Czech Technical University in Prague

Faculty of Nuclear Sciences and Physical Engineering

Department of Physics

Research Assignment

Simulations and tests of higher-spin ultra-light dark matter

Simulace a testování modelů ultra-lehké temné hmoty s vyšším spinem

Author: Martin Šmíd

Supervisor: Federico Urban, Ph.D.

Consultants: Eric Munive-Villa, Ph.D.; Jessica López-Sánchez, Ph.D.; Ing. Marek Matas, Ph.D.

Academic Year: 2024/2025

FAKULTA JADERNÁ A FYZIKÁLNĚ INŽENÝRSKÁ
KATEDRA FYZIKY

VÝZKUMNÝ ÚKOL

Akademický rok: 2024/2025

Student:	**Bc. Martin Šmíd**
Studijní program:	Jaderná a částicová fyzika
Vedoucí úkolu:	Federico Urban, Ph.D.
Konzultant:	Ing, Marek Matas, Ph.D; Jessica López-Sánchez, Ph.D; Eric Munive-Villa, Ph.D
Název úkolu (česky):	Simulace a testování modelů ultra-lehké temné hmoty s vyšším spinem
(anglicky):	Simulations and tests of higher-spin ultra-light dark matter
Jazyk VÚ:	Angličtina

Pokyny pro vypracování:
1. Studium literatury o problému temné hmoty a zaměřit se na modely ultralehké temné hmoty.
2. Naučit se používat a upravovat výpočetní nástroje, které simulují kosmologické struktury ultralehké temné hmoty.
3. Implementovat nové simulace ultralehké temné hmoty.
4. Získat užitečné informace ze simulací a definovat pozorovací testy.

1. Reading the literature on the dark matter problem and focus on ultra-light dark matter models.
2. Learning how to use and modify the computational tools that simulate ultra-light dark matter cosmological structures.
3. Implement new ultra-light dark matter simulations.
4. Extract useful information from the simulations and define observational tests.

Literatura:
[1] E. G. M. Ferreira, ``Ultra-light dark matter,'' Astron. Astrophys. Rev. 29 (2021) no.1, 7
[2] J. C. Niemeyer, ``Small-scale structure of fuzzy and axion-like dark matter,'' Prog. Part. Nucl. Phys. 113 (2020), 103787
[3] S. Dodelson and F. Schmidt, ``Modern Cosmology,'' Elsevier (2020)
[4] M. Vogelsberger, F. Marinacci, P. Torrey and E. Puchwein, ``Cosmological Simulations of Galaxy Formation,'' Nature Rev. Phys. 2 (2020) no.1, 42

Datum zadání: 21. 10. 2024

Datum odevzdání: 15. 08. 2025

Výzkumný úkol odevzdá student v elektronické podobě nahráním souboru ve formátu pdf do příslušné události v systému Indico. Vedoucí úkolu přiloží své písemné vyjádření k práci studenta s doporučením pro hodnocení.

.....
garant programu

.....
vedoucí katedry

Title: **Simulations and tests of higher-spin ultra-light dark matter**

Author: Martin Šmíd

Study Programme: Nuclear and Particle Physics

Supervisor: Federico Urban, Ph.D.FZÚ AV ČR

Abstract: This work presents the development of a simulation framework for studying the dynamics of ultralight axion-like dark matter, utilizing the Fourier split-step method to solve the Schrödinger–Poisson system of equations. The core objective of this work was to implement an efficient and modular numerical solver using the split-step Fourier method in Python, which would be capable of simulating axion-like dark matter with arbitrary integer spin. The framework allows users to initialize systems of self-gravitating solitons, evolve them in time under various potentials (including self-interactions), and extract observables such as energy, radial profiles, and eventually the matter power spectrum. The theoretical background motivating this approach is provided in Chapter 1, while the specifics of the numerical implementation are described in Chapter 2.

The correctness of the code is verified through a series of tests summarized in Chapter 3, including comparisons with analytical solutions and conservation checks. Although the results shown in Chapter 4 are currently limited to preliminary density profiles and energy diagnostics, they demonstrate the intended functionality and stability of the framework. Future developments are discussed in Chapter 5. These include tracking of virialization, study of self-interaction effects, and adding cosmological expansion.

Key words: Axion-like particles, ultralight dark matter, Schrödinger–Poisson system, split-step Fourier method, numerical simulation, wave dark matter.

Název práce: **Simulations and tests of higher-spin ultra-light dark matter**

Autor: Martin Šmíd

Studijní program: Nuclear and Particle Physics

Vedoucí práce: Federico Urban, Ph.D.FZÚ AV ČR

Abstrakt: Tato práce představuje vývoj simulačního frameworku pro studium dynamiky ultralehké axion-like temné hmoty. K řešení Schrödingerovy-Poissonovy soustavy rovnic využívá Fourierovu split-step metodu. Hlavním cílem této práce bylo vytvořit efektivní a modulární numerický nástroj v Pythonu založený na split-step Fourierově metodě, který by dokázal simulovat axion-like temnou hmotu s libovolným celočíselným spinem. Framework umožňuje uživatelům inicializovat systémy solitonů podléhající gravitaci, sledovat jejich časový vývoj v různých potenciálech (včetně self-interakcí) a extrahovat pozorovatelné veličiny jako energii nebo radiální profily.

Teoretické motivace pro tento přístup je uvedena v kapitole s názvem "Theoretical Background", podrobnosti vysvětlující implementaci numerických nástrojů jsou popsány v kapitole s názvem "Code Implementation". Správnost fungování kódu je ověřena prostřednictvím série testů, které zahrnují porovnání výsledků simulace s analytickými řešeními a kontroly zachování veličin. Ačkoli dosavadní výsledky jsou omezeny jen na předběžnou analýzu profilů hustoty a energetickou diagnostiku, demonstrují funkcionalitu a stabilitu kódu. V kapitole s názvem "Future Plans" jsou diskutovány plánované vylepšení pro simulační framework.

Klíčová slova: axion-like částice, ultra-lehká temná hmota, Schrödinger–Poissonův systém, Fourierova split-step metoda, numerické simulace, vlnová temná hmota.

Acknowledgement

With this, I would like to thank Erick Munive Villa, Ph.D., who spent a substantial amount of time going through the code with me and searching for bugs. Furthermore, I would like to thank Jessica López-Sánchez, Ph.D., for proofreading and consulting, and Federico Urban, Ph.D., and Ing. Marek Matas, Ph.D., for supervising this work.

Contents

Introduction

The aim of this research task was to develop a numerical simulation framework for studying ultra-light dark matter, with a particular focus on axion-like particles. This work involved learning how to apply numerical methods, specifically the split-step Fourier method, to evolve the Schrödinger–Poisson system and simulate the wave-like behaviour of dark matter at galactic scales.

The broader motivation stems from the fact that conventional cold dark matter models face challenges on small scales [1, 2], such as the core cusp problem and the missing satellites issue [3]. Ultra light dark matter models, especially those involving light scalar fields, offer an intriguing alternative by exhibiting quantum effects at astrophysical distances [4]. This work contributes to the field by implementing a modular, extensible simulation tool capable of evolving ultra-light dark matter wave functions, including support for higher spin generalizations and self-interactions, and by analysing the preliminary outputs of such simulations.

This thesis is structured as follows:

- In Chapter 1, we review the theoretical background behind dark matter and its alternatives, with an emphasis on fuzzy and axion-like dark matter. The Schrödinger–Poisson system, its derivation, physical meaning, and numerical handling using the split-step Fourier method are introduced here.
- Chapter 2 describes the structure and design of the simulation code, implemented in Python. This chapter outlines the main classes and their responsibilities, such as managing the wave functions, propagators, and system evolution.
- The accuracy and reliability of the code are tested in Chapter 3. Several benchmarks are performed, including conservation checks, harmonic oscillator evolution, and tests of motion under various potentials to verify the correctness of the simulation.
- Preliminary outputs of the simulation are presented in Chapter 4. These include density profiles and energy diagnostics of solitonic structures formed during the evolution, showing that the code reproduces expected physical behaviour and is numerically stable.
- Finally, Chapter 5 outlines possible extensions and improvements to the simulation framework. These include the inclusion of cosmological expansion, studying the matter power spectrum, incorporating angular momentum and vortex dynamics, and further investigation of self-interaction effects.

This research task serves both as an introduction to the numerical simulation of ultra-light dark matter and as documentation of the developed code, which is intended to serve as a foundation for future study of its phenomenology. The implemented framework provides a solid base upon which more specialised and complex simulations of fuzzy dark matter can be built.

Chapter 1

Theoretical Background

1.1 Dark Matter and Ultralight ALPs

Over the past few decades, scientists have gathered observational evidence that points in the direction of the existence of dark matter (DM) in our universe. The need for dark matter first became clear through studies of galactic rotation curves. When astronomers measured the rotational speeds of stars within galaxies, they noticed an unexplained trend. The speeds stayed nearly constant even at large distances from the galactic centre. This did not match predictions made based on the estimates of visible matter within the galaxy and Newtonian gravity. This suggested that there must exist an additional, different, invisible form of matter which contributes to the galaxy's gravity. This invisible form of matter was descriptively named dark matter.

1.2 Introduction to Dark Matter

The existence of DM was first proposed by Fritz Zwicky in his work on galaxy clusters in 1933. While studying the Coma cluster, Zwicky used spectroscopic methods to measure the velocities within the individual galaxies in this dense system of approximately 800 nebulae. He observed large differences in velocities, with individual galaxies showing recession speeds ranging from about 5100 to 8500 km/s, differences of at least 1500 to 2000 km/s from the cluster's mean velocity. To understand whether this system could be gravitationally bound, Zwicky used the virial theorem to compare the observed kinetic energies to the gravitational potential energy that would arise from visible matter alone. His calculations revealed a discrepancy showing that the cluster would need to be at least 400 times more massive than the luminous matter suggested it is to maintain gravitational equilibrium and prevent the system from dispersing. This led Zwicky to conclude the following: dark matter is present in much greater amounts than luminous matter. His analysis provided the first quantitative evidence for the existence of a non-luminous mass dominating the dynamics of cosmic structures decades before galactic rotation curves would reinforce this mystery [5].

As observational techniques and instruments have improved, the need for dark matter has appeared across a wide range of cosmic scales. Discrepancies between the amount of visible matter and the gravitational effects we observe show up not just within galaxies and smaller structures, but also in galaxy clusters and even across the large-scale structure of the universe. Measurements of the cosmic microwave background [6], along with luminosity distances from Type Ia supernovae, provide precise estimates of the total matter energy content of the universe.

When these observations are combined with predictions from Big Bang Nucleosynthesis

about the primordial abundances of light elements, a clear picture emerges: most of the matter in the universe must be non-baryonic, and it barely interacts with light. Because dark matter does not emit, absorb, or reflect electromagnetic radiation, it must interact with regular matter mainly through gravity. Although interactions through other forces are possible, DM´s charge would have to be incredibly small to explain the observed lack of its interactions with the surrounding matter [7, 8, 9, 10]. This also fits with the fact that we have never observed dark matter directly.

All of these points point to the existence of a non-luminous, gravitationally active component making up about 85% of the universe's matter. Without it, we cannot explain how the large-scale structure of the universe, the cosmic web of galaxies and clusters, formed the way we see it today. [11, 3].

To account for the wide array of cosmological observations, a phenomenological framework known as the ΛCDM model has been developed. This model currently represents the most successful and widely accepted description of the universe. Over the years, it has shown remarkable agreement with observational data [11], allowing precise determination of its parameters, often at the per cent or even subpercent level. One of the main advantages of the model is that, despite its success, the model remains remarkably simple, relying on just six free parameters to characterise a broad range of cosmological phenomena.

According to the ΛCDM model, the universe is spatially flat and seeded by nearly scale-invariant perturbations made up of both baryonic matter and dark matter [12, 13]. We can study these early perturbations through the cosmic microwave background, using the observable imprints left by baryons. Dark matter is expected to have experienced the same initial fluctuations. The perturbations likely began in the dark matter regions, with both components later collapsing together under gravity.

In the ΛCDM universe, baryonic matter makes up only about 5% of the total energy density. The rest is dominated by two dark components: dark matter and dark energy. Dark energy, which drives the accelerated expansion of the universe, is modelled as a cosmological constant Λ and contributes roughly 70% of the total energy content and dark matter, which on large scales accounts for about 25% of the total energy density. More precise values of these cosmological parameters can be found in [14].

Dark matter is described as a non-baryonic component that, in the hydrodynamical limit, behaves like a perfect fluid with negligible pressure. This means that the ratio $w = P/\rho$ of the pressure of DM and its density ρ is effectively zero. This leads to a very low sound speed defined as $c_s^2 = \frac{\delta P}{\delta \rho}$, meaning that pressure perturbations propagate extremely slowly through the dark matter fluid.

It is expected that this component would interact weakly with the observable baryonic matter. Many dark matter candidates have no interactions with the Standard Model of particles beyond gravitational effects [3].

Within the theory laid by the ΛCDM model, the Cold Dark Matter (CDM) component describes the energy density responsible for the formation of cosmic structures via gravitational collapse in an expanding universe. It is treated as a cold, massive, and collisionless perfect fluid, where "cold" means it was already non-relativistic when large structures began to form. This hydrodynamical description of CDM has proven very effective in reproducing a wide range of cosmological data, from the cosmic microwave background and large-scale structure to galaxy clusters and the general statistical properties of galaxies [15].

Despite this success in determining the hydrodynamical properties of DM on large scales, the fundamental microphysical nature of dark matter remains unknown. Worse still, the fact that dark matter can be described hydrodynamically on large scales reveals nothing about its "particle identity" [16]. This lack of constraint opens the door to a wide variety of theoretical

models, many of which aim to reproduce the large-scale behaviour of ΛCDM while proposing radically different particle candidates or underlying mechanisms [3].

To test and constrain the wide range of viable dark matter models, the matter power spectrum is often used [17]. This observable, shown in Figure 1.1, shows the amplitude of matter density fluctuations across different length scales, showing the dependence of $\Delta^2(k)$, which represents variations in the matter power spectrum dependent on k which is the comoving wave number corresponding to the inverse of the size of the object.

