Towards Domain Formation in the Self-Organisation Phase Transition

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Towards Domain Formation in the Self-Organisation Phase Transition

Semester project

by

Kilian Kirchner
kkirchner@student.ethz.ch

Quantum Optics Group
Departement of Physics, D-PHYS
ETH Zürich

Advisors: Justyna Stefaniak
Simon Hertlein

Monday 14th October, 2024

Abstract

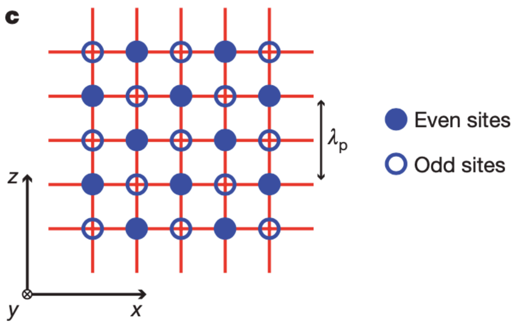
In this Semester project, we explore the formation of domains in the self-organisation phase transition. The domains are identifiable by a shift in the density wave by half a transverse pump wavelength, as well as a phase jump in the light-field scattered from the illuminated region of the density lattice as the transverse pump scans across the domain wall. We extend upon a theoretical proposal, proving the concept for a one dimensional system with periodic boundary conditions, by simulating a trapped Bose-Einstein condensate in two dimensions. To achieve domain formation, we implemented a scanning transverse pump and the correct equations of motion in the limit of a narrow-waisted Gaussian beam on top of the `TorchGPE` python package. For a transverse pump with the waist size of $\sqrt{2}$ times the pump wavelength λ, we observe the formation of two long-lived domains starting from a cigar-shaped cloud of length 156.45 λ by scanning with a frequency of 60 times the recoil frequency ω_r. We confirm a wide range of scanning frequency can lead to domain formation and a quick ramping speed leads to stable self-organisation throughout the whole cloud. We also observe the formation of domains for a larger waist size of 5 λ. In order to fully support an experimental realisation, more scenarios such as different trap geometries and a running wave via an imbalanced pump need to be explored.

Contents

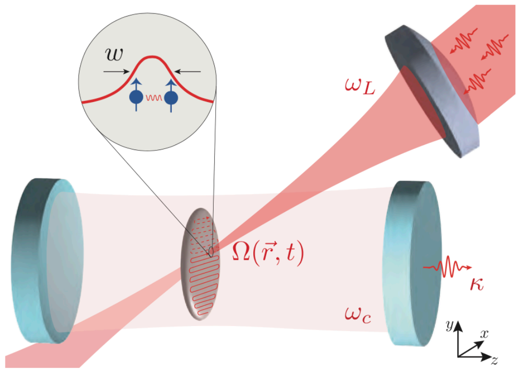
1 Introduction

Cavity quantum electro dynamics with Bose-Einstein condensates have been a key platform to reaching the strong coupling regime between atom and light due to collective ground state occupied [4]. Specifically the transversely driven cavity exhibits infinite-range interaction mediated by cavity photons [8]. Driven by these infinite range interactions, the system exhibits a superradiant Dicke quantum phase transition, which was experimentally first realised over a decade ago at ETH [2]. In the superfluid gas, the phase transition induces the formation of a crystalline lattice as a critical pump power is crossed [2]. The phase transition spontaneously breaks $\mathbf{Z}_2$ symmetry, with atoms ordering on either even or odd sites of a checkerboard lattice [2], pictured in figure 1.1a. While the flat condensate prohibits scattering into the cavity, the lattice structure collectively enhances the process. The phase of the light-field is also spontaneously broken and thus allows for a readout of the parity.

A recent proposal by T. Donner, M. Bonifacio and F. Piazza [3] introduces a novel pumping scheme leading to tunable interactions. A red-detuned transverse pump is rapidly and periodically scanned across the cloud in a single-mode dissipative cavity leading to an attractive time-averaged potential. A visual depiction of the experimental scheme is given in figure 1.1b. While Atoms move on timescales given by the recoil frequency of the pump, the dynamics of the light-field are determined by the dissipation rate of the cavity. Scanning over the cloud at frequencies above the cavity's loss rate reestablishes global order. Lower scanning frequencies give rise to a regime of tunable interactions. The transverse drive beam shape determines the range and shape of interactions. By tightly focusing the waist we limit the range of photon-mediated interactions in the gas. Due to effective short-range interactions, the system can form domains scattering different phases of light after the phase transition. The Kibble-Zurek mechanism describes the appearance of the domains.



(a)



(b)

Figure 1.1: (a) The two possible atomic density checkerboard-lattices after the quantum phase transition are shown. Figure taken from [2]. (b) A Bose-Einstein condensate is placed within a cavity, with the single mode ω_c and dissipation rate κ. A transverse beam with waist w and beam-shape $\Omega(\vec{r},t)$ and single-mode ω_L is periodically scanned across the condensate. The inset shows the microscopic photon-mediated interaction between two atomic dipoles within the beam. Figure taken from [3].

In this semester project, we explore the formation of these domains using the `TorchGPE` Python package [6]. To this end we extend upon the existing package by implementing a moving transverse pump and adjust the time-dependent potential simulated to match our needs. In contrast to the simulations performed in the proposing work, simulating a 1D system with periodic boundary conditions, we simulate a trapped Bose-Einstein condensate in 2D modelling the experimental setup. In the second chapter we first go through the derivation of the renormalization of atomic contact-interaction from 3D to 2D. Next, we state the the mean-field equations of motion for the atomic wave function and light-field, which are propagated in time. We show how the beam-shape tunes the interactions. We end the chapter by intuitively explaining how the driving scheme leads to the formation of domains and the necessary conditions. In Chapter 3 we first briefly describe the package used for simulating the dynamics and show the extensions performed in this work. Through a tutorial we explain the individual steps of a simulation. Next, we motivate the parameters chosen to simulate. Finally, we discuss various simulations performed, showing the formation of domains, exploring the limits of varying the scanning frequency, ramping speed and the effect of changing waist size. Lastly, we give an outlook on possible future steps and scenarios to simulate to ensure a successful experiment in the laboratory.

2
Analytics

In this chapter we provide the analytical calculations of the effective interaction strength employed due to the dimensional reduction of the simulation from 3D to 2D as well as the mean-field equations of motion of the light-field $\alpha(t)$ and the matter wave function $\psi(\mathbf{r},t)$ which is simulated in the program. We end this chapter by giving an intuitive non-analytic description of the pumping scheme, which allows for domains to form.

2.1. Effective interaction strength

Due to computational complexity, we perfrom simulations in 2D. The simulation effectively approximates the 3D picture in the case where the out-of-plane trapping frequency, i.e. ω_z, is much larger than the trapping frequencies in-plane ω_x and ω_y. We approximate 3D scattering theory in 2D by rescaling contact-interactions, which can be done when the out-of-plane trapping frequency ω_z is much larger than the trapping frequencies in-plane ω_x and ω_y [9]. In the following section we show the derivation of the effective atom-atom 2D contact interaction $g_{aa,2D}$ matching the 3d s-wave scattering length a_s, based on two approaches resulting in a different scaling factors.

We start with the three-dimensional Gross-Pitaevskii equation (GPE):

$$i\hbar\frac{\partial\psi_{3D}(\mathbf{r},t)}{\partial t} = (-\frac{\hbar^2}{2m}\nabla^2 + \frac{m\omega_z^2}{2}z^2 + V_{trap}(x,y) + Ng_{aa,3D}|\psi_{3D}(\mathbf{r},t)|^2)\psi_{3D}(\mathbf{r},t), \tag{2.1}$$

where $g_{aa,3D} = \frac{4\pi\hbar^2 a_s}{m}$ is the 3D contact interaction and $V_{trap}(x,y)$ is the trapping potential in transverse direction. In the scenario of $\omega_x, \omega_y \ll \omega_z$, we use the trial function $\psi_{3D}(\mathbf{r},t) \approx \phi_{2D}(x,y,t)\psi_\perp(z,t;\sigma(x,y,t))$, where $\sigma(x,y,t)$ gives the extension of the BEC in the x-y plane. To maintain normalization, we require both partial functions to be separately normalized. In the case of a harmonic trapping potential [9] we find the 1D gaussian function,

$$\psi_\perp(z,t;\sigma(x,y,t)) = \frac{1}{\pi^{1/4}\sqrt{\sigma(x,y,t)}}\exp\left(-\frac{z^2}{2\sigma^2(x,y,t)}\right). \tag{2.2}$$

In the approach of [9] the reduced contact-interaction is derived from the action-functional, whose Euler-Lagrange equation is the 3D GPE,

$$S = \int dtd\mathbf{r}\psi_{3D}^*(\mathbf{r},t)[i\hbar\frac{\partial}{\partial t} + \frac{\hbar^2}{2m} - V_{trap}(x,y) - \frac{m\omega_z^2}{2}z^2 - \frac{1}{2}g_{aa,3D}N|\psi_{3D}(\mathbf{r},t)|^2]\psi_{3D}(\mathbf{r},t). \tag{2.3}$$

After inserting the trial function (2.2), we integrate the functional out in z to arrive at an effective 2D functional. As in the full 3D picture the Euler-Lagrange equations of the effective 2D action functional will give a 2D GPE, with rescaled contact interactions. The effective 2D Lagrangian is given by [9],

$$L_{2D} = \phi_{2D}^*[i\hbar\frac{\partial}{\partial t} + \frac{\hbar^2}{2m}\nabla_\perp^2 - V_{trap}(x,y) - \frac{Ng_{aa,3D}}{\sqrt{2\pi}\sigma}|\phi_{2D}|^2 - \frac{\hbar^2}{2m}\sigma^{-2} - \frac{m\omega_z^2}{2}\sigma^2]\phi_{2D}, \tag{2.4}$$

and the Euler Lagrange equation for the extension in transverse direction $\phi_{2D}(x, y, t)$ now reads as

$$i\hbar\frac{\partial\phi_{2D}}{\partial t} = [-\frac{\hbar^2}{2m}\nabla_\perp^2 + V_{trap}(x, y) + \frac{Ng_{aa,3D}}{\sqrt{2\pi}\sigma}|\phi_{2D}|^2 + \frac{\hbar^2}{2m}\sigma^{-2} + \frac{m\omega_z^2}{2}\sigma^2]\phi_{2D} \tag{2.5}$$

The experiments performed by the IMPACT team use bose-einstein condensates of Rb-87. These condensates are formed by cooling an atomic dilute gas [1] and we can easily calculate that the 3D contact interactions are weak in a sense that $g_{aa,3D}N \ll 1$. In this weak interaction regime, we can take $\sigma = a_z$ [9], with a_z being the harmonic oscillator length in z direction. With a harmonic trapping potential in transverse direction, i.e. $V_{trap}(x, y) = \frac{m}{2}(\omega_x^2 x^2 + \omega_y^2 y^2)$, the Euler Lagrange equation for ϕ_{2D} reduces to

$$i\hbar\frac{\partial\phi_{2D}}{\partial t} = [-\frac{\hbar^2}{2m}\nabla_\perp^2 + V_{trap}(x, y) + \frac{g_{aa,3D}N}{\sqrt{2\pi}a_z}|\phi_{2D}^2| + \hbar\omega_z]\phi_{2D}. \tag{2.6}$$

From this we can readoff that the rescaled interaction strength in 2D is

$$g_{aa,2D} = \frac{g_{aa,3D}}{\sqrt{2\pi}a_z} = \frac{\sqrt{8\pi}\hbar^2 a_s}{m a_z}. \tag{2.7}$$

This form of rescaling the contact interaction strength is the readily implemented in `TorchGPE`.

