C and Fortran OpenMP programs for rotating Bose–Einstein condensates☆

Ramavarmaraja Kishor Kumar [a], Vladimir Lončar [b], Paulsamy Muruganandam [c,d], Sadhan K. Adhikari [e,*], Antun Balaž [b]

[a] *Instituto de Física, Universidade de São Paulo, 05508-090 São Paulo, Brazil*
[b] *Scientific Computing Laboratory, Center for the Study of Complex Systems, Institute of Physics Belgrade, University of Belgrade, Serbia*
[c] *Department of Physics, Bharathidasan University, Palkalaiperur Campus, Tiruchirappalli 620024, Tamilnadu, India*
[d] *Department of Medical Physics, Bharathidasan University, Palkalaiperur Campus, Tiruchirappalli 620024, Tamilnadu, India*
[e] *Instituto de Física Teórica, UNESP – Universidade Estadual Paulista, 01.140-70 São Paulo, São Paulo, Brazil*

ABSTRACT

We present OpenMP versions of C and Fortran programs for solving the Gross–Pitaevskii equation for a rotating trapped Bose–Einstein condensate (BEC) in two (2D) and three (3D) spatial dimensions. The programs can be used to generate vortex lattices and study dynamics of rotating BECs. We use the split-step Crank–Nicolson algorithm for imaginary- and real-time propagation to calculate stationary states and BEC dynamics, respectively. The simulation input parameters for the C programs are provided via input files, while for the Fortran programs they are given at the beginning of each program and therefore their change requires recompilation of the corresponding program. The programs propagate the condensate wave function and calculate several relevant physical quantities, such as the energy, the chemical potential, and the root-mean-square sizes. The imaginary-time propagation starts with an analytic wave function with one vortex at the trap center, modulated by a random phase at different space points. Nevertheless, the converged wave function for a rapidly rotating BEC with a large number of vortices is most efficiently calculated using the pre-calculated converged wave function of a rotating BEC containing a smaller number of vortices as the initial state rather than using an analytic wave function with one vortex as the initial state. These pre-calculated initial states exhibit rapid convergence for fast-rotating condensates to states containing multiple vortices with an appropriate phase structure. This is illustrated here by calculating vortex lattices with up to 61 vortices in 2D and 3D. Outputs of the programs include calculated physical quantities, as well as the wave function and different density profiles (full density, integrated densities in lower dimensions, and density cross-sections). The provided real-time propagation programs can be used to study the dynamics of a rotating BEC using the imaginary-time stationary wave function as the initial state. We also study the efficiency of parallelization of the present OpenMP C and Fortran programs with different compilers.

Program summary
Program title: BEC-GP-ROT-OMP, consisting of: (1) BEC-GP-ROT-OMP-C package, containing programs (i) bec-gp-rot-2d-th and (ii) bec-gp-rot-3d-th; (2) BEC-GP-ROT-OMP-F package, containing programs (i) bec-gp-rot-2d-th and (ii) bec-gp-rot-3d-th.
Program files doi: http://dx.doi.org/10.17632/cw7tkn22v2.2
Licensing provisions: Apache License 2.0
Programming language: OpenMP C; OpenMP Fortran. The C programs are tested with the GNU, Intel, PGI, Oracle, and Clang compiler, and the Fortran programs are tested with the GNU, Intel, PGI, and Oracle compiler.
Nature of problem: The present Open Multi-Processing (OpenMP) C and Fortran programs solve the time-dependent nonlinear partial differential Gross–Pitaevskii (GP) equation for a trapped rotating Bose–Einstein condensate in two (2D) and three (3D) spatial dimensions in a fully anisotropic traps.

Solution method: We employ the split-step Crank–Nicolson algorithm to discretize the time-dependent GP equation in space and time. The discretized equation is then solved by imaginary- or real-time propagation, employing adequately small space and time steps, to yield the solution of stationary and non-stationary problems, respectively.

1. Introduction

Previously published Fortran [1] and C [2] programs, and their OpenMP extensions [3,4] are now popular tools for solving the Gross–Pitaevskii (GP) equation and are enjoying widespread use. These programs have been later extended to the more complex scenario of dipolar atoms [5]. The C programs have been adapted to run even faster on modern multi-core computers using general-purpose graphic processing units with Nvidia CUDA and computer clusters using Message Passing Interface (MPI) [6]. In this paper, we present new OpenMP C and Fortran programs to solve the GP equation for a rotating trapped Bose–Einstein condensate (BEC) and to generate a vortex lattice, based on our earlier work [3,4]. This is a problem of general interest for both theoreticians [7,8] and experimentalists [9].

The GP equation for a rotating trapped BEC can be conveniently solved by the imaginary- [10–12] and real-time evolution [13] methods. The solution algorithms rely on transforming the GP equation to the rotating frame, where the rotating BEC with vortices becomes a stationary state [7] and the standard imaginary-time approach can be applied [10]. In the real-time approach [13], a dissipation has to be included in the GP equation to generate the vortices. The imaginary-time approach [10] does not require any dissipation, is simpler to implement and is found to converge faster and lead to accurate results. Here we provide combined imaginary- and real-time programs in two (2D) and three (3D) spatial dimensions without any dissipation [10]. The present imaginary-time program already involves complex variables and is hence combined together with the real-time program. The choice of the type of propagation is made through an input parameter. The imaginary-time approach should be used to solve the GP equation for the rotating BEC and to generate the stationary vortex lattice. A subsequent study of the non-stationary dynamics of the rotating BEC should be done using the real-time propagation. Here we provide C and Fortran programs for the solution of the GP equation for a rotating BEC in a fully anisotropic 3D trap by imaginary- and real-time propagation. We also present C and Fortran programs for the reduced GP equation in 2D, appropriate for a disk-shaped BEC under a tight axial (z-direction) trapping. We use the split-step Crank–Nicolson scheme for solving the GP equation, as in Refs. [1,2].

The imaginary-time algorithm employs a time iteration loop of an initial state until the convergence is reached [1]. The usual initial states are analytic wave functions, generally with one vortex at the center of the trap. However, such an analytic initial function may exhibit slow convergence and often may lead to an inappropriate final vortex lattice structure. We will use an analytic initial function modulated by a random phase at different space points and show that this procedure is essential in addressing the convergence issues, as well as in obtaining the correct vortex lattice structure for a given set of system parameters. Moreover, the GP equation of a rapidly rotating BEC with a very large number of vortices, viz. Figs. 2(c) and (d) with 37 and 61 vortices, faces a convergence difficulty even after random phase modulation. In this latter case, when a pre-calculated converged wave function of the rotating BEC with a smaller number of vortices is used as the initial state, the convergence of the algorithm is vastly improved, resulting in the reduction of more than 90% in execution time.

In Section 2 we present the GP equation for a rotating BEC in an anisotropic trap. We present the mean-field model and a general scheme for its numerical solution. The reduced 2D GP equation appropriate for a disk-shaped rotating BEC is also presented there. The details about the computer programs, and their input/output files, etc. are given in Section 3. The numerical method and results are given in Section 4, where we illustrate the generation of vortex lattices by employing the imaginary-time propagation in rapidly rotating trapped BECs with different angular frequencies and interaction strengths (nonlinearities). The stability of these vortex lattices is demonstrated in real-time propagation using the corresponding converged solution obtained by the imaginary-time propagation as initial states. The efficiency of parallelization of the present OpenMP programs in multi-core computers using the GNU and Intel compilers is also demonstrated there. Finally, a brief summary is given in Section 5.