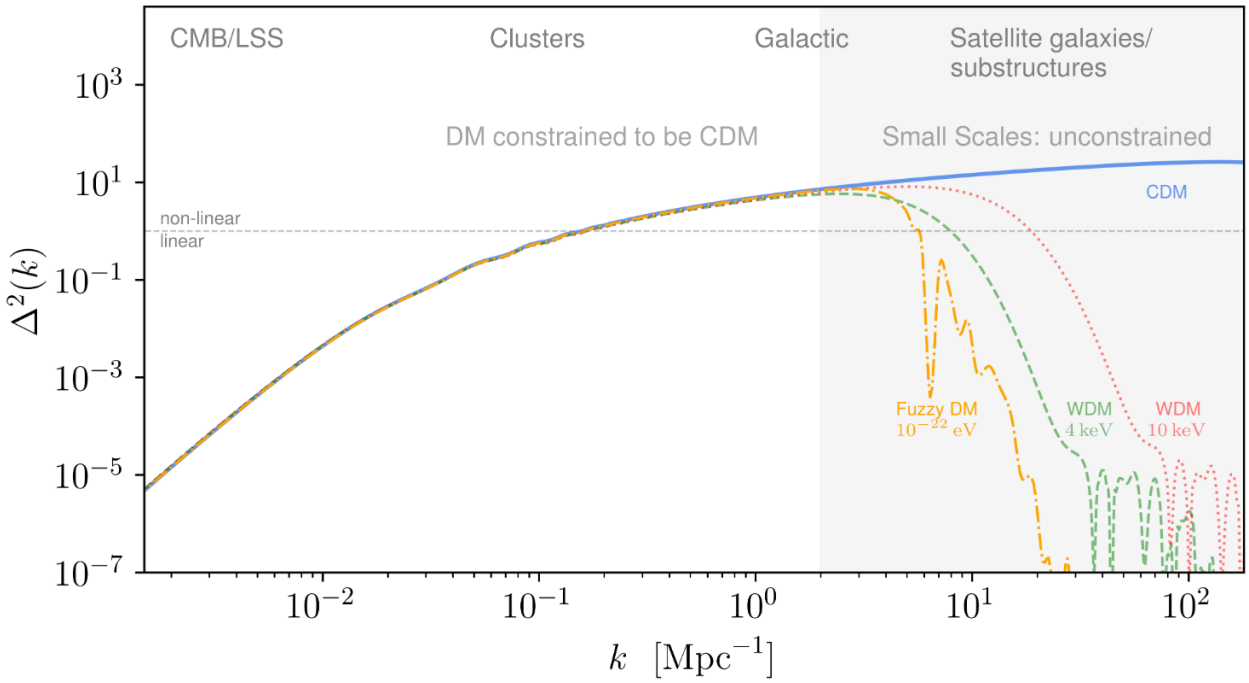

Figure 1.1: Dimensionless matter power spectrum $\Delta^2(k)$ as a function of comoving wave number k, illustrating predictions from various dark matter models and observational constraints. The blue full line shows the power spectrum for a ΛCDM universe. Red and green dashed lines show the power spectra for Warm Dark Matter with masses 10 keV and 4 keV. The orange dash-dotted line shows the values for the model of Fuzzy Dark Matter (FDM) with mass 10^{-22} eV. The grey dotted horizontal line indicates the transition between the linear and non-linear regimes of structure formation. Top labels correspond to structures in the universe depending on the inverse of their size k. Figure was taken from [3].

In Figure 1.1, small k corresponds to large cosmological distances (e.g. superclusters), while large k means smaller structures such as galaxies and subhalos.

At large scales ($k \lesssim 1\,\mathrm{Mpc}^{-1}$), the different dark-matter models converge, since these scales are well constrained by observations of the cosmic microwave background, baryon acoustic oscillations (BAO), and large-scale galaxy clustering. In this regime, gravity dominates and the specific particle properties of dark matter are mostly irrelevant [17].

At smaller scales ($k \gtrsim 2\,\mathrm{Mpc}^{-1}$), however, significant differences emerge, which depend on the different model being taken into account:

- **Cold Dark Matter (CDM)** predicts a nearly scale-invariant or slowly increasing power spectrum, allowing for the formation of structure across a wide range of scales, including very small dark matter halos [3].

- **Warm Dark Matter (WDM)** exhibits a characteristic suppression of power at high k due to free streaming: thermal motion in the early universe smooths out small-scale density fluctuations. This leads to a cutoff point in the formation of structures below a certain mass scale [3].
- **Fuzzy Dark Matter (FDM)** introduces a suppression due to quantum pressure, a consequence of the ultralight nature of its bosonic particles (e.g., axion-like particles). The large de Broglie wavelength prevents structure formation below a characteristic length scale [3].

This suppression at small scales is highly relevant for addressing well-known challenges of the ΛCDM model, including:

- the *missing satellites problem*, wherein CDM predicts more small-scale substructures than are observed [3];
- the *cusp-core problem*, where observed dwarf galaxy profiles show flatter cores in contrast to the cuspy profiles predicted by CDM [3];
- and the *too-big-to-fail problem*, where simulated subhalos are too massive to be devoid of stars, yet no such galaxies are observed [3].

These problems, discussed in more detail in [18, 19, 20], appeared when considering CDM CDM-only simulation. However, the incorporation of baryons into these simulations proved to solve some of these issues. The study of CDM + baryons simulations has gained popularity in the recent decade, revealing that stellar feedback and galactic outflows can significantly change the central density profiles of dark matter halos. These baryonic processes can transform the predicted cuspy cores into the observed flat cores through mechanisms such as supernova-driven gas outflows that create time-changing gravitational potentials, effectively heating the dark matter and reducing central densities [21, 22, 23].

Additionally, Figure 1.1 highlights a region on the left (low k - large scales), where models are tightly constrained by large-scale observations, and a shaded region on the right (high k - smaller scales), where current constraints are weak. This observational blind spot provides an opportunity for new models to diverge from CDM predictions without contradicting existing data, leading to the appearance of new models for DM [3, 24].

By diverging from the ΛCDM model, which predicts an overabundance of small-scale structures, the alternative models try to explain the discrepancies between observations and the ΛCDM model. Discrepancies such as a smaller number of observed dwarf galaxies and satellites, the predicted cuspy central density profiles in dark matter halos at high k values, while observations of dwarf galaxies often favour shallower "cores" [25].

1.3 Alternative Dark Matter Models

One of the key unknowns in identifying the nature of dark matter is its mass. Particle candidates for dark matter span an enormous range of mass scales, illustrated in Figure 1.2, covering more than 80 orders of magnitude. The figure presents various broad classes of dark matter models, each encompassing several specific candidates. These range from hypothetical elementary particles to composite states [26, 27], and even extend to objects on an astrophysical scale such as primordial black holes [3].

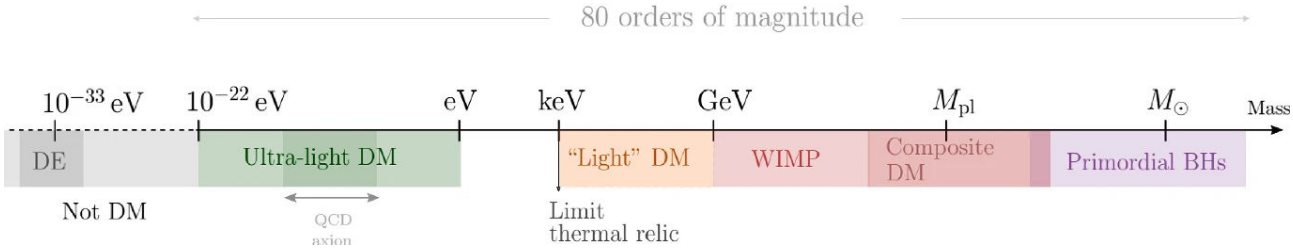

Figure 1.2: Diagram illustrating the wide range of proposed dark matter models, spanning many orders of magnitude in mass. These candidates represent fundamentally different physical phenomena, from new elementary particles to astrophysical-scale objects such as black holes. Figure adopted from [3]

1.3.1 Weakly Interacting Massive Particles

Classes of models hypothesising that DM is composed of long-existing particles are particularly compelling, especially when such candidates were to arise in extensions of the Standard Model of particle physics. Among these, one of the most studied classes is that of weakly interacting massive particles (WIMPs), a hypothetical class or single elementary particle that interacts with baryons not only gravitationally, but also via the weak nuclear force or a new force of comparable strength [28]. A key motivation for WIMPs is the so-called "WIMP miracle": If these particles were thermally produced in the early universe, then a particle with a mass near the electroweak scale and a coupling of order one would yield a relic abundance in remarkable agreement with the observed dark matter density [3].

The potential for WIMPs to be detected directly through laboratory experiments has driven substantial experimental efforts over recent decades. These efforts have significantly constrained the parameter space available for WIMPs, although no conclusive detections have yet been made. Due to the diverse range of theoretical WIMP models and their varying interactions with the standard model of particle physics, translating these experimental constraints into exclusions of specific models is not straightforward.

Among the most popular WIMP candidates are heavy neutrinos, which provide an elegant explanation for the observed low masses of Standard Model neutrinos through the seesaw mechanism. Heavy scalar particles arising from Kaluza-Klein compactification, motivated by string theory considerations, represent another well-studied class of WIMP candidates [16, 28]. Perhaps the simplest extension to the Standard Model involves the addition of an extra inert singlet scalar field, which naturally provides a stable dark matter candidate while requiring minimal modifications to the existing theoretical framework. Cosmologically, all viable WIMP models behave similarly to cold dark matter, and so their phenomenology is best probed through a combination of direct, indirect, and collider-based detection strategies. For a more comprehensive overview of WIMP models, one may want to refer to [29, 3].

1.3.2 Axions and Axion-like Particles

Another well-motivated dark matter candidate emerging from extensions of the Standard Model is the QCD axion and, from it, axion-like particles (ALPs). It was originally proposed by Pecceni and Quinn [30] as a solution to the strong CP problem in quantum chromodynamics (QCD), and later developed by Weinberg and Wilczek [31, 32]. The axion arises as a pseudo-Goldstone boson of a spontaneously broken global $U(1)$ symmetry and has extremely weak interactions with Standard Model particles. If produced non-thermally, QCD axions can act as cold dark matter, making them the focus of extensive theoretical and experimental interest [33, 3]. However, while QCD axions are naturally predicted when solving the CP problem, their

masses are typically too large for certain dark matter scenarios of interest. This limitation has motivated the study of axion-like particles, which are proposed scalars that maintain the essential shift symmetry of axions manifested through characteristic $\cos(\theta)$ potentials - but can have different mass scales and coupling strengths. These ultra-light dark matter candidates share the fundamental symmetry properties of QCD axions while offering greater flexibility in their parameters. I will focus on this class of DM candidates in later parts of this chapter [4, 3].

1.3.3 Ultralight Dark Matter

In recent years, ultralight dark matter (ULDM) models have emerged as compelling alternatives to the cold dark matter paradigm. These models, which include the QCD axion, axion-like particles, and fuzzy dark matter, preserve the large-scale successes of CDM while introducing distinctive quantum phenomena on smaller, galactic scales. In ULDM scenarios, dark matter consists of extremely light bosonic particles whose minute masses give them de Broglie wavelengths comparable to astrophysical structures, in some cases as large as dwarf galaxies [4]. This wave-like nature allows them to form coherent states that can be described in a fluid-like language via the Madelung transformation, which I will mention in more detail in Section 1.6. Such behaviour can alter the inner structure of halos and potentially address several long-standing small-scale issues in the CDM model [3]. A more detailed discussion of ULDM, including its mass range and other characteristics, is given in Section 1.4.

1.4 Ultralight Dark Matter

Ultralight dark matter (ULDM) denotes a class of models in which dark matter is composed of extremely light bosonic particles, with masses typically ranging between 10^{-25} eV and 2 eV [3]. These models were originally introduced as a possible resolution to several long-standing small-scale challenges of the ΛCDM model, but more importantly, they offer a novel and testable phenomenology in galactic environments. Due to their minute masses, ULDM particles possess macroscopic de Broglie wavelengths on astrophysical scales. The de Broglie wavelength for a non-relativistic particle is given by:

$$\lambda_{dB} = \frac{\hbar}{mv}, \tag{1.1}$$

where m_a is the particle mass and v is the velocity. For $v = 300$ km/s $= 3 \times 10^5$ m/s, which is an estimate of the speed of DM particles based on the speeds of the stars around which the DM is orbiting. Plugging in the mass we study, $m \approx 10^{-22}$ eV, into Equation 1.1, we get a wavelength of about 60 pc. This macroscopic de Broglie wavelength results in the emergence of collective quantum behaviour that is absent in conventional particle-like dark matter scenarios [24].

This wave-like behaviour leads to the formation of large-scale condensates that can alter the inner structure of dark matter halos. Theoretical considerations impose both lower and upper bounds on the mass of the ULDM particle [3]. The lower bound follows from the requirement that the de Broglie wavelength cannot exceed the size of the halo, ensuring that the condensate forms only on galactic scales and not beyond. By evaluating this condition at virialization ($z_{\rm vir} \sim 2$) for halos of mass $M \sim 10^{12} M_\odot$, one finds the following lower mass limit:

$$m \gtrsim 10^{-25} \text{ eV}. \tag{1.2}$$

The upper bound arises from requiring that the wave properties of ULDM remain relevant on galactic scales. Specifically, what is the largest de Broglie wavelength DM can have so that it forms a core inside the galaxy? This can be answered by studying the Jeans condition of a

specified model, but if the maximum mass is not of much importance, which in this work it is not, since I will mostly work with masses of order $m_a \approx 10^{-22}$ eV, one can infer an upper bound limit on ULDM by requiring that the inter-particle distance allows for macroscopic wave behaviour through the superposition of individual particle wave functions [3]. Using the halo density and velocity dispersion at virialization, this condition leads to:

$$m \lesssim 2 \text{ eV} \tag{1.3}$$

Together, these considerations constrain the ULDM particle mass to lie approximately within the interval:

$$10^{-25} \text{ eV} \lesssim m \lesssim 2 \text{ eV} \tag{1.4}$$

This mass range is just an estimate of the maximal upper and lower mass limit and may be different for each ULDM mode. However, the mass must always fall within this range, considering that ULDM is the only DM in the Universe, so that ULDM exhibits the desired non-CDM behaviour on kiloparsec scales while still forming coherent structures consistent with observed galactic dynamics [4].