Alternatively, one can directly integrate out the 3D GPE in z, after replacing $\psi_{3D}(\mathbf{r}, t)$ with the above stated trial wave function (2.2), and arrive at an effective 2D GPE of the form [10]:

$$i\hbar\frac{\partial\phi_{2D}}{\partial t} = [-\frac{\hbar^2}{2m}\nabla_\perp^2 + V_{trap}(x, y) + \frac{Ng_{aa,3D}}{\sqrt{3\pi}a_z}|\phi_{2D}|^2]\phi_{2D}. \tag{2.8}$$

We can immediately read off that the 3D interaction strength is renormalized by the factor $\sqrt{3\pi}$, giving an effective 2D interaction strength of

$$g_{aa,2D} = \frac{g_{aa,3D}}{\sqrt{3\pi}a_z}. \tag{2.9}$$

We implemented this form of rescaling of interaction strength into the code and used it in all simulations presented here in this report. We chose this renormalization to make our scattering lengths comparable to the 1D scattering length employed in [3].

2.2. Equations of motion

In this section, we model the interaction between the atoms and the light-field. We start with the mean-field description of the atoms using the Gross-Pitaevskii equation and describe all included terms. We then show the light-field equations of motion. Through adiabatic field elimination we will bring the latter field into a steady-state form. Inserting this back into the Matter-wave function we arrive at the analytical equation propagated in time by the simulation.

The equation of motion for the matter wave function $\psi(x, y, t)$ in 2D is given by the Gross-Pitaevskii equation [3]:

$$i\hbar\frac{\partial\psi(x, y, t)}{\partial t} = [-\frac{\hbar^2}{2m}\nabla^2 + g_{aa,2D}|\psi(x, y, t)|^2 + V_{trap}(x, y) + V_{lat}(x, y)]\psi(x, y, t) \tag{2.10}$$

The first term gives the kinetic energy of the cloud and the second term in the above equation models contact interactions between atoms in the cloud using the reduced interaction strength derived in the section above. The third term is the trapping potential and the fourth contains the lattice potentials formed by the transverse pump (TP), cavity photons and the interference of the two. Explicitly, the trapping potential is harmonic and given by

$$V_{trap}(x, y) = \frac{\hbar m}{2}(\omega_x^2 x^2 + \omega_y^2 y^2) \tag{2.11}$$

The lattice potential formed by the light from the TP and in the cavity is given by:

$$\begin{aligned}
V_{lat}(x,y) =& \hbar\frac{g(\mathbf{r})^2}{\Delta_a}|\alpha|^2 + \hbar\frac{h(\mathbf{r})^2}{\Delta_a} + \hbar\frac{g(\mathbf{r})h(\mathbf{r})}{\Delta_a}(\alpha(t)+\alpha^*(t)) && (2.12)\\
=& \hbar\frac{g_0^2}{\Delta_a}\cos^2(k_c x)|\alpha|^2 + \hbar\frac{\Omega_p^2}{\Delta_a}\Omega(x,y)\cdot\cos^2(k_p y) && (2.13)\\
&+ \hbar\frac{\Omega_p g_0}{\Delta_a}\cos(k_c x)\sqrt{\Omega(x,y)}\cos(k_p y)\cdot 2\mathrm{Re}(\alpha(t)) && (2.14)\\
=& \hbar|\alpha|^2\frac{g_0^2}{\Delta_a}\cos^2(k_c x) + V_{TP}\Omega(x,y)\cdot\cos^2(k_p y) && (2.15)\\
&+ \hbar\frac{g_0\sqrt{\Omega(x,y)V_{TP}|\Delta_a|/\hbar}}{\Delta_a}\cos(k_c x)\cos(k_p y)\cdot 2\mathrm{Re}(\alpha(t)), && (2.16)
\end{aligned}$$

where Δ_a is the atomic detuning of the TP and we express the transverse drive power, or in other words the lattice depth of the standing wave, as $V_{TP} = \hbar\Omega_p^2/|\Delta_a|$. The last reformulation of V_{lat} is done here to arrive at an expression in terms of V_{TP}, as this is the parameter one specifies in the code. The cavity-mode forms a standing wave, $g(\mathbf{r}) = g_0\cos(k_c x)$, g_0 being the cavity-atom coupling and $k_c = 2\pi/\lambda_c$ being the modulus of the cavity-mode wave-vector. The TP, also being a standing wave, has the mode-function $h(\mathbf{r}) = \Omega_p\sqrt{\Omega(x,y)}\cos(k_p y)$, with the gaussian shape factor of the transverse lattice being,

$$\Omega(x,y) = \frac{1}{1+(\frac{\lambda y}{\pi w})^2}\exp\left(\frac{-2x^2}{w^2+(\frac{\lambda y}{\pi w})^2}\right). \qquad (2.17)$$

The waist of the beam is given by w. Since we define $\Omega(x,y)$ to be the mode function of the pump lattice, the square root of $\Omega(x,y)$ defines the mode function of the entering pump beam.

We now move to the light field $\alpha(t)$ in the dissipative cavity with atoms. It is comprised of a dispersive shift, a dissipative term and a self-consistent coupling term with the cloud plus an interaction term of the cavity and pump photons with the cloud of atoms [8]:

$$\begin{aligned}
i\hbar\frac{\partial\alpha(t)}{\partial t} =& \hbar\big(-\Delta_c - i\kappa + \frac{\hbar}{\Delta_a}\int d\mathbf{r}g(\mathbf{r})|\psi(\mathbf{r},t)|^2\big)\alpha(t) + \frac{\hbar}{\Delta_a}\int d\mathbf{r}g(\mathbf{r})h(\mathbf{r})|\psi(x,y,t)|^2\\
=& -\hbar(\Delta_c + i\kappa)\alpha(t) + \hbar\frac{g_0^2}{\Delta_a}\alpha(t)\int d\mathbf{r}\cos^2(k_c x)|\psi(x,y,t)|^2\\
&+ \hbar\frac{\sqrt{V_{TP}|\Delta_a|/\hbar}g_0}{\Delta_a}\int d\mathbf{r}\cos(k_c x)\sqrt{\Omega(x,y)}\cos(k_p y)|\psi(\mathbf{r},t)|^2
\end{aligned}$$

where Δ_c is the dispersive shift of the cavity frequency via the pump frequency, κ is the dissipation rate of the cavity .

Since the evolution of the cavity field happens on timescales (on the order of $1/\kappa \lesssim 1.1$ ms) faster than the atoms' movement (on the order of $1/\omega_r \approx 42$ ms) we can assume the light field to always be in the steady state. The steady-state solution of the light-field is found by setting it's time evolution to zero:

$$\begin{aligned}
\alpha_{st}(t) =& \frac{\int d\mathbf{r}g(\mathbf{r})h(\mathbf{r})|\psi(x,y,t)|^2}{\Delta_a(\Delta_c + i\kappa - Bg_0^2/\Delta_a)} && (2.18)\\
=& \frac{\int d\mathbf{r}g_0\cos(k_c x)\Omega_p\sqrt{\Omega(x,y)}\cos(k_p y)|\psi(x,y,t)|^2}{\Delta_a(\Delta_c + i\kappa - Bg_0^2/\Delta_a)}, && (2.19)
\end{aligned}$$

where we introduce the bunching parameter $B = \int dxdy|\psi|^2\cos^2(k_c x)$ in the last step, which evaluates the degree of overlap between the cloud wave function and the standing wave of the cavity mode. In the fully condensed super-fluid phase $B = 1/2$ and will go to $B = 1$ in the super-radiant self-organised phase. We insert the steady-state cavity light-field into the matter equation and arrive at a single coupled equation due to adiabatic field elimination:

$$i\hbar\frac{\partial\psi(x,y,t)}{\partial t} = [-\frac{\hbar^2}{2m}\nabla^2 + g_{aa,2D}|\psi(x,y,t)|^2 + V_{trap}(x,y) + \hbar\frac{g(\mathbf{r})^2}{\Delta_a}|\alpha|^2 + \hbar\frac{h(\mathbf{r})^2}{\Delta_a} \tag{2.20}$$

$$+\hbar g(\mathbf{r})h(\mathbf{r})\cdot\frac{2\Delta_c}{\Delta_a^2((\Delta_c + Bg_0^2/\Delta_a)^2 + \kappa^2)}\int d\mathbf{r'}g(\mathbf{r}')h(\mathbf{r}')|\psi(x',y',t)|^2]\psi(x,y,t). \tag{2.21}$$

From the last term in the above equation, we can see that only atoms that are within the TP can interact, as $h(\mathbf{r})$ goes to zero far away from the waist of the beam. For this we should keep in mind that the area of interaction is tunable via the gaussian shape factor $\Omega(x,y)$ of the pump lattice, contained in the TP mode $h(\mathbf{r}) = \Omega_p\sqrt{\Omega(x,y)}\cos(k_p y)$.

2.3. Domain-formation

We now intuitively describe the pumping scheme, proposed in [3], that allows for domains with different symmetry breaking to form. In the case of static TP with an infinite waist, the atoms self-organise into a checker-board lattice as a critical pump power is crossed. Hereby, a $\mathbb{Z}_2$ symmetry is broken as the atoms organise onto either odd or even sites of the checker-board lattice created by the interference of pump and cavity fields.

We have already established a difference in timescales between the dissipation of photons in the cavity and movement of the atoms in the cloud, which allowed us to find a steady state of the light field. We have also seen we can limit the range of interactions between atoms via the cavity light-field by reducing the waist size of the TP. If we now scan the beam over the cloud, while quenching the pump power over the critical value, we can see that due to the finite range of interactions the formation of domains is conceivable if the cavity field dissipates faster than the movement of the TP. The necessity of finite range interactions imposes the limit of the waist size being smaller than the size of the cloud, $w \ll L$ in the direction of scanning. In the case where the scanning frequency is on the order of the cavity detuning, the potential felt by the atoms due to the moving beam is analogous to a static beam with infinite waist. In the red-detuned case, i.e. $\Delta_a < 0$ the atoms will feel an attractive potential towards positions of the checkerboard lattice given by the interference term in line (2.16). The timescale of atomic movements is given by the recoil frequency ω_r. We can see that we require $\omega_{scan} \gg \omega_r$ in order to prevent the back-action of the atomic density onto the light-field, which determines the phase of the interference pattern determining the symmetry of the organisation lattice. In summary domains can only form, if the following conditions are met

$$(1) \quad w \ll L \tag{2.22a}$$

$$(2) \quad \omega_r \ll \omega_{scan} \ll \Delta_c \tag{2.22b}$$

3 Numerics

In this chapter we cover the code we used and created to simulate the Bose gas transversely driven in a dissipative cavity and discuss various simulations performed. First, we give a qualitative description of the simulation tool, `TorchGPE` [6] and show how the scanning pump was implemented in the existing code. We briefly introduce the class we implemented that stores a density cut and plots it's evolution over time. Next, we show how the simulations are structured in the form of a tutorial. Finally, we discuss select simulations and the effect of varying parameters towards the goal of finding domains.