2. The Gross–Pitaevskii equation for a rotating condensate

A non-rotating BEC made up of N atoms, each of mass m, can be described by the following mean-field GP equation for a wave function $\phi(\mathbf{r}, t)$ at the space point $\mathbf{r}$ at time t [8]

$$
i\hbar \frac{\partial \phi(\mathbf{r}, t)}{\partial t} = \left[-\frac{\hbar^2}{2m} \nabla_{\mathbf{r}}^2 + \frac{1}{2} m\omega^2 (\gamma^2 x^2 + \nu^2 y^2 + \lambda^2 z^2) \right.
$$
$$
\left. + \frac{4\pi\hbar^2 aN}{m} |\phi(\mathbf{r}, t)|^2 \right] \phi(\mathbf{r}, t), \quad i = \sqrt{-1}, \tag{1}
$$

where $\mathbf{r} \equiv (\boldsymbol{\rho}, z) \equiv (x, y, z)$, a is the atomic s-wave scattering length, and ω is the reference trapping frequency, with γ, ν, λ representing the trap anisotropies along the x, y, z directions, respectively. The normalization condition is $\int d\mathbf{r} |\phi(\mathbf{r}, t)|^2 = 1$. This equation can be derived from the energy functional [8]

$$
E[\phi] = \int d\mathbf{r} \left[\frac{\hbar^2}{2m} |\nabla_{\mathbf{r}} \phi|^2 + \frac{1}{2} m\omega^2 (\gamma^2 x^2 + \nu^2 y^2 + \lambda^2 z^2) |\phi|^2 \right.
$$
$$
\left. + \frac{2\pi\hbar^2 aN}{m} |\phi|^4 \right]. \tag{2}
$$

The formation of a vortex lattice in a rapidly rotating BEC can be conveniently calculated in the rotating frame, where the generated vortex lattice forms a stationary state, which can be obtained by the imaginary-time propagation method. Such a dynamical equation in the rotating frame can be written if we note that the Hamiltonian in the rotating frame is given by $H = H_0 - \Omega L_z$ [14], where H_0 is the laboratory frame Hamiltonian, Ω is the angular frequency of rotation around the z axis, and $L_z = i\hbar(y\partial/\partial x - x\partial/\partial y)$ is the z component of the angular momentum. Consequently, the GP equation in the rotating frame has the explicit form [8,10,11,13,15,16]

$$
i\hbar \frac{\partial \phi(\mathbf{r}, t)}{\partial t} = \left[-\frac{\hbar^2}{2m} \nabla_{\mathbf{r}}^2 + \frac{1}{2} m\omega^2 (\gamma^2 x^2 + \nu^2 y^2 + \lambda^2 z^2) \right.
$$
$$
\left. + \frac{4\pi\hbar^2 aN}{m} |\phi(\mathbf{r}, t)|^2 - \Omega L_z \right] \phi(\mathbf{r}, t). \tag{3}
$$

Using the transformations $\mathbf{r}' = \mathbf{r}/l$, $l = \sqrt{\hbar/(m\omega)}$, $a' = a/l$, $t' = \omega t$, $\phi' = l^{-3/2}\phi$, $\Omega' = \Omega/\omega$, and $L'_z = L_z/\hbar$, we obtain the following convenient dimensionless form of the above equation:

$$i\frac{\partial\phi(\mathbf{r}, t)}{\partial t} = \left[-\frac{1}{2}\nabla_{\mathbf{r}}^2 + \frac{1}{2}(\gamma^2 x^2 + \nu^2 y^2 + \lambda^2 z^2) \right.$$
$$\left. + g_{3D}|\phi(\mathbf{r}, t)|^2 - \Omega L_z \right]\phi(\mathbf{r}, t), \quad g_{3D} = 4\pi Na, \quad (4)$$

where we have dropped the primes from the transformed dimensionless variables. We note that Eq. (4) can also be derived from the dimensionless energy functional [8]

$$E[\phi] = \int d\mathbf{r}\left[\frac{1}{2}|\nabla_{\mathbf{r}}\phi|^2 + \frac{1}{2}(\gamma^2 x^2 + \nu^2 y^2 + \lambda^2 z^2)|\phi|^2 \right.$$
$$\left. + \frac{1}{2}g_{3D}|\phi|^4 - \phi^*\Omega L_z\phi \right], \quad (5)$$

obtained using the same transformations and expressing the energy in units of $\hbar\omega$. All derivations and results presented in the following are using these dimensionless variables.

A convenient equation for a quasi-2D disk-shaped BEC under a strong harmonic confinement in the z direction ($\lambda \gg \gamma, \nu$) can be derived using the following ansatz for the wave function [17]:

$$\phi(\mathbf{r}, t) = \psi(\boldsymbol{\rho}, t) \times \frac{1}{(\pi d_z^2)^{1/4}} \exp\left(-\frac{z^2}{2d_z^2} \right), \quad d_z = \sqrt{\frac{1}{\lambda}}, \quad (6)$$

where we assume that because of the strong confinement the dynamics in the z direction will be frozen to a time-independent Gaussian of width d_z, and that the relevant dynamics will evolve only in the x-y plane. If we substitute the ansatz (6) to Eq. (4), we can integrate out the z variable and obtain the corresponding dynamical equation in 2D, valid for a quasi-2D rotating BEC in a disk-shaped trap [1,17]:

$$i\frac{\partial\psi(\boldsymbol{\rho}, t)}{\partial t} = \left[-\frac{1}{2}\nabla_{\boldsymbol{\rho}}^2 + \frac{1}{2}(\gamma^2 x^2 + \nu^2 y^2) + g_{2D}|\psi(\boldsymbol{\rho}, t)|^2 \right.$$
$$\left. - \Omega L_z \right]\psi(\boldsymbol{\rho}, t), \quad g_{2D} = \frac{4\pi aN\sqrt{\lambda}}{\sqrt{2\pi}}, \quad (7)$$

with the normalization condition $\int d\boldsymbol{\rho}|\psi(\boldsymbol{\rho}, t)|^2 = 1$. The energy functional corresponding to Eq. (7) is

$$E[\psi] = \int d\boldsymbol{\rho}\left[\frac{1}{2}|\nabla_{\boldsymbol{\rho}}\psi|^2 + \frac{1}{2}(\gamma^2 x^2 + \nu^2 y^2)|\psi|^2 \right.$$
$$\left. + \frac{1}{2}g_{2D}|\psi|^4 - \psi^*\Omega L_z\psi \right]. \quad (8)$$

We use the split-step Crank–Nicolson algorithm for the solution of the GP equations (4) and (7). This approach has been elaborated in detail in Ref. [1]. In the following we describe the necessary modifications for the 2D equation (7). We follow the identical prescription in 3D. Noting that $L_z = i\hbar(y\partial/\partial x - x\partial/\partial y)$, we split the Hamiltonian into three parts:

$$H \equiv H_1 + H_2 + H_3, \quad (9)$$

$$H_1 = \frac{1}{2}(\gamma^2 x^2 + \nu^2 y^2) + g_{2D}|\psi|^2, \quad (10)$$

$$H_2 = -\frac{1}{2}\frac{\partial}{\partial x^2} - i\Omega y\frac{\partial}{\partial x}, \quad (11)$$

$$H_3 = -\frac{1}{2}\frac{\partial}{\partial y^2} + i\Omega x\frac{\partial}{\partial y}. \quad (12)$$

In this approach we perform the time propagation over infinitesimally small time step first over only the part H_1, and then over the part H_2, and finally over the part H_3 of the Hamiltonian. Essentially, we split Eq. (7) into

$$i\frac{\partial\psi}{\partial t} = H_1\psi, \quad i\frac{\partial\psi}{\partial t} = H_2\psi, \quad i\frac{\partial\psi}{\partial t} = H_3\psi, \quad (13)$$

and perform the time propagation over these three sub-equations successively and independently of each other, in the given order.

