quTARANG: A High-performance computing Python package to study turbulence using the Gross-Pitaevskii equation

Sachin Singh Rawat[a,*], Shawan Kumar Jha[b], Mahendra Kumar Verma[a], Pankaj Kumar Mishra[b]

[a]*Department of Physics, Indian Institute of Technology Kanpur, Kanpur 208016, India*
[b]*Department of Physics, Indian Institute of Technology Guwahati, Guwahati 781039, India*

Abstract

We present quTARANG, a robust GPU-accelerated Python package developed for a comprehensive study of turbulence problems in Bose-Einstein condensates (BECs). It solves the mean-field Gross-Pitaevskii equation (GPE) using a Time-splitting pseudo-spectral scheme. An imaginary time approach is used to obtain the ground state of the system. quTARANG also has post-processing tools that can compute different statistical properties of turbulent Bose-Einstein condensates, such as kinetic energy spectra, particle number spectrum and corresponding fluxes. This paper provides detailed descriptions of the code, along with specific examples for calculating the ground state and turbulent state of the condensate under different initial conditions for both 2-D and 3-D cases. We also present results on the dynamics of the GPE in 2-D and 3-D used to validate our code. Finally, we compare the performance of quTARANG on different GPUs to its performance on a CPU, demonstrating the speedup achieved on various GPU architectures.

Keywords: Quantum turbulence, Bose-Einstein condensates, Quantum fluids, Gross-Pitaevskii Equation Solver, High-performance computing.

PROGRAM SUMMARY

Program Title: **quTARANG**
CPC Library link to program files: (to be added by Technical Editor)
Developer's repository link: `https://github.com/sachinrawat2207/quTARANG`
Code Ocean capsule: (to be added by Technical Editor)
Licensing provisions: MIT
Programming language: Python
Nature of problem: This software is designed to compute the ground state and dynamical evolution of the Gross-Pitaevskii equation for 2-D and 3-D cases with GPU acceleration.
Solution method: We have used a Time-splitting pseudo-spectral (TSSP) scheme to compute the dynamics of the condensate and the ground state of the system is obtained by minimizing the energy functional through the evolution of GPE in imaginary time.

1. Introduction

Turbulence is a phenomenon associated with the chaotic nature of flows in space and time arising from the nonlinear nature of the interactions. Turbulence in classical fluids, best characterized by the Navier-Stokes equation, remains unresolved to this day. The complex nonlinear interactions at various length and time scales make it a difficult problem to handle analytically as well as numerically. A recent approach to shed some light on this long-standing problem has been the study of turbulence in quantum fluids. The zero viscosity and presence of quantized vortices distinguish quantum fluid systems like Bose-Einstein condensates (BECs) from their classical counterparts. BEC is a state of matter where a large fraction of particles in a dilute Bose gas occupy the ground state upon cooling to a very low temperature and thus can be represented by a macroscopic wave function. The theoretical idea of BECs was first

*Corresponding author.
E-mail address: sachinr@iitk.ac.in

conceived by S.N. Bose and Albert Einstein almost a century ago, while the first experimental realization took place in 1995 using ^{87}Rb [1] and ^{23}Na atoms [2]. Since then, BECs have been realized using different atoms, molecules and photons [2–5] making it a viable candidate for the exploration of a large variety of phenomena at the macroscopic scale.

In recent years, the mean-field Gross-Pitaevskii model has been considered quite widely to simulate the ground state and dynamics of weakly interacting BECs. Solving these equations to investigate different states of BECs is quite challenging. Several open-source software packages, such as GPELab [6, 7], a sequential code in Matlab, GPUE [8], a CUDA based code and a Fortran code [9] are available which can simulate the ground state and dynamical features of BECs. However, a code that can be used for a comprehensive study of different turbulent states of BECs lacks in the literature.

In this paper, we present detailed features of quTARANG, a Python package that can efficiently compute the ground state structure and can simulate the dynamics of different complex turbulent BECs in 2-D and 3-D. To solve the GPE, we utilize a Time-splitting pseudo-spectral scheme (TSSP) [10]. We also provide post-processing tools to calculate different statistical quantities of interest, such as spectra and fluxes that characterize the turbulent BECs. The package is CPU-GPU agnostic allowing a significant improvement in performance for large grid sizes on GPUs. We demonstrate the effectiveness of our code by comparing its performance on a CPU to different GPUs and also with the CUDA package GPUE [8].

The paper is organized as follows. Section 2 introduces the mean-field Gross-Pitaevskii model used in our code to study the ground state and the dynamics in particular the turbulent states of BECs. We also discuss the non-dimensionalization scheme used within the code to solve the GPE. In section 3, we present the numerical scheme implemented in quTARANG. Section 4 provides a comprehensive exploration of quTARANG's functionalities. Section 5 presents the validation of quTARANG with different sources in the literature. In section 6, we provide examples for different use cases of quTARANG while in section 7, a performance analysis as well as benchmarking of the code with GPUE has been presented. Finally, in section 8, we conclude the paper.

2. The Mean-field model: Gross-Pitaevskii Equation

At temperature T much lower than the critical temperature T_c for condensation, the dynamics of BECs can be well described by a single macroscopic wave function $\psi(\boldsymbol{r}, t)$ which is governed by the mean-field Gross-Pitaevskii equation (GPE) [11, 12] given by

$$i\hbar\partial_t\psi(\boldsymbol{r},t) = -\frac{\hbar^2}{2m}\nabla^2\psi(\boldsymbol{r},t) + V(\boldsymbol{r},t)\psi(\boldsymbol{r},t) + U_0|\psi(\boldsymbol{r},t)|^2\psi(\boldsymbol{r},t), \tag{1}$$

where $V(\boldsymbol{r}, t)$ is the trapping potential, $U_0 = 4\pi\hbar^2 a_s/m$ is the strength of the interaction between atoms, m is the mass of an atom and a_s is the scattering length of the atoms. The wavefunction in the GPE model obeys the normalization condition, $\int |\psi(\boldsymbol{r}, t)|^2 = N$, where N is the total number of particles in the system. The total energy of the system in this model is given by

$$E = \int \left[\frac{\hbar^2}{2m}|\nabla\psi(\boldsymbol{r},t)|^2 + V(\boldsymbol{r},t)|\psi(\boldsymbol{r},t)|^2 + \frac{1}{2}U_0|\psi(\boldsymbol{r},t)|^4\right] d^3\boldsymbol{r}. \tag{2}$$

The total number of particles and energy remain conserved during the time evolution of the condensate.

The ground state solutions of the GPE (1) are obtained by considering $\psi(\boldsymbol{r}, t) = \varphi(\boldsymbol{r})\exp(-i\mu t/\hbar)$ in Eq. (1), where μ is the chemical potential of the system. This leads to the time-independent GPE given by

$$-\frac{\hbar^2}{2m}\nabla^2\varphi(\boldsymbol{r}) + V(\boldsymbol{r})\varphi(\boldsymbol{r}) + U_0|\varphi(\boldsymbol{r})|^2\varphi(\boldsymbol{r}) = \mu\varphi(\boldsymbol{r}). \tag{3}$$

The chemical potential μ can be obtained by using Eq. (3) as

$$\mu = \frac{1}{N}\int \left[\frac{\hbar^2}{2m}|\nabla\varphi(\boldsymbol{r})|^2 + V(\boldsymbol{r})|\varphi(\boldsymbol{r})|^2 + U_0|\varphi(\boldsymbol{r})|^4\right] d^3\boldsymbol{r}. \tag{4}$$

The conservation of the particle number N results in a continuity equation for the particle density (ρ) given by

$$\partial_t \rho + \nabla . \boldsymbol{j} = 0. \tag{5}$$

Here $\boldsymbol{j}$ is the particle number current density:

$$\boldsymbol{j} = \frac{i\hbar}{2m} (\psi \nabla \psi^* - \psi^* \nabla \psi) = \rho \boldsymbol{v}, \tag{6}$$

where $\boldsymbol{v} = \hbar \nabla \theta / m$ is the fluid velocity. The particle number density and phase θ can be obtained from the wavefunction by the Madelung transformation,

$$\psi(\boldsymbol{r}, t) = \sqrt{\rho(\boldsymbol{r}, t)} e^{i\theta(\boldsymbol{r},t)}. \tag{7}$$