3.1. GPE code - simulation setup

All simulations done in the course of this work are performed with a GPU accelerated GPE solver `TorchGPE` [6], which we expanded on by implementing various pumping schemes. The solver uses imaginary time evolution to reach the ground state. For real-time evolution the fourier-split-step method is utilized, which alternately propagates the wave function in real-space and fourier-space. The system simulated is a Rb-87 gas in a dispersive cavity, trapped harmonically with radio frequencies. The gas is simulated on a $N_{\text{grid}} \times N_{\text{grid}}$, with the cavity and pump field pre-evaluated at each step in the imaginary or real-time evolution:

$$R_{\text{pump}} = X \cdot \cos(\alpha_p) + Y \cdot \sin(\alpha_p) \tag{3.1}$$

$$R_{\text{cavity}} = X \cdot \cos(\alpha_c) + Y \cdot \sin(\alpha_c), \tag{3.2}$$

$$\tag{3.3}$$

where X and Y are matrices of the size $N_{grid} \times N_{grid}$, giving the respective x- and y-coordinate at each position in the grid, α_p is the angle of the pump and α_c is the angle of the cavity. In all simulations we simulate a transverse pump by choosing $\alpha_c = 0$ and $\alpha_p = \pi/2$, reducing the pre-evaluated matrices to $R_{\text{pump}} = Y$ and $R_{\text{cavity}} = X$. The implementation of the modified dissipative cavity is given in the appendix. The geometry of the simulation is pictured in figure 3.1.

3.2. Implementing the scanning pump

A feature that had to be implemented in the existing modules was the movement of the pump lattice. In this section we briefly show and explain the implementation of this novel feature.

Initially the pump was always assumed to be at a fixed position in time and the pump field was axially symmetric along the chosen pump axis, while being centered at the origin. The simplest implementation introduces time-dependent coordinates for the center of the transverse pump. The pre-evaluated matrices for the pump field is shifted as $X \to \tilde{X}(t) = X - x_{offset}(t)$ and $Y \to \tilde{Y}(t) = Y - y_{offset}(t)$. By passing functions to the variables $x_{offset}(t)$ and $y_{offset}(t)$ the movement of the pump is fully controllable in space and time.

Due to the geometry of the simulation, we let the pump beam move only along the cavity axis, the x-axis. We implement a scanning function, which periodically sweeps over the cloud in the same direction. It's implementation in python is given by:

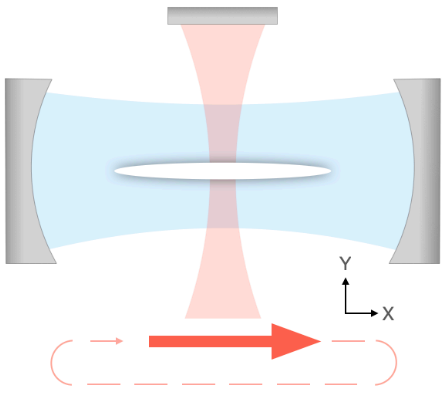


Figure 3.1: A cigar-shaped Bose-Einstein condensate (white) is trapped in a dispersive single-mode (blue) cavity (silver along the x-axis), while being transversely pumped by a standing wave orthogonally (red) along the y-axis. The beam is then periodically scanned from left to right over the cloud.

```python
from scipy import signal
def offset_periodic_sweep(span=1, w_scan=1, t0=0):
    return lambda t: -span/2 if t < t0 else span/2 *signal.sawtooth(w_scan * (t-t0), 1)
```

The parameters passed to the above function are the range of motion `span`, the scanning frequency `w_scan` and the starting time of scanning `t_0`. Note, that the span is a length scale and the scanning motion will range from `-span/2` to `+span/2` along the x- or y-axis.

3.3. Implementing a density monitor

The `TorchGPE` provides multiple possibilities to track the dynamics of the system and visualize the results. In this section we describe the density monitor class we implemented.

While we can store the data of the cavity parameters, the lattice depth and complex light-field α, we are not able to do the same for the density. For a grid of $2^{11} \times 2^{11}$ points, the density of the matter-wave function into a `.csv` format, takes up 3.4 MB of storage. Doing this for multiple snapshots during a simulation is thus unfeasible. If we want to store the density for closer inspection, we do this once at the end of a simulation, and we limit the region saved to the Thomas-Fermi diameter of the cloud. In order to capture the time evolution of the density in a static image, we implemented the class `density_monitor`, see appendix B, which temporarily stores the density cut along the horizontal direction at a given height over time. After the simulation ends, the density monitor assembles the density cuts into a figure as is plotted in ([3] figure 4 (a)). The domain is split into odd and even sites, i.e integer and half-integer wavelengths, and shaded in blue and red respectively. An example can be seen in figure 4.3. This type of plot allows for the quick evaluation of density-wave formation, as in the self-organisation phase, the density is highly localised around the lattice sites, saturating the color.

3.4. Tutorial

In this subsection we present a step-by-step tutorial of the simulation script for a scanning pump run leading to self-organisation. This will demonstrate how the numerical experiment is performed and the data is obtained without detailing the specific code for utility functions.

First, we import various modules from `TorchGPE` and general python libraries and define physical constants:

```python
import os
import sys
sys.path.insert(1, "/home/kkirchner/sp_qg/gpe-main")

from gpe.bec2D import Gas
from gpe.bec2D.callbacks import CavityMonitor, Animation, DensityMonitor
from gpe.bec2D.potentials import Contact, DispersiveCavity, Trap
from gpe.utils import parse_config
from gpe.utils.potentials import offset_periodic_sweep, offset_periodic_cos,
    offset_periodic_triangle, offset_periodic_arccos, offset_cos, linear_ramp, tanh_ramp

import numpy as np
import torch
import matplotlib.pyplot as plt
from matplotlib import ticker
from scipy.constants import hbar
from tqdm.auto import tqdm
from datetime import datetime
from pathlib import Path
import pandas as pd
from scipy.constants import physical_constants

OMEGA_R = 23659.55675860723
A_BOHR =  physical_constants["Bohr␣radius"][0]
```

The main module `gpe.bec2D.Gas` intializes the discretized grid and wavefunction as well as implementing the imaginary and real time propagation. The `gpe.bec2D.potentials` class includes the potentials of the GPE described in section 2.2 and `gpe.utils.potentials` encompasses helper functions such as the movement and ramping function of the transverse drive. Data storing of the cavity and the density of the wave function over time are found in `gpe.bec2D.callbacks` as well as an animation of the cloud evolution in the cavity. We pre-define the recoil frequency given by $\omega_r = \hbar k_p^2/(2m)$, where the pump wave-vector is defined over the frequency of the D2-line of Rb-87 and the atomic detuning $\Delta_a = \omega_L - \omega_{D2}$ as

$$k_p = \frac{2\pi\times}{\lambda} = \frac{2\pi\times}{c(\omega_{D2} + \Delta_a)}.$$

We start the main function of the script by defining path variables for the working directory and data directory as well as stamping the time for identifying runs, initializing random seeds and parsing the configuration file.

```python
if __name__ == "__main__":
    dir_path = "/scratch3/kkirchner/domains_change_waist/"
    save_folder = "tutorial"
    plot_save_folder = os.path.join(dir_path, save_folder)

    config = parse_config("configuration.yaml")

    time = datetime.now().strftime("%d%m_%H-%M")
    np.random.seed(config["random_seed"])
    torch.manual_seed(config["random_seed"])
```

The configuration file a YAML file used to bundle inputs for various modules, and is read-in via the `parse_config` function into a standard python dictionary. The configuration file, `configuration.yaml`, for this tutorial is given by,

```yaml
random_seed: 10032023
gas:
  element: 87Rb
  N_particles: 2e5
  N_grid: !eval [2**11]
  grid_size: 16e-5
  adimensionalization_length: 780e-9
boundaries:
  cavity_detuning: -10.5e6
  lattice_depth: 4000
potentials:
  contact:
    a_s: 100
    a_orth: 5.39e-7
  trap:
    omegax: 10
    omegay: 100
  cavity:
    atomic_detuning: -76.6e9
    cavity_decay: 800e3
    cavity_coupling: 1.95e6
    cavity_angle: 0
    pump_angle: !eval [pi/2]
    waist: 780e-9
initial_wavefunction:
  gaussian_sigma: 1e-6
propagation:
  imaginary_time:
    time_step: !eval [-1e-6j]
    N_iterations: 20000
    leave_progress_bar: False
  real_time:
    final_time: 1e-3
    leave_progress_bar: False
```

It is important to note that all frequencies stated in `configuration.yaml` are multiplied in the `TorchGPE` module by 2π to convert to angular frequencies. All length scales are adimensionalized by the `adimensionalization_length`.

Next, we define the contact interaction, the harmonic trapping potential and initialize the Bose-gas with a Gaussian wave-function. The aforementioned system of a gas with contact interactions in a harmonic trap is propagated in imaginary time to reach the ground state, a Bose-Einstein condensate. We determine the ground state to be reached if the condensate profile sufficiently matches the Thomas-Fermi profile. In preparation for the scanning range of motion, we determine the grid indices along the main-axes that mark the edge of Thomas-Fermi profile and the diameters. We visually check the state of condensation by plotting the Thomas-Fermi profile and the density of the condensate. An example image can be seen in figure 4.1b.

```
    contact = Contact(**config["potentials"]["contact"])
    trap = Trap(**config["potentials"]["trap"])

    bec = Gas(**config["gas"], float_dtype=torch.float32, complex_dtype=torch.complex64)
    bec.psi = torch.exp(-(bec.X**2 + bec.Y**2)/(2*(config["initial_wavefunction"]["
        gaussian_sigma"] / bec.adim_length)**2))

    bec.ground_state(potentials=[trap, contact], callbacks=[], **config["propagation"]["
        imaginary_time"])

    tf_ind_x, span, tf_ind_y, tf_diam_y = get_TF_diameter()
    plot_density(bec, plot_save_folder, f"{time}_gs", tf_ind_x, tf_ind_y)

    w_scan = 60          # [recoil frequency]
    ramp_up_time = 5e-4 # [s]
```

Next, we declare variables for the starting and stopping times of the TP lattice depth ramping function as well as the scanning frequency of the transverse drive. We initialise the cavity potential with the aforementioned functions as well as static parameters from the configuration file, such atomic detuning, dissipation rate, atom-light coupling, angle of the transverse pump and it's waist size.

```
    detuning = config["boundaries"]["cavity_detuning"]
    lattice_depth = config["boundaries"]["lattice_depth"]
    ramp = linear_ramp(0, ramp_up_start, lattice_depth, ramp_up_time)
    offset_scan_x = offset_periodic_sweep(span, w_scan * OMEGA_R, 0)
    cavity = DispersiveCavity(lattice_depth=ramp, cavity_detuning=detuning, waist_offset_x=
        offset_scan_x, **config["potentials"]["cavity"])
```

Afterward, we initialise the monitor objects that store the cavity data, density cuts over time and the animation class which bundles various parameters over the course of the evolution and afterwards assembles the data in a `gif` format. We then specify the time-step and propagate the total system, described by the equation 2.21, via split-step Fourier propagation. In a final step, we save the density of the final step and the cavity data of each time-step to a `csv` file for further analysis and plot the density from a top-down view as well as a density cut in a given range along a specified axis for quick visual confirmation of self-organisation.

```python
    animation = Animation(output_file=os.path.join(plot_save_folder, f"{time}" + ".mp4"),
        cores=10, save_every=save_every_var, phase = False, density_cut=True, cavities=[
        cavity])
    cavity_monitor = CavityMonitor(cavity, save_every=1)
    density_monitor = DensityMonitor(output_file=os.path.join(plot_save_folder, f"{time}" + "
        .png"), y_cut_index = len(bec.y)//2, save_every=1, border_ind = tf_ind_x)

    delta_t = 2*np.pi / (w_scan*OMEGA_R) / span / 2
    bec.propagate(potentials=[trap, contact, cavity], callbacks=[cavity_monitor,
        density_monitor, animation], time_step=delta_t, final_time=config["propagation"]["
        real_time"]["final_time"])

    plot_density(bec, plot_save_folder, f"{time}_scan", tf_ind_x, tf_ind_y)
    save_density(bec, plot_save_folder, f"{time}_monitor", border_ind_x=tf_ind_x,
        border_ind_y = tf_ind_y)
    save_cavity(cavity_monitor, plot_save_folder, f"{time}_monitor")
```

3.5. Simulation parameters

The goal of this section is to motivate the choice of parameters of the tutorial and of the results obtained. We will explain the choice of parameters for each of the separate modules of the system, in order of: Grid parameters, cavity detuning and pump power, potentials, initial wave function and propagation parameters.