We first solve the first of Eqs. (13) starting from an initial state $\psi(\boldsymbol{\rho}, t_0)$ at $t = t_0$ to obtain the first intermediate solution after an infinitesimal time step Δ. Then this intermediate solution is used as an initial value to solve the second of Eqs. (13), yielding the second intermediate solution at the time $t = t_0 + \Delta$, which is then used to propagate the third of Eqs. (13) over the infinitesimal time Δ to yield the final solution at $t = t_0 + \Delta$, after one full time iteration of Eq. (7). This procedure is repeated n times to get the final solution at time $t_{\text{final}} = t_0 + n\Delta$.

The first equation of (13) with H_1 has the analytic solution [1], which we denote by $\psi^{k+1/3}$ when propagating between the time steps k and $k + 1$. Similarly, we denote by $\psi^{k+2/3}$ the wave function after the time propagation with respect to H_2, and finally by ψ^{k+1} after additional propagation with respect to H_3, i.e., after one full time iteration. Following Ref. [1] and using notations therein, we discretize the second equation of (13) for H_2 alone as

$$i\frac{\psi_i^{k+2/3} - \psi_i^{k+1/3}}{\Delta} = -\frac{1}{2}\frac{1}{2h_x^2}\left\{ \left(\psi_{i+1}^{k+2/3} - 2\psi_i^{k+2/3} \right. \right.$$
$$+ \psi_{i-1}^{k+2/3} \right) + \left(\psi_{i+1}^{k+1/3} - 2\psi_i^{k+1/3} + \psi_{i-1}^{k+1/3} \right) \bigg\}$$
$$- \frac{i\Omega y_j}{4h_x}\left\{ \left(\psi_{i+1}^{k+2/3} - \psi_{i-1}^{k+2/3} \right) + \left(\psi_{i+1}^{k+1/3} - \psi_{i-1}^{k+1/3} \right) \right\}, \quad (14)$$

where $\psi_i^t = \psi(x_i, y_j, t)$ refers to the wave function value at the spatial grid point determined by $x \equiv x_i = -N_x h_x/2 + ih_x$, $y_j = -N_y h_y/2 + jh_y$, $i = 0, 1, 2, \ldots, N_x$, and $j = 0, 1, 2, \ldots, N_y$. Here h_x, h_y are the space steps along the x and y directions, respectively, and $t = k + 1/3$ or $k + 2/3$ refers to the time iteration [1], connecting the present $(k + 1/3)$ to the future $(k + 2/3)$ in propagation with respect to H_2.

The above procedure results in a set of tridiagonal equations (14) in $\psi_{i+1}^{k+2/3}$, $\psi_i^{k+2/3}$, and $\psi_{i-1}^{k+2/3}$ at time $t_{k+2/3}$, which are solved using the proper boundary conditions [1]. The tridiagonal equations are written explicitly as $A_i^-\psi_{i-1}^{k+2/3} + A_i^0\psi_i^{k+2/3} + A_i^+\psi_{i+1}^{k+2/3} = b_i$, where

$$b_i = \frac{i\Delta}{4h_x^2}\left(\psi_{i+1}^{k+1/3} - 2\psi_i^{k+1/3} + \psi_{i-1}^{k+1/3} \right)$$
$$- \frac{\Delta\Omega y_j}{4h_x}\left(\psi_{i+1}^{k+1/3} - \psi_{i-1}^{k+1/3} \right) + \psi_i^{k+1/3}, \quad (15)$$

$$A_i^0 = 1 + \frac{i\Delta}{2h_x^2}, \quad A_i^- = -\frac{i\Delta}{4h_x}\left(\frac{1}{h_x} - i\Omega y_j \right),$$

$$A_i^+ = -\frac{i\Delta}{4h_x}\left(\frac{1}{h_x} + i\Omega y_j \right). \quad (16)$$

The discretization for H_3 is performed similarly. The tridiagonal set of equations above is very similar to Eqs. (34) and (35) of Ref. [1], and the real-time propagation routine is programmed and solved in identical fashion after a straightforward modification to include the extra terms due to a non-zero value of Ω in these equations. The imaginary-time propagation routine corresponds to a transformation $t \to -it$ or $\Delta \to -i\Delta$ [1] and hence can be obtained by replacing $i\Delta \to \Delta$ in Eqs. (15) and (16) in the real-rime routine, which is performed in our combined real- and imaginary-time programs by the selection parameter OPTION_RE_IM.

Instead of evaluating the real energies from Eqs. (5) and (8) in 3D and 2D involving complex algebra over complex wave functions, it is convenient to write a real expression for the energy. To calculate the energy and the chemical potential, we write the two coupled nonlinear equations for the real and imaginary parts

of the wave function ($\psi = \psi_R + i\psi_I$), viz. Eqs. (2.1) of Ref. [15]. The equation satisfied by the real part is

$$i\frac{\partial \psi_R(x,y;t)}{\partial t} = \left[-\frac{1}{2}\nabla^2 + \frac{1}{2}(\gamma^2 x^2 + \nu^2 y^2) \right.$$
$$\left. + g_{2D}|\psi(x,y;t)|^2 \right] \psi_R(x,y;t) + \Omega \left(y\frac{\partial}{\partial x} - x\frac{\partial}{\partial y} \right) \psi_I(x,y;t).$$
$$(17)$$

In this equation ψ_R is not normalized to unity. Using Eq. (17), the energy and the chemical potential can be expressed in 2D as

$$\frac{1}{\int dx dy\, \psi_R^2} \int d\boldsymbol{\rho} \left[-\frac{1}{2}(\nabla_{\boldsymbol{\rho}}\psi_R)^2 + \frac{1}{2}(\gamma^2 x^2 + \nu^2 y^2)\psi_R^2 \right.$$
$$\left. + \alpha g_{2D}(\psi_R^2 + \psi_I^2)\psi_R^2 + \Omega \psi_R \left(y\frac{\partial}{\partial x} - x\frac{\partial}{\partial y} \right) \psi_I \right], \qquad (18)$$

where the value $\alpha = 1$ corresponds to the chemical potential μ, and the value $\alpha = 1/2$ to the energy E per atom. A similar expression for energy and chemical potential in 3D is

$$\frac{1}{\int d\mathbf{r}\, \phi_R^2} \int d\mathbf{r} \left[-\frac{1}{2}(\nabla_{\mathbf{r}}\phi_R)^2 + \frac{1}{2}(\gamma^2 x^2 + \nu^2 y^2 + \lambda^2 z^2)\phi_R^2 \right.$$
$$\left. + \alpha g_{3D}(\phi_R^2 + \phi_I^2)\phi_R^2 + \Omega \phi_R \left(y\frac{\partial}{\partial x} - x\frac{\partial}{\partial y} \right) \phi_I \right]. \qquad (19)$$

Eqs. (18) and (19) are equivalent to Eqs. (5) and (8) and involve algebra of real functions. Hence these equations lead to far more accurate numerical results than the previous set of expressions. Specifically, the calculation of the rotational energy and the kinetic energy term involving derivatives and gradients of a complex wave function in Eqs. (5) and (8) can be numerically problematic.

The initial wave function in the imaginary-time programs is taken to be one containing a single vortex at the center, aligned with the z axis. Explicitly, for the 2D and 3D programs, we take, respectively

$$\psi_{\text{initial}}(x,y) = \frac{x + iy}{\sqrt{\pi d_{xy}^2}} \exp\left(-\frac{x^2 + y^2}{2d_{xy}^2} + 2\pi i\mathcal{R}(x,y) \right),$$

$$\phi_{\text{initial}}(x,y,z) = \psi_{\text{initial}}(x,y) \frac{1}{(\pi d_z^2)^{1/4}} \exp\left(-\frac{z^2}{2d_z^2} \right), \qquad (20)$$

where d_{xy} and d_z are width parameters in the x-y plane and in the z direction, and $\mathcal{R}(x,y)$ is a random number. In numerical calculation, the random phase ensures that the number of vortices changes by units of one, as parameters, e.g., nonlinearity and angular frequency, are changed. Without the random phase, the number of vortices changes by units of two or multiples of two. In fact, any localized normalizable initial function modulated by a random phase at different space points, e.g., a Gaussian function without any vortices, obtained by setting $(x + iy) = 1$ in Eq. (20), will lead to the same vortex lattice as the initial function (20) with one vortex. Without the random phase these functions usually will lead to different results [16].