The mass range for ULDM covers particles significantly lighter than those typically considered for dark matter candidates. Such light particles cannot be produced thermally in the early universe. Consequently, ULDM must be a non-thermal relic, requiring creation through non-thermal mechanisms to remain cold today and function as dark matter [4].

A key prediction of these ULDM models is that, inside gravitationally bound structures known as virialised dark-matter halos appear [34]. These regions that have reached dynamical equilibrium through gravitational collapse and relaxation can undergo a process of effective thermalisation. This leads to the formation of dense and coherent central regions often referred to as solitonic cores. Schematic sketch of how a ULDM behaves inside galaxies can be seen in Figure 1.3. In the central regions of the galaxy, a condensate core appears, while on the outskirts of the galaxy, where the ULDM is not so tightly bound by gravity, it exhibits particle-like behaviour [4]. These cores can be described within the framework of a Bose-Einstein condensate (BEC) [35], as a macroscopic quantum state in which a large number of bosons occupy the same ground-state. In certain extensions of the basic ULDM framework, the condensed phase may also exhibit properties analogous to a superfluid, characterised by collective excitations and hydrodynamic behaviour governed by the presence of long-range coherence and suppressed viscosity.

Importantly, this quantum structure affects the internal dynamics of galaxies. On cosmological scales, the evolution of density perturbations proceeds similarly to cold dark matter, with differences appearing primarily in the suppression of structure formation below a characteristic length scale set by the particle mass [1]. This is apparent on galactic scales, the wave nature of ULDM leads to new effects. The quantum pressure associated with the gradient energy of the wave function acts to prevent the formation of singular density cusps in the centres of halos. This could potentially solve the so-called cusp core problem by preventing the gravitational collapse of DM in the core of a halo. Similarly, the suppression of small-scale fluctuations in the power spectrum may explain the observed lack of low-mass satellite galaxies around the Milky Way and other galaxies [34], which is a problem of the CDM model commonly referred to as the missing satellite problem, this was mentioned in more detail in Section 1.2 [3].

This distinctive phenomenology of ULDM can be compared with observational data (e.g. [14]) across a range of astrophysical systems by, for example, studying the matter power spectrum shown on Figure 1.1. Constraints and signatures arise from the internal kinematics of dwarf spheroidal galaxies, precise measurements of galactic rotation curves, and high-resolution mapping of low-surface-brightness galaxies, an example can be found here [36]. In addition, gravitational lensing and large-scale structure surveys may reveal subtle imprints of the modified

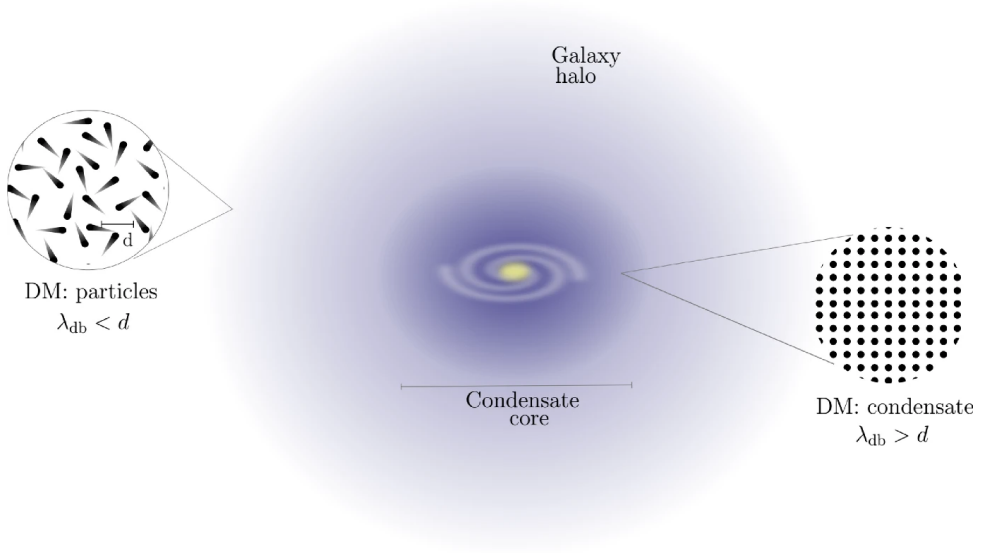

Figure 1.3: Schematic sketch of how ULDM behaves within a galaxy, forming a condensate core in the central part of a galaxy while on the outskirts exhibiting particle-like behaviour. Taken from [3].

dynamics of the ULDM. Consequently, these models provide a framework in which quantum-mechanical principles appear on astrophysical scales, opening new possibilities for testing dark matter theories through observations of small-scale structure formation and galactic dynamics [3, 24].

1.5 Classification of Ultralight Dark Matter Models

Dark matter condensation on small scales has been discussed for decades [37], and numerous models have been proposed in the literature to describe dark matter which would form coherent, wave-like condensate cores in galactic halos. These models aim to capture the quantum nature of ULDM on kiloparsec scales while remaining consistent with CDM phenomenology at larger scales [1].

The existing models can be broadly categorised into two types: those with a microscopic description and phenomenological models. Microscopic models, such as the QCD axion and axion-like particles, are grounded in fundamental scalar field theories and possess a well-defined cosmology [38]. While these theories can behave as dark matter over a wide range of parameters, their distinctive small-scale phenomenology typically arises only within a narrower mass window. On the other hand, phenomenological models provide a more flexible framework that incorporates additional interactions or modifications in the dynamics, allowing for a diverse astrophysical behaviour on small scales [35].

Despite their different origins, both classes often reduce to the same non-relativistic limit governed by the Schrödinger–Poisson system or its general form. This shared dynamical structure offers a classification based on the non-relativistic dynamics that manifest on galactic scales [2]. In what follows, I adopt the classification introduced in [3], which organises ULDM models according to the effective description they provide on small scales. Each class may encompass both microscopic and phenomenological realisations that converge to the same dynamical equations in the non-relativistic regime.

This classification also distinguishes between different condensation conditions and resulting structures: whether a Bose–Einstein condensate (BEC) forms, or whether the system behaves as a superfluid in the presence of self-interactions. The type of condensation and the nature of

the core depend sensitively on the underlying dynamics captured by the non-relativistic model.

1.5.1 Fuzzy Dark Matter

The first subclass, which also forms the basis of the simulation framework this work aims to document, is characterised by a gravitationally bound scalar field model. It is described by a Schrödinger equation coupled to a gravitational potential, which itself satisfies the Poisson equation:

$$i\frac{\partial\psi}{\partial t} = -\frac{1}{2m}\nabla^2\psi + V_{\rm grav}\psi, \tag{1.5}$$

where the gravitational potential is associated with the scalar field by the Poisson equation:

$$\nabla^2\Phi = 4\pi G m|\psi|^2. \tag{1.6}$$

Here, $m|\psi|^2$ corresponds to the mass density, so the quantity $|\psi|^2$ represents the number density of the scalar field. The Poisson equation links the average gravitational potential in a local region to the local mass density, derived from the squared amplitude of the wave function.

In this model, condensation occurs under gravitational confinement, where gravitational attraction is balanced by quantum pressure. This class of models is widely known as Fuzzy Dark Matter, a term that has become standard for gravitationally bound Bose–Einstein condensates in this context. In FDM, dark matter is modelled as a light scalar particle with mass $m \sim 10^{-22}$ eV, as originally proposed by Hu et al. [1]. This mass scale leads to macroscopic de Broglie wavelengths on kiloparsec scales, enabling FDM to address several of the small-scale challenges of the standard ΛCDM model while remaining consistent with large-scale structure observations. A mass of order 10^{-22} eV serves as a standard benchmark value [3] and will be adopted in my simulations. However, the mass remains a free parameter that requires observational constraints, as different values of m produce distinct small-scale phenomenology. Thus, the FDM model is effectively characterised by a single free parameter: the particle mass (m).

Microscopic scalar theories such as QCD axions and axion-like particles also fall within this class, as they exhibit the same effective dynamics on small scales, namely those governed by the Schrödinger–Poisson system in the non-relativistic limit. These axionic models are particularly appealing because they arise naturally as pseudo-Nambu–Goldstone bosons from spontaneously broken global symmetries, and their masses are protected by symmetry, making them radiatively stable [24]. Axions as DM will be discussed further in this work in Section 1.6.

Despite its appeal, FDM is subject to increasingly tighter and tighter constraints across a wide range of scales. Observations of the Lyman-α forest [39, 40], Milky Way satellite galaxies and stellar streams [41], as well as galaxy formation at high redshift [42] have narrowed the possible mass range of the scalar field to around 10^{-22} eV. These results form boundaries to how much FDM can deviate from the ΛCDM model without conflicting with already existing data.

1.5.2 Self-Interacting Fuzzy Dark Matter (SIFDM):

The second category is known as self-interacting fuzzy dark matter (SIFDM), though it also appears in the literature under various other names such as repulsive dark matter, scalar field dark matter, or fluid dark matter. Like FDM, these models describe dark matter as a scalar field under the influence of gravity, but with the addition of self-interactions, typically modelled as two-body contact interactions [24]. The presence of these interactions significantly changes the condensation dynamics and the internal structure of dark matter halos.

Self-interactions change the properties of the condensate, leading to new phenomenology which is not truly described by the purely gravitational case [43]. In particular, repulsive interactions can stabilise the condensate core and result in superfluid behaviour, characterised by long-range coherence. As a consequence, SIFDM models depend not only on the mass of the dark matter particle, as in FDM, but also on the strength and sign of the self-interaction coupling. The presence of self-interactions enables a wider range of density profiles and galactic core structures [43].

The dynamics of SIFDM can be captured by an extended non-linear Schrödinger equation that includes both gravitational and self-interaction potentials. A general form of the governing equation is:

$$i\frac{\partial\psi}{\partial t} = -\frac{1}{2m}\nabla^2\psi + V_{\text{grav}} + g|\psi|^2\psi + g_3|\psi|^4\psi + \dots, \tag{1.7}$$

where the coefficients g, g_3, etc., represent the strength of the self-interaction terms. The specific form and coefficients depend on the underlying microscopic model or effective theory used. These interactions open the possibility of describing dark matter as a superfluid in galactic cores, leading to phenomena such as modified rotation curves or vortex formation under specific conditions [3].

1.6 Axion-Like Particles as Dark Matter

Axion-like particles are a type of light scalar field that can behave as dark matter. These particles are similar to the well-known QCD axion, which was originally proposed to solve the strong CP problem, but ALPs are more general: they can have a wide range of masses and interaction strengths and are not tied to QCD [38]. ALPs arise as the angular part (or phase) of a complex scalar field when a global symmetry is broken. This symmetry breaking happens at a high energy scale, typically denoted by f_a, and leads to the appearance of a new light particle, the ALP [4].

In the early universe, the ALP field is initially displaced from the minimum of its potential. As the universe expands and cools, the Hubble rate H drops below the ALP mass m, and the field begins to oscillate. These coherent oscillations behave like cold dark matter. This process is called the misalignment mechanism, and it naturally produces a non-thermal population of ALPs that can make up the dark matter today [4].

The potential for the ALP field is periodic and often written as:

$$V(\phi) = \Lambda_a^4\left[1 - \cos\left(\frac{\phi}{f_a}\right)\right], \tag{1.8}$$

where Λ_a sets the energy scale of the potential [44]. When the field value stays close to the minimum, this potential can be approximated by a simple quadratic form:

$$V(\phi) \approx \frac{1}{2}m^2\phi^2, \tag{1.9}$$

with the mass given by $m = \Lambda_a^2/f_a$. The dynamics of ALPs in this regime are then the same as those of a free massive scalar field.

ALPs are interesting because they cover a very wide range of masses, ranging from around 10^{-33} eV to above the eV scale. Depending on the mass and production scenario, they can behave as dark matter, dark radiation, or have observable effects on cosmological structure formation. In many standard scenarios, their self-interactions are negligible compared to gravity, so they act like fuzzy dark matter on small scales while remaining cold and collisionless on large scales [45, 4].

An important distinction is whether the symmetry breaking that creates the ALP field happens before or after inflation:

- In the pre-inflationary scenarios, the ALP field is homogeneous across the observable universe, and the initial misalignment angle determines the dark matter abundance. This scenario also leads to isocurvature fluctuations, which are constrained by observations of the CMB.
- In the post-inflationary scenario, where the symmetry breaks after the period of Inflation, different parts of the universe can have different initial field values, leading to large differences in the local ALP density. This creates small-scale fluctuations that can lead to the formation of dense gravitationally bound clumps of ALPs, known as miniclusters.

ALPs provide a flexible class, meaning that they offer a wide range of different properties of dark matter candidates. They combine simple field dynamics with rich cosmological phenomenology and can act as fuzzy dark matter on small scales, while also being embedded in high-energy theories such as string theory and extensions of the Standard Model [24].

In particular, axion fuzzy dark matter inherits all the wave-like features of the generic fuzzy dark matter scenario, including the formation of solitonic cores at the centres of dark matter halos shown in Figure 1.3 and the suppression of small-scale structure below the axion's de Broglie wavelength. These effects arise naturally because of the quantum pressure associated with the coherent field. Quantum pressure will be discussed in the next Subsection 1.6.1. What sets axion FDM apart is that these features are realised within a well-defined cosmological and particle physics framework since it was introduced to solve a problem arising in QCD. This makes axion FDM not only a phenomenologically good model for structure formation, but also a promising candidate for connecting astrophysical observations with high-energy theories [4, 3].

1.6.1 Dynamics of Axion-like Dark Matter

To understand how ALP DM and FDM differ dynamically from standard cold dark matter and form small-scale structures, it is necessary to study the equations of the evolution of scalar fields under gravity. In the weak-field, non-relativistic limit appropriate for cosmological structure formation of these scales, the dynamics of axion fields can be described using the SP system of equations as was mentioned in Section 1.5.