The atoms simulated are ^{87}Rb as this is the atom isotope used in the laboratory of the IMPACT team. The number of atoms simulated, `N_particles` $= 2 \times 10^5$, are a reasonable choice for the current experimental setup. The grid size is chosen to be larger than the Thomas-Fermi radii, which are determined by the trapping frequencies, of the BEC and there is sufficient space to the border to mitigate finite-size effects. The grid size is closely tied to the choice of `N_grid`, since it effects the resolution of the density patterns we want to observe. The discretization value was chosen such that there are approximately 10 points between the λ_p spacing of two density wave peaks. The adimensionalization length, $l = 780\,\text{nm}$, is chosen to be close to $\lambda_p \approx \lambda_c \approx 780.4\,\text{nm}$.

The s-wave scattering length a_s is chosen to be 100 a_B, which is the experimental value of Rb-87 [7]. We choose a trapping frequency in the z-direction (out-of-plane in the simulation) of $2\pi \times 400\,\text{Hz}$, which results in a harmonic oscillator length $a_z = \sqrt{\hbar/(m\omega_z)} \approx 0.539\,\mu\text{m}$. The in-plane trapping frequencies were chosen in an experimentally feasible regime, in an attempt to make the cloud greatly spread out along the cavity axis, while being in a regime $\omega_x, \omega_y \ll \omega_z$. We thus chose only a small trapping frequency in x, $\omega_x = 100\,\text{Hz}$, and an order of magnitude larger trapping frequency in y, $\omega_y = 100\,\text{Hz}$, which further squeezes the cloud.

We choose a cavity detuning of $\Delta_c = -2\pi \times 0.5\,\text{MHz}$, as this is slightly lower than the maximum dispersive-shift $Ng_0^2/\Delta_a \approx -2\pi \times \cdot 10\,\text{MHz}$ bounding an unstable self-organisation phase from below ([2]). The maximum depth of the pump lattice is chosen after multiple trials until self-organisation was achieved. We define self-organisation if the maximum modulus of $|\alpha(t)|$ is larger than one. The critical pumping strength depends on the dissipation rate, cavity detuning and waist size to cloud length ratio. The atomic detuning $\Delta_a = -2\pi \times 76.6\,\text{GHz}$ is given by the laser frequency and the D2-line of ^{87}Rb. The atom-light coupling $g_0 = 2\pi \times 1.95\,\text{MHz}$ and is the first cavity's Rabi-vacuum coupling [5]. For the decay rate of photons in the cavity, we choose to match the value of cavity 2 of the IMPACT setup, of $\kappa \approx 2\pi \times \cdot 800\,\text{kHz} \approx 212.5\ \omega_r$, which leaves a broader window for scanning below the dissipation rate. As mentioned before, the cavity was chosen to lie along the x-direction and the transverse pump was positioned orthogonally, i.e. along the y-direction. For individual simulations described in the following section, specific parameters will be motivated and specified.

The cavity monitor saves the cavity's parameters at every snapshot to allow for a greater resolution of dynamics. However as the time step is decreased or the length of the simulation is increased, we recommend increasing the `save_every` parameter to avoid extremely large file sizes. The density monitor saves a cut along a specified axis at a given position through the cloud after a given number of time steps. This reduces the amount of data stored in a temporary folder while still allowing to observe the formation of domains in a single image. During a run of 2 ms and a time step covering $\lambda/2$ in distance, roughly 150000 time steps are performed. The assembly of all the density cuts towards a single image is inefficient as it is currently implemented. For this reason we increase the number of time steps, after which the density is saved, to 9 or 113, greatly reducing the image assembly time. We did not notice a visual loss of resolution in the image, when changing the density monitor's `save_every` parameter for the aforementioned values, given the number of time steps performed at $\omega_{scan} = 60\ \omega_r$. For longer simulations we recommend to further increase this parameter. Another possible option to speed up the image creation, is to explore if the assembly of an image can be distributed to multiple threads.

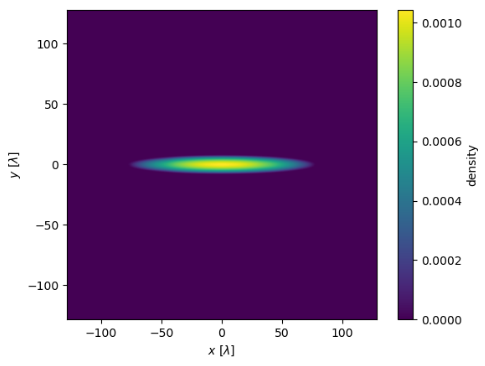
As described in the tutorial, we first find the ground state of the Bose gas in the trap until it assumes the Thomas-Fermi profile using imaginary time evolution. For the chosen trapping frequencies, sufficient condensation was achieved after 20000 steps at an imaginary time step of $-1\mathrm{i}\,\mu\mathrm{s}$.

4 Results

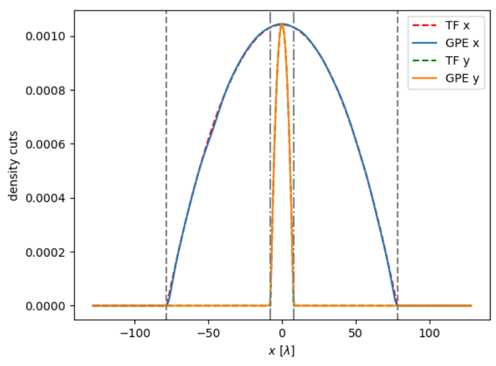
In this chapter we present the results of select simulations showing the formation of domains. First, we show the result of imaginary-time evolution, which condenses the cloud of atoms into a Bose-einstein condensate. Next, we analyze the behaviour of the light-field and formation of domains with a smaller time step. Afterwards, we test the conditions of domain formation, postulated in equation (2.22). Finally, we vary physical parameters such as the ramping speed, and increasing the waist size.

4.1. Imaginary time-evolution

We start the simulation by condensing the Bose-gas in the trap into the ground state via imaginary-time evolution. Figure 4.1a shows the density of the wave-function in the simulation grid, assuming a cigar-shape. By looking at cuts through the middle of the cloud along the x-axis and y-axis, we can see that the GPE wave function matches the Thomas-Fermi profile with slight deviation at the edges of the cloud due to the healing length.



(a)



(b)

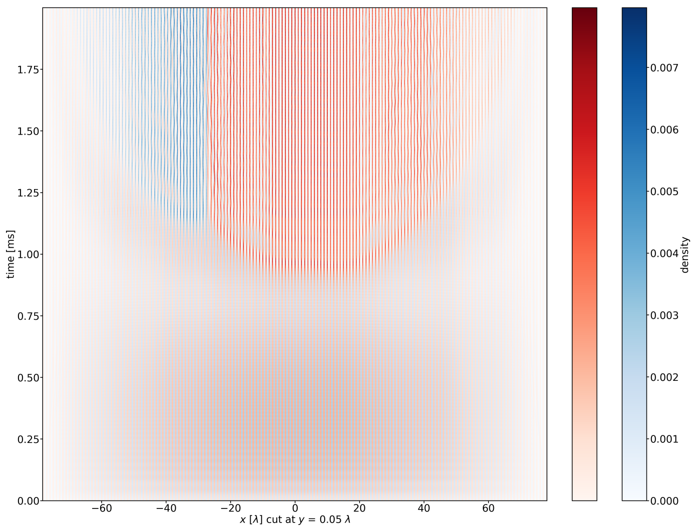
Figure 4.1: (a) Shows the density of the condensed wave function. (b) Shows density cuts along the x-axis (blue) and y-axis (orange). The dashed colored lines show the Thomas-Fermi profile along the same respective cut (red along x, green along y). The determined length of the Thomas-Fermi diameter along the major axis is $L_{TF} = 156.45\ \lambda$.

4.2. Domain formation

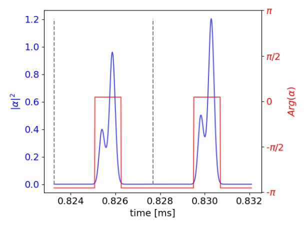
We start this section by describing and analyzing a simulation, leading to domain formation. We then go on to vary the time step, scanning frequency and waist size of the transverse pump.

We ramp the transverse pump power linearly, with a final value of 3000 E_r and a ramping time of $t_{ramp} = 10/\omega_r \approx 0.42\,\text{ms}$. While in the 1D scenario [3] the power is ramped smoothly via a smooth tanh function, whereas here we ramp linearly. The transverse lattice was swept over the cloud periodically from left to right via the sawtooth function with a scanning frequency of 60 ω_r. The diameter of the

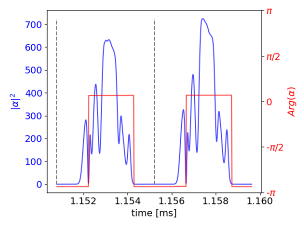
Thomas-Fermi profile along x in figure 4.1b is 156.45 λ. We set the range for the scanning function to be 156 λ, i.e. we sweep over the full cloud. For the real time propagation we chose a time-step that moves the beam by half a wave-length every step. This stroboscopic movement was chosen to reduce simulation time, while still having a resolution of the cavity-field that is able to resolve a domain wall spaced by $\lambda/2$. The waist of the transverse pump was $w = \sqrt{2} \cdot 780\,\text{nm} \approx 1.1\,\mu\text{m}$. This specific waist size was chosen, such that the standing wave created has a waist size of λ. We propagate the system for 2 ms, which roughly equals 452 periods of the beam scanning over the cloud at a frequency of 60 ω_r. This specific scanning frequency, is well above the recoil frequency while still being beneath the cavity dissipation rate, which according to the results of [3] is well suited to establish domains, given we use a similar cavity dissipation rate. Higher scanning frequencies would lead to shorter periods, which would increase the number of time steps performed. The evolution of the density cut through the middle of the cloud along x is made visible in figure 4.2a.