2.1. Dimensionless GPE

We have non-dimensionalized Eq. (1) by using the characteristic length and time scales corresponding to the harmonic trap, $V(\boldsymbol{r}) = \frac{1}{2} m(\omega_x^2 x^2 + \omega_y^2 y^2 + \omega_z^2 z^2)$. We have chosen the inverse of oscillator frequency ω_m^{-1} as the characteristic time scale, oscillator length $a_0 = \sqrt{\hbar / m\omega_m}$ as the characteristic length scale, and the harmonic oscillator ground state energy $\hbar\omega_m$ as the characteristic energy scale [9], where ω_m is $\min(\omega_x, \omega_y, \omega_z)$. Using the dimensionless variables $t' = \omega_m t$, $\boldsymbol{r}' = \boldsymbol{r}/a_0$ and $\psi' = a_0^{3/2} \psi$ in Eq. (1) and then removing all ($'$), the GPE takes the following dimensionless form,

$$i\partial_t \psi(\boldsymbol{r}, t) = -\frac{1}{2} \nabla^2 \psi(\boldsymbol{r}, t) + V(\boldsymbol{r}) \psi(\boldsymbol{r}, t) + g |\psi(\boldsymbol{r}, t)|^2 \psi(\boldsymbol{r}, t), \tag{8}$$

where

$$V(\boldsymbol{r}) = \frac{1}{2} (\gamma_x^2 x^2 + \gamma_y^2 y^2 + \gamma_z^2 z^2), \quad \gamma_{x,y,z} = \omega_{x,y,z} / \omega_m, \quad g = (4\pi a_s U_0)/a_0. \tag{9}$$

The scaled energy of the system for dimensionless GPE (2) is given by

$$E = \int \left[\frac{1}{2} |\nabla \psi(\boldsymbol{r}, t)|^2 + V |\psi(\boldsymbol{r}, t)|^2 + \frac{1}{2} g |\psi(\boldsymbol{r}, t)|^4 \right] d^3 \boldsymbol{r}. \tag{10}$$

The time-independent non-dimensionalized GPE takes the form,

$$-\frac{1}{2} \nabla^2 \varphi(\boldsymbol{r}) + V(\boldsymbol{r}) \varphi(\boldsymbol{r}) + g |\varphi(\boldsymbol{r})|^2 \varphi(\boldsymbol{r}) = \mu \varphi(\boldsymbol{r}). \tag{11}$$

and the chemical potential μ is given by

$$\mu = \frac{1}{N} \int \left[\frac{1}{2} |\nabla \varphi(\boldsymbol{r})|^2 + V |\varphi(\boldsymbol{r})|^2 + g |\varphi(\boldsymbol{r})|^4 \right] d^3 \boldsymbol{r}. \tag{12}$$

For the case of a disc-shaped trap with strong axial binding ($\omega_x \approx \omega_y$, and $\omega_z \gg \omega_x, \omega_y$), the excitation along the $z-$ direction gets highly suppressed. As a result of this, the system remains in the ground state along the $z-$ direction, and the condensate can be treated as a quasi-2-D system. For this trap, the dynamical evolution of the condensate is confined in the $x - y$ plane [13]. Thus, the wavefunction for this trap can be written as $\psi(x, y, z, t) = \phi(x, y, t) \phi_0(z, \gamma_z) e^{-i\gamma_z t/2}$ with $\phi_0(z, \gamma_z) = (\gamma_z / \pi)^{1/4} e^{-\gamma_z z^2/2}$. Using this wavefunction in Eq. (8) and further multiplying the equation with $\phi_0^*(z)$ and integrating over the whole range of z, the equation reduces to the GPE in two dimensions given by

$$i\partial_t \phi(x, y) = -\frac{1}{2} (\partial_{xx} + \partial_{yy}) \phi(x, y, t) + \frac{1}{2} (\gamma_x^2 x^2 + \gamma_y^2 y^2) \phi(x, y, t) + g_{2D} |\phi(x, y, t)|^2 \phi(x, y, t), \tag{13}$$

with $g_{2D} = 2a_s U_0 \sqrt{2\pi\gamma_z} / a_0$ and normalization $\int |\phi(x, y, t)|^2 dx dy = N$.

Using highly elongated cigar-shaped trap ($\omega_y, \omega_z \gg \omega_x$), quasi 1-D systems can be realized in the laboratory. The dynamics in such systems occur only along the $x-$ direction. However, the condensate remains in the ground state along the $y-$ and $z-$ directions. The wavefunction for such a system can be expressed as $\psi(x, y, z, t) = \phi(x, t) \phi_0(y, \gamma_y) \phi_0(z, \gamma_z) e^{-i\gamma_y t/2} e^{-i\gamma_z t/2}$, where ϕ_0 follows the same expression as the 2D case. As we use this ansatz in Eq. (8) and subsequently multiply the equation with $\phi_0^*(y, \gamma_y)\phi_0^*(z, \gamma_z)$ and integrating the equation over y and z, we obtain GPE in one dimension,

$$i\partial_t\phi(x,t) = -\frac{1}{2}\partial_{xx}\phi(x,t) + \frac{1}{2}\gamma_x^2 x^2\phi(x,t) + g_{1D}|\phi(x,t)|^2\phi(x,t), \tag{14}$$

with $g_{1D} = 2a_s U_0\sqrt{\gamma_y\gamma_z}/a_0$ and normalization $\int|\phi(x,t)|^2 dx = N$.

2.2. Energy decomposition

The total energy, given by the Eq. (10), can be decomposed into four components [14] using the Madelung transformation (7),

$$E = \int\left[\frac{1}{2}\rho(\boldsymbol{r},t)|\boldsymbol{v}(\boldsymbol{r},t)|^2 + \frac{1}{2}|\nabla\sqrt{\rho(\boldsymbol{r},t)}|^2 + \rho(\boldsymbol{r},t)V(\boldsymbol{r},t) + \frac{1}{2}g\rho^2(\boldsymbol{r},t)\right]d^3\boldsymbol{r}, \tag{15}$$

where the four terms in Eq. (15) correspond to the kinetic, quantum, potential and interaction energy of the system respectively viz.

$$E_{kin} = \int\frac{1}{2}\rho(\boldsymbol{r},t)|\boldsymbol{v}(\boldsymbol{r},t)|^2 d^3\boldsymbol{r}, \tag{16}$$

$$E_{qnt} = \int\frac{1}{2}|\nabla\sqrt{\rho(\boldsymbol{r},t)}|^2 d^3\boldsymbol{r}, \tag{17}$$

$$E_{pot} = \int\rho(\boldsymbol{r},t)V(\boldsymbol{r},t)d^3\boldsymbol{r}, \tag{18}$$

$$E_{int} = \int\frac{1}{2}g\rho(\boldsymbol{r},t)^2 d^3\boldsymbol{r}, \tag{19}$$

The kinetic energy of the system can be further decomposed into a compressible and an incompressible part by decomposing the effective velocity $\boldsymbol{w}(\boldsymbol{r},t) = \sqrt{\rho(\boldsymbol{r},t)}\boldsymbol{v}(\boldsymbol{r},t)$ into a divergence-free component ($\boldsymbol{w}^i$) and a potential component ($\boldsymbol{w}^c$) through Helmholtz decomposition [15] given by

$$E_{kin}^{[i,c]} = \int\frac{1}{2}|\boldsymbol{w}(\boldsymbol{r},t)^{[i,c]}|^2 d^3\boldsymbol{r}. \tag{20}$$

2.3. Spectra and fluxes

The spectra and fluxes of the energy components and the particle number density are important quantities used to analyze turbulent behavior in BECs. The incompressible ($\varepsilon_{kin}^i(k)$) and compressible ($\varepsilon_{kin}^c(k)$) kinetic energy spectra of the system are defined as

$$E_{kin}^{\{i,c\}}(t) = \frac{1}{2}\int\left|\mathcal{F}_{\boldsymbol{k}}\left[\boldsymbol{w}^{\{i,c\}}\right]\right|^2 d^3\boldsymbol{k} = \int\varepsilon_{kin}^{\{i,c\}}(k)dk, \tag{21}$$

where $\boldsymbol{w}^{\{i,c\}} = \sqrt{\rho}\boldsymbol{v}^{\{i,c\}}$ represents the density-weighted velocity field of the system corresponding to the incompressible and compressible components and $\mathcal{F}_{\boldsymbol{k}}[\zeta]$ represents the Fourier transformation of a given field ζ. Furthermore, the quantum energy ($\varepsilon_{qnt}(k)$) and particle number ($N(k)$) spectra are defined as

$$E_{qnt}(t) = \frac{1}{2}\int\left|\mathcal{F}_{\boldsymbol{k}}[\nabla\sqrt{\rho}]\right|^2 d^3\boldsymbol{k} = \int\varepsilon_{qnt}(k)dk, \tag{22}$$

$$N(t) = \int\left|\mathcal{F}_{\boldsymbol{k}}[\psi(\boldsymbol{r},t)]\right|^2 d^3\boldsymbol{k} = \int N(k)dk. \tag{23}$$