First, one must begin with a scalar field ϕ evolving in a flat, perturbed Friedmann–Lemaître–Robertson–Walker (FLRW) metric. On small scales, the Newtonian potential V_N is identified with the scalar perturbation Ψ due to the absence of anisotropic stress in minimally coupled scalar fields. The scalar field action, combined with gravity in this limit, leads to an effective action where fast oscillations of ϕ at frequency m can be separated using the ansatz:

$$\phi(x,t) = \frac{1}{\sqrt{2ma^3}} \left(\psi(x,t)e^{-imt} + \psi^*(x,t)e^{imt}\right), \tag{1.10}$$

where ψ is a slowly varying complex field [4].

Neglecting rapidly oscillating terms and assuming that the velocity field is much smaller than the density field, i.e. $\dot{\psi} \ll m\psi$ and $m \gg H$, the resulting dynamics are described by this form of Schrödinger-Poisson equations:

$$i\hbar\partial_t\psi = -\frac{\hbar^2}{2ma^2}\nabla^2\psi + mV_N\psi, \tag{1.11}$$

$$\nabla^2 V_N = \frac{4\pi G}{a}\left(\rho - \langle\rho\rangle\right), \tag{1.12}$$

where $\rho = m|\psi|^2$ is the density and $\langle\rho\rangle$ denotes its spatial average.

These equations reveal the inherently wave-like nature of axion dark matter on small scales. The gradient term in the Schrödinger equation, $-(\hbar^2/2ma^2)\nabla^2\psi$, gives rise to a quantum pressure that counteracts gravitational collapse. To illustrate this behaviour more clearly, one can rewrite the complex field ψ using the Madelung transformation:

$$\psi = \sqrt{\rho/m}\, e^{im\theta/\hbar} = \sqrt{n}\, e^{im\theta/\hbar}, \tag{1.13}$$

where n is the comoving number density and θ is the phase of the wave function. The velocity field is then defined as the gradient of the phase, $\vec{v} = \nabla\theta$ [45, 4].

Substituting this into the Schrödinger-Poisson system leads to a set of hydrodynamic-like equations:

$$\partial_t\rho + \frac{1}{a^2}\nabla\cdot(\rho\vec{v}) = 0, \tag{1.14}$$

$$\partial_t\vec{v} + \frac{1}{a^2}(\vec{v}\cdot\nabla)\vec{v} = -\nabla V_N + \frac{\hbar^2}{2m^2a^2}\nabla\left(\frac{\nabla^2\sqrt{\rho}}{\sqrt{\rho}}\right), \tag{1.15}$$

where the last term is often referred to as the quantum pressure or quantum potential. It appears from spatial gradients in the amplitude of the density and captures the dispersive behaviour of the scalar field [4].

This additional pressure-like contribution becomes dynamically significant on scales where density gradients are steep, typically at or below the de Broglie wavelength $\lambda_{\rm dB}$, meaning in the central part of the soliton. To estimate the relative importance of this term, one should use a halo of mass M and radius R, and compare the quantum pressure gradient Q/R to the typical inertial term v^2/R. Doing this, one finds the following relation:

$$\frac{Q/R}{v^2/R} \sim \left(\frac{\lambda_{\rm dB}}{R}\right)^2, \tag{1.16}$$

which implies that deviations from classical cold dark matter (CDM) dynamics are controlled by the ratio of the de Broglie wavelength to the halo size. For ultra-light axions, $\lambda_{\rm dB}$ can reach kiloparsec scales, making the quantum pressure term relevant for the internal dynamics of dwarf galaxies and solitonic cores [4].

As shown in [46], the inclusion of this pressure term is necessary to accurately model the stability and structure of fuzzy dark matter halos. It suppresses small-scale fluctuations and supports the formation of coherent solitonic cores, thus playing a defining role in the nonlinear evolution of axion dark matter.

In this framework, gravitational clustering and structure formation can be viewed through the lens of wave turbulence. The process of Bose-Einstein condensation, gravitational collapse of DM, leads to the emergence of coherent solitonic structures. These effects occur classically in the field description and do not require quantum statistical mechanics. Consequently, this classical wave dynamics offers a powerful alternative to the collisionless particle picture of CDM, with distinct implications for galactic structures [4].

1.6.2 What is the Aforementioned Soliton?

There exists a Newtonian scalar field stationary solution to the SP system in Equation 1.12 called a soliton, which is a gravitationally bound configuration of scalar particles. How such a state should look is shown in Figure 1.3. These solitonic solutions are sometimes referred to in the literature as Bose stars [47, 48], (dilute) axion stars [49, 50], and other names. The terminology used often depends on the context this can be for example, Bose stars referring to solitons formed in the early Universe, while those formed inside dark matter halos today are called solitonic cores.

Mathematically, solitons correspond to stationary (i.e. time-independent) solutions of the SP system, obtained by substituting $\psi(\mathbf{r}, t) = \psi(\mathbf{r})e^{-iEt/\hbar}$ into the time-dependent Schrödinger equation. This leads to finding the eigenvalues in these equations:

$$mE\psi = -\frac{\hbar^2}{2m}\nabla^2\psi + mV_N\psi, \tag{1.17}$$

$$\nabla^2 V_N = 4\pi G m|\psi|^2, \tag{1.18}$$

where E is the energy eigenvalue and V_N is the Newtonian gravitational potential. To describe isolated, self-gravitating structures, a boundary condition is imposed $\psi(r \to \infty) = 0$. The resulting density $\rho = |\psi|^2$ then describes a nearly Gaussian-shaped profile with a flat central core and a sharp fall-off at large radii [4].

Assuming spherical symmetry simplifies this system further. Defining $\psi = \psi(r)$ and $V_N = \Phi(r)$, we can rewrite the equations as:

$$-\frac{\hbar^2}{2m}\frac{1}{r}\frac{d^2}{dr^2}(r\psi) = (\omega - m\Phi)\,\psi, \tag{1.19}$$

$$\frac{1}{r}\frac{d^2}{dr^2}(r\Phi) = 4\pi G\rho_0\psi^2, \tag{1.20}$$

where $\omega = E/\hbar$, and ρ_0 is a reference density used for normalization.

It's often useful to work in dimensionless units. Using the following scaling relations obtained from [51], we arrive at dimensionless units:

$$\hat{\psi} = \frac{m\sqrt{G\rho_0}}{\hbar}\psi, \quad \hat{\Phi} = \frac{m^2}{\hbar^2}\Phi, \quad \hat{\omega} = \frac{m\omega}{\hbar^2}. \tag{1.21}$$

With these rescaled quantities, the equations become:

$$\frac{1}{2}\frac{d^2}{dr^2}(r\hat{\psi}) = r\hat{\psi}(\hat{\Phi} - \hat{\omega}), \tag{1.22}$$

$$\frac{d^2}{dr^2}(r\hat{\Phi}) = 4\pi r\hat{\psi}^2, \tag{1.23}$$

where $\hat{\omega}$ plays the role of the eigenvalue of the system. We can use boundary conditions to find physically meaningful solutions:

$$\hat{\psi}(r \to \infty) \to 0, \qquad \hat{\Phi}(r \to \infty) = -\frac{GM}{\tilde{\hbar}^2 r}, \tag{1.24}$$

$$\hat{\psi}(0) = 1, \qquad \left.\frac{d\hat{\psi}}{dr}\right|_{r=0} = 0, \tag{1.25}$$

$$\left.\frac{d\hat{\Phi}}{dr}\right|_{r=0} = 0, \qquad \left.\frac{d\hat{\psi}}{dr}\right|_{r\to\infty} \to 0, \tag{1.26}$$

Where $M(r) = 4\pi \int_0^r \rho(r')r'^2 dr'$ is the enclosed mass within the radius r.

Solving this system numerically gives a localised soliton that is gravitationally bound and static. Some of its important properties can be approximated by these relations that characterise its structure: [4, 2]:

1. **Half-mass radius**:
$$R_{1/2} \simeq \frac{4\hbar^2}{GMm^2}, \tag{1.27}$$
where M is the total mass of the soliton.

2. **Central density**:
$$\rho_c \simeq 4 \times 10^{-3} \left(\frac{Gm^2}{\hbar^2}\right)^3 M^4 \simeq 2\bar{\rho}_{1/2}, \tag{1.28}$$
with
$$\bar{\rho}_{1/2} = \frac{3(M/2)}{4\pi R_{1/2}^3} \tag{1.29}$$
as the mean density within the half-mass radius.

3. **Virial velocity**:
$$v_{\rm vir}^2 \simeq 0.1 \left(\frac{GMm}{\hbar}\right)^2 \simeq 0.4\frac{GM}{R_{1/2}} \simeq -0.3V_{N,c}, \tag{1.30}$$
where $V_{N,c}$ is the central gravitational potential, approximated as
$$V_{N,c} \simeq -0.3 \left(\frac{GMm}{\hbar}\right)^2. \tag{1.31}$$

4. **Coherence (de Broglie) length**:
$$\lambda_{\rm dB} = \frac{\hbar}{mv_{\rm vir}} \simeq 0.8R_{1/2}, \tag{1.32}$$
showing that the soliton size is closely tied to the wavelength of its constituent particles.

An important property of the SP equations is their invariance under a scaling transformation:
$$\{t, x, V_N, \psi, \rho\} \to \{\lambda^{-2}\hat{t}, \lambda^{-1}\hat{x}, \lambda^2\hat{V}_N, \lambda^2\hat{\psi}, \lambda^4\hat{\rho}\}, \tag{1.33}$$
where λ is an arbitrary scaling parameter. This symmetry enables rescaling soliton profiles to different astrophysical contexts, from galactic cores to miniclusters, while preserving the underlying structure of the solutions [52, 53].

1.6.3 Wave Turbulence and Bose-Einstein Condensation

Bosonic dark matter, like axions described by a self-gravitating scalar field, can go through a process that's kind of like Bose–Einstein condensation, except it happens due to gravity [54]. Over time, the axion field can become more organised as the particles interact gravitationally and start forming clumps.

This kind of behaviour is studied using wave turbulence theory, which looks at how classical waves change over time. It's a bit like kinetic theory in gases, just with waves instead of particles [4]. Even though this is all classical physics, the wave nature of fuzzy dark matter leads to effects that look a lot like quantum condensation.

One thing that makes this more complicated is that gravity is a non-local force: what happens in one place depends on mass in other places. That's different from things like plasmas or light waves, where the interactions are usually local. But interestingly, similar behaviour has been seen in nonlinear optics, where things called "incoherent solitons" can form when the interactions are also non-local. This is kind of like how solitonic cores form in dark matter halos.

Recent simulations have shown that axion dark matter can really condense like this [55]. They show that small clumps, or solitons, can appear and grow over time by pulling in mass from nearby fluctuations. This process is called an "inverse mass cascade," where smaller clumps of matter connect to bigger ones to create even bigger clumps as the system evolves [56].

To describe this process more precisely, one can use a kinetic approach based on the Wigner distribution function [4]:

$$f_W(x,p) = \int \frac{d^3\xi}{(\pi\hbar)^3} e^{-2ip\cdot\xi/\hbar} \langle \psi(x+\xi)\psi^*(x-\xi)\rangle, \tag{1.34}$$

which tells us how the wave energy is distributed in phase space (position x and momentum p). Its evolution is governed by an equation similar to the classical Vlasov-Poisson system, but with additional terms that account for gravitational scattering:

$$\partial_t f_W + \frac{p}{a^2 m} \cdot \nabla_x f_W = \text{gravitational scattering terms.} \tag{1.35}$$

The key point is that relaxation and condensation happen due to these scattering terms, which arise from correlations between different parts of the wave field [57]. These terms become important when the de Broglie wavelength $\lambda_{\rm dB}$ is small compared to the typical size of the structures being formed ($\lambda_{\rm dB} \ll R$) [4]. In that case, corrections to the evolution are suppressed by a small parameter $\varepsilon = \lambda_{\rm dB}/R$.

At leading order, the dynamics reduce to those of CDM, but higher-order corrections introduce the distinctive features of FDM. The relaxation time τ, the timescale on which Bose-Einstein condensation proceeds, is governed by gravitational scattering and controls how quickly coherent structures like solitons form:

$$\text{Scattering term} \sim \frac{f_W}{\tau}. \tag{1.36}$$

This provides a kinetic description of how wavelike dark matter condenses into localised, gravitationally bound objects over cosmological timescales. More detailed descriptions can be found in [55, 4].

1.6.4 Formation and Growth of Solitonic Cores

In the fuzzy dark matter scenarios, the wavelike nature of ultra-light bosons leads to the creation of gravitationally bound structures called solitonic cores. These cores arise due to interactions between the gravitational attraction and quantum pressure [4]. Quantum pressure of the system appears from the gradient energy term in the Schrödinger-Poisson system as shown in Equation 1.15.

The formation of solitons begins with gravitational collapse in overly dense regions, where quantum pressure stops any further contraction and stabilises as a core. This process has been confirmed in numerous cosmological simulations [55], which reveal that the innermost regions of FDM halos are consistently described by soliton-like profiles, while the outer regions resemble the Navarro-Frenk-White (NFW) profile typical of CDM.

The NFW profile is an empirical fitting function derived from N-body simulations of hierarchical structure formation in a CDM universe [58, 59]. It describes the characteristic density distribution of dark matter halos over a wide range of masses and redshifts. The density profile is given by

$$\rho_{\mathrm{NFW}}(r) = \frac{\rho_s}{\left(\frac{r}{r_s}\right)\left(1+\frac{r}{r_s}\right)^2}, \tag{1.37}$$

where r is the radial coordinate, ρ_s is the density, and r_s is a scale radius at which the logarithmic slope of the density profile equals -2.

This profile exhibits a central cusp, $\rho(r) \sim r^{-1}$ as $r \to 0$, and falls off as $\rho(r) \sim r^{-3}$ at large distances. The total mass diverges logarithmically, which is physically regulated by imposing a virial cut-off at a radius r_{200}, where the average enclosed density is 200 times the critical density of the universe.