3. Details about the programs

All input data (number of atoms, scattering length, harmonic oscillator trap length, trap anisotropy, etc.) are conveniently placed at the beginning of each Fortran program, as before [3]. Hence after changing the input data in a Fortran program a recompilation is required. The C programs use external input files that contain all parameters, and their adjustment does not require a recompilation. The source programs are located in the directory `src` within the corresponding package directory (BEC-GP-ROT-OMP-C for the C programs and BEC-GP-ROT-OMP-F for

the Fortran ones). They can be compiled by the `make` command using the `makefile` in the corresponding package root directory. The examples of produced output files can be found in the directory `output`, although some large density files are omitted, to save space. The programs use an initial state with repeatable random phase. A different random phase can be generated by changing the variable SEED in the subroutine Initialize for the Fortran programs, or in the corresponding input file for the C programs. The provided Fortran output files are calculated with SEED = 13 using the one-vortex initial function (20). The change of the variable SEED implies a different initial function, thus changing the output files. In the Fortran programs, the random phase is included by the integer parameter RANDOM: the value 0 excludes the random phase and 1 includes it. The integer parameter FUNCTION permits the selection of a Gaussian or a one-vortex initial function: the value 0 selects a Gaussian function and 1 selects the one-vortex function (20). For the C programs, the input files contain variables providing the same functionality, which is explained there. After running a program and obtaining the results, one can use the file `fig*.gnu` in the directory `output` to visualize the density profiles, relying on a popular software package gnuplot. These files are used by invoking the command `gnuplot fig*.gnu` to obtain an eps figure of the generated vortex lattice. Depending on the density file to be plotted, one has to adjust the corresponding line in the `fig*.gnu` file. Currently it is set to use the density file provided as an example and already present in the BEC-GP-ROT-OMP distribution.

The output files are conveniently named such that their contents can be easily identified, following the naming convention introduced in Ref. [3]. For example, a file named `<code>-out.txt`, where `<code>` is a name of the individual program, represents the general output file containing input data, time and space steps, nonlinearity, energy, and chemical potential. A file named `<code>-den2d.txt` is the output file with the reduced (integrated) 2D condensate density. There are output files for reduced (integrated) 1D densities for different programs. Typically, a user first solves the stationary problem using the imaginary-time programs, and then uses the real-time programs to read the pre-calculated stationary wave function and to study the dynamics. To read the pre-calculated wave function the parameter NSTP should be set to zero. In this way one can also run the imaginary time program with a pre-calculated wave function. The supplied programs have the pre-defined value NSTP = 1 and use the analytic wave function (20) as the initial state. In each program the selection for imaginary- or real-time propagation is done by setting the parameter OPTION_RE_IM to 1 or 2, respectively. If the imaginary-time propagation is thus selected, the programs run either by using an initial analytic input function (if NSTP is not set to zero) or by employing a pre-calculated wave function (if NSTP is set to zero). The real-time propagation can successfully work only with a meaningful initial wave function, usually assuming that NSTP = 0 is set, and that the program will read a pre-calculated wave function by the earlier performed imaginary-time propagation. The reader is advised to consult our previous publication where a complete description of the output files is given [4]. The calculation is essentially done in the NPAS time loop, which are in the Fortran programs conveniently divided into 10 equal intervals (NPAS/10). The output files for the reduced 2D densities at the end of each of these intervals are saved as files `<code>*-den-j.txt`, where j=1,...,10. If necessary, one can further customize this by changing and recompiling the Fortran programs. In the C programs the selection of output files is done through the input file, when one can set the desired frequency of saving the output densities, as well as the types of density profiles to be saved. A README.md file, included in the corresponding root directory for C and Fortran, explains the procedure to compile and run the programs in more detail.

The supplied 2D programs are preset to run the imaginary-time propagation using the space steps DX=DY =0.05, numbers of space points NX=NY=256, $g_{2D} = 100, \Omega = 0.8$, the trap parameters $\gamma = \nu = 1$. The 3D programs use DX=DY =0.05, DZ=0.025, NX=NY=256, NZ=32, $\gamma = \nu = 1, \lambda = 100, g_{2D} = 100, g_{3D} = g_{2D}\sqrt{2\pi/\lambda} = 25.0662827$. The large trap parameter λ ensures a disk-shaped BEC, which enables a comparison of the 3D results for the integrated density over the z coordinate with the 2D density profile. This also reduces a transversal instability of the 3D vortex lines. The time steps used are $\Delta = 0.00025$ (imaginary time) and 0.0001 (real time), numbers of time iterations are NPAS=3,000,000 and NSTP=1 (imaginary time) and NSTP=0 (to run real- or imaginary-time propagation with a pre-calculated wave function as an input). To achieve the convergence in some cases (large nonlinearity g_{2D}, g_{3D} and Ω), one may need to increase the values of NX, NY, NPAS, and reduce the space and time steps DX, DY, DZ and DT accordingly. Note that the actual spatial grid used contains (NX+1)×(NY+1) or (NX+1)×(NY+1)×(NZ+1) points, since in each dimension the grid index takes the values from 0 to NX, etc. Therefore, the produced output files also contain the data for such grid sizes.

The function (20) always leads to a converged solution after a large number of time iterations in imaginary-time propagation. A Gaussian wave function given as an input in imaginary-time propagation would sometimes face a convergence difficulty and should not be used. Therefore the programs by default use a better initial state, containing one vortex at the center. Once a stationary vortex lattice is obtained for a specific nonlinearity and angular frequency by imaginary-time propagation, the final wave function so obtained should be used as the initial state for the generation of vortex lattices by imaginary-time propagation with larger nonlinearities and/or angular frequencies. For example, to generate closed hexagonal vortex lattices of 19, 37, and 61 vortices in the panels (b), (c) and (d) of Figs. 2 and 5 in the next section, respectively, we have used the previously calculated initial states of 7, 19, and 37 vortices in the corresponding panels (a), (b), and (c), respectively. Such a choice of dynamically generated multi-vortex initial state with a proper phase distribution enhances the convergence of the numerical scheme enormously compared to the propagation starting from a single-vortex initial state. The reduction in the execution time for the calculation done in this fashion could be as much as 99%. The size of the condensate increases as the nonlinearity and/or the angular frequency Ω are increased. To accommodate a larger condensate, the number of space points NX, NY, etc. should be appropriately increased. To read a pre-calculated wave function by setting NSTP to zero, the grid size in the used wave function file should match exactly the number of points used in the current program. The supplied programs assume equal numbers of space step points in both imaginary- and real-time propagation, and in C programs this is configurable through the input files. If the grid sizes in the two calculations are different, the user can customize the programs to accommodate this. For instance, in Fortran programs the READ statement in the subroutine INITIALIZE should be changed, for instance, from I=0,NX to I= NX2−NXOLD2,NX2+NXOLD2,1, where NXOLD2 is the NX2 value of the previous calculation with a smaller number of grid points.