The spectrum of a quantity is usually computed by the angular binning of the data with respect to the wavenumber k in Fourier space. This approach is an inherently crude one, especially at low wavenumbers where only a few data points are available to each bin. In [16], Bradley et. al. suggested an alternate approach that can be used to compute highly resolved spectra of these quantities. The approach depends on using the angle-averaged Wiener-Khinchin theorem to calculate the power spectra and is given by

$$\varepsilon_{kin}^{\{i,c\}}(k) = \frac{1}{2}\int d^d\boldsymbol{r}\Lambda_d(k,|\boldsymbol{r}|)C[\boldsymbol{w}^\alpha,\boldsymbol{w}^\alpha](\boldsymbol{r}), \tag{24}$$

where $C[\boldsymbol{w}^\alpha,\boldsymbol{w}^\alpha](\boldsymbol{r})$ is the two-point auto-correlation function in position for a velocity field $\boldsymbol{w}^\alpha$ and $\Lambda_d(k,|\boldsymbol{r}|)$ is a kernel function given by

$$\Lambda_d(k,|\boldsymbol{r}|) = \begin{cases}\frac{1}{2\pi}J_0(kr), & \text{for } d=2\\ \frac{1}{2\pi^2}k^2\text{sinc}(kr), & \text{for } d=3.\end{cases} \tag{25}$$

In quTARANG, we have provided functions to calculate energy spectra using both these approaches. Further, the fluxes corresponding to the incompressible ($\Pi_{kin}^i(k)$) and compressible ($\Pi_{kin}^c(k)$) kinetic energy can be defined as [17],

$$\Pi_{kin}^{[i,c]}(k) = -\frac{d}{dt}\int_{k_0}^{k}\varepsilon_{kin}^{[i,c]}(k')dk', \tag{26}$$

where k_0 represents the wavenumber corresponding to the largest length scale. In turbulent BECs, another important quantity of interest is the exchange of particle number occupancy among different wavenumber modes. To quantify these exchanges across time and among modes, we define the corresponding transfer function and flux below. We begin by taking the Fourier transform of the GPE (8) which yields,

$$i\partial_t\hat{\psi}(\boldsymbol{k},t) = \frac{\boldsymbol{k}^2}{2}\hat{\psi}(\boldsymbol{k},t) + \hat{R}(\boldsymbol{k}), \quad \hat{R}(\boldsymbol{k}) = \mathcal{F}_{\boldsymbol{k}}\left[V(\boldsymbol{r},t)\psi(\boldsymbol{r},t) + g|\psi(\boldsymbol{r},t)|^2\psi(\boldsymbol{r},t)\right], \tag{27}$$

where $\hat{\psi}(\boldsymbol{k},t)$ is the Fourier transform of $\psi(\boldsymbol{r},t)$. Multiplying the complex conjugate of $\hat{\psi}(\boldsymbol{k},t)$ to the above equation and subtracting the resulting equation with its complex conjugate, we obtain,

$$\partial_t|\hat{\psi}(\boldsymbol{k},t)|^2 = \hat{T}(\boldsymbol{k}), \quad \hat{T}(\boldsymbol{k}) = -i(\hat{\psi}^*(\boldsymbol{k},t)\hat{R}(\boldsymbol{k}) - \hat{\psi}(\boldsymbol{k},t)\hat{R}^*(\boldsymbol{k})), \tag{28}$$

where $\hat{T}(\boldsymbol{k})$ represents the transfer function of the particle number. The particle number flux of the system is defined in terms of the transfer function as follows,

$$\Pi_N(k) = -\int_{k_0}^{k}\hat{T}(\boldsymbol{k}')d\boldsymbol{k}'. \tag{29}$$

3. Numerical scheme: Time-splitting pseudo-spectral scheme

In quTARANG, We have used the Time-splitting pseudo-spectral (TSSP) scheme to solve the GPE. In this section, we present the detailed methodology involved in the implementation of this scheme to GPE used in our code.

Let us consider the following time-dependent Partial Differential Equation (PDE):

$$\begin{cases}\partial_t\psi(\boldsymbol{r},t) = A\psi(\boldsymbol{r},t) + B\psi(\boldsymbol{r},t),\\ \psi(\boldsymbol{r},0) = \psi_0(\boldsymbol{r}).\end{cases} \tag{30}$$

where A and B represent operators. The solution of the above equation can be denoted by $\psi(\boldsymbol{r},t) = e^{(A+B)t}\psi_0(\boldsymbol{r})$, for all $t > 0$. The Time-splitting scheme involves approximating the solution ψ by splitting the operator $e^{(A+B)t}$ into a product of operators $e^{\alpha_i At}$ and $e^{\beta_i Bt}$ where α_i and β_i are constants that depend on the order of splitting. For a time step δt, a second-order splitting scheme called Strang splitting [7] approximates the solution as

$$\psi(\boldsymbol{r},t+\delta t) \approx e^{A\delta t/2}e^{B\delta t}e^{A\delta t/2}\psi(\boldsymbol{r},t), \tag{31}$$

where $\delta t < 1$. For the case of the GP equation (8), the Strang splitting works via the operators,

$$A = -iV(\boldsymbol{r}) - ig|\psi(\boldsymbol{r},t)|^2, \quad B = i\frac{1}{2}\nabla^2. \tag{32}$$

The advantage of using the Strang-TSSP scheme over other finite difference schemes is that it is spectrally accurate in space and second order in time [7]. Further, it is time reversible, time transverse invariant, mass conserving and unconditionally stable. To solve Eq. (8) for the initial condition $\psi(\boldsymbol{r}, 0)$ on the time interval $[0, T]$, we discretize the time domain into N equally spaced parts with $\delta t = T/N$ and $t_n = n\delta t$, where $\delta t < 1$. For $t_n < t < t_{n+1}$ and $a \leq x, y, z \leq b$, the first step involves solving the differential equation corresponding to operator A for half the time step:

$$i\frac{\partial\psi(\boldsymbol{r}, t)}{\partial t} = V(\boldsymbol{r}, t)\psi(\boldsymbol{r}, t) + g|\psi(\boldsymbol{r}, t)|^2\psi(\boldsymbol{r}, t). \tag{33}$$

It can be shown that for $t_n < t \leq t_{n+1}$, $|\psi(\boldsymbol{r}, t)^2|$ in Eq. (33) remains invariant [18, 19]. As a result, it is exactly solvable in real space. The second step involves solving the differential equation corresponding to operator B for a full time step:

$$i\partial_t\psi(\boldsymbol{r}, t) = -\frac{1}{2}\nabla^2\psi(\boldsymbol{r}, t), \tag{34}$$

which can be efficiently solved by using FFTs [19]. Finally, the first step is repeated for another half time. The detailed steps for a solution in 1-D in a domain $x\epsilon[a, b]$ is given by

$$\psi^{(1)}(x_n, t_n) = \psi(x_n, t_n)e^{-i(V(x_n,t_n)+g|\psi(x_n,t_n)|^2)\frac{\delta t}{2}}, \tag{35}$$

$$\hat{\psi}^{(2)}(k_p, t_n) = \hat{\psi}^{(1)}(k_p, t_n)e^{-i\frac{k_p^2}{2}\delta t}, \tag{36}$$

$$\psi(x_n, t_{n+1}) = \psi^{(2)}(x_n, t_n)e^{-i(V(x_n,t_n)+g|\psi^{(2)}(x_n,t_n)|^2)\frac{\delta t}{2}}, \tag{37}$$

where $\hat{\psi}^{(1)}(k_p, t_n)$ is the Fourier transform of $\psi^{(1)}(x_n, t_n)$ and $\psi^{(2)}(x_n, t_n)$ is the inverse Fourier transform of $\hat{\psi}^{(2)}(k_p, t_n)$ defined as

$$\hat{\psi}^{(1)}(k_p, t_n) = \sum_{i=0}^{M-1}\psi^{(1)}(x_i, t_n)e^{ik_p(x_i-a)}, \quad p = -\frac{M}{2}, \ldots, \frac{M}{2} - 1, \quad k_p = \frac{2\pi p}{b-a}, \tag{38}$$

$$\psi^{(2)}(x_n, t_n) = \sum_{p=-\frac{M}{2}}^{\frac{M}{2}-1}\hat{\psi}^{(2)}(k_p, t_n)e^{-ik_p(x_n-a)}, \quad n = 0, 1, 2, \ldots, M-1. \tag{39}$$

Ground state solution for a given system can be obtained using the TSSP scheme with imaginary time propagation ($t \to -it$) [20]. When the wavefunction propagates in imaginary time, it will result in an exponential decay of all the energy states of the system, with a decay rate proportional to their respective energies. This decay process effectively “filters” out the higher energy components, leaving behind the ground state dominant in the system. By iteratively applying renormalization, the wavefunction converges to the ground state of the system. In the following section, we provide the detailed description of our code.

