in a fluid of light. *Nat. Comm.*, *9*(1), 2108. https://doi.org/10.1038/s41467-018-04534-9

Okuta, R., Unno, Y., Nishino, D., Hido, S., & Loomis, C. (2017). CuPy: A NumPy-compatible library for NVIDIA GPU calculations. *Proceedings of Workshop on Machine Learning Systems (LearningSys) in the Thirty-First Annual Conference on Neural Information Processing Systems (NIPS)*. http://learningsys.org/nips17/assets/papers/paper_16.pdf

Pitaevskij, L. P., & Stringari, S. (2016). *Bose-einstein condensation and superfluidity*. Oxford University Press. https://doi.org/10.1017/cbo9780511524240.005

Schloss, J. R., & O'Riordan, L. J. (2018). GPUE: Graphics processing unit gross–pitaevskii equation solver. *Journal of Open Source Software*, *3*(32), 1037. https://doi.org/10.21105/joss.01037