4. Numerical results

To test the programs and to demonstrate their usage, we have generated vortex-lattice structures using the imaginary-time programs and then ran the real-time programs starting from the previously obtained imaginary-time wave functions as inputs. First, we numerically calculate the critical angular frequency Ω_c, for the generation of a single vortex, using the initial
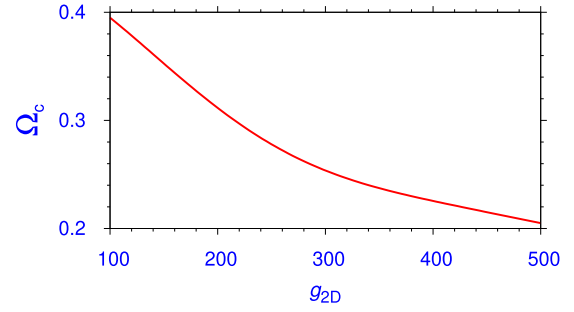

Fig. 1. Critical angular frequency Ω_c for the generation of a single vortex using function (20) with random phase versus nonlinearity g_{2D} for a rotating BEC in 2D. For $\Omega < \Omega_c$ no vortex is generated.

function (20), for a rotating BEC in 2D for different nonlinearities g_{2D}. Without the random phase in the initial wave function this threshold cannot be calculated, as, then, a single vortex continues to exist for $\Omega < \Omega_c$. The result is displayed in Fig. 1. The displayed result is the average over several runs.

We next numerically study the 2D vortex lattice in a rotating BEC using the imaginary-time propagation. The imaginary-time propagation with the supplied 2D program bec-gp-rot-2d-th uses the wave function (20) as the initial state and the parameters $g_{2D} = 100$ and $\Omega = 0.8$. The generated vortex lattice with seven vortices arranged in a triangular lattice in the shape of a closed hexagon is exhibited in Fig. 2(a) through the contour density plot. In Fig. 2(b) we illustrate the 2D vortex lattice with 19 vortices arranged in a triangular lattice in the shape of a closed hexagonal form obtained with parameters $g_{2D} = 100, \Omega = 0.95$. To illustrate the convergence of the imaginary-time propagation we show in Figs. 3(a)–(d) the 2D density profiles at different times, using the analytic wave function (20) as the initial state and employing the parameters $g_{2D} = 100, \Omega = 0.95$, the same as in Fig. 2(b). This scheme shows a slow convergence and the vortex lattice structure practically remains the same from the panel 3(a) for 2×10^5 time steps to the panel 3(c) for 8×10^5 time steps with 19 vortices, before converging to the desired solution in the panel 3(d) after 12×10^5 time steps, containing 19 vortices. The convergence can be highly enhanced if we use the final converged state with a smaller number of vortices as the initial state of a calculation where a larger number of vortices is expected, either because the parameters g_{2D} or Ω or both are larger. In Fig. 3(e)–(h) we demonstrate this and show the vortex lattice evolution of the rotating BEC for the same parameters $g_{2D} = 100, \Omega = 0.95$ as in the panels 3(a)–(d), but starting from the initial state with seven vortices, obtained in Fig. 2(a) for $g_{2D} = 100, \Omega = 0.8$. In Fig. 3 we see that the convergence in this case is achieved much faster. In practical terms, in panels 3(c) after 20,000 time steps or 3(d) after 30,000 time steps of the imaginary-time propagation the convergence is already reached. The reduction in execution time in the later scheme resulting in Figs. 3(e)–(h) compared to the former resulting in Figs. 3(a)–(d) could be very large, viz. 12×10^5 time iterations and 30,000 time iterations in the two schemes.

In Figs. 2(b)–(d) we illustrate 2D vortex lattices with 19, 37, and 61 vortices, respectively, arranged in triangular lattices in the shape of a closed hexagonal form obtained with parameters $g_{2D} = 100, \Omega = 0.95$ in 2(b), $g_{2D} = 500, \Omega = 0.92$ in 2(c), and $g_{2D} = 500, \Omega = 0.978$ in 2(d). As already suggested above, the vortex lattices of Figs. 2(b), (c), and (d) were obtained using the final wave functions of Fig. 2(a), (b), and (c), respectively, as the initial states, to speed up the convergence. We demonstrate the stability of the obtained vortex lattices using the real-time propagation for 500 time units in Figs. 2(e)–(h).
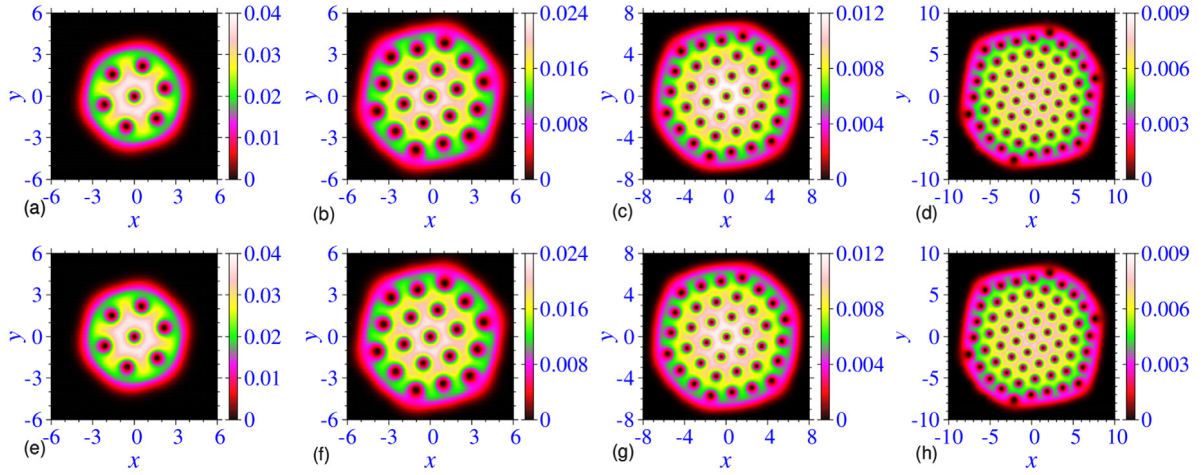
Fig. 2. Contour plots of the density $|\psi(x, y)|^2$ for the generated vortex lattices by the 2D imaginary-time propagation of Eq. (7) for (a) $g_{2D} = 100$, $\Omega = 0.8$, (b) $g_{2D} = 100$, $\Omega = 0.95$, (c) $g_{2D} = 500$, $\Omega = 0.92$, and (d) $g_{2D} = 500$, $\Omega = 0.978$. Panels (e), (f), (g), and (h) display these vortex lattices, respectively, after the additional real-time propagation for 500 units of time using the corresponding imaginary-time wave function as input. The employed trap parameters are $\nu = \gamma = 1$, the space steps are DX=DY=0.05, and the time steps are 0.00025 in imaginary time and 0.0001 in real time. The size of the condensate increases as Ω increases from (a) to (b) and from (c) to (d), and as g_{2D} increases from (b) to (c). The space grids used are (a) 257×257, (b) 321×321, (c) 401×401, and (d) 441×441.
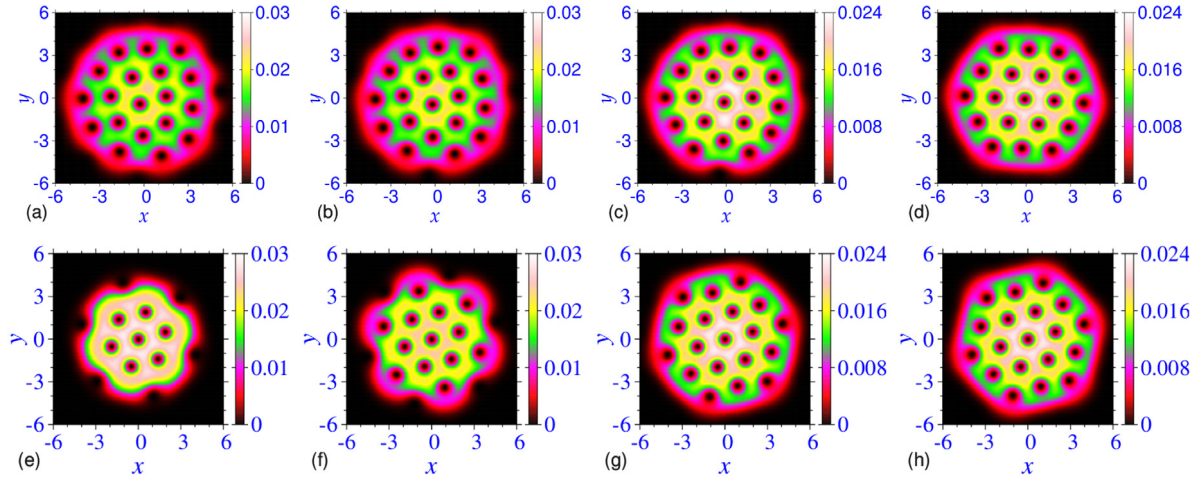
Fig. 3. The convergence of calculation from snapshots at different time steps during the imaginary-time propagation of the 2D equation (7) to generate a vortex lattice for parameters $g_{2D} = 100$, $\Omega = 0.95$. Numerical simulation used the initial state (20), and the panels correspond to (a) 2×10^5, (b) 4×10^5, (c) 8×10^5, and (d) 12×10^5 time steps. For the same parameters, a much faster convergence is obtained in a simulation using as the initial function the converged wave function from Fig. 2(a), obtained for $g_{2D} = 100$, $\Omega = 0.8$. The panels correspond to (e) 5000, (f) 10,000, (g) 20,000, and (h) 30,000 time steps. The employed time step is 0.00025, the space steps DX=DY=0.05, and the grid size used is 321×321 in all panels.