4. Description and Execution of quTARANG

The quTARANG package uses Python’s `Numpy` and `CuPy` libraries for computation and execution of code on the CPU and the GPU, respectively along with several other libraries detailed in it’s GitHub repository [21]. The execution of quTARANG is straightforward. On the user end, there are two files which are important for the execution of the code: (a) `para.py` and (b) `main.py`. To execute quTARANG, the user has to set the input parameters and the initial conditions inside the `para.py` file and then run the `main.py` file. A code snippet of the parameters inside the `para.py` has been shown in List. (1).

```
#para.py
device = "cpu"
device_rank = 1
Nx, Ny, Nz = 256, 256, 1
Lx, Ly, Lz = 16, 16, 1
tmax = 8
dt = 0.001
```

```
g = 2
inp_type = "fun"
typ = "rp"
in_path = "/path/to/input_directory"
op_path = "/path/to/output_directory"
scheme = "TSSP"
imgtime = False
delta = 1e-12
```

Listing 1: An example of the `para.py` file showing the input parameters of a 3-D GPE simulation.

```
#main.py
from quTARANG.src.lib import gpe
from quTARANG.src import evolution
from quTARANG.src.univ import grid, fns
from quTARANG.config.config import ncp
import quTARANG.config.mpara as para

V = 0.5*(grid.x_mesh**2+grid.y_mesh**2+z_mesh**2)
def wfc():
    return 1/ncp.sqrt(ncp.pi)*ncp.exp((-grid.x_mesh**2+grid.y_mesh**2+z_mesh**2)/2)
def pot(t):
    return V + ncp.sin(t)

G = gpe.GPE(wfcfn=wfc, potfn=pot)
evolution.time_advance(G)
```

Listing 2: An example of the `main.py` file showing initialisation using user-defined initial wavefunction and potential for a 3-D system.

The description of the parameters inside the `para.py` file is as follows:

1. `device`: Sets the device (CPU or GPU) on which the code will be executed by using keyword <`"cpu"`> and <`"gpu"`>, respectively.
2. `device_rank`: Selects the GPU on which the code will be run if a system has more than one GPU. For example, if a system has 4 GPUs, then this parameter can take a value between <`0-3`>.
3. (`Nx, Ny, Nz`) and (`Lx, Ly, Lz`): Sets the grid size and box length along the different axes, respectively. To run the code for a 2-D system, one needs to set the parameters `Nz` and `Lz` equal to <`1`>.
4. `tmax, dt`: `tmax` sets the final time upto which the code will be executed while `dt` is the step size that will be used for the time discretization.
5. `g`: Sets the strength of the nonlinearity of the system.
6. `inp_type and typ`: `inp_type` sets how quTARANG initializes the initial conditions for a simulation. It will take one of the values among <`"fun"`>, <`"pdf"`> and <`"dat"`>. <`"fun"`> corresponds to setting initial conditions by functions in `main.py`, <`"pdf"`> corresponds to setting initial conditions from those defined in quTARANG, while <`"dat"`> corresponds to setting initial conditions through two separate input files containing the data for initial wavefunction and potential. For selecting initial conditions from those defined in quTARANG, `typ` sets specific types of predefined initial conditions while for the initial conditions through the input files, the user needs to provide the path of the directory containing the data for the initial wavefunction and the potential in the form of two separate `HDF5` files, named `wfc.h5` and `pot.h5`, respectively to `inp_path` parameter. The datasets inside `wfc.h5` and `pot.h5` storing the wavefunction and potential data should be named as `wfc` and `pot`, respectively.
7. `op_path`: Sets the path of directory where generated data will be stored.
8. `imgtime, delta`: Set `imgtime` = <`True`> to compute the stationary state while for evolving the system set it to <`False`>. The `delta` parameter sets the threshold error needed to stop the execution of the code while computing the ground state and it is the absolute value of energy difference between current time step and the previous time step ($|E_{n+1} - E_n|$).

There are a multitude of other parameters as well which allow the user to further control over the simulation. These parameters along with the detailed description of quTARANG are given in its GitHub repository [21].

5. Validation

In this section, we present the validation of the results obtained using our code by comparing them with the earlier results for both 2-D and 3-D cases.

5.1. Validation of ground state

To quantify the ground state and the corresponding dynamical state of the condensate, we compute the chemical potential of the condensate (12) as well as its rms size defined by

$$\sigma_i = \sqrt{\langle (i - \langle i \rangle)^2 \rangle} \tag{40}$$

where $i \in \{r, x, y, z\}$ representing the radial size and size along the three coordinate axes respectively and $\langle f \rangle = \int f(\boldsymbol{x})|\psi(\boldsymbol{x}, t)|^2 d\boldsymbol{x}$.

```
#==============================================================================
#                        Change the following parameters
#==============================================================================
real_dtype = "float64"
complex_dtype = "complex128"

pi = 3.141592653589793

# Device Setting
device = "gpu"
device_rank = 0

# Set grid size
Nx = 256
Ny = 256
Nz = 1

# Set box length
Lx = 32
Ly = 32
Lz = 1

# Set maximum time and dt
tmax = 1
dt = 0.001

# Choose the value of the non linearity
g = 3.1371

inp_type = "fun"
typ = "rp"

# If inp_type = "dat" then set the input path
in_path = "/path/to/input_directory"

# Set output folder path
op_path = "../output_gstate2D"

scheme = "TSSP"
imgtime = True
delta = 1e-12
overwrite = True

# Wavefunction save setting
save_wfc = False
wfc_start_step = 0
wfc_iter_step = 500

```

```
# Rms save setting
save_rms = False
rms_start_step = 0
rms_iter_step = 10

# Energy save setting
save_en = False
en_start_step = 0
en_iter_step = 100

# Printing iteration step
t_print_step = 6000
```

Listing 3: The `para.py` file for the ground state computation of a 2-D condensate in a harmonic potential well.

```
#main.py
from quTARANG.src.lib import gpe
from quTARANG.src import evolution
from quTARANG.src.univ import grid, fns
from quTARANG.config.config import ncp
import quTARANG.config.mpara as para

##########################################################################
V = 0.5*(grid.x_mesh**2 + grid.y_mesh**2)
def gstate_wfc2d():
    return 1/ncp.sqrt(ncp.pi)*ncp.exp(-(grid.x_mesh**2 + grid.y_mesh**2)/2)

def gstate_pot2d(t):
    return V + 0*t
G = gpe.GPE(wfcfn = gstate_wfc2d, potfn = gstate_pot2d)
##########################################################################

evolution.time_advance(G)
```

Listing 4: The `main.py` file for the ground state computation of a 2-D condensate in a harmonic potential well.

We obtain the ground state of the system in a harmonic potential well for different nonlinearities and compare the results with those from [9] to validate our code. For the case of a 2-D condensate, we considered a box size of $[-16, 16]^2$ with grid resolution as 256×256 and have obtained the ground state using the imaginary time evolution starting from the following initial conditions:

$$\psi(\boldsymbol{r}, 0) = \frac{1}{\sqrt{\pi}} e^{-(x^2+y^2)/2}, \quad V(\boldsymbol{r}) = \frac{1}{2}(x^2 + y^2). \tag{41}$$

```
[Nx, Ny, Nz]:  256 256 1
[Lx, Ly, Lz]: 32 32 1
dimension:  2
g:  3.1371
Scheme: TSSP
Imaginary time: True
dt:  0.001
device:  gpu
Stoping criteria for energy: delta E< 1e-12
Elapsed time(s):  0.005561584949493408
***** Initialization completed *****

***** Time advence started *****
Iteration:  0
Stoping criteria for energy: delta E< 1e-12
delta E:  0.0001654110114968077
Chemical Potential:  1.4989539298525165
```

```
Energy:  1.2494770749752955
rms:  1.000249029833995
Particle Number:  1.0000000000000002
Elapsed time(s):  0.0626481056213379
----------------------------------------

Iteration:  5168
Stoping criteria for energy: delta E< 1e-12
delta E:  9.99644811372491e-13
Chemical Potential:  1.4200945462992156
Energy:  1.2213323947910935
rms:  1.1050410876627483
Particle Number:  0.9999999999999998
Elapsed time(s):  5.7030869140625
----------------------------------------

----------------------------------------
Time taken for initialization(s):  0.005561584949493408
Time take for execution(s):  5.7008735351562505
Total run time(s):  5.706435120105744
----------------------------------------

***** Run Completed *****
```

Listing 5: Output generated by the `main.py` file, shown in List (4), for the ground state computation of a 2-D condensate.