The importance of the NFW profile lies in its universality and its predictive success in describing the outer regions of dark matter halos in CDM simulations. It is used as the base model in both observational and theoretical studies of galaxy formation, gravitational lensing, and cosmological simulations. [59]

However, in the context of FDM and ultra-light axion models, the inner regions of halos exhibit behaviour different from the NFW profile due to quantum pressure effects. The central cusp predicted by the NFW profile is therefore replaced by a solitonic core with a flatter density distribution. For the system that will be used in this work, meaning resulting from N-body mergers, a composite profile is better suited to fit the full halo:

$$\rho_{\mathrm{halo}}(r) = \Theta(r_\epsilon - r)\rho_{\mathrm{sol}}(r) + \Theta(r - r_\epsilon)\rho_{\mathrm{NFW}}(r), \tag{1.38}$$

where Θ is the Heaviside step function, r_ϵ is the transition radius between the solitonic core and the NFW outer halo. The solitonic profile is given by

$$\rho_{\mathrm{sol}}(r) = \frac{\rho_c}{[1+\alpha(r/r_c)^2]^8}. \tag{1.39}$$

Here, ρ_c is the central density of the soliton, r_c is the characteristic radius of the core, and α is a dimensionless parameter that controls the steepness of the transition from the flat core to the outer power law behaviour. I shall adopt $\alpha = 0.091$ [60].

During cosmic structure formation, solitons grow by merging with smaller cores or by accreting ambient matter. This gradual buildup leads to a tight correlation between the core mass M_c and the mass of the dark matter halo M_h. The relation follows a scaling law of the form:

$$M_c \propto a^{-1/2} M_h^{1/3}, \tag{1.40}$$

where a is the cosmic scale factor. This relation was originally derived from cosmological simulations of the formation of the FDM structure [34] and reflects the average behaviour of the DM halos. However, taking into account mergers of idealised solitons, a different scaling emerges:

$$M_c \propto M_h^{5/9}, \tag{1.41}$$

as shown in [61]. This has since been reproduced by several other authors as well.

Both relations show how FDM halos grow over time and the non-relativistic, wave-like behaviour of the solitons. In simulations, these processes also create interference patterns and relaxation characteristics, which are often referred to as "gravitational cooling" [4].

1.6.5 Characteristic Scales and Natural Units

Following the conventions of [2], the Schrödinger-Poisson equations can be written in physical units as shown in Equation 1.5 and Equation 1.6 where ψ is the wave function of the ultralight bosonic dark matter particle of mass m, and V is the Newtonian gravitational potential.

To get the idea of typical scales of structures in these systems, a natural length is defined by finding the value for which the quantum pressure and gravitational potential energy are equal:

$$\frac{\hbar^2}{2mx^2} \sim \frac{GmM}{x} \quad \Rightarrow \quad x_c \sim \frac{\hbar^2}{Gm^2M}. \tag{1.42}$$

This defines the characteristic core radius x_c of a self-gravitating soliton of mass M.

The associated time scale is obtained from the kinetic term:

$$t_c \sim \frac{\hbar}{Gm^2M}. \tag{1.43}$$

These expressions reveal that the system is indeed of a non-relativistic nature: The characteristic size of the bound object increases as the particle mass m decreases. For reference, FDM mass $m \sim 10^{-22}$ eV and the core mass $M \sim 10^8 M_\odot$, the core radius is of the order of kiloparsecs and the time scale is of the order 10^9 years.

1.7 Simulating Fuzzy and Axion-like Dark Matter

Fuzzy Dark Matter, modelled as a massive scalar field governed by the Schrödinger–Poisson equation shown in Equation 1.5, offers a compelling alternative to the Cold Dark Matter model as was mentioned in Section 1.3. Due to the extremely low particle mass $m \sim 10^{-22}$ eV, quantum mechanical effects manifest on astrophysical scales due to large de Broglie wavelengths, leading to phenomena such as solitonic cores and interference patterns in dark matter halos [1, 34].

Numerical simulations are crucial to explore the non-linear regime of FDM dynamics. Unlike standard N-body simulations for CDM, FDM requires solving the SP system, which describes the wave behaviour of the field. These simulations reveal that virialized halos develop a distinct core–halo structure as is shown on Figure 1.3: a central soliton supported by quantum pressure, surrounded by a turbulent halo formed through wave interference [34, 61].

Several numerical methods have been developed over the years to integrate the SP system. Among them, the split-step Fourier method stands out for its efficiency and accuracy in evolving the wave function in time [61]. These simulations not only help validate the theoretical predictions of FDM but also provide direct observables such as the halo mass function, core–halo mass relation, and power spectrum suppression on small scales. Consequently, simulation-based

approaches are essential in constraining fuzzy and axion-like dark matter models and linking them to observations of structure formation.

The FDM model is equivalent to a scalar field theory describing an ultralight bosonic particle. At a fundamental level, this is represented by a real scalar field ϕ governed by a relativistic action. In the non-relativistic and weak-field limit, this field can be re-expressed as a complex scalar wave function ψ using Equation 1.10, which then can be evolved using the Schrödinger–Poisson Equations 1.5 [4].

To simulate the formation of cosmic structures, the SP system is often expressed in the so-called "comoving coordinates", using a rescaled wave function $\psi_c = a^{3/2}\psi$, a comoving gradient $\nabla_c = a\nabla$, and a comoving gravitational potential $\Phi_c = a\Phi$. The evolution equations then take the form:

$$i\hbar\partial_t\psi_c(t,\mathbf{x}) = -\frac{\hbar^2}{2ma(t)^2}\nabla_c^2\psi_c(t,\mathbf{x}) + \frac{m}{a(t)}\Phi_c(t,\mathbf{x})\psi_c(t,\mathbf{x}), \tag{1.44}$$

$$\nabla_c^2\Phi_c(t,\mathbf{x}) = 4\pi Gm\left(|\psi_c(t,\mathbf{x})|^2 - \langle|\psi_c|^2\rangle(t)\right), \tag{1.45}$$

where the subtraction of the mean density ensures compatibility with periodic boundary conditions.

The wave function ψ_c carries both density and velocity information. The mass density is once again given by $\rho_c = m|\psi_c|^2$, and the velocity field can be derived from the phase of the wave function, $\psi_c = \sqrt{\rho_c/m}e^{i\alpha}$, as $\mathbf{v}_c = \hbar/m\nabla_c\alpha$. This polar decomposition allows for a fluid-like interpretation of the system via the Madelung transformation [62]. It results in a continuity equation and a modified Euler equation, where a quantum pressure term introduced in Section 1.6 appears:

$$\partial_t\mathbf{v}_c + \frac{1}{a^2}\nabla_c\mathbf{v}_c^2 = -\frac{1}{a}\nabla_c\Phi_c + \frac{\hbar^2}{2m^2a^2}\nabla_c\left(\frac{\nabla_c^2\sqrt{\rho_c}}{\sqrt{\rho_c}}\right). \tag{1.46}$$

This quantum pressure leads to a characteristic suppression of structure formation on Jeans scales, preventing the DM structure from collapsing further. The comoving Jeans wave number depends on redshift and particle mass as:

$$k_J = \left(\frac{6m}{1+z}\right)^{1/4}\left(\frac{mH_0}{\hbar}\right)^{1/2}, \tag{1.47}$$

Implying that only perturbations larger than the associated Jeans length can get bigger gravitationally [62]. In our simulations, I shall adopt a redshift of $z = 0$ and scaling factor $a = 1$ for simplicity, but this can easily be changed in the continuation of this work [62, 1, 34].

1.7.1 Split-Step Fourier Method for the Schrödinger–Poisson System

The foundation of our numerical approach lies in the Fourier split-step method [63], which uses the separability of the Hamiltonian into kinetic and potential components. This technique enables stable and efficient evolution of the wave function by alternating between operations in different representations: Fourier space for kinetic terms and real space for potential terms.

Considering first the vacuum Schrödinger equation in one spatial dimension:

$$i\frac{\partial\psi(x,t)}{\partial t} = -\frac{1}{2}\partial_x^2\psi(x,t). \tag{1.48}$$

The analytical solution over a small time step h can be written as:

$$\psi(x, t+h) = \exp\left(i\frac{h}{2}\partial_x^2\right)\psi(x,t). \tag{1.49}$$

The key insight is that the exponential of the differential operator can be computed using the Fourier transform. In Fourier space, the spatial derivatives become just a multiplication by the associated wave number k. Applying the Fourier transform $\mathscr{F}$ to our previous solution in Equation 1.49, we obtain:

$$\widetilde{\psi}(k, t+h) = \exp\left(-i\frac{h}{2}k^2\right)\widetilde{\psi}(k,t), \tag{1.50}$$

where $\widetilde{\psi}$ denotes the Fourier transformed ψ and k is the wave number. The resulting wave function in real space is then recovered via the inverse transform:

$$\psi(x, t+h) = \mathscr{F}^{-1}\left[\exp\left(-i\frac{h}{2}k^2\right)\widetilde{\psi}(k,t)\right]. \tag{1.51}$$

When an external potential $V(x)$ is added to the Schrödinger equation, the complete time-dependent Schrödinger equation becomes:

$$i\frac{\partial\psi(x,t)}{\partial t} = \left(-\frac{1}{2}\partial_x^2 + V(x)\right)\psi(x,t). \tag{1.52}$$

For finite time steps, we can employ symmetric operator splitting (Strang splitting) [64] to maintain second-order accuracy:

$$\psi(x, t+h) \approx e^{-i\frac{h}{2}V(x)}\,\mathscr{F}^{-1}\left[e^{-i\frac{h}{2}k^2}\,\mathscr{F}\left(e^{-i\frac{h}{2}V(x)}\psi(x,t)\right)\right]. \tag{1.53}$$

This solves the equation by alternating between real space for potential operations and Fourier space for kinetic operations. The method is very stable and preserves the norm of the wave function to machine precision, making it ideal for long-time integrations required in cosmological simulations [65, 62].

1.7.2 Extension to the Schrödinger-Poisson System

To model fuzzy dark matter accurately, we must solve the coupled Schrödinger-Poisson system that incorporates gravitational self-interactions. In comoving coordinates with scale factor $a(t)$, this system takes the form:

$$i\frac{\partial\psi_c(x,t)}{\partial t} = -\frac{\hbar}{2m}\frac{1}{a(t)^2}\nabla_c^2\psi_c(x,t) + \frac{m}{\hbar}\frac{1}{a(t)}\Phi_c(x,t)\psi_c(x,t), \tag{1.54}$$

$$\nabla_c^2\Phi_c(x,t) = \frac{4\pi G}{a(t)}\left[m|\psi_c(x,t)|^2 - \langle m|\psi_c|^2\rangle\right], \tag{1.55}$$

where ψ_c is the comoving wave function, Φ_c is the comoving gravitational potential, and $\langle\cdot\rangle$ denotes spatial averaging. The subtraction of the mean density in the Poisson equation accounts for the homogeneous background expansion.

The formal solution over a time step Δt can be written as:

$$\psi_c(t+\Delta t, x) = \mathcal{T}\exp\left[-i\int_t^{t+\Delta t}\hat{H}(t')dt'\right]\psi_c(t,x), \tag{1.56}$$

where $\mathcal{T}$ denotes time ordering and $\hat{H}(t')$ is the time-dependent Hamiltonian operator [62].

For small time steps, we approximate the evolution of the scale factor and potential as quasi-static, allowing us to apply the split-step method. Using symmetric operator splitting, the evolution over one time step becomes [62, 65]:

$$\psi_c(t+\Delta t, x) \approx \underbrace{e^{-i\frac{m}{\hbar}\frac{1}{a(t)}\frac{\Delta t}{2}\Phi_c(t+\Delta t,x)}}_{\text{kick}} \underbrace{e^{i\frac{\hbar}{m}\frac{1}{a(t)^2}\frac{\Delta t}{2}\nabla_c^2}}_{\text{drift}} \underbrace{e^{-i\frac{m}{\hbar}\frac{1}{a(t)}\frac{\Delta t}{2}\Phi_c(t,x)}}_{\text{kick}} \psi_c(t,x). \tag{1.57}$$

This kick-drift-kick structure separates the evolution into three computationally efficient steps:

- **Kick steps**: Applying the gravitational potential in real space through multiplication by an exponential
- **Drift step**: Appling the kinetic operator in Fourier space, where $\nabla_c^2 \to -k^2$, by jumping in and out of Fourier space

1.7.3 Computational Algorithm

The complete numerical procedure for each time step consists of the following operations:

1. **Density computation**: Calculating the matter density from the wave function:

$$\rho(x,t) = m|\psi_c(x,t)|^2 \tag{1.58}$$

2. **Poisson solver**: Computing the gravitational potential using the squared absolute value of the wave function and using a Fourier transform:

$$\widetilde{\rho}(k,t) = \mathscr{F}[\rho(x,t) - \langle\rho\rangle] \tag{1.59}$$

$$\widetilde{\Phi}_c(k,t) = \frac{4\pi G}{a(t)}\frac{\widetilde{\rho}(k,t)}{k^2} \tag{1.60}$$

$$\Phi_c(x,t) = \mathscr{F}^{-1}[\widetilde{\Phi}_c(k,t)] \tag{1.61}$$

   where the $k = 0$ mode is set to zero to prevent any computational errors arising from dividing by zero.

3. **Wave function evolution**: Appling the split-step evolution:

$$\psi_c^{(1)} = \exp\left(-i\frac{m}{\hbar}\frac{1}{a(t)}\frac{\Delta t}{2}\Phi_c(t,x)\right)\psi_c(t,x) \tag{1.62}$$

$$\widetilde{\psi}_c^{(2)} = \exp\left(-i\frac{\hbar}{m}\frac{1}{a(t)^2}\frac{\Delta t}{2}k^2\right)\mathscr{F}[\psi_c^{(1)}] \tag{1.63}$$

$$\psi_c^{(3)} = \mathscr{F}^{-1}[\widetilde{\psi}_c^{(2)}] \tag{1.64}$$

$$\psi_c(t+\Delta t, x) = \exp\left(-i\frac{m}{\hbar}\frac{1}{a(t)}\frac{\Delta t}{2}\Phi_c(t+\Delta t,x)\right)\psi_c^{(3)} \tag{1.65}$$

4. **Repeating the steps**

This implementation is heavily inspired by [60, 62, 65].

Higher Order Methods

While the second-order symmetric split step method presented above is widely used due to being computationally inexpensive, it is possible to employ higher-order methods. This can provide improved accuracy for long-term simulations at the cost of additional computational steps. The fourth-order and even one step further going sixth-order split step method developed by Yoshida [66] offers significantly enhanced precision through a more complex decomposition of the time evolution operator. This method uses a sequence of eight alternating kick and drift steps with carefully optimized coefficients, achieving $\mathcal{O}(\Delta t^7)$ accuracy while maintaining the desirable properties of unitarity and time-reversibility. This is very desirable for long-term simulations requiring stability in gravitational wave simulations [55]. Higher-order methods become particularly valuable when there is a demand for increased computational precision. However, for many simulations of structure formation on cosmological scales, the second-order method provides a great balance between computational cost and accuracy [67].