(a)



(b)



(c)

Figure 4.2: (a) The evolution of the density cut through the middle of the cloud is plotted over time. Red highlights the density of even lattice sites, whereas blue highlights the odd lattice sites. The simulation parameters are: $\omega_{scan} = 60\ \omega_r$ with waist $w = 1.1\,\mu\text{m} = \sqrt{2}\lambda$, linear-ramp up to 3000 E_r in $10/\omega_r \approx 0.42\,\text{s}$ at a cavity detuning of $\Delta_c = -10.5\,\text{Mhz}$. (b) The photon number (blue) and the phase (red) of the cavity light field plotted over 2 periods of scanning over the cloud at the onset of self-organisation. The vertical dashed lines (grey) mark the beginning of a period, i.e. where the center of the waist of the transverse pump is at the left edge of the cloud determined by the Thomas-Fermi radius. The phase is manually set to the minimal value if the photon number is smaller than 0.1. Although two distinct peaks appear, these are not the two domains. This can be verified in the light's phase, which remains the same. (c) The photon number and phase of the light-field plotted over 2 scanning periods at a later time. A second domain with opposite parity has emerged in the left tail of the cloud, as visible in the phase. The phase is manually set to the minimal value if the photon number is smaller than 0.1

In figure 4.2a we can observe an onset of self-organisation, defined here by the maximum photon number per period to be greater than one, at 0.83 ms, which starts in the middle of the cloud due to the higher number of atoms participating in the overlap-integral of the interference term in V_{lat}. Interestingly, this is after maximum depth of the transverse pump has already been achieved. We plot the cavity light-field and it's phase at the onset in figure 4.2b. The photon number significantly decreases as the transverse pump crosses the domain wall, see figure 4.2c. This is expected as the domain wall is spaced $\lambda/2$, a distance at which dipole radiation destructively interferes. Looking at the real-space density of the bose gas, we can clearly observe the emergence of a domain wall at roughly $-27.75\ \lambda$ in figure 4.3b. To the left of the domain wall, the cloud organises on odd lattice sites (spaced λ apart), whereas to the right of the domain wall the density peaks on even lattice sites. Looking at the density cut along x at $y = 0.55\ \lambda$, we can see the opposite symmetry breaking due to the formation of a checkerboard lattice throughout the distinct domains, visible in figure 4.3c.

We examine the stability of the self-organised phase by performing the same simulation for 10 ms. In the density cut through the middle we can observe a stable domain wall after the onset of self-organisation. We also look at the average photon number exiting the cavity per scanning period in figure 4.4. Here

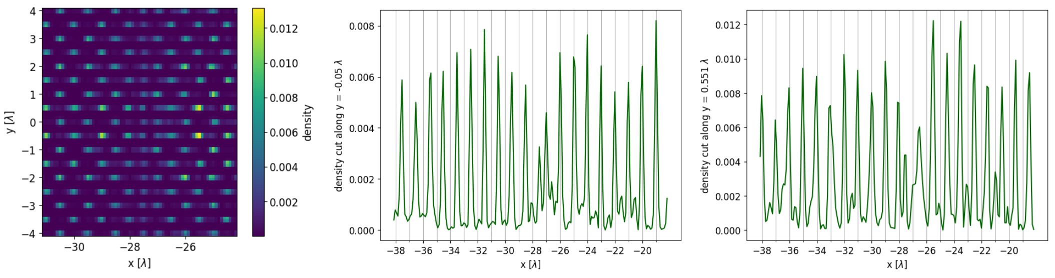


Figure 4.3: (a) The density of the cloud in the x-y plane is plotted after 2 ms. Two distinct domains forming a checkerboard lattice are visible with a domain wall formed at $-27.75\ \lambda$. The simulation parameters were $\omega_{scan} = 60\ \omega_r$, $w = \sqrt{2}\lambda$, $V_{TP} = 3000\ E_r$, $t_{ramp} = 5\,\text{ms}$, $\Delta_c = -2\pi \times 10.5\,\text{MHz}$ and $\Delta t = T\lambda/(2L_{TF})$, $L_{TF} = 156\ \lambda$. (b) The density of the Bose-gas cut along the middle of the cloud against x. (c) The density of the Bose-gas cut along $y = 0.55\ \lambda$ of the cloud against x. Here symmetry breaking of the domains is opposite to the cut along $y = 0.05\ \lambda$ in (a) as expected to form a checkerboard lattice.

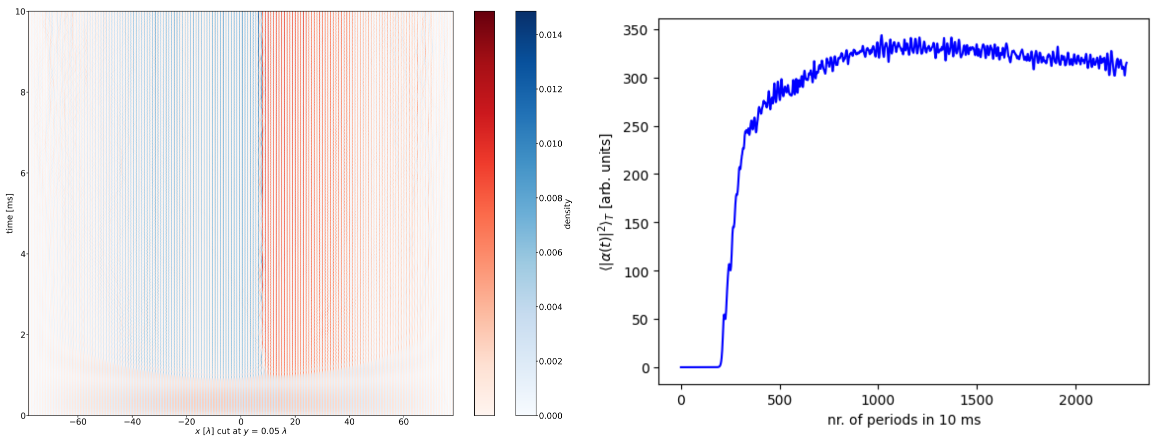
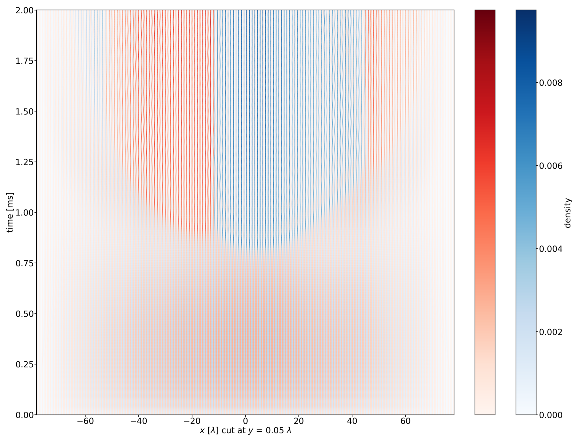


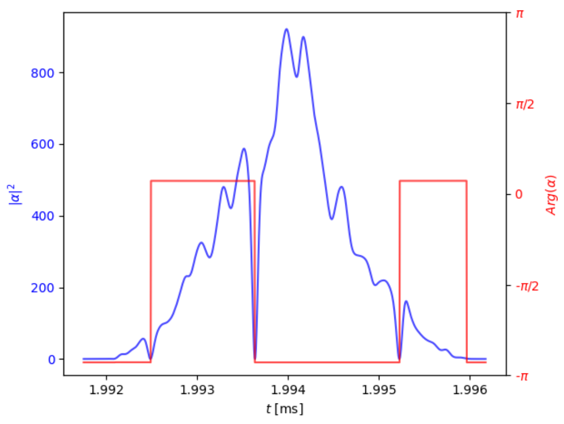
Figure 4.4: (a) Two clear domains can be seen in the density cut through the cloud after an initial symmetry breaking. The simulations parameters were $\omega_{scan} = 60\ \omega_r$, $w = \sqrt{2}\lambda$, $V_{TP} = 3000\ E_r$, $t_{ramp} = 10/\omega_r$, $\Delta_c = -2\pi \times 10.5\,\text{MHz}$ and $\Delta t = T\lambda/(10L_{TF})$, $L_{TF} = 156\ \lambda$. (b) The averaged photon number over a single scanning period, $\langle|\alpha(t)|^2\rangle_T = \sum_{i=0}^{\lfloor T/dt \rfloor} |\alpha(i \cdot \Delta t)|^2$, is plotted for each period in the course of 10 ms. After the onset of the phase transition, the average photon number per period starkly rises in combination with the formation of the density-wave seen in figure (a). After roughly 5 ms the average photon number per period decreases.

the average photon number increases drastically as the cloud self-organizes and rounds out until roughly 5 ms. Thereafter, the mean photon number slightly decreases. This hints at the fact that atoms are being dispersed from the strongly localised checkerboard lattice configuration.

4.2.1. Decreasing the time step



(a)

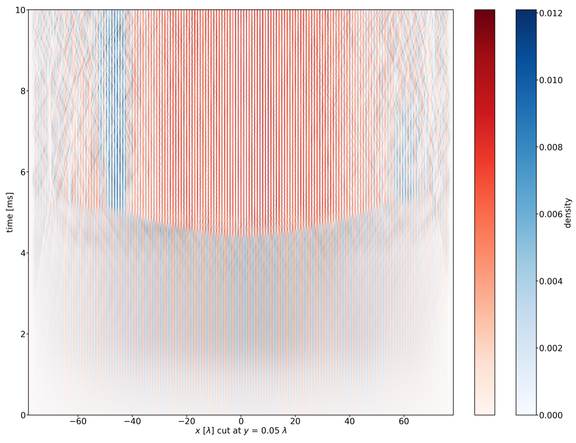


(b)

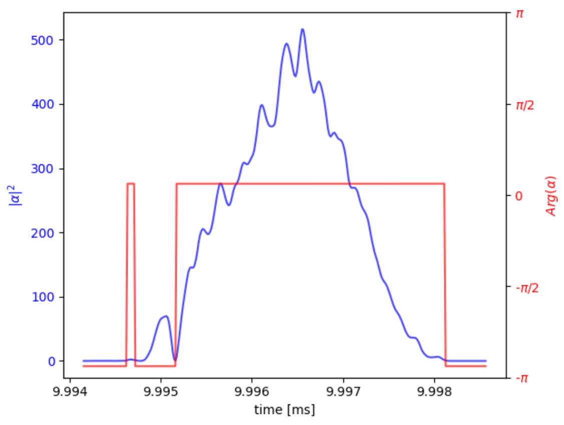
Figure 4.5: (a) The evolution of the density cut for 2 ms for $\omega_{scan} = 60\ \omega_r$, $w = \sqrt{2}\lambda$, $V_{TP} = 3000\ E_r$, $t_{ramp} = 10/\omega_r$, $\Delta_c = -2\pi \times 10.5\,\text{MHz}$ and $\Delta t = T\lambda/(10L_{TF})$, $L_{TF} = 156\ \lambda$ (b) The photon number (blue) and phase (red) of the light-field for the last full period. Five distinct domains are visible. The phase was set to the minimal value for $|\alpha|^2 < 1$ to remove fluctuations at the edge of the scanning region.

We run the same simulation as above, however with a 5 times smaller time step, i.e. in one Δt the beam moves $\lambda/10$ in distance. The result differs from the above simulation, see figure 4.5 and compare with 4.2. Self-organisation sets in earlier, i.e. $|\alpha_{max}|^2 \geq 1$ at $t = 0.728\,\text{ms}$. Secondly, over the course of the simulation we can see the emergence of up to 5 distinct domains, see figure 4.5, compared to only 2 domains to the former simulation 4.2. As we lower the time step we expect the simulations to converge towards the true ground state of the system. However as long as domains form for the larger time step, we can conclude that domain formation is possible for a specific parameter set. For further simulations, we took the largest sensible time step, i.e. moving the beam by half the wavelength per timestep, which reduced the simulation time for the grid size of 2^{11} to 1 hour.