The Lists (3) and (4) show the `para.py` file and `main.py` file for the above-mentioned initial conditions with $g = 3.1371$. The corresponding output in terminal is shown in the List (5). We have run a set of similar simulations with different values for the nonlinearity of the condensate in 2-D and 3-D. For the 3-D cases, we have run the simulations on a box size of $[-16, 16]^3$ with a grid size of $256 \times 256 \times 256$ using the following initial condition:

$$\psi(\boldsymbol{r}, 0) = \frac{1}{\pi^{3/4}} e^{-(x^2+y^2+z^2)/2}, \quad V(\boldsymbol{r}) = \frac{1}{2}(x^2 + y^2 + 16z^2). \tag{42}$$

Table (1) shows the rms size (σ_r) and the chemical potential (μ) of the ground state for a set of nonlinearities in 2-D and 3-D which have been obtained using quTARANG. A comparison with the results presented in [9] shows that the two are in good agreement with each other.

5.2. *Validation of dynamics*

We further validate the dynamics of the condensate by comparing the oscillations in the condensate width along different axes, starting from the non-interacting ground state, with those reported by Bao et. al. [10] for both 2-D and 3-D systems. For the 2-D case, the system is subjected to a trapping potential given by

$$V(\boldsymbol{r}, 0) = \frac{1}{2}(x^2 + 4y^2), \tag{43}$$

with an initial wavefunction given by

$$\psi(\boldsymbol{r}, 0) = \left(\frac{1}{\sqrt{2}\pi}\right)^{1/2} e^{-(x^2+2y^2)/4}, \tag{44}$$

and nonlinearity $g = 2$. The system is evolved within a box size of $[-8, 8]^2$ with a grid size of 256×256. In the 3-D case, the trapping potential takes the form

$$V(\boldsymbol{r}, 0) = \frac{1}{2}(x^2 + 4y^2 + 16z^2), \tag{45}$$

with an initial state,

$$\psi(\boldsymbol{r}, 0) = \left(\frac{8}{\pi}\right)^{3/4} \exp(-2x^2 - 4y^2 - 8z^2), \tag{46}$$

and nonlinearity $g = 0.1$. The box size is chosen as $[-8, 8]^3$ box with a grid resolution of $256 \times 256 \times 256$.

Dimension	g	σ_r	σ_r **[9]**	μ	μ **[9]**
	-2.5097	0.87771	0.87759	0.49987	0.49978
	0	1.00000	1.00000	1.00000	1.00000
	3.1371	1.10504	1.10513	1.42009	1.42005
2	12.5484	1.30656	1.30687	2.25609	2.25583
	62.742	1.78722	1.78817	4.61136	4.60982
	313.71	2.60122	2.60441	10.07639	10.06825
	627.42	3.07914	3.08453	14.20569	14.18922
	0	1.0606	1	3.0000	3.0000
	18.81	1.3778	1.3249	4.3618	4.3611
	94.05	1.8222	1.7742	6.6824	6.6797
3	188.1	2.0881	2.0411	8.3718	8.3671
	940.5	2.8912	2.8424	14.9663	14.9487
	1881	3.3268	3.2758	19.5058	19.4751
	7524	4.3968	4.3408	33.5623	33.4677
	15048	5.0497	4.9922	44.1894	44.0234

Table 1: The chemical potential (μ) and the rms radius (σ_r) for 2-D and 3-D cases with different values of nonlinearity (g) computed using quTARANG and those reported by Murugananadam *et al.* [9].

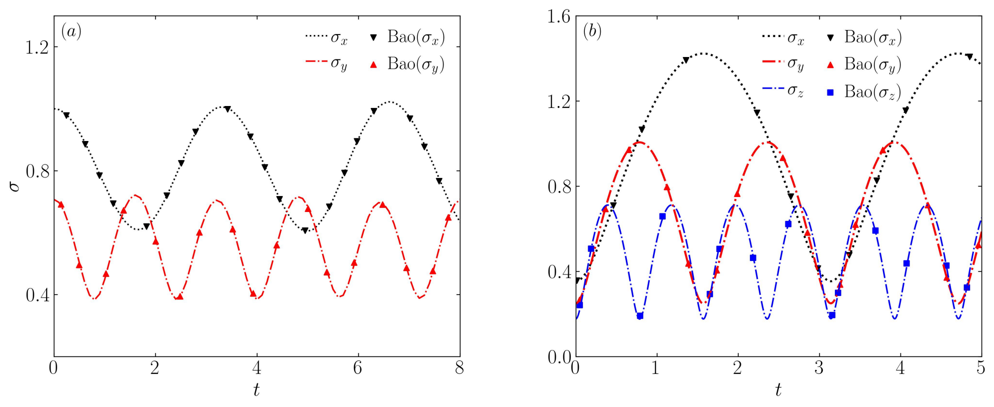

Figure 1: Comparison of the dynamical evolution of the root-mean-square size (σ_x, σ_y and σ_z) of a condensate along the $x-$, $y-$ and $z-$ directions obtained using quTARANG (black, red and blue curves) and those obtained by Bao et. al. [10] (black, red and blue markers) is shown for the (a) 2-D and (b) 3-D case, respectively. A good agreement between our results and those of Bao et. al. has been obtained.

Fig. (1) shows the rms width of the condensate along x-axis (σ_x), y-axis (σ_y) and z-axis (σ_z) computed for the (a) 2-D case and the (b) 3-D case respectively by using quTARANG and corresponding results from Ref. [10]. It is evident that the two results are in good agreement with each other.

6. Numerical Example: Turbulent BECs

6.1. Example 1: Vortex lattice case

One can generate various distributions of vortices in a BEC using phase imprinting methods [22, 23] or by stirring the condensate with different types of potentials [24, 25]. In quTARANG, a vortex lattice initial configuration can be generated by setting the parameter `typ` as `<"vl">`. quTARANG then automatically generates the appropriate phase for the initial condition and uses imaginary time evolution to heal the vortices according to the phase. One can change the number of vortices in the system inside the `dimension2.py` file. For the current example, the number of vortices in the system are `100` with equal numbers of vortices and antivortices pairs. The simulation is performed for $g = 1$ in a box of length $[-128, 128]^2$ and grid size 256×256 for a periodic boundary condition. Fig. (2) (a-d) shows the density distribution and (e-h) corresponding phase of the wavefunction generated using quTARANG at time steps $t = 0$, 3000, 10000 and 100000 respectively.