1.7.4 Numerical Properties and Advantages

This method offers several key advantages for fuzzy dark matter simulations. The method is stable for choices of time step that satisfy the requirement that the phase difference in the exponentials must not exceed 2π, beyond which problems arising from enforcing periodic boundary conditions would occur. Both the kicks and drift operations yield separate constraints that must be simultaneously fulfilled:

$$\Delta t < \min\left(\frac{4\hbar}{3\pi m}\frac{a^2}{(\Delta x)^2}, \frac{2\pi\hbar}{m|\Phi_{c,\max}|a}\right), \tag{1.66}$$

where $\Delta x = L/N$ is the spatial resolution and $\Phi_{c,\max}$ is the maximum value of the potential. The constraint involving the resolution appears from the drift operation, while the constraint with the potential results from the kick operation. The dependence $\Delta t \propto (\Delta x)^2$ is typical for numerical approaches to the Schrödinger-Poisson system and reflects the relation of the Schrödinger equation to diffusion problems [62, 60]. Another benefit of using the split-step method is its efficiency of the method. The use of FFTs reduces the computational complexity of spatial derivatives from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$ per dimension, where N is the number of grid points. The existence of ready-to-use Fourier transform libraries in Python is another big plus. Although the derivation of the Schrödinger equation solution in Equation 1.48 was sketched in one dimension for simplicity, the framework naturally generalizes to higher dimensions. The next big benefit the split-step method provides is easy incorporation of additional physics, such as self-interactions or modified gravity. From which an important note arises that this method applies equally well to nonlinear Schrödinger equations, because the density $|\psi(t+\Delta t)|^2$ can be computed at the half-step $t+\Delta t/2$ using the property that $|\psi(t+\Delta t/2)|^2 = |\psi(t+\Delta t)|^2$ for the symmetric splitting scheme. This allows the gravitational potential $\Phi(t+\Delta t)$ to be determined before the full wave function $\psi(t+\Delta t)$ is computed, maintaining the self-consistency of the coupled system [62].

The combination of these properties makes the Fourier split-step method the method of choice for large-scale cosmological simulations of fuzzy dark matter, allowing for the simulation of gravitationally bound DM systems.

1.7.5 Manipulation with Small Numbers

Since this project focuses on simulating extremely light ALP dark matter with $m \sim 10^{-22}$ eV, and we work in astrophysical units such as $M_\odot$, the relevant physical quantities, especially those involving the reduced Planck constant $\hbar$, can become extremely small. These small values may lead to numerical precision issues, as they approach the limits of floating-point accuracy in standard computational implementations.

To address this, we introduce a rescaled constant defined as $\tilde{\hbar} = \hbar/m$. For our scalar field with mass $m \sim 10^{-22}\,\mathrm{eV}$, and working in units of eV, Gyr, and kpc, this yields $\tilde{\hbar}_{m_{22}} \approx 19.6\,\mathrm{kpc}^2/\mathrm{Gyr}$. This value is significantly more suitable for numerical computation, mitigating precision errors associated with extremely small numbers.

Using this constant, we rescale the variables as:

$$\hat{\psi}_c = \frac{\sqrt{G}}{\tilde{\hbar}}\psi_c, \quad \hat{\Phi} = \frac{\Phi}{\tilde{\hbar}^2},$$

which improves numerical stability and precision. This approach follows the rescaling procedure adopted in [60].

1.7.6 Spin Implementation in the Simulation

This work aims to simulate the dynamics between DM structures for different spins in the ALP DM model and to examine their effects on survival time, structural configurations, and mass transfer. These simulations may reveal spin-dependent behaviour which could be of interest to different theoretical models. For this purpose, a method to implement higher spins in numerical simulations is necessary. In the literature, such as [60], one finds that this can be done by decomposing the wave function into polarization states as follows:

$$\Psi(t, \mathbf{x}) = \sum_p \psi_p(t, \mathbf{x})\epsilon^p, \tag{1.67}$$

where ψ_p is the field in the polarization state p. Choosing the polarization state can be done using BUNO[1] by setting the initial value $\psi_p(t = 0, \mathbf{x})$ to be equal to the spin-0 case, with the solution equal to ψ_{sol} multiplied by a real polarization coefficient and a random phase θ_p such that:

$$\psi_p(t = 0, \mathbf{x}) = \psi_{\mathrm{sol}}(\mathbf{x})\, c_p e^{-i\theta_p}. \tag{1.68}$$

These polarization coefficients are numbers that arise when coupling the angular momenta of quantum particles in the ALP DM framework. They appear as expansion coefficients of total angular momentum eigenstates on an uncoupled tensor product basis. These are the so-called Clebsch–Gordan (CG) coefficients, which define how two particles combine into a total angular momentum eigenstate, i.e.:

$$(j_1, m_1) \otimes (j_2, m_2) \to (J, M), \tag{1.69}$$

where j_i are the total angular momenta of the particles, m_i are their magnetic quantum numbers, and J, M are the resulting total angular momentum and magnetic quantum number of the state. Mathematically, they can be expressed as:

$$|J, M\rangle = \sum_{m_1, m_2} \langle j_1, m_1; j_2, m_2 | J, M\rangle\, |j_1, m_1\rangle |j_2, m_2\rangle,$$

where $\langle j_1, m_1; j_2, m_2 | J, M\rangle$ is the *Clebsch–Gordan coefficient*.

[1] without the loss of generality

We use these coefficients to "lift„ the scalar wave function, which solves the Schrödinger equation from spin 0 to a tensor-valued wave function that represents a particle with higher spin. Essentially, a scalar field spin-0 solution is decomposed and recombined into several components, each associated with a particular spin projection. A big plus of using Clebsch–Gordan coefficients is their symmetry: some states differ only by the complex conjugation. In simulations, this allows for a reduction in memory demands, and the fact that a Python library with precomputed CG coefficients already exists.

To illustrate how this procedure works, we will show cases for spin 0, 1, and 2:

Spin 0 This is the simplest case, as the field is defined by a single component ψ_p in Equation 1.68, with $c_p = 1$:

$$\psi_0 = \psi_{\text{sol}} e^{-i\theta_0}. \tag{1.70}$$

Spin 1 The basis is represented by the following set of orthonormal vectors associated with three polarization states $p = \pm 1, 0$:

$$\boldsymbol{\epsilon}(\pm 1) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ \pm i \\ 0 \end{pmatrix}, \quad \boldsymbol{\epsilon}(0) = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}. \tag{1.71}$$

Two of the coefficients c_p are chosen randomly, and the third is then fixed by requiring that $\sum_p |c_p^2| = 1$. This ensures that the resulting vector is normalised. In practice, we are creating an orthonormal random vector basis. If we take a look at the polarisation states $\boldsymbol{\epsilon}(\pm 1)$, we can see that they differ only by the sign of the imaginary part. That means when we compute physical observables that depend only on the modulus squared of the wave function $|\Psi|^2$ these two states contribute equally. Meaning, instead of treating them separately, we can compute one and count it twice. This effectively cuts the number of components needed from three to two, which saves memory. And that's exactly what we need, since the only observable required to solve the SP system is the density $\rho = |\Psi|^2$ and not the full complex wave function.

This trick is especially handy because we can exploit the natural symmetry of rank-s tensors in 3D space. In three dimensions, a rank-s tensor has 3^s components if no symmetry is assumed. But when the tensor is fully symmetric, meaning it is unchanged under permutation of its indices, the number of unique components drops to only $\binom{s+2}{s}$. So instead of needing to store all components, we can just keep the symmetric ones and from them reconstruct the full tensor.

When we decompose a spin-s state using CG coefficients, the resulting tensor can be written as a linear combination of products of the polarisation vectors. I will describe this in more detail in Subsection 2.2.4. This means that many of the components in this decomposition are related by symmetry and do not need to be computed, just taken into account by raising their symmetric counterparts' multiplicity.

Spin 2 For spin-2, there are five independent polarisation states, corresponding to $p = \pm 2, \pm 1, 0$. These states form an orthonormal basis and are represented by the following maximally polarised symmetric tensors:

$$\boldsymbol{\epsilon}(\pm 2) = \frac{1}{2}\begin{pmatrix} 1 & \pm i & 0 \\ \pm i & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \tag{1.72}$$

$$\boldsymbol{\epsilon}(\pm 1) = \frac{1}{2}\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & \pm i \\ 1 & \pm i & 0 \end{pmatrix}, \tag{1.73}$$

$$\boldsymbol{\epsilon}(0) = \frac{1}{\sqrt{6}}\begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 2 \end{pmatrix}. \tag{1.74}$$

As before, the five coefficients c_p are randomly assigned (with the normalisation constraint of $\sum_p |c_p^2| = 1$) for each soliton. For example, in this spin-2 case, we do not distinguish between T_{01} and T_{10}, we just need to store one of them and factor in how many times it appears when computing the density. Using our method for reducing the number of wave functions, we only need six independent components instead of nine.

This pattern gets even more useful for higher spins. For spin-3, we reduce from 27 components to just 10, and for spin-4, from 81 to 15. In general, the number of components drops from 3^s to $\binom{s+2}{s}$, which follows from counting combinations with replacement of the three spatial indices. So instead of tracking every component of the full tensor, we only need to store the unique symmetric ones and their multiplicity.

This approach should[2] still preserve all the necessary physical information to simulate gravitational interactions, while cutting down the memory needed to store the wave function. More on how CG coefficients are used to construct spin states will be covered in the next chapter, especially in Subsection 2.2.4. For a more detailed overview of CG coefficients, the author recommends revisiting reader's favourite quantum mechanics textbook[3].

[2] Author's wish

[3] Mine is this one ❤️

Chapter 2

Code Implementation

The aim of this work is to produce a simulation tool for the behaviour of axion-like particle dark matter using the Fourier split step method described in Subsection 1.7.1. The simulation framework uses Python due to the language's simplicity, extensive ecosystem of useful libraries, and even includes some for scientific computing. Python's nature as a high-level programming language allows for quick development and debugging, while libraries such as NumPy, SciPy, and Matplotlib provide perfect numerical and visualization tools. Plus, it is the only programming language with which the author has some experience beyond making the code write out a sad "hello world" message.

Most importantly, Python's compatibility with the CuPy library enables GPU acceleration. CuPy, developed by Preferred Networks, provides a NumPy-compatible interface for CUDA programming, allowing existing NumPy code to be easily transferred to a GPU with minimal modifications. This library handles memory management and data transfers between CPU and GPU automatically, significantly reducing the required code complexity.

GPU programming offers advantages over CPU-based computation for computationally heavy tasks such as this simulation framework. While CPUs excel at sequential processing with typically 4-16 cores, modern GPUs contain thousands of smaller cores optimized for parallel computation. The Fourier split-step method involves operations on large arrays and Fast Fourier Transforms (FFTs), both of which are highly parallelizable operations that benefit enormously from GPU architecture. Basically, instead of trying to solve many simple tasks with "smart" CPU cores, I was advised to attack the tasks with an armada of thousands (5120) of "dumb" GPU cores. For this project, development was primarily done using an NVIDIA GeForce RTX 3070 laptop GPU with 8GB of VRAM, which provided sufficient memory for smaller simulation grids.

2.1 Project Structure

The simulation framework uses Python's object-oriented programming (OOP) capabilities to create a modular and maintainable codebase. Object-oriented design provides several key advantages for scientific computing applications: encapsulation allows separation of data structures and their associated methods to be bundled together, and inheritance of classes in Python enables code reuse. For a more complex code such as this one, OOP provides a great way to write readable and well-structured code into objects which are manageable one by one[1].

[1] One is allowed to challenge the author's claims made about his code only if it doesn't come to the author's attention.

The main building blocks of the simulation framework are the `Simulation_class` and the `Wave_vector_class` class, with supporting classes for handling wave function creation, the `Wave_function` and `Packet` classes, time evolution done by `Evolution_Class`, and the `Propagator_Class`, which handles the calculation of the propagators needed for wave function evolution. Each class serves its purpose within the code with unique operations that are needed throughout the simulation. We will now discuss the specifics of each class, outlining its purpose, how it is implemented in the code, its properties, and the rationale behind its design.

2.1.1 Repository Structure

Since this report will partly serve as documentation for possible future reference of the simulation framework, let me give a brief overview of the GitHub repository. The repository can be found at the URL https://github.com/Martin-Smid/Research-task.

One should mainly focus on the **resources** directory in the repository. The main bulk of the code is located in **resources/Classes**, which contains all the classes, each in its own `.py` file. Additionally, there are some custom errors defined in the **Errors** directory for cleaner error handling within the code. The subdirectory **Functions** contains various helper functions used throughout the project, although some of these might be outdated. Lastly, the **solitons** subdirectory holds `.dat` files with data static soliton profiles. These were provided by Erick Munive Villa, Ph.D, and were obtained by a process described in Subsection 1.6.2.

Static soliton profiles were obtained by solving the time-independent SP system in Equation 1.18 assuming spherical symmetry of the solitons. This SP system can then be rewritten to spherical coordinates and, using the so-called "god's given" units ($\hbar = c = 1$) [68], the resulting system of equations looks like this:

$$E\,\phi(r) = -\frac{1}{2m^2}\left(\frac{1}{r^2}\frac{d}{dr}\left(r^2\frac{d\phi}{dr}\right)\right) + V_N(r)\,\phi(r),$$

$$\frac{1}{r^2}\frac{d}{dr}\left(r^2\frac{dV_N}{dr}\right) = 4\pi G m\,|\phi(r)|^2.$$

To find the ground state, a real-valued wave function $\phi(r)$ vanishing at infinity is assumed, and the equations are solved iteratively using finite-difference methods on a radial grid. The wave function is initialized with a trial profile (e.g., a Gaussian), and the gravitational potential is updated at each step by solving the Poisson equation.

The resulting output should be a static, spherically symmetric solution characterized by a localized wave function $\phi(r)$ and the corresponding gravitational potential.

2.1.2 Code Structure

Here, I'll give a brief overview of how to interact with the code. A more detailed description of each class will be provided in the next Section 2.2.