In Figs. 4 we show the increase of the number of vortices with the increase of the angular frequency Ω for a fixed $g_{2D} = 100$ as obtained with the one-vortex initial function and the Gaussian initial function, both modulated by a random phase at different space points. The number of vortices and their orientation in space are identical with both functions, although the energy varies a little from one initial function to another. If the random-phase modulation is removed, these two functions lead to different number of vortices, whereas with the random-phase modulation these functions usually lead to the same number of vortices, viz. Fig. 4.

In Figs. 5 we present the z-integrated reduced 2D density $\int dz |\phi(x, y, z)|^2$, calculated from the 3D imaginary-time runs, with 7, 19, 37, and 61 vortices for the parameters: (a) $g_{2D} = 100$, $\Omega = 0.8$, (b) $g_{2D} = 100$, $\Omega = 0.95$, (c) $g_{2D} = 500$, $\Omega = 0.92$, and (d) $g_{2D} = 500$, $\Omega = 0.978$. The vortex lattices of Figs. 5(b)–(d) were generated, as before, by the imaginary-time propagation of Eq. (4) until the convergence using the final wave function of Figs. 5(a)–(c) as the initial states, respectively. Figs. 5(e)–(h)

illustrate the same reduced densities obtained from the 3D real-time runs after 100 time units using as inputs the final converged imaginary-time wave function of Figs. 5(a)–(d), respectively. The agreement between the imaginary- and the real-time densities demonstrates the stability of the vortex-lattice structures and the employed algorithm. The 2D densities of Figs. 5 are quite similar to those in Fig. 2 with the same 2D nonlinearity and the same angular frequency. To the best of our knowledge, such a clean 61-vortex lattice, viz. Fig. 5(d), is obtained for the first time here in the simulation of the 3D GP equation (4).

In Table 1 we show the energy and the chemical potential of the BECs of Figs. 2(a) and 5(a) calculated starting from the analytic function (20) as the initial state. We also give the energy and the chemical potential of the BECs of Figs. 2(b)–(d) and 5(b)–(d), calculated with the converged wave functions of Figs. 2(a)–(c) and 5(a)–(c), respectively, as the initial states. The 2D energy values $E = 3.190$ and 2.209 shown in Table 1 for $g_{2D} = 100$ and $\Omega = 0.8$ and 0.95, respectively, are in good agreement with the energies $E = 3.1904$ and 2.2106 reported in Fig. 6 of Ref. [15]. The authors
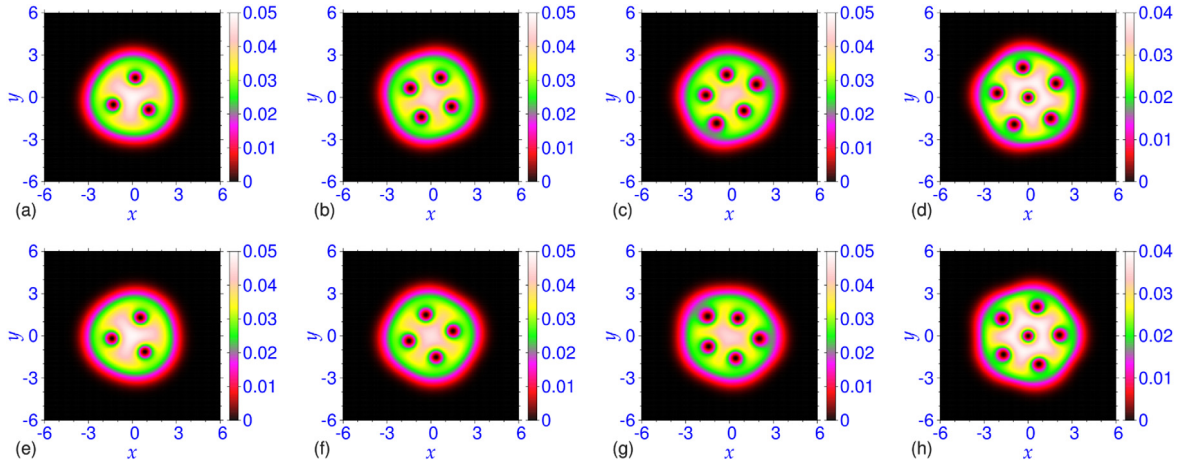
Fig. 4. Contour plots of the density $|\psi(x, y)|^2$ for the generated vortex lattices by the 2D imaginary-time propagation of Eq. (7) for $g_{2D} = 100$, and (a) $\Omega = 0.65$, (b) $\Omega = 0.74$, (c) $\Omega = 0.76$, and (d) $\Omega = 0.78$ obtained with the one-vortex initial state (20). Panels (e), (f), (g), and (h) display these vortex lattices, respectively, obtained with the Gaussian initial state. The employed trap parameters are $\nu = \gamma = 1$, the space steps are DX=DY=0.05, the time step is 0.00025 and the space grid is 257 × 257.
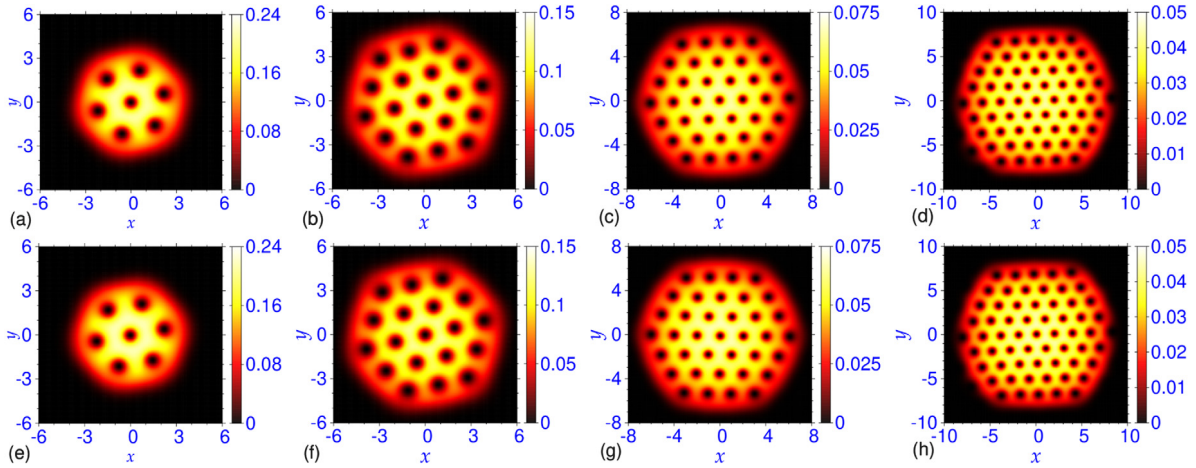
Fig. 5. Contour plots of the density profiles for the generated vortex lattices by the 3D imaginary-time propagation of Eq. (4) for (a) $g_{2D} = 100$, $g_{3D} \equiv g_{2D}\sqrt{2\pi/\lambda} = 25.0662827$, $\Omega = 0.8$, (b) $g_{2D} = 100$, $g_{3D} = 25.0662827$, $\Omega = 0.95$, (c) $g_{2D} = 500$, $g_{3D} = 125.33141$, $\Omega = 0.92$, and (d) $g_{2D} = 500$, $g_{3D} = 125.33141$, $\Omega = 0.978$. Panels (e), (f), (g), and (h) display these vortex lattices, respectively, after the additional real-time propagation for 100 units of time using the corresponding imaginary-time wave function as input. The employed trap parameters are $\nu = \gamma = 1$, $\lambda = 100$, the space steps are DX=DY=0.05, DZ=0.025, and the time steps are 0.00025 in imaginary time and 0.0001 in real time. The space grids used are (a) 257 × 257 × 33, (b) 321 × 321 × 33, (c) 401 × 401 × 33, and (d) 451 × 451 × 33.