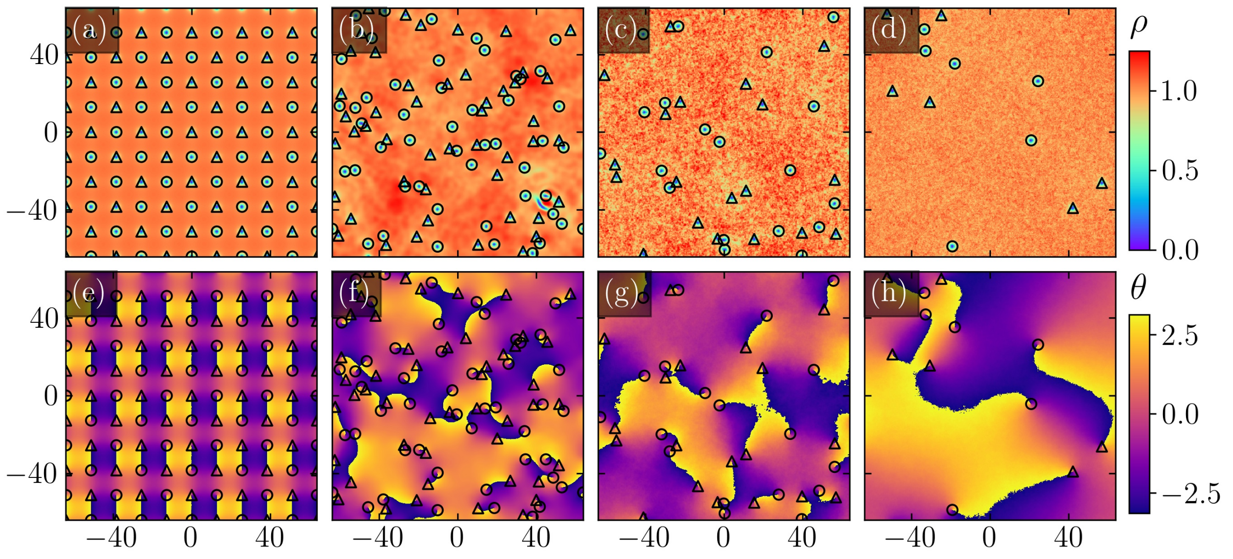

Figure 2: The temporal evolution of an initial wavefunction with a grid of vortices and anti-vortices created using the phase imprinting method. The figures $(a-d)$ show the density and $(e-h)$ show the phase of the wavefunction at time $t = 0$, $t = 3000$, $t = 10000$ and $t = 100000$, respectively. Square markers represent the positions of vortices, while triangular markers denote the positions of anti-vortices within the system.

6.2. *Example 2: Random phase initial condition*

To generate turbulent flow, an initial condition with random phase state $\psi(\boldsymbol{r}, 0) = e^{i\theta(\boldsymbol{r},0)}$ has been chosen with

$$\tilde{\theta}(\boldsymbol{k}, 0) = \begin{cases} \theta_0 e^{i\alpha(\boldsymbol{k})} & (\delta k \leq |\boldsymbol{k}| \leq 3\delta k, \\ 0 & \text{otherwise,} \end{cases} \tag{47}$$

where $\tilde{\theta}(\boldsymbol{k}, 0)$ is the Fourier transform of $\theta(\boldsymbol{r}, 0)$, $\tilde{\theta}(\boldsymbol{k}, 0) = \tilde{\theta}^*(-\boldsymbol{k}, 0)$, $\alpha(\boldsymbol{k})$ is chosen randomly for each $\boldsymbol{k}$ in interval $[-\pi, \pi]$ and $\delta k = 2\pi/L$ [15]. One can simulate this in quTARANG by setting `typ` to `"rp"` and box size to $[-14.92, 14.92]^2$, grid size to 256×256 and the nonlinearity to $g = 4$ in `para.py` file. Fig. (3) shows the density and phase profiles for the simulation with the above initial condition at time $t = 0$, 10, 100 and 600, while the corresponding different component of energies values at different time steps are listed in Table (2).

Time step (t)	E^i_{kin}	E^i_{kin}	E_{qnt}	E_{int}
0	0	2,684.53	0	1778.46
50	68.49	1,276.63	1,063.71	2,052.51
100	57.22	1,292.20	1,135.76	1,972.25
300	43.23	1,299.98	1,198.86	1,906.49
600	28.53	1,318.31	1,215.08	1,883.02

Table 2: Compressible kinetic energy (E^c_{kin}), incompressible kinetic energy (E^i_{kin}), quantum energy (E_{qnt}), and internal energy (E_{int}) at different time steps for smooth random-phase initial conditions.

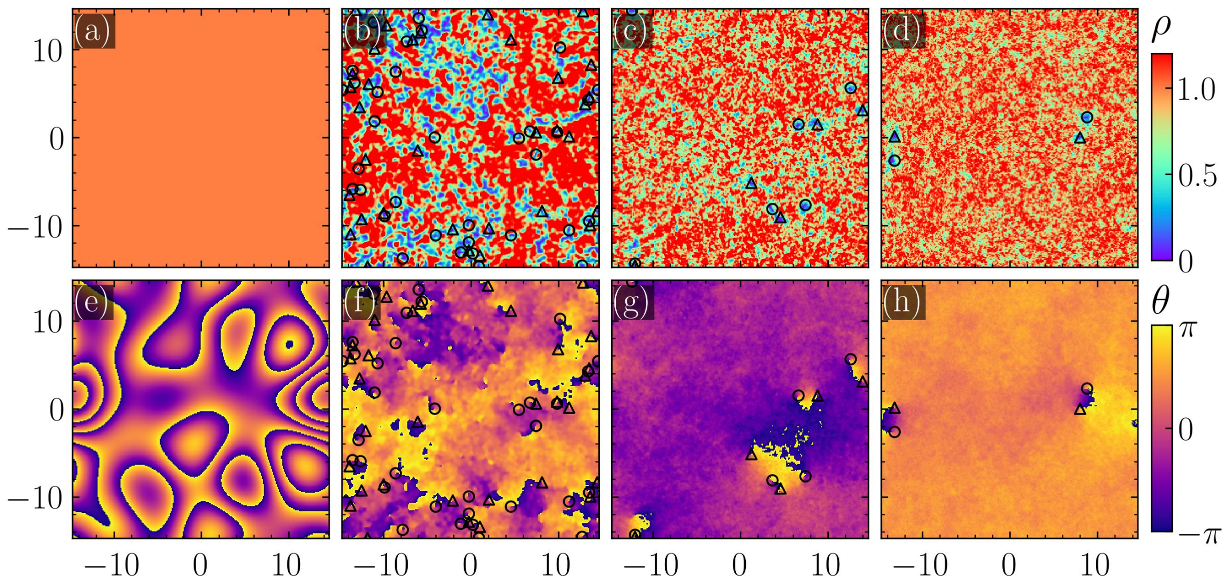

Figure 3: The temporal evolution of the condensate with a random phase and uniform density as the initial condition demonstrating the attainment of a turbulent state. The initial state leads to the creation of a large number of vortices and anti-vortices during the evolution of the system. The density profile $(a-d)$ and phases $(e-h)$ of the condensate are shown at t = 0, 10, 100 and 600, respectively. The square markers show the location of vortices, while triangular markers show the location of anti-vortices.

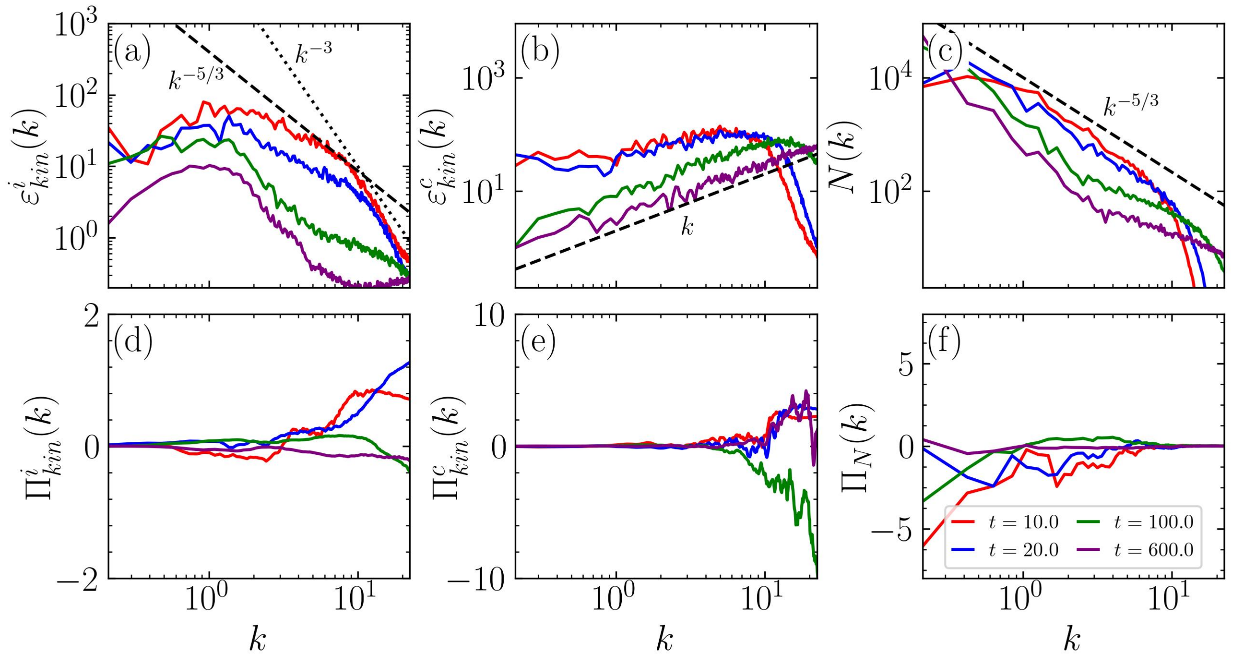

Figure 4: Energy spectra $(a-c)$ and fluxes $(d-f)$ corresponding to the incompressible, compressible kinetic energy and particle number density at various times for a case of the random phase initial condition. The spectra corresponding to the incompressible kinetic energy show a $k^{-5/3}$ scaling in the intermediate range consistent with a turbulent regime, while at large wavenumbers, it shows a k^{-3} scaling due to the vortex core structure. The compressible kinetic energy spectra show a k scaling at t = 600, suggesting the thermalization of the system at large times. Consecutively, the particle number spectra show a k^{-1} scaling.