To set up the simulation and define its type, one needs to create an instance of the `Simulation_class`, let's call it `sim` for this explanation, and pass in the appropriate parameters. Most of these parameters are optional, but a few are essential, as they define the base of the simulation. These include `dim`, which sets the dimensionality of the grid; `boundaries`, which defines the size of the grid; `N` - resolution, which specifies how many spatial points the grid should be divided into along one axis; and `total_time` and `h`, which control how long the simulation runs and what time step is used.

Once the simulation is set up, it's time to populate it. To do that, you create one or more instances of the `Wave_vector_class`. These need to know which simulation they belong to, so the `Simulation_class` instance, our `sim`, must be passed in. Similarly, it is necessary to tell the `Wave_vector_class` where the wave should be located and what momentum it should carry, which is done through lists of values named `means` and `momenta`, with the length equal to the simulation dimension. Another parameter required is `spin`, which determines the spin of the resulting wave function. To choose the "identity" of the wave, one uses the attribute `packet_type`, which requires a string. Right now, the simulation supports three packet types selected by `'gaussian', 'LHO'` or passing in a path to a file with data. To create a `Wave_vector_class` instance, two supporting classes are used. One which handles reading of the data from data files and placing a grid of values which represents the soliton - `Packet` class, and a second which then adopts this grid of values and handles logic of individual wave function components which are created within the wave vector to create a higher spin state.

After creating a wave function, it must be added to the simulation using `sim.add_wave_vector(wave_vector)`. This method expects a `Wave_vector_class` instance.

Once the simulation is set up and populated, it can be run using `sim.evolve(save_every)`, where `save_every` is an integer indicating how often the simulation should produce a snapshot (i.e., after how many time steps). These snapshots can then be used for further analysis. The plotting and analysis capabilities of the code will be discussed later. The Evolve method uses the suggestively called `Evolve` method, which employs the split step technique. This class uses a helper class named `Propagator_Class`.

This process is schematically shown in Figure 2.1

2.2 Classes

This Section will give the reader a more detailed look into each one of the classes, how they work and interact together to produce the results which will be discussed later in Chapter 4.

2.2.1 Simulation Class

The central piece of this simulation framework is fittingly named `Simulation_Class`, which I will continue to call from now on for the sake of brevity `sim`. It is used to keep track of all parameters, such as desired scalar field mass, units used within the simulation, and so on. After initialization and population with a wave function, this object takes care of evolving the simulation using a helper class. Unfortunately, due to the central role `sim` plays in its initialization throughout the work has become quite complex to account for all the functionality required from this framework. To guide users and for better error handling, there is a helper function before initialization of `sim` which checks the types of parameters passed to `sim`.

Initialization

A simplified version of the class constructor is shown in the following Listing 1 and will now be briefly discussed:

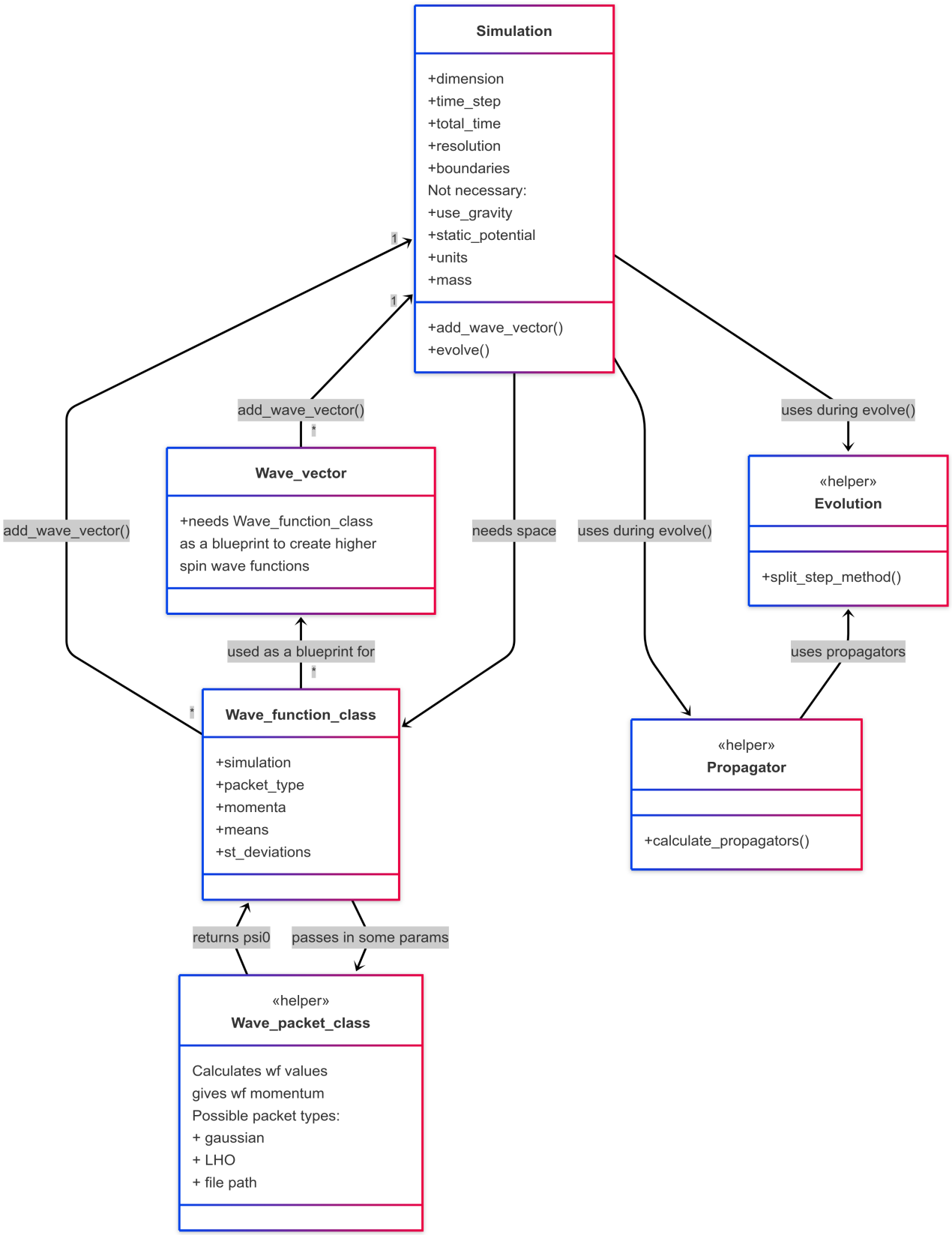

Figure 2.1: Simplified schema of how the building blocks of the code work together to produce simulations.

Potentials
Other than some already mentioned parameters in Subsection 2.1.2, there are the `use_gravity`, which enables or disables the self-gravity within solitons. This is done by either solving the full SP system or setting the kick step propagator to be the identity. Next is the `static_potential`, which now supports either a quadratic potential or additional gravitational potential from a point source with specified mass in the middle of the grid. Both of these potentials were used for testing of the framework and will be covered later in this work in Chapter 3.

Simulation Space
Next, the `sim` takes care of setting up the simulation grid. Both `sim.grids` and `sim.dx` are created by a method within `sim`, which takes a list of boundary intervals, defining where each axis should start and end. For example, in a 3D simulation with a grid size of $a + b$, one should provide `boundaries = [[a, b], [a, b], [a, b]]`[2]. The method then unpacks these intervals and creates linearly spaced points between each pair of values using `np.linspace`. Specifically, for each dimension, it creates `N` equally spaced points between the lower and upper bounds.

The spacing between the points in each dimension is calculated as $dx = (b - a)/(N - 1)$. These values are stored in the list `sim.dx`. The multidimensional coordinate grids are then constructed using `np.meshgrid` and stored in `sim.grids`. These grids are the space where solitons are placed. The attribute `k_space` is just these grids transformed to Fourier space.

Storing Solitons
An important side note before continuing: in the next few paragraphs, I will be using two different names for objects that may refer to similar things. In the code, there's a `Wave_vector` class, which represents a soliton with spin. As mentioned in Subsection 1.7.6, this is done by "splitting" the soliton's wave function into components with different coefficients. These will be referred to as wave vectors, since they contain multiple components. The individual components of the wave vector are created using the `Wave_function` class and will be referred to as wave functions.

With that nomenclature in mind, we can address storage of solitons in `sim`, so they can be placed on the grid and evolved. The way these wave vectors are stored is a bit peculiar, which is why both a `wave_vectors` dictionary and a `wave_functions` list are used.

The simulation allows for adding multiple wave vectors, either with the same spin or different ones, to allow for simulations of more complex systems. If two or more wave vectors with the same spin are added to `sim`, they are stored in the dictionary under the same key (their spin). In this case, their corresponding components are summed together (e.g., the first component of soliton one + the first component of soliton two) and then each of these wave functions is evolved as a single, combined wave function.

On the other hand, wave vectors with different spins are stored under separate keys in the dictionary, so they aren't mixed together. If there are multiple same-spin solitons in one `sim`, their components are summed, otherwise, they are treated as separate wave vectors. Then all wave functions in `wave_vectors` dictionary are collected into the `wave_functions` list. This list allows the simulation to handle each wave function it received individually during evolution. For example, all the first components of spin-1 wave vectors are grouped and evolved together, separately from the first component of a spin-2 wave vector.

[2]The simulation does support different grid sizes in each dimension, so in the most general case one should declare boundaries as: `boundaries = [[a, b], [c, d], [e, f]]`.

```python
class Simulation_Class:

    @parameter_check(int, list, int, (int, float), (int, float), int, float, bool, object, bool, dict,bool,bool,float)
    def __init__(self, dim, boundaries, N, total_time, h,order_of_evolution = 2, m_s=2.5e-22, use_gravity=False,
                 static_potential=None, save_max_vals=False,
                 sim_units={"dUnits": "kpc", "tUnits": "Gyr", "mUnits": "Msun", "eUnits": "eV"},use_units=True,self_int=True,a_s=-10e-80,):
        # Setup parameters
        self.dim = dim
        self.boundaries = boundaries
        self.N = N
        self.total_time = total_time
        self.h = h
        self.num_steps = int(self.total_time / self.h)
        self.order_of_evolution = order_of_evolution

        if order_of_evolution not in (2, 4, 6):
            raise ValueError("order_of_evolution must be either 2, 4, 6. Raised while initializing Simulation_Class")

        # Gravity and potential settings
        self.use_gravity = use_gravity
        self.static_potential = static_potential

        # Initialize spatial grids
        self.dx = []
        self.grids = []
        self.dx, self.grids = self.unpack_boundaries()

        # Initialize k-space for Fourier methods
        self.k_space = self.create_k_space()

        # Initialize wave functions storage
        self.wave_vectors = {}
        self.wave_functions = []  # List to store Wave_function instances

        # Setup physical units
        self.setup_units(sim_units, m_s)

        self.use_self_int =self_int
        self.a_s = a_s


```

Listing 1: Initialization of the Simulation class

Physical Units
For creating a physically meaningful simulation, it is necessary to use physical units, which should be passed in as a dictionary or not at all to continue with the default ones. Default units used are kpc, Gyr, eV, and $M_\odot$. To implement them, a Python library named `astropy` is used. In these units `sim` then calculates $\tilde{\hbar}$ which is used to rescale required quantities as was described in Subsection 1.7.5. Connected to this is the mass of the scalar field by default set to $2.5{\cdot}10^{22}$ eV/c^2, but can be changed using the parameter `m_s`.

Self-interaction
Final option for the simulation is to enable or disabled self-interaction using a boolean named `use_self_int` with a parameter called `a_s`. This is used to add self-interaction to the underlying scalar field from which solitons emerge. Prescription for this self-interaction was taken from [43], where `a_s` is the "strength" coefficient of the self-interaction.

Further Functionality

After initialization sets up the simulation grid, one can populate it with wave vectors and then use the `sim.evolve()` method to begin the time evolution. This is the main functionality of the `Simulation_class` within the code. I will now describe how the creation of wave vectors works.

2.2.2 Packet Class

Although for soliton creation one uses `Wave_vector_class`, this one uses the `Wave_function` to create its components, and the numerical values which are then used to represent each wave function are created by the `Packet` class. So, for soliton creation in this simulation framework, the user only ever interacts with `Wave_vector_class`, but to understand its functionality, we first need to discuss two "helper classes".

As was mentioned in the Subsection 2.1.2 `Wave_vector_class` requires the `Simulation_Class` instance to be passed in and the `packet_name` parameter. These are then passed into the creator of `Wave_function` and from there to the `Packet` class. This is so that the `Packet` has access to the `grids` attribute mentioned in Section 2.2.1. This is so that `Packet` can create a grid of the same dimensions populated with values. These depend on the `packet_type`. The type of packet created can be chosen between three options: Gaussian packet `packet_type = "gaussian"`, linear harmonic packet `packet_type = "LHO"`, and a file with values for soliton in spherical coordinates `packet_type = "path to the data file"`. An example of how the Gaussian packet type is created can be found in Listing 2.

The Gaussian packet is created using the means and standard deviations inputted by the user in the `Wave_vector_class` constructor. These should be given as lists, with a length equal to the number of simulation dimensions. For example, in a 3D simulation: `means = [mean1, mean2, mean3]`. The `st_deviations` should be given to the constructor in a similar manner. For each spatial dimension, a one-dimensional Gaussian is generated on the corresponding grid axis using the given mean and standard deviation. These 1D Gaussians are then multiplied together to form the full multidimensional wave packet.

```python
def _create_gaussian_packet(self):
        """
        Creates a Gaussian wave packet based on the provided means and standard deviations.
        Returns:    np.ndarray: Gaussian wavefunction.
        """
        means_arr = np.array(self.means)
        st_deviations_arr = np.array(self.st_deviations)
        psi_0 = np.ones_like(self.grids[0])
        for i in range(len(self.grids)):
            psi_0 *= gaussian_packet(self.grids[i], means_arr[i], st_deviations_arr[i])

        dx_total = np.prod(np.array(self.dx))
        return psi_0 + 0j

def gaussian_packet(x, x_0, sigma_0):
    """Function which returns gaussian packet at position x.
    params are x, x_0 - mean, sigma_0 - standard deviation"""
    return cp.exp(-(x - x_0) ** 2 / (2 * sigma_0 ** 2)) / (cp.sqrt(2 * cp.pi) * sigma_0)

```

Listing 2: Function for generating an initial Gaussian wave packet over a multidimensional grid defined by `Simulation_Class.grids`, using passed-in means and standard deviations.