Table 1
Energy E and chemical potential μ for the rotating BECs in 2D and 3D shown in Figs. 2 and 5, respectively. For parameters $g_{2D} = 100$, $\Omega = 0.8$ the calculation is performed with the initial state (20). For the BECs from panels (b), (c), and (d) in Figs. 2 and 5 the calculation is performed with the converged wave functions of the corresponding panels (a), (b), and (c) as the initial states.

|	$g_{2D} = 100$ $\Omega = 0.8$	$g_{2D} = 100$ $\Omega = 0.95$	$g_{2D} = 500$ $\Omega = 0.92$	$g_{2D} = 500$ $\Omega = 0.978$
μ (2D)	4.351	2.871	6.257	4.198
E (2D)	3.190	2.209	4.424	2.951
μ (3D)	54.32	52.85	56.20	54.17
E (3D)	53.17	52.19	54.40	52.94

of Ref. [16] also calculated the 2D energy and the chemical potential and we verified using the same parameters that the present energies and chemical potentials are in qualitative agreement with their calculations.

We have tested the performance and scalability of our programs on a modern 8-core Intel Xeon E5-2670 CPUs with 32 GB of RAM. The nodes used for testing contain two CPUs, which allowed us to study the performance of our programs on up to 16 CPU cores. The testing was done at the PARADOX supercomputing facility of the Institute of Physics Belgrade.

For both the C and the Fortran programs the execution time in the beginning reduces rapidly as the number of threads (used CPU cores) is increased. But eventually the gain in the execution time saturates. This is illustrated in Fig. 6, where we plot the execution time versus the number of threads for both the C and the Fortran programs using GNU 7.2.0 and Intel 17.0.4 compilers, respectively. For both compilers, for a large number of threads the C programs are faster. For a small number of threads (four or less), the Fortran programs compiled with the GNU compiler are faster, whereas for the Intel compiler all programs have similar performance, with the C programs being slightly faster.

For a quantitative estimate of the performance we now study the speedup and the efficiency of the programs using different compilers for a calculation: GNU GCC 7.2.0, Intel C 17.0.4, GNU Fortran 7.2.0, and Intel Fortran 17.0.4. The speedup is defined as the ratio $T(1)/T(n)$ where $T(n)$ is the execution time of a run with n threads. The efficiency is the ratio $T(1)/[nT(n)]$, indicating how many of the threads the computer is effectively utilizing. These are illustrated in Fig. 7 for GNU GCC, Intel GCC, GNU Fortran,
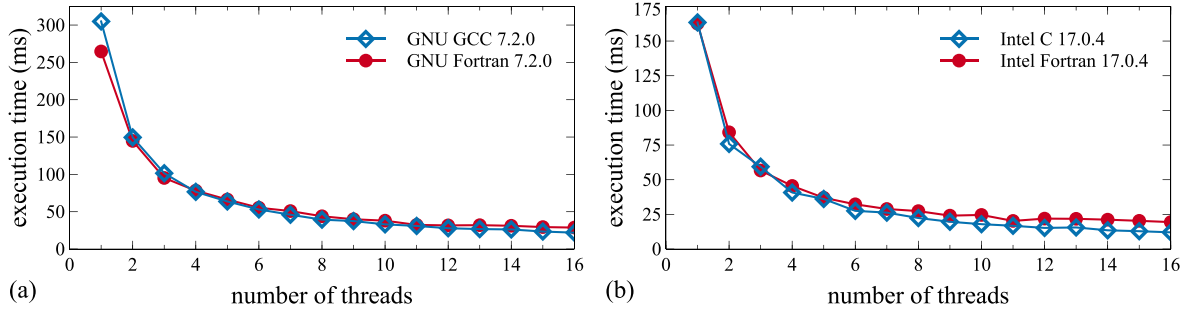
Fig. 6. Wall-clock execution times of BEC-GP-ROT-OMP programs for imaginary-time propagation in 3D (bec-gp-rot-3d-th), compiled with (a) GNU compiler and (b) Intel compiler, as functions of the number of OpenMP threads. The execution times given here are for one iteration, calculated as averages using runs with 1000 iterations (in milliseconds, excluding initialization and input/output operations, as reported by each program) and with the grid size $257 \times 257 \times 33$.
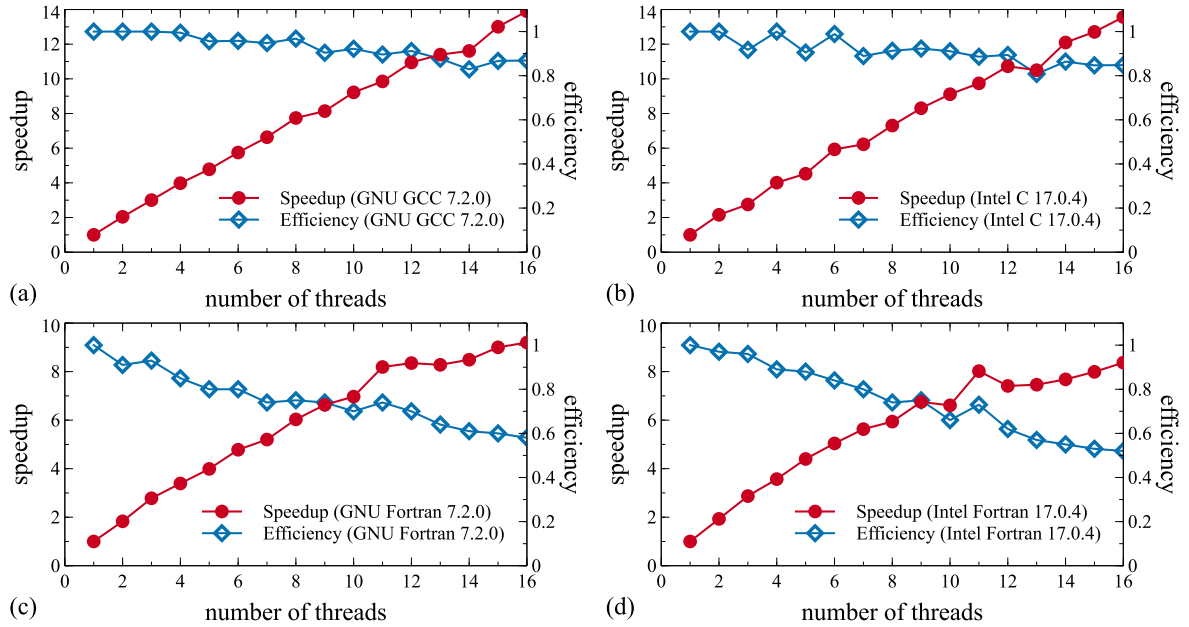
Fig. 7. Speedup in the execution time and scaling efficiency of BEC-GP-ROT-OMP programs for imaginary-time propagation in 3D (bec-gp-rot-3d-th), compared to single-threaded runs for (a) C program compiled with the GNU compiler, (b) C program compiled with the Intel compiler, (c) Fortran program compiled with the GNU compiler, and (d) Fortran program compiled with the Intel compiler. The speedup is calculated as a ratio of the wall-clock execution times $T(1)/T(n)$ for a single-threaded run and a run with n threads, and the scaling efficiency is calculated as a fraction of the obtained speedup compared to a theoretical maximum n. Grid size used for testing is $257 \times 257 \times 33$.