In quTARANG, there are functions that can be used to calculate statistical quantities of importance, such as spectra and fluxes. The spectra can be calculated using both, the binning method as well as the alternate approach suggested by Bradley et. al. [16]. Fig. (4) $(a-c)$ shows the spectra for the incompressible kinetic energy ($\varepsilon^i_{kin}(k)$), the compressible kinetic energy ($\varepsilon^c_{kin}(k)$) and the particle number density ($N(k)$) respectively. In the panels $(d-f)$ we show the incompressible kinetic energy flux ($\Pi^i_{kin}(k)$), compressible kinetic energy flux ($\Pi^c_{kin}(k)$) and particle number flux $\Pi_N(k)$, respectively. The energy spectra have been calculated using the angle-averaged Wiener-Khinchin method and the particle number spectra have been calculated using the binning method. We find that the incompressible kinetic energy exhibits a Kolmorgorov like scalings [$\varepsilon^i_{kin}(k) \propto k^{-5/3}$] in the Infrared regime while at large wavenumbers (ultravoilet regime) it shows a scaling of k^{-3} which is due to the presence of vortex core [16] varies linearly with k, suggesting the thermalization of the system after a longer period of time.

7. Performance of quTARANG on CPU and GPU

In this section, we present the performance of our code when executed on a CPU and the performance gain when the same code is run on the GPU. GPUs possess significantly greater processing power than CPUs, primarily due to their extensive array of parallel processing units. This parallelism is particularly advantageous for computational tasks involving Fast Fourier Transforms (FFTs). Given that FFTs are the most computationally expensive part of the TSSP scheme, leveraging the GPU's architecture to accelerate these computations results in faster processing times compared to traditional CPU-based implementations. By utilizing the CuPy library, quTARANG offloads these intensive operations onto the GPU, thereby achieving significant speedups in both single and double-precision computations. This capability is crucial for high-resolution simulations where both precision and performance are critical.

For a performance evaluation of our code, starting from the ground state of the GPE in a harmonic potential well, we have evolved the system for different numbers of iterations and compared the execution times on a single core of the AMD EPYC 7713P processor with those on AMD MI210 and NVIDIA A100 GPUs. In Table (3), we present the execution times for different 2-D and 3-D grid sizes. The results demonstrate the performance gains (Fig. (5) (a, b)) achieved by quTARANG on the A100 and MI210 GPUs for both 2-D and 3-D computations across various grid sizes. In the 2-D case, we have achieved a maximum speedup of approximately 210 times on the MI210 GPU and 267 times on the A100 GPU for a 4096^2 grid. For 3-D computations on a 256^3 grid, we have observed a speedup of approximately 162 times on the MI210 GPU and 218 times on the A100 GPU compared to a single core of the CPU. These findings underscore the substantial advantage of using GPU acceleration with quTARANG for GPE simulation.

Grid Size	Iterations	Processing device		
		AMD EPYC 7713P CPU (in sec)	MI210 GPU (in sec)	A100 GPU (in sec)
256^2	1000	9.92	0.52	0.82
512^2	1000	41.47	0.61	0.77
1024^2	100	17.20	0.24	0.23
2048^2	100	87.15	0.59	0.42
4096^2	100	419.97	2.00	1.56
64^3	100	3.09	0.17	0.32
128^3	100	39.90	0.42	0.25
256^3	100	349.15	2.15	1.57

Table 3: Execution times of quTARANG on a AMD EPYC 7713P CPU, AMD MI210 GPU and NVIDIA A100 GPU for different grid sizes in 2-D and 3-D.

T_{gpu} (in sec) for 10000 iterations			T_{gpu} (in sec) for 100 iterations		
Grid Size	quTARANG	GPUE [8]	Grid Size	quTARANG	GPUE [8]
64^2	7.34	0.61	64^3	0.09	0.06
128^2	6.67	0.86	128^3	0.24	0.46
256^2	6.48	1.97	256^3	1.51	3.78
512^2	6.33	6.1			
1024^2	8.40	22.02			
2048^2	37.86	92.01			

Table 4: Execution times of quTARANG and GPUE [8] for different 2-D and 3-D grid sizes on NVIDIA A100 GPU.

We have also compared the performance of our code quTARANG with the CUDA code GPUE [8]. Table (4) shows the execution timings of quTARANG and GPUE for 10000 iterations in 2-D and 100 iterations in 3-D on NVIDIA A100 GPU for different grid sizes. Our results indicate that despite of quTARANG being written purely in python it achieves speedup of 2.5 times over CUDA based GPUE code for larger grid sizes, highlighting its efficiency in handling extensive computational workloads despite being written completely in Python. Fig. (5) (c, d) shows this comparison for different grid sizes for 2-D and 3-D cases, respectively. We observe that the performance of quTARANG significantly improves as we increase the grid size of the system. This improvement is likely due to a more efficient utilization of the GPU's parallel processing capabilities when dealing with larger datasets. As the grid size grows, the overhead of GPU communication and memory access is better amortized, leading to enhanced computational efficiency. These comparisons underscore the robustness of quTARANG in large-scale simulations of quantum turbulence.

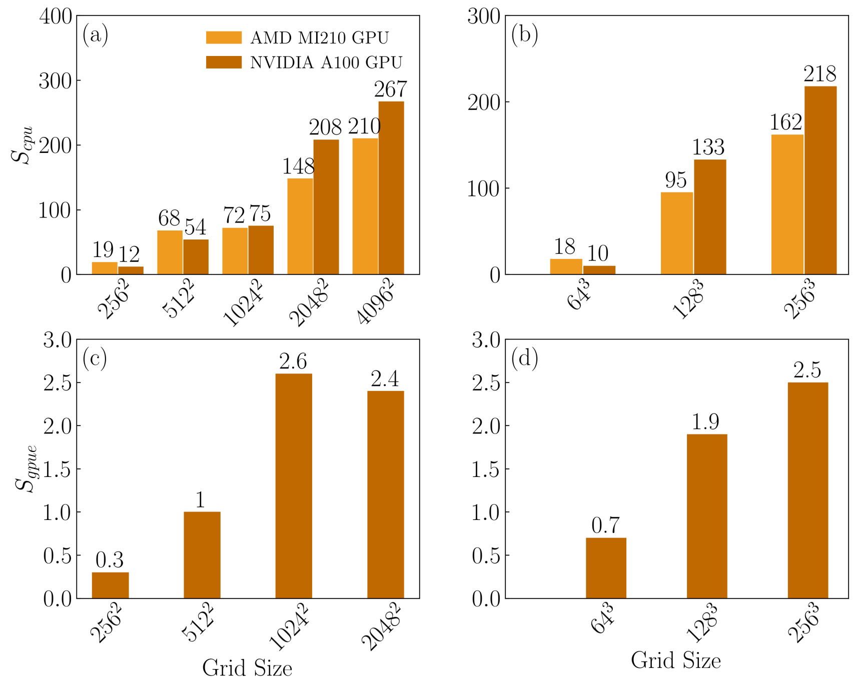

Figure 5: Speedup of quTARANG on AMD MI210 GPU and NVIDIA A100 GPU compared to the single core of the AMD EPYC 7713P processor for (a) 2-D and (b) 3-D cases, respectively. (c) and (d) show the speedup comparison of quTARANG with GPUE [8] on NVIDIA A100 GPU for 2-D and 3-D cases, respectively.