4.2.2. Slow ramp



(a)

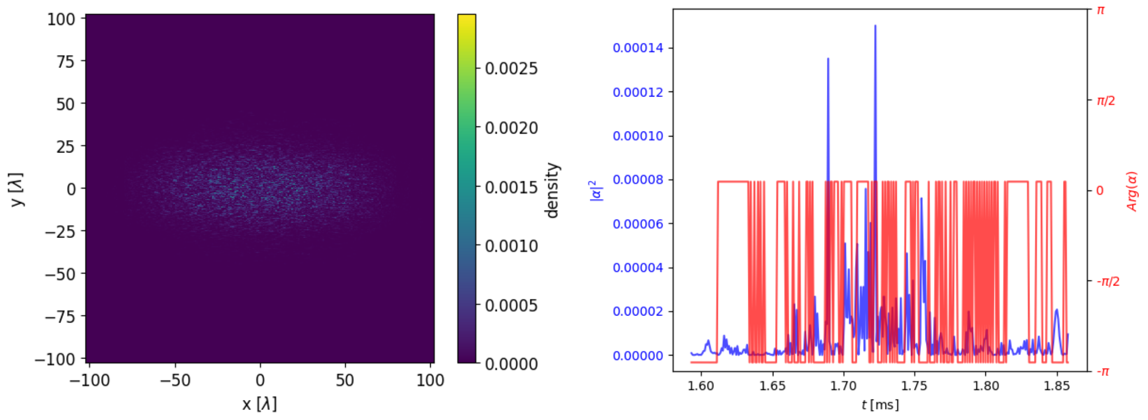


(b)

Figure 4.6: (a) The evolution of the density cut for 2 ms for $\omega_{scan} = 60\ \omega_r$, $w = \sqrt{2}\lambda$, $V_{TP} = 3000\ E_r$, $t_{ramp} = 5\,\text{ms}$, $\Delta_c = -2\pi \times 10.5\,\text{MHz}$ and $\Delta t = T\lambda/(10L_{TF})$, $L_{TF} = 156\ \lambda$. (b) The photon number (blue) and phase (red) of the light-field for the last full period is plotted. The phase was set to the minimal value for $|\alpha|^2 < 1$ to remove fluctuations at the edge of the scanning region. Three phase jumps are visible over the whole period.

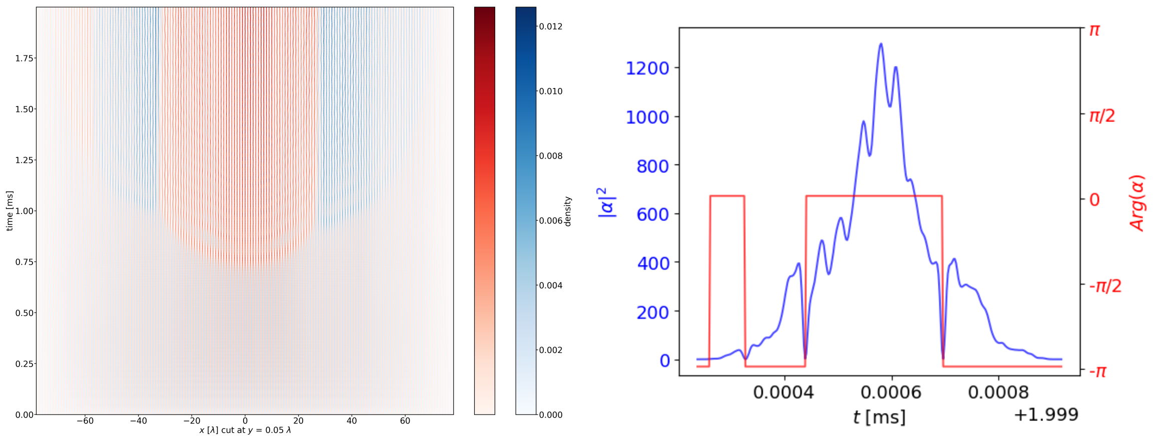
In the before mentioned simulations, we ramp the pump power within 0.42 ms. This is an order of magnitude faster than usually performed in an experiment, where the ramping time is 5 ms. [5]. We run a simulation with parameters as described in figure 4.2, but increase the ramping time to 5 ms. In the density cut along x through the middle of the cloud we can identify a large domain sitting on integer sites, and a small domain sitting on half-integer sites at around $-47.5\ \lambda$. The edges of the cloud do not self-organise even though there density modulations occur. In the phase of the light-field, we can see that there is also a further jump in phase to the left of the already identified domains, although the photon numbers are very low in comparison. While we do see the establishment of at least two domains, the edges of the cloud do not self-organise. We conclude the quench of the lattice depth works better.

4.2.3. Varying the scanning frequency



(a) **(b)**

Figure 4.7: (a) Density of the real-space wave function after 2 ms for simulation parameters $\omega_{scan} = \omega_r$, $w = \lambda$, $V_{TP} = 3000\ E_r$, $t_{ramp} = 10/\omega_r$, $\Delta_c = -2\pi \times 10.5\,\text{MHz}$ and $\Delta t = T\lambda/(2L_{TF})$, $L_{TF} = 156\ \lambda$. No self-organisation is achieved. (b) The photon number (blue) and phase (red) of the light-field over the last full period.



(a) **(b)**

Figure 4.8: (a) The evolution of the density cut for 2 ms for $\omega_{scan} = 390\ \omega_r$, $w = \sqrt{2}\lambda$, $V_{TP} = 3000\ E_r$, $t_{ramp} = 10/\omega_r$, $\Delta_c = -2\pi \times 10.5\,\text{MHz}$ and $\Delta t = T\lambda/(10L_{TF})$, $L_{TF} = 156\ \lambda$ (b) The photon number (blue) and phase (red) of the light-field for the last full period. Five distinct domains are visible. The phase was set to the minimal value for $|\alpha|^2 < 0.01$ to remove fluctuations at the edge of the scanning region.

Next, we test the condition (2.22a) of domain formation by varying the scanning frequency, while keeping the waist size at 1.1 µm and ramping to 3000 E_r in $10/\omega_r$. When scanning over the cloud at a frequency of $\omega_{scan} = \omega_r$, the particles occupy higher density states and do not self organise, which can be seen in figure 4.7 in both the scattered density of atoms and chaotic light-field. Due to the pump moving on the time-scale of the atoms, the beam and atoms This validates the lower limit of the scanning frequency, although the exact boundary has not been determined.

Conversely, we probe the other limit of condition (2.22b) by scanning at a higher frequency than the dissipation rate of the cavity. To be precise, we choose $\omega_{scan} = 10 \cdot \kappa_1 \approx 390\ \omega_r \ll \Delta_c$, with $\kappa_1 = 2\pi \times \cdot 147\,\text{MHz}$ being the dissipation of cavity 1 in the IMPACT group's setup [5]. In fact, we observe the formation of 4 domains, seen in figure 4.8. The faster scanning seems to facilitate the formation of more domains, as the critical point is crossed in a shorter time frame as the pump sweeps across the cloud. This validates the boundaries set for the scanning frequencies and demonstrating a large window of scanning frequencies in which domains can form.

4.2.4. Increasing the waist size

We finally test the influence of changing the waist size of the transverse drive. So far we've only looked at examples with $w = \sqrt{2} \cdot \lambda$ which was chosen create a standing wave mode with a waist of λ such that the beam width of the standing wave is λ. While this tight focussing is possible in an experiment, demonstrating that broader waists can also lead to domain formation provides a window in which experimental uncertainties of the waist size are tolerated.

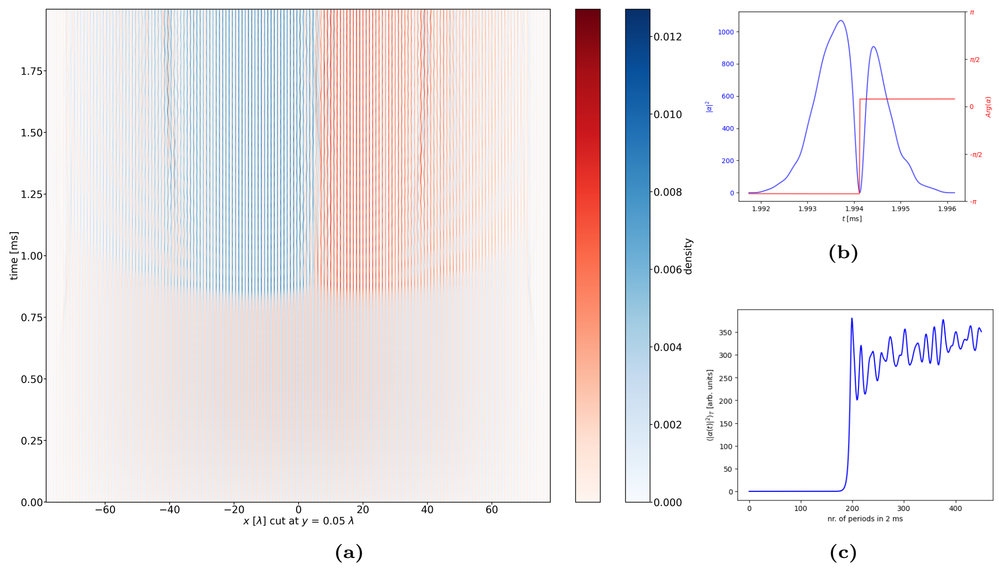


Figure 4.9: (a) The evolution of the density cut through the middle of the cloud is plotted over time. The simulation parameters are: $\omega_{scan} = 60\ \omega_r$ with waist $w = 5\lambda$, linear-ramp up to 300 E_r in 0.5 ms at a detuning of $\Delta_c = -10.5\,\text{Mhz}$. (b) The photon number (blue) and the phase (red) of the cavity light field plotted over the last full period of scanning over the cloud. The phase is manually set to the minimal value if the photon number is smaller than 0.1. (c) The averaged photon number over a single scanning period, $\langle|\alpha(t)|^2\rangle_T = \sum_{i=0}^{\lfloor T/dt \rfloor} |\alpha(i \cdot \Delta t)|^2$, is plotted for each period in the course of 2 ms. We can see a stark rise in average photon number after the phase transition. The average photon numbers then oscillate.

We increase the waist size to $w = 5\ \lambda$, i.e. 3.9 µm, while keeping the other parameters the same as for the simulation shown in figure 4.2. The larger waist illuminates more atoms simultaneously, facilitating self-organisation at lesser lattice depths, i.e. we ramp only to 300 E_r. The critical power needed to observe self-organisation for specific waist and cloud sizes has not yet been determined. In figure 4.9 we can see the formation of domains. The average photon number per period increases sharply after the onset of the phase transition and then oscillates for the remainder of the simulation, which shows that for the duration of the simulation the self-organised phase is stable and there is no drastic heating present.

We want to conclude this chapter by stating that all simulations were performed at a specific cloud size given by the trap geometry of $\omega_x = 10\,\text{Hz}$, $\omega_y = 100\,\text{Hz}$ and $\omega_z = 400\,\text{Hz}$. For the case of $w = 5\ \lambda$ we increased the trap frequency ω_x to 20 Hz and 25 Hz, and to $\omega_x = 20$ Hz for a waist of $w = \sqrt{2}\ \lambda$. In neither case did we observe domain formation. Since we kept the ramp power the same, we saw that increasing ω_x lead to an earlier onset of the phase transition. The self-organisation was also unstable in the sense that atoms were dispersing. A finer tuning of the pump power is needed to ensure we ramp equivalently deep into the self-organised phase for different trap geometries.

