Reply to Referee 1

Comment A.1: *For each of the examples studied the authors should present a study of how well their algorithm converges to the correct result with eg the size of the grid, along with the required computing resources.*

Response: The Referee is correct in pointing out that we didn't mention the topic of convergence in our results. Indeed, we have verified that all results presented in the paper have reached convergence. Specifically, for Figure 4, we display hereunder the convergence of $|\alpha|$ at $V_0 = 6E_r$ as a function of the grid's linear extent L. We fix the ratio $r = N/L = 2^{10}/30\mu$m, where $N = N_x = N_y$ is the grid size. The results presented in the main text are taken at $L = 30\mu$m and $N = 2^{10}$. As can be observed from Fig. 1, for sizes smaller than the physical extent of the cloud the integration fails as spurious interference effects dominate the numerics. This manifests as a spuriously vanishing amplitude of the emitted field. Upon increasing L past the physical spatial extension of the cloud (dashed red line) convergence is gradually reached.

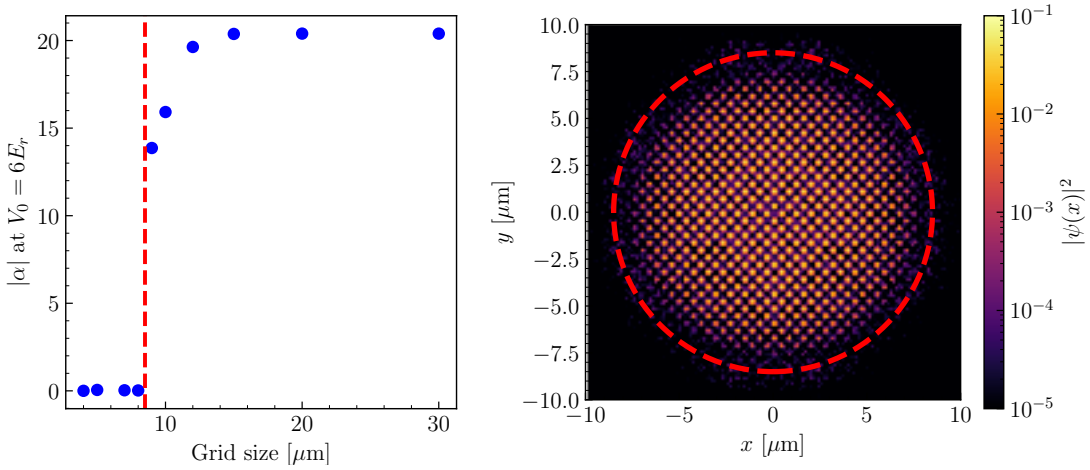

FIG. 1. Convergence of self-organization phase transition. (a) Amplitude of the emitted field as a function of the linear spacial extent L of the grid. (b) Density profile of the atomic cloud at convergence ($L = 20\mu m$). We identify in red the spatial extension of the cloud beyond which $|\psi(x)|^2 < 10^{-4}$.

In the implementation of Kapitza-Dirac diffraction, shown in Figure 3, we focused on the convergence with respect to the time step used in the simulation while keeping the grid size fixed. In this case, it is clear that a grid size large enough to support up to $4k$ peaks scattering is necessary. We observe that the accuracy does not significantly improve beyond this grid size. Moreover, the short duration of the simulation limits the possibility for long-term comparisons.

However, if the time step is not chosen carefully, numerical errors can accumulate during the time evolution. In Figure 2, we present the differences observed for a fixed grid size $N = 512$ with varying time steps, compared to the most precise time step of $dt = 1 \times 10^{-8}$. The results demonstrate that our choice of time step is reasonable.

We do not accompany these plots with the computing resources required for each point as these are already detailed in Figure 2 of the main text. We have verified that they accurately predict the walltime of the presented simulations.

Changes:

- Added commments on convergence to Figure 4.
- Added a comment on converegence to Figure 3.

Comment A.2: *It would also be useful to add some comparisons to other libraries. How much more efficient is the code here?*

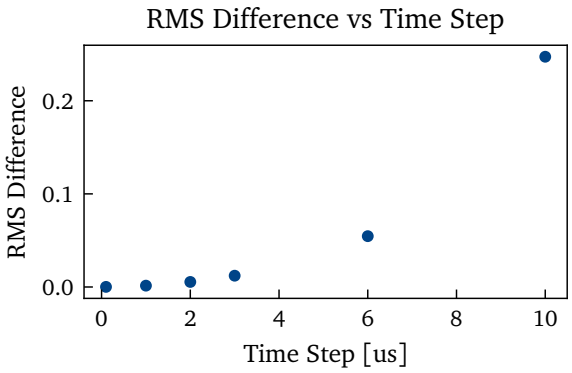

FIG. 2. Root mean square error of momentum peak occupations at different time steps compared to a run with a time step of $dt = 1 \times 10^{-8}$. The momentum occupation was calculated by summing the region of interest (ROI) in momentum space after each time step throughout the evolution. For all results, a grid size of 512 was used, allowing the simulation to include momenta higher than $4k$.

Response: We thank the Referee for this comment. We agree that a comparison to another library would effectively highlight the performance improvements offered by our package. Given its popularity, we chose to compare it to the `GPELab` library in MATLAB, specifically focusing on its *Backward Euler pseudoSPectral* (BESP) scheme [1].

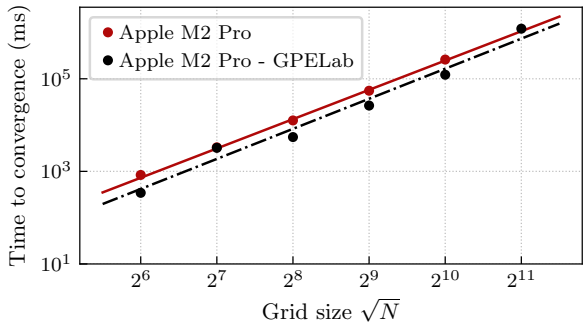

FIG. 3. Comparison of the wall-clock time required by `TorchGPE` and `GPELab` to reach convergence in the simulations of Fig. 2 in the main text.

Although there are some similarities between the BESP method implemented in `GPELab` and the split-step Fourier algorithm included in `TorchGPE`, these are distinct algorithms that may require different numbers of iterations to achieve convergence. To ensure a fair comparison, we allowed both software packages to run the simulations in Fig. 2 of the main text for an unlimited number of iterations, stopping only when the energy converged to a stationary point.

In figure 3 we report the runtime for both codes, showing that the convergence time on CPU are indeed comparable. However, as highlighted in the main text, `GPELab` does not support GPU execution and therefore cannot benefit from the significant speed-up that our package demonstrates when running on these processors.

Changes:

- In response to the Referee's suggesion, we have now included a comparison with the `GPELab` library in MATLAB. Figure 2 in the main text has been updated to include the comparison data, and Sec. 4.1 in the main text has been extended with comments on the results obtained.

Comment A.3: *The link at the end of the paper does not have the correct address*

Response: We thank the Referee for pointing us to the broken link.

Changes:

- The link has now been updated in the revised version.

---

[1] X. Antoine and R. Duboscq, Computer Physics Communications **185**, 2969 (2014).