and Intel Fortran compilers, respectively. For a large number of threads, the C programs, viz. plots 7(a)–(b), are more scalable, with large speedup and efficiency compared to the Fortran programs, viz. plots 7(c)–(d). The programs in both programming languages are quite efficient and optimized, but a user should use the specific program and compiler with which he/she has more experience and feels more comfortable.

5. Summary and conclusions

We have presented the efficient OpenMP C and Fortran programs for solving the GP equation for a rotating BEC and use them to calculate the vortex lattices of a rotating BEC by solving the GP equation in the rotating frame. We provide two sets of programs — one for a 3D BEC and the other for a quasi-2D BEC. Each of these programs is capable of executing both the imaginary- and the real-time propagation. We use the split-step Crank–Nicolson algorithm and the programs are based on our earlier OpenMP C and Fortran programs of Ref. [4] for a non-rotating BEC. We solve the GP equation by the imaginary-time propagation with the analytic wave function (20) as the initial state to generate a vortex lattice with a small number of vortices. To solve the GP equation

with a large number of vortices it is much more efficient to use a converged wave function with a smaller number of vortices as the initial state, rather than the analytic function (20). However, the solution can be obtained with any initial state. Nevertheless, the convergence with one initial state could be much faster than with another initial state. For example, to solve the 2D GP equation (7) with parameters $g_{2D} = 100$ and $\Omega = 0.95$ by the imaginary-time propagation using the initial function (20) and obtain the vortex lattice with 19 vortices, one needs 12×10^5 time iterations, viz. Fig. 3. For the same calculation using the pre-calculated vortex lattice with 7 vortices it is sufficient to use only 30,000 time iterations. Although both the C and the Fortran programs produce equivalent results, on a multi-core computer with more than 8 cores, the C programs compiled with both the GCC and the Intel compiler yield a more efficient and faster performance.

The localized normalizable initial function (20) has a random phase at each grid point (x, y) which is necessary to obtain a converged vortex lattice with any number of vortices – even or odd – independent of the initial function. If the random phase is removed from the initial function, the one-vortex initial function (20) leads to a vortex lattice with an odd number of vortices and a Gaussian initial function leads to a vortex lattice with an even

number of vortices. Any localized normalizable initial function with random phase as in Eq. (20), e.g., a Gaussian function or a function with one vortex, usually leads to the same vortex lattice. Without the random phase these functions lead to different vortex lattices.

Acknowledgments

References

[1] P. Muruganandam, S.K. Adhikari, Comput. Phys. Comm. 180 (2009) 1888.
[2] D. Vudragović, I. Vidanović, A. Balaž, P. Muruganandam, S.K. Adhikari, Comput. Phys. Comm. 183 (2012) 2021.
[3] L.E. Young-S., D. Vudragović, P. Muruganandam, S.K. Adhikari, A. Balaž, Comput. Phys. Comm. 204 (2016) 209.
[4] L.E. Young-S., P. Muruganandam, S.K. Adhikari, V. Lončar, D. Vudragović, Antun Balaž, Comput. Phys. Comm. 220 (2017) 503.
[5] R. Kishor Kumar, L.E. Young-S., D. Vudragović, A. Balaž, P. Muruganandam, S.K. Adhikari, Comput. Phys. Comm. 195 (2015) 117.
[6] V. Lončar, A. Balaž, A. Bogojević, S. Škrbić, P. Muruganandam, S.K. Adhikari, Comput. Phys. Comm. 200 (2016) 406;
V. Lončar, L.E. Young-S., S. Škrbić, P. Muruganandam, S.K. Adhikari, A. Balaž, Comput. Phys. Comm. 209 (2016) 190;
B. Satarić, V. Slavnić, A. Belić, A. Balaž, P. Muruganandam, S.K. Adhikari, Comput. Phys. Comm. 200 (2016) 411.
[7] A.L. Fetter, Rev. Modern Phys. 81 (2009) 647.
[8] A.L. Fetter, J. Low Temp. Phys. 161 (2010) 445.
[9] J.R. Abo-Shaeer, C. Raman, J.M. Vogels, W. Ketterle, Science 292 (2001) 476;
V. Schweikhard, I. Coddington, P. Engels, S. Tung, E.A. Cornell, Phys. Rev. Lett. 93 (2004) 210403;
J.R. Abo-Shaeer, C. Raman, W. Ketterle, Phys. Rev. Lett. 88 (2002) 070409.
[10] D.L. Feder, C.W. Clark, B.I. Schneider, Phys. Rev. Lett. 82 (1999) 4956;
D.L. Feder, C.W. Clark, B.I. Schneider, Phys. Rev. a 61 (1999) 011601(R).
[11] R. Zeng, Y.-Z. Zhang, Comput. Phys. Comm. 180 (2009) 854;
A. Aftalion, Q. Du, Phys. Rev. a 64 (2001) 063603;
A. Aftalion, I. Danaila, Phys. Rev. A 68 (2003) 023603.
[12] I. Danaila, F. Hecht, J. Comput. Phys. 229 (2010) 6946;
G. Vergez, I. Danaila, S. Auliac, F. Hecht, Comput. Phys. Comm. 209 (2016) 144;
T. Mizushima, Y. Kawaguchi, K. Machida, T. Ohmi, T. Isoshima, M.M. Salomaa, Phys. Rev. Lett. 92 (2004) 060407.
[13] A.A. Penckwitt, R.J. Ballagh, C.W. Gardiner, Phys. Rev. Lett. 89 (2002) 260402;
M. Tsubota, K. Kasamatsu, M. Ueda, Phys. Rev. a 65 (2002) 023603;
C. Lobo, A. Sinatra, Y. Castin, Phys. Rev. Lett. 92 (2004) 020403.
[14] L.D. Landau, E.M. Lifshitz, Mechanics, Pergamon Press, Oxford, 1960, Section 39.
[15] B.-W. Jeng, Y.-S. Wang, C.-S. Chien, Comput. Phys. Comm. 184 (2013) 493.
[16] W. Bao, H. Wang, P.A. Markowich, Commun. Math. Sci. 3 (2005) 57.
[17] L. Salasnich, A. Parola, L. Reatto, Phys. Rev. A 65 (2002) 043614.