5 Conclusion

The formation of domains in the self-organisation phase of a Bose-Einstein condensate in a far detuned dissipative cavity was demonstrated for a 1D system with periodic boundary conditions in the theoretical proposal [3]. This behaviour occurs due to the range of interactions being limited by the narrow waist of the transverse pump, much smaller than the diameter of the cloud, and the quick scanning of the transverse pump across the cloud, faster than the timescale of atomic motion. The effectively engineered short-range interactions allow for domain formation. This phenomenon has not been observed in the self-organisation phase due to the usual global range interactions. The simulations shown in this work demonstrate domain formation for a trapped Bose-Einstein condensate in 2 dimensions. To this end, we expanded upon the existing `TorchGPE` python package by implementing the scanning transverse pump and the correct equations of motion in the limit of a narrow-waisted transverse drive. As far as we know, this is the first time this behavior has been seen in a 2D simulation.

On top of observing domains in the self-organisation phase transition with the scanning pump protocol, we varied various parameters, to test their influence towards the finding of domains. The time step that propagates the pump by half a wavelength provides the minimal resolution of a possible domain wall and we have seen various examples of domain formation. We observed that a faster ramping time of 0.42 ms lead to a more stable self-organisation throughout the whole cloud, due to the faster crossing of the critical lattice depth while the transverse pump scans over the cloud. We validated that scanning must occur at frequencies much larger than the recoil frequency and there is a large window for which domains can occur, at least up to 390 $\omega_r = 10 \cdot \kappa_1$, the highest scanning frequency shown here. Lastly, domains can also form for larger waists of the transverse pump beam of 5 λ for the specific frequencies chosen.

For the waist of $w = 5\lambda$ and $w = \sqrt{2}\ \lambda$ we increased the trapping frequency in x direction to $\omega_x = 20$ Hz and $\omega_x = 25$ Hz and respectively to $\omega_x = 20$ Hz. We did not observe domain formation as well as stable self-organisation. In order to ramp equivalently deep into the phase transition, we will need to ramp to lower powers as we increase the trapping frequencies. We should further explore different trap geometries to identify a specific waist size needed for a given cloud size to form domains.

Following up on this work we should simulate the experimentally relevant case of an imbalanced pump, the imbalance being between the in-going and reflected beam shape. It will be difficult to align the transverse pump such that it remains perfectly overlapping while scanning across the cloud. Instead, if the reflected beam has a larger waist than the incoming beam, the region of overlap will exhibit a standing wave, whereas the outer regions will exhibit a running wave. To test this pumping scheme, the imbalance pump would need to be implemented in the `TorchGPE` package, which only provides an imbalanced pump in amplitude.

References

[1] M. H. Anderson et al. “Observation of Bose-Einstein Condensation in a Dilute Atomic Vapor”. In: *Science* 269.5221 (1995), pp. 198–201. DOI: 10.1126/science.269.5221.198. eprint: https://www.science.org/doi/pdf/10.1126/science.269.5221.198. URL: https://www.science.org/doi/abs/10.1126/science.269.5221.198.

[2] Kristian Baumann et al. “Dicke quantum phase transition with a superfluid gas in an optical cavity”. In: *Nature* 464.7293 (Apr. 2010), pp. 1301–1306. ISSN: 1476-4687. DOI: 10.1038/nature09009. URL: https://doi.org/10.1038/nature09009.

[3] Mariano Bonifacio, Francesco Piazza, and Tobias Donner. *Laser-painted cavity-mediated interactions in a quantum gas.* 2024. arXiv: 2405.07492 [cond-mat.quant-gas]. URL: https://arxiv.org/abs/2405.07492.

[4] Ferdinand Brennecke et al. “Cavity QED with a Bose–Einstein condensate”. In: *Nature* 450.7167 (Nov. 2007), pp. 268–271. ISSN: 1476-4687. DOI: 10.1038/nature06120. URL: https://doi.org/10.1038/nature06120.

[5] D. Dreon, A. Baumgärtner, and X. et al. Li. “Self-oscillating pump in a topological dissipative atom–cavity system”. In: *Nature* 608 (7923 Aug. 2022), p. 494. DOI: 10.1038/s41586-022-04970-0. URL: https://doi.org/10.1038/s41586-022-04970-0.

[6] Lorenzo Fioroni et al. *A Python GPU-accelerated solver for the Gross-Pitaevskii equation and applications to many-body cavity QED.* 2024. arXiv: 2404.14401 [physics.comp-ph]. URL: https://arxiv.org/abs/2404.14401.

[7] K. M. Mertes et al. “Nonequilibrium Dynamics and Superfluid Ring Excitations in Binary Bose-Einstein Condensates”. In: *Phys. Rev. Lett.* 99 (19 Nov. 2007), p. 190402. DOI: 10.1103/PhysRevLett.99.190402. URL: https://link.aps.org/doi/10.1103/PhysRevLett.99.190402.

[8] Farokh Mivehvar et al. “Cavity QED with quantum gases: new paradigms in many-body physics”. In: *Advances in Physics* 70.1 (Jan. 2021), pp. 1–153. ISSN: 1460-6976. DOI: 10.1080/00018732.2021.1969727. URL: http://dx.doi.org/10.1080/00018732.2021.1969727.

[9] L. Salasnich, A. Parola, and L. Reatto. “Effective wave equations for the dynamics of cigar-shaped and disk-shaped Bose condensates”. In: *Phys. Rev. A* 65 (4 Apr. 2002), p. 043614. DOI: 10.1103/PhysRevA.65.043614. URL: https://link.aps.org/doi/10.1103/PhysRevA.65.043614.

[10] R Zamora-Zamora et al. “Validity of Gross–Pitaevskii solutions of harmonically confined BEC gases in reduced dimensions”. In: *Journal of Physics Communications* 3.8 (Aug. 2019), p. 085003. DOI: 10.1088/2399-6528/ab360f. URL: https://dx.doi.org/10.1088/2399-6528/ab360f.

A

Source Code: Dispersive Cavity with Scanning Pump

```python
from __future__ import annotations
from typing import Union, Callable

import scipy.constants as spconsts
import numpy as np
import torch

from ..utils.potentials import LinearPotential, NonLinearPotential,
    any_time_dependent_variable, time_dependent_variable


class DispersiveCavity(NonLinearPotential):
    """Transversally pumped dispersive cavity potential with scanning pump

    Args:
        lattice_depth (Union[float, Callable]): The lattice depth in units of the recoil
            energy. It can be set to be either a constant or a function of time.
        atomic_detuning (float): The atomic frequency detuning with respect to the pump.
        cavity_detuning (Union[float, Callable]): The cavity's frequency detuning with
            respect to the pump. It can be set to be either a constant or a function of time.
        cavity_decay (float): The cavity's decay rate.
        cavity_coupling (float): The coupling constant between the gas and the cavity.
        cavity_angle (float, optional): The angle in the 2D plane of the cavity. Defaults to
            :math:`0`
        imbalance_factor (float, optional): The imbalance factor for the pumping beam.
            Defaults to 1.
        waist (float, optional): the waist of the gaussian beam. Defaults to infinity
        waist_offset_x: (float, optional): the offset of the waist of the gaussian beam in x-
            axis in units of adim. length. Defaults to 0
        waist_offset_y: (float, optional): the offset of the waist of the gaussian beam in y-
            axis in units of adim. length. Defaults to 0
    """

    def __init__(self, lattice_depth: Union[float, Callable], atomic_detuning: float,
        cavity_detuning: Union[float, Callable], cavity_decay: float, cavity_coupling: float,
         cavity_angle: float = 0, pump_angle: float = np.pi/3, imbalance_factor: float = 1,
        waist: float = np.inf, waist_offset_x: Union[float, Callable] = 0, waist_offset_y:
        Union[float, Callable] = 0):

        super().__init__()

        self.lattice_depth = lattice_depth
        self.atomic_detuning = atomic_detuning
        self.cavity_detuning = cavity_detuning
        self.cavity_decay = cavity_decay
        self.cavity_coupling = cavity_coupling
        self.cavity_angle = cavity_angle
        self.pump_angle = pump_angle
        self.gamma = imbalance_factor
        self.waist = waist
        self.adim_offset_x = waist_offset_x
        self.adim_offset_y = waist_offset_y

    def on_propagation_begin(self):
        self.is_time_dependent = any_time_dependent_variable(
            self.cavity_detuning, self.lattice_depth, self.adim_offset_x, self.adim_offset_y)

        self._cavity_detuning = time_dependent_variable(self.cavity_detuning)
        self._lattice_depth = time_dependent_variable(self.lattice_depth)
        self._adim_offset_x = time_dependent_variable(self.adim_offset_x)
        self._adim_offset_y = time_dependent_variable(self.adim_offset_y)

        self.g0 = 2*np.pi*self.cavity_coupling
        self._atomic_detuning = 2*np.pi*self.atomic_detuning
        self.kappa = 2*np.pi*self.cavity_decay
        self.freq_d2 = self.gas.d2_pulse
        self.lambda_pump = 2*np.pi*spconsts.c / (self.freq_d2+self._atomic_detuning)
        self.adim_lambda_pump = self.lambda_pump/self.gas.adim_length
        self.k_pump = 2*np.pi/self.lambda_pump
        self.adim_k_pump = 2*np.pi/self.adim_lambda_pump
        self.Er = 0.5 * (spconsts.hbar*self.k_pump)**2 / self.gas.mass
        self.U0 = self.g0**2 / self._atomic_detuning
        self._adim_waist = self.waist / self.gas.adim_length

        self.R_cavity = self.gas.X * np.cos(self.cavity_angle) + self.gas.Y * np.sin(self.cavity_angle)
        self.COS2 = torch.cos(self.adim_k_pump*self.R_cavity)**2
        self.c1 = self.gas.N_particles*self.gas.dx*self.gas.dy
        self.c3 = self.c1*self.U0
        self.eta1_prefactor = np.sqrt(self.Er*np.abs(self._atomic_detuning)/spconsts.hbar) * self.g0/self._atomic_detuning*0.5*(self.gamma+1/self.gamma)
        self.eta2_prefactor = np.sqrt(self.Er*np.abs(self._atomic_detuning)/spconsts.hbar) * self.g0/self._atomic_detuning*0.5*(self.gamma-1/self.gamma)
        self._cavity_lattice = self.COS2 * self.U0 / self.gas.adim_pulse

    def get_alpha(self, psi: torch.tensor, time: float = None):
        """Return the intracavity field

        Args:
            psi (torch.tensor): The wave function of the gas
            time (float, optional): The time at which to compute the intracavity field. Defaults to None.

        Returns:
            float: The intracavity field :math:`\\alpha`
        """
        order1 = (torch.abs(psi)**2*self.COS*torch.sqrt(self._gaussian_profile)).sum()
        order2 = (torch.abs(psi)**2*self.SIN*torch.sqrt(self._gaussian_profile)).sum()
        bunching = (torch.abs(psi)**2*self.COS2).sum()
        self._cavity_detuning_tilde = 2*np.pi * self._cavity_detuning(time)-self.c3*bunching
        self.c6 = self.c2-self.c3*bunching

        self.eta1 = np.sqrt(self._lattice_depth(time))*self.eta1_prefactor
        self.eta2 = np.sqrt(self._lattice_depth(time))*self.eta2_prefactor

        alpha = self.c1*(self.eta1*order1+1j*self.eta2*order2)/self.c6

        return alpha

    def potential_function(self, X: torch.tensor, Y: torch.tensor, psi: torch.tensor, time: float = None):
        R_pump = (self.gas.X - self._adim_offset_x(time)) * np.cos(self.pump_angle) + (self.gas.Y - self._adim_offset_y(time)) * np.sin(self.pump_angle)
        R_pump_orth = - (self.gas.X - self._adim_offset_x(time)) * np.sin(self.pump_angle) + (self.gas.Y - self._adim_offset_y(time)) * np.cos(self.pump_angle)
        self.COS = torch.cos(self.adim_k_pump*R_pump) * torch.cos(self.adim_k_pump*self.R_cavity)
        self.SIN = torch.sin(self.adim_k_pump*R_pump) * torch.cos(self.adim_k_pump*self.R_cavity)
        self._gaussian_profile = 1/(1+(self.adim_lambda_pump*R_pump/(np.pi*self._adim_waist
            **2))**2)* \
            torch.exp(-2 * R_pump_orth**2/(self._adim_waist**2 + (self.adim_lambda_pump*
                R_pump/(np.pi*self._adim_waist))**2))
        self._pump_lattice = np.sign(self._atomic_detuning) * self.Er * torch.cos(self.
            adim_k_pump*R_pump)**2 / (spconsts.hbar * self.gas.adim_pulse) * self.
            _gaussian_profile

        self.c2 = 2*np.pi*self._cavity_detuning(time)+1j*self.kappa
        alpha = self.get_alpha(psi, time)

        self.pump_lattice = self._lattice_depth(time) * self._pump_lattice
        self.cavity_lattice = torch.abs(alpha)**2 * self._cavity_lattice
        self.interaction = 2 * torch.sqrt(self._gaussian_profile) / self.gas.adim_pulse * (
            self.eta1*self.COS*torch.real(alpha) + self.eta2*self.SIN*torch.imag(alpha))

        return self.pump_lattice + self.cavity_lattice + self.interaction
```