8. Conclusion

In this paper, we have presented the detailed features of our code quTARANG, a GPU-accelerated pseudo-spectral GPE solver that can accurately compute the ground state and simulate the behavior of different turbulent states of BECs. The pseudo-spectral TSSP scheme implemented in quTARANG (using FFTs), leads to a robust and efficient code to obtain the ground state structure and dynamics of BECs. Our code is equipped to simulate and characterize different turbulent states of BECs as well. Simulations with user-defined initial conditions and trapping potentials can be easily implemented in quTARANG using the description presented in this paper. In quTARANG, we have used the imaginary time propagation with the TSSP scheme to obtain the ground state, while the dynamics and turbulent states have been computed using the same scheme with real-time propagation. The statistical properties like spectra and fluxes presented in this paper, which are highly useful to study quantum turbulence, are also implemented in quTARANG. Two methods, the conventional binning approach and the angle-averaged Wiener-Khinchin approach have both been implemented in quTARANG, the later allowing high-resolution spectra to be computed without requiring a large grid size. We have presented a speedup comparison of quTARANG on two different GPUs with a CPU and the results show a substantial performance gain on GPUs. At the end, we have presented a comparison between the speedup of our code with the CUDA based code GPUE which reveals that despite quTARANG being written completely in Python, it performs better the GPUE for large grid simulations.

References

[1] M. H. Anderson, J. R. Ensher, M. R. Matthews, C. E. Wieman, E. A. Cornell, Observation of bose-einstein condensation in a dilute atomic vapor, science 269 (5221) (1995) 198–201. doi:10.1126/science.269.5221.198.

[2] K. B. Davis, M. O. Mewes, M. R. Andrews, N. J. van Druten, D. S. Durfee, D. M. Kurn, W. Ketterle, Bose-einstein condensation in a gas of sodium atoms, Phys. Rev. Lett. 75 (1995) 3969–3973. doi:10.1103/PhysRevLett.75.3969.
URL https://link.aps.org/doi/10.1103/PhysRevLett.75.3969

[3] A. Görlitz, J. Vogels, A. Leanhardt, C. Raman, T. Gustavson, J. Abo-Shaeer, A. Chikkatur, S. Gupta, S. Inouye, T. Rosenband, et al., Realization of bose-einstein condensates in lower dimensions, Physical review letters 87 (13) (2001) 130402. doi:10.1103/PhysRevLett.87.130402.

[4] M. W. Zwierlein, C. A. Stan, C. H. Schunck, S. M. Raupach, S. Gupta, Z. Hadzibabic, W. Ketterle, Observation of bose-einstein condensation of molecules, Physical review letters 91 (25) (2003) 250401. doi:10.1103/PhysRevLett.91.250401.

[5] J. Klaers, J. Schmitt, F. Vewinger, M. Weitz, Bose–einstein condensation of photons in an optical microcavity, Nature 468 (7323) (2010) 545–548. doi:10.1038/nature09567.

[6] X. Antoine, R. Duboscq, GPELab, a Matlab toolbox to solve Gross–Pitaevskii equations I: Computation of stationary solutions, Computer Physics Communications 185 (11) (2014) 2969 – 2991. doi:10.1016/j.cpc.2014.06.026.
URL http://www.sciencedirect.com/science/article/pii/S0010465514002318

[7] X. Antoine, R. Duboscq, Gpelab, a matlab toolbox to solve gross–pitaevskii equations ii: Dynamics and stochastic simulations, Computer Physics Communications 193 (2015) 95–117.

[8] J. R. Schloss, L. J. O'Riordan, Gpue: Graphics processing unit gross–pitaevskii equation solver, Journal of Open Source Software 3 (32) (2018) 1037.

[9] P. Muruganandam, S. K. Adhikari, Fortran programs for the time-dependent gross–pitaevskii equation in a fully anisotropic trap, Computer Physics Communications 180 (10) (2009) 1888–1912. doi:10.1016/j.cpc.2009.04.015.

[10] W. Bao, D. Jaksch, P. A. Markowich, Numerical solution of the gross–pitaevskii equation for bose–einstein condensation, Journal of Computational Physics 187 (1) (2003) 318–342. doi:10.1016/S0021-9991(03)00102-5.
URL https://www.sciencedirect.com/science/article/pii/S0021999103001025

[11] F. Dalfovo, S. Giorgini, L. P. Pitaevskii, S. Stringari, Theory of bose-einstein condensation in trapped gases, Reviews of modern physics 71 (3) (1999) 463.

[12] C. J. Pethick, H. Smith, Bose–Einstein Condensation in Dilute Gases, Cambridge University Press, 2008. doi:10.1017/CBO9780511802850.

[13] C. F. Barenghi, N. G. Parker, A primer on quantum fluids, Springer, 2016.

[14] C. Nore, M. Abid, M. E. Brachet, Kolmogorov turbulence in low-temperature superflows, Phys. Rev. Lett. 78 (1997) 3896–3899. doi:10.1103/PhysRevLett.78.3896.
URL https://link.aps.org/doi/10.1103/PhysRevLett.78.3896

[15] R. Numasato, M. Tsubota, V. S. L'vov, Direct energy cascade in two-dimensional compressible quantum turbulence, Phys. Rev. A 81 (2010) 063630. doi:10.1103/PhysRevA.81.063630.
URL https://link.aps.org/doi/10.1103/PhysRevA.81.063630

[16] A. S. Bradley, R. K. Kumar, S. Pal, X. Yu, Spectral analysis for compressible quantum fluids, Phys. Rev. A 106 (2022) 043322. doi:10.1103/PhysRevA.106.043322.
URL https://link.aps.org/doi/10.1103/PhysRevA.106.043322

[17] A. D. García-Orozco, L. Madeira, L. Galantucci, C. F. Barenghi, V. S. Bagnato, Intra-scales energy transfer during the evolution of turbulence in a trapped bose-einstein condensate (a), Europhysics Letters 130 (4) (2020) 46001. doi:10.1209/0295-5075/130/46001.

[18] W. Bao, S. Jin, P. A. Markowich, On time-splitting spectral approximations for the schrödinger equation in the semiclassical regime, Journal of Computational Physics 175 (2) (2002) 487–524. doi:https://doi.org/10.1006/jcph.2001.6956.
URL https://www.sciencedirect.com/science/article/pii/S0021999101969566

[19] W. Bao, S. Jin, P. A. Markowich, Numerical study of time-splitting spectral discretizations of nonlinear schrödinger equations in the semi-classical regimes, SIAM Journal on Scientific Computing 25 (1) (2003) 27–64. doi:10.1137/S1064827501393253.
URL https://doi.org/10.1137/S1064827501393253

[20] W. Bao, Q. Du, Computing the ground state solution of bose–einstein condensates by a normalized gradient flow, SIAM Journal on Scientific Computing 25 (5) (2004) 1674–1697. doi:10.1137/S1064827503422956.
URL https://doi.org/10.1137/S1064827503422956

[21] S. Rawat, S. K. Jha, qutarang (2024).
URL https://github.com/sachinrawat2207/quTARANG

[22] G. Andrelczyk, M. Brewczyk, L. Dobrek, M. Gajda, M. Lewenstein, Optical generation of vortices in trapped bose-einstein condensates, Phys. Rev. A 64 (2001) 043601. doi:10.1103/PhysRevA.64.043601.
URL https://link.aps.org/doi/10.1103/PhysRevA.64.043601

[23] A. E. Leanhardt, A. Görlitz, A. P. Chikkatur, D. Kielpinski, Y. Shin, D. E. Pritchard, W. Ketterle, Imprinting vortices in a bose-einstein condensate using topological phases, Phys. Rev. Lett. 89 (2002) 190403. doi:10.1103/PhysRevLett.89.190403.
URL https://link.aps.org/doi/10.1103/PhysRevLett.89.190403

[24] G. Gauthier, M. T. Reeves, X. Yu, A. S. Bradley, M. A. Baker, T. A. Bell, H. Rubinsztein-Dunlop, M. J. Davis, T. W. Neely, Giant vortex clusters in a two-dimensional quantum fluid, Science 364 (6447) (2019) 1264–1267. arXiv:https://www.science.org/doi/pdf/10.1126/science.aat5718, doi:10.1126/science.aat5718.
URL https://www.science.org/doi/abs/10.1126/science.aat5718

[25] S. Das, K. Mukherjee, S. Majumder, Vortex formation and quantum turbulence with rotating paddle potentials in a two-dimensional binary bose-einstein condensate, Phys. Rev. A 106 (2022) 023306. doi:10.1103/PhysRevA.106.023306.
URL https://link.aps.org/doi/10.1103/PhysRevA.106.023306