B

Source code: Density monitor

```python
import torch
import numpy as np
import numpy.ma as ma

import fcntl
import json
import warnings
import matplotlib.pyplot as plt
import tempfile
from os import path
import ffmpeg
from shutil import rmtree
from abc import ABCMeta
from matplotlib import ticker
from .potentials import DispersiveCavity
from ..utils.plotting import pi_tick_formatter
from matplotlib.gridspec import GridSpec
from ..utils import prompt_yes_no, enumerate_chunk
import atexit
import signal
import psutil
from ..utils.callbacks import Callback

class DensityMonitor(Callback):
    """Callback monitoring the cut through the middle of the density in y.

    During the simulation, the values of cavity detuning, pump strength and cavity field are stored. Once the simulation is finished, the saved parameters are accessible via the :py:attr:`gpe.bec2D.callbacks.CavityMonitor.alpha`, :py:attr:`gpe.bec2D.callbacks.CavityMonitor.pump` and :py:attr:`gpe.bec2D.callbacks.CavityMonitor.cavity_detuning` tensors.

    Args:

        dispersive_cavity (DispersiveCavity): The cavity to be monitored.
        save_every (int): Optional. The number of epochs after which the parameters should be saved. Defaults to 1.
    """

    def __init__(self, output_file, y_cut_index, save_every=1, border_ind=np.array([0, -1])) -> None:
        super().__init__()
        self.save_every = save_every
        self.output_file = output_file
        self.output_folder = path.dirname(output_file)
        self.y_cut_index = y_cut_index
        self.start_ind = border_ind[0]
        self.stop_ind = border_ind[1]
        #: list(float): A list of the times at which the parameters were saved. It is a list of lists, where each inner list contains the times for a single propagation. At the end of the simulation, it is converted to a PyTorch tensor.
        self.times = []

        if not path.exists(self.output_folder):
            raise Exception("The␣output␣folder␣does␣not␣exist")
        if path.exists(self.output_file):
            if not prompt_yes_no("The␣specified␣file␣already␣exists.␣Are␣you␣sure␣you␣want␣to␣overwrite␣it?␣Y/n", True):
                raise Exception("The␣output␣file␣already␣exists")


    def _register_run(self):
        with open(path.join(path.expanduser("~"), ".density_cuts_cleanup.json"), 'a+') as file:
            fcntl.flock(file, fcntl.LOCK_EX)  # Acquire exclusive lock
            file.seek(0)
            try:
                existing_data = json.load(file)
            except (json.JSONDecodeError, EOFError):
                existing_data = {}
                warnings.warn("The␣executions␣register␣file␣does␣not␣exist␣and␣it␣will␣be␣created.␣If,␣before␣now,␣you␣have␣run␣the␣animation␣callback␣and␣the␣process␣has␣been␣interrupted,␣there␣might␣be␣some␣leftover␣temporary␣folders.␣Please,␣check␣the␣folder␣/tmp/␣and␣delete␣eventual␣temporary␣folders␣manually.")

            existing_data.setdefault(str(psutil.Process().pid), []).extend([self.temp_dir])
            file.truncate(0)
            file.seek(0)
            json.dump(existing_data, file)
            file.flush()
            fcntl.flock(file, fcntl.LOCK_UN)  # Release lock

    def _deregister_run(self, pid=None, folder=None):
        if pid is None:
            pid = str(psutil.Process().pid)

        with open(path.join(path.expanduser("~"), ".density_cuts_cleanup.json"), 'a+') as file:
            fcntl.flock(file, fcntl.LOCK_EX)  # Acquire exclusive lock
            file.seek(0)
            try:
                existing_data = json.load(file)
            except (json.JSONDecodeError, EOFError):
                existing_data = {}
            if pid in existing_data:
                if folder is not None:
                    self.clear_dir(folder)
                    existing_data[pid] = [f for f in existing_data[pid] if f != folder]
                    if len(existing_data[pid]) == 0:
                        del existing_data[pid]
                else:
                    for f in existing_data[pid]:
                        self.clear_dir(f)
                    del existing_data[pid]
            file.truncate(0)
            file.seek(0)
            json.dump(existing_data, file)
            file.flush()
            fcntl.flock(file, fcntl.LOCK_UN)  # Release lock

    def _clean_leftovers(self):
        try:
            with open(path.join(path.expanduser("~"), ".density_cuts_cleanup.json"), "r") as f:
                runs = json.load(f)
                for key, value in runs.items():
                    if not psutil.pid_exists(int(key)):
                        self._deregister_run(key)
        except (json.JSONDecodeError, FileNotFoundError):
            return

    def clear_dir(self, dir):
        if path.exists(dir):
            rmtree(dir)

    def on_propagation_begin(self):
        self.tensor_index = 0

        self.temp_dir = tempfile.mkdtemp()

        # Register the temporary folder for deletion at exit and on SIGINT. Check if the folder exists before deleting it to avoid errors

        self._clean_leftovers()
        self._register_run()

        atexit.register(self.clear_dir, self.temp_dir)
        signal.signal(signal.SIGINT, lambda sig, frame: (self.clear_dir(self.temp_dir), self._deregister_run(), signal.default_int_handler(signal.SIGINT, None)) )
        signal.signal(signal.SIGTERM, lambda sig, frame: (self.clear_dir(self.temp_dir), self._deregister_run(), signal.default_int_handler(signal.SIGTERM, None)) )


    def on_epoch_end(self, epoch):
        if epoch % self.save_every != 0:
            return

        time = epoch*self.propagation_params["time_step"]
        self.times.append(time*1000)
        torch.save(self.gas.density[self.y_cut_index, self.start_ind:self.stop_ind].to(torch.float16), path.join(self.temp_dir, f"density_cut_{self.tensor_index}.torch"))

        self.tensor_index += 1

    def plot_density_trace(self):
        mask_even = torch.zeros_like(self.gas.x.cpu()[self.start_ind:self.stop_ind])
        mask_odd = torch.zeros_like(self.gas.x.cpu()[self.start_ind:self.stop_ind])

        self.density_trace = torch.load(path.join(self.temp_dir, f"density_cut_0.torch"), map_location="cpu")
        for i in range(1, self.tensor_index):
            density_cut = torch.load(path.join(self.temp_dir, f"density_cut_{i}.torch"), map_location="cpu")
            self.density_trace = torch.vstack((self.density_trace, density_cut))

        x_values = self.gas.x.cpu()[self.start_ind:self.stop_ind].numpy()
        mask_odd = np.array([0.25 <= np.mod(x, 1) < 0.75 for x in x_values])

        # Apply masks to the density data
        density_odd = ma.masked_array(self.density_trace.cpu().numpy(), mask=np.tile(mask_odd, (len(self.times), 1)))
        density_even = ma.masked_array(self.density_trace.cpu().numpy(), mask=~np.tile(mask_odd, (len(self.times), 1)))

        plt.rcParams.update({'font.size': 42})

        fig, ax = plt.subplots(figsize=(4*10, 3*10), layout="constrained")

        # Add even density mesh
        mesh_even = ax.pcolormesh(x_values, self.times.cpu().numpy(), density_even, shading='nearest', cmap="Blues")
        colorbar_even = plt.colorbar(mesh_even, ax=ax, label="density")

        # Add odd density mesh
        mesh_odd = ax.pcolormesh(x_values, self.times.cpu().numpy(), density_odd, shading='nearest', cmap="Reds")
        colorbar_odd = plt.colorbar(mesh_odd, ax=ax)
        colorbar_odd.set_ticks([])  # Remove ticks from the second colorbar

        # Set axis labels
        ax.set_xlabel(r"$x$␣[$\lambda$]␣cut␣at␣$y$␣=␣" + str(np.round(self.gas.y[self.y_cut_index].item(), 3)) + r"␣$\lambda$")
        ax.set_ylabel(r"time␣[ms]")

        ax.tick_params(width = 3, length = 10)
        colorbar_even.ax.tick_params(width = 3, length = 10)
        colorbar_even.outline.set_linewidth(3)
        colorbar_odd.outline.set_linewidth(3)
        for axis in ['top','bottom','left','right']:
            ax.spines[axis].set_linewidth(3)

        fig.savefig(self.output_file)

        plt.rcParams.update({'font.size': 10})

        print(f"Density␣trace␣saved␣to␣{self.output_file}")


    def on_propagation_end(self):
        self.times = torch.tensor(self.times)
        self.plot_density_trace()
        self.clear_dir(self.temp_dir)
        self._deregister_run(folder=self.temp_dir)
```

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every written paper or thesis authored during the course of studies. In consultation with the supervisor, one of the following three options must be selected:

I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used no generative artificial intelligence technologies[1].

I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used and cited generative artificial intelligence technologies[2].

I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used generative artificial intelligence technologies[3]. In consultation with the supervisor, I did not cite them.

Title of paper or thesis:

Authored by:
If the work was compiled in a group, the names of all authors are required.

Last name(s):	**First name(s):**

With my signature I confirm the following:
- I have adhered to the rules set out in the Citation Guide.
- I have documented all methods, data and processes truthfully and fully.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for originality.

Place, date	**Signature(s)**

If the work was compiled in a group, the names of all authors are required. Through their signatures they vouch jointly for the entire content of the written work.

[1] E.g. ChatGPT, DALL E 2, Google Bard
[2] E.g. ChatGPT, DALL E 2, Google Bard
[3] E.g. ChatGPT, DALL E 2, Google Bard