A linear harmonic oscillator packet is constructed in a manner similar to that described above. Creating a wave packet from a soliton file is more complicated and involves extrapolating the spherical soliton data onto the Cartesian grid. The code is lengthy and will not be covered in more detail here. The most important part for the user is that they can find the path to the soliton packages in the GitHub repository under `resources/solitons`. The file itself should have two columns: the first one named r, representing the radial coordinate, and the second one *phi*, which contains the values to be extracted.

The last functionality of the `Packet` class is to compute the soliton "mass" from a given file. It reads the data from the file and numerically integrates the spherically symmetric values ϕ.

First, values for r and ϕ are extracted, and then the density is calculated as $\rho(r) = |\phi(r)|^2$, and since the data is spherical symmetry, the total mass of the soliton is given by:

$$M = 4\pi \int_0^\infty \rho(r)\, r^2\, dr = 4\pi \int_0^\infty |\phi(r)|^2\, r^2\, dr \tag{2.1}$$

This integral is computed numerically using the trapezoidal rule on the discrete data set provided in the file [69]. Usage of this method will now be discussed in the `Wave_function` class.

```
def rescale_psi_to_phys_units(self):
        # Converts wave function: _sol = ^_sol * (ħ/√G)
        self.conversion_factor = self.simulation.h_bar_tilde /
        ↪  np.sqrt(self.simulation.G)
        self.psi *= self.conversion_factor
def calculate_density(self):
        return cp.abs(self.multiplicity*self.psi).astype(cp.float32) ** 2

def _rescale_psi_to_new_scale_based_on_mass(self):
        data = self.packet_creator.read_ground_state_data(self.packet_type)
        self.soliton_mass = self.calculate_soliton_mass_from_spherical_data()
        self.scaling_lambda = self.desired_soliton_mass / self.soliton_mass

        r_values = data[:, 0]
        r_values /= self.scaling_lambda
        print(max(r_values))
        phi_values = data[:, 1]
        phi_values *=  self.simulation.h_bar_tilde / np.sqrt(self.simulation.G)
        phi_values *= self.scaling_lambda**2
        interp_phi = interp1d(r_values, phi_values, kind='linear',
                              bounds_error=False, fill_value=0.0)

        r_distance = np.zeros_like(self.grids[0])
        for dim in range(self.dim):
            r_distance += (self.grids[dim] - self.means[dim]) ** 2
        r_distance = np.sqrt(r_distance)
        psi = interp_phi(r_distance)

        return psi.astype(np.complex64)
```

Listing 3: Key methods for the functionality of `Wave_function` class within the code.

2.2.3 Wave function Class

This class represents the components of the wave vector and contains the numerical values representing the soliton on a grid. Once a `Wave_vector_class` is initialized, it creates an instance of `Wave_function` as its `wave_blueprint` from which all the components of the wave vector are created. Meaning that the `Wave_function` represents a spin-0 soliton. Since this class contains the data grid representing the components of a soliton, it is used for rescaling the scalar field to physical units as was mentioned in Subsection 1.7.5. This part of the code is shown with other key parts for `Wave_function` class in Listing 3. The `rescale_psi_to_phys_units` takes the grid with values representing the wave, which are saved in its `.psi` attribute and rescales it by a conversion factor which is calculated using constants saved in the `simulation_Class` instance, which are expressed in the desired units.

The next functionality of this class is the computation of the density, which is later used for solving the SP system in `evolution_class`. This is simply done by taking the absolute square value of the `.psi` attribute. Since some components of the wave vector may be unnecessary to compute, as was explained in Subsection 1.7.6, the density is multiplied by the multiplicity of the component to compensate for the reduction in the number of wave functions used.

Another important function the `Wave_function` has is the ability to rescale the soliton to a desired mass using the scaling transformation relations shown in Equation 1.33. This not only allows to set the desired soliton mass and passing it into a `Wave_vector_class` constructor as attribute `desired_soliton_mass` but also rescales the size of the soliton.

Rescaling is done using the `Packet.compute_original_soliton_mass` method, which calculated the mass of the soliton in the file as was explained in the Subsection talking about the `Packet` class and then comparing it to the desired soliton mass set by the user. This fraction is then used to rescale the data, as can be seen in Listing 3.

2.2.4 Wave Vector Class

To finalize a part concerned with soliton creation within the simulation framework, we have to talk in more detail about the `Wave_vector_class`. This is the class the user will interact with to create solitons for their simulation. During its initialization, which was discussed in Subsection 2.1.2, the class passes received attributes into a wave function creator, which uses the `Packet` class to create a `Wave_function` class instance that is saved as `wave_blueprint`. This blueprint will then be used to create all of the wave vector components as was described in Subsection 1.7.6. The process of creation of wave vector using the KG coefficients is quite complex and employs methods from various Python libraries such as `itertools` or `sympy`.

This construction is implemented in the method `generate_spin_basis`. It relies on a recursive helper function named `build`, which iterates over all possible sequences of spin projections $(m_1, m_2, \ldots, m_s)$ that satisfy the constraint:

$$m_1 + m_2 + \cdots + m_s = m$$

for a given total magnetic quantum number $m \in \{-s, \ldots, +s\}$.

Each valid sequence corresponds to a tensor product:

$$\mathbf{e}_{m_1} \otimes \mathbf{e}_{m_2} \otimes \cdots \otimes \mathbf{e}_{m_s}$$

which contributes to the overall spin eigenstate $|s, m\rangle$. This mirrors the formal angular momentum coupling procedure in quantum mechanics, where Clebsch–Gordan coefficients determine how elementary spin-0 states combine to form a total spin s explained in Subsection 1.7.6.

In the code, provided in Listing 5, this tensor construction goes as follows:

1. For each valid tuple of spin projections $m_s = (m_1, m_2, \ldots, m_s)$, the corresponding basis vectors are gathered in a list:

   ```
   vecs = [e_m[mi] for mi in ms]
   ```

   Best explained by showing an example: for m_s `= (-1, 1)` which corresponds to spin-1, we obtain:

   $$\texttt{vecs} = \left[\frac{1}{\sqrt{2}}(1, -i, 0), \quad \frac{1}{\sqrt{2}}(-1, -i, 0)\right]$$

2. The tensor product is initialized with the first vector:

   ```
   t = vecs[0]
   ```

3. The remaining vectors are combined iteratively using `np.tensordot` with `axes=0`, which computes the outer product:

```
for v in vecs[1:]:
    t = np.tensordot(t, v, axes=0)
```

The shape of the tensor grows with each multiplication:

- (3,) ⊗ (3,) yields (3,3)
- (3,3) ⊗ (3,) yields (3,3,3), and so on

Again showing examples:

- `ms = (-1, 1)` (spin-2): $t = \mathbf{e}_{-1} \otimes \mathbf{e}_{+1}$, shape (3,3)
- `ms = (-1, 0, +1)` (spin-3): $t = \mathbf{e}_{-1} \otimes \mathbf{e}_0 \otimes \mathbf{e}_{+1}$, shape (3,3,3)

4. The resulting tensor is added to an array:

```
tensor[:] += t
```

This step then sums over all valid combinations for a given magnetic quantum number m, ensuring that the resulting tensor correctly represents the quantum superposition corresponding to $|s, m\rangle$.

Example: for spin-2, $m = 0$, the following combinations contribute:

- `(-1, +1)`: $\mathbf{e}_{-1} \otimes \mathbf{e}_{+1}$
- `(+1, -1)`: $\mathbf{e}_{+1} \otimes \mathbf{e}_{-1}$
- `(0, 0)`: $\mathbf{e}_0 \otimes \mathbf{e}_0$

giving:

$$\texttt{tensor} = \mathbf{e}_{-1} \otimes \mathbf{e}_{+1} + \mathbf{e}_{+1} \otimes \mathbf{e}_{-1} + \mathbf{e}_0 \otimes \mathbf{e}_0$$

This process creates a properly normalized tensor for each m, and repeating this process across all $m \in \{-s, \ldots, +s\}$ generates the complete set of $2s + 1$ spin basis tensors.

Once the full spin tensor is constructed, the method `_create_combined_wave_function` selects the individual tensor components corresponding to specific spatial directions. This is done using combinations with replacement of (range(3), spin), which produces all unique symmetric index combinations, e.g.:

- Spin-1: (0), (1), (2)
- Spin-2: (0,0), (0,1), (1,1), etc.

The function `get_index_multiplicities` computes the number of equivalent arrangements. This is used to reduce the necessary number of wave functions in the system's memory using the KG coefficients symmetry, as was mentioned in Subsection 1.7.6. The corresponding wave function is then normalized by dividing by the square root of this multiplicity:

$$\psi \mapsto \frac{\psi}{\sqrt{N}}$$

This normalization ensures that the field obeys the correct symmetry and normalization properties expected of a spin-s bosonic field. Using this method to create a wave vector leaves us with an object that has an attribute containing a list of all necessary components to represent a soliton with spin s. It is during the creation of the wave vector in method `_create_combined_wave_function` that the momentum is applied to all wave functions using the `Packet.momentum_propagator`, which is computed from the inputted momentum and applied to the wave values as

$$e^{i\frac{px}{\hbar}}\psi.$$

In the code, this is done as shown in Listing 4. Thanks to this algorithm, it is possible to create a soliton with any integer spin, $s \in \mathbb{N}_0$.

```
def compute_momentum_propagator(self):
        """Compute the kinetic propagator based on Fourier space components."""
        momenta = [
            1j * cp.array((momentum / self.h_bar_tilde) * grid, dtype=cp.float32)
            for momentum, grid in zip(self.momenta, self.grids)]
        summed_momenta = cp.zeros_like(momenta[0])
        for momentum in momenta:
            summed_momenta += momentum


        return cp.exp(summed_momenta)
```

Listing 4: Computation of the momentum propagator within `Packet` class used to give wave functions momentum.

2.2.5 Evolution Class

After finally creating a soliton and adding it into a simulation, the evolution can begin after a quick check of the selected time step `h`. This check verifies if the time step satisfies the constraint in Equation 1.66. If yes, the simulation will continue if not, the user has a chance to stop the simulation. After tha,t the simulation passes the saved wave vectors into the `Evolution_Class` as a list of wave functions.

In the main event loop, the simulation shown on Listing 6 the code goes one step at a time going from time $t = 0$ to time $t =$ `total_time` computing all the necessary quantities. First thing is to compute the total density of the system so it can be used to solve the SP system of equations. This is done by simply looping over all available wave functions. After that, a `total_energy` at a given step is set to 0 and later computed.

```python
def generate_spin_basis(self):
    if self.spin == 0:
        return [np.array(1.0)]
    e_m = {
        -1: np.array([1, -1j, 0]) / np.sqrt(2), 0: np.array([0, 0, 1]), 1:
        ↪  np.array([-1, -1j, 0]) / np.sqrt(2)    }
    basis = []
    for m in range(-self.spin, self.spin + 1):
        tensor = np.zeros([3] * self.spin, dtype=complex)
        def build(ms=(), total=0):
            if len(ms) == self.spin:
                if total != m:
                    return
                vecs = [e_m[mi] for mi in ms]
                t = vecs[0]
                for v in vecs[1:]:
                    t = np.tensordot(t, v, axes=0)
                tensor[:] += t
            else:
                for mi in [-1, 0, 1]:
                    build(ms + (mi,), total + mi)
        build()
        norm = np.linalg.norm(tensor)
        basis.append(tensor / norm if norm > 0 else tensor)
    return basis

def _create_combined_wave_function(self):
    result = []
    full_tensor = sum(
        coeff * np.exp(-1j * phase) * basis
        for coeff, phase, basis in zip(
            self.polarization_coefficients,
            self.polarization_phases,
            self.polarization_bases)   )
    for idx in self.index_combinations:
        value = full_tensor[idx]
        raw_psi = self.wave_blueprint.psi * value *
        ↪  self.wave_blueprint.packet_creator.momentum_propagator
        multiplicity = self.index_multiplicities[idx]
        psi = raw_psi / np.sqrt(multiplicity)
        new_wf = self.wave_blueprint.softcopy_psi(psi, multiplicity=multiplicity)
        result.append(new_wf)
    return result

def get_index_multiplicities(index_combinations):
    multiplicities = {}
    for idx in index_combinations:
        perms = set(permutations(idx))
        multiplicities[idx] = len(perms)
    return multiplicities
```

Listing 5: Methods used to create wave vector using the `wave_blueprint` and CG coefficients, including a method that computes the multiplicities of reduced symmetric tensor components.

```
1
2  for step in range(self.num_steps):
3          total_density = self._compute_total_density(wave_functions)
4          self.total_energy = 0
5          save_step=False
6          current_time = step * self.h
7          self.compute_kinetic_energy(wave_functions)
8          self._compute_potential_energy(wave_functions, total_density,
           ↪  current_time)
9          # Save snapshots at specified intervals
10         if step % save_every == 0 and step > 0:
11             save_step = True
12         wave_functions =
           ↪  self._perform_evolution_step(wave_functions,total_density, step,
           ↪  save_step)
13
14         if step % save_every == 0 and step > 0:
15             self.compute_radial_density_profile(total_density,current_time)
16             self._save_snapshots(wave_functions, step, save_every)
17         # Memory cleanup
18         cp.get_default_memory_pool().free_all_blocks()
```

Listing 6: Main event loop driving the simulation forward in `Evolution_Class`.

Next, parameters `save_step current_time` are purely for saving purposes and this functionality will not be covered in this work for the sake of brevity. The energy, both the potential energy of the system and the kinetic energy, is computed.

Kinetic Energy

The method `compute_kinetic_energy` calculates the kinetic energy of a system based on the spatial gradients of the wave amplitudes. This is done by computing the expectation value of the kinetic energy operator in position space using:

$$E_{\text{kin}} = \frac{\hbar^2}{2m} \int |\nabla \psi(\vec{x})|^2 \, d^n x, \tag{2.2}$$

where $\psi(\vec{x})$ is the wave function, $\nabla\psi$ is its gradient, and the integral is taken over the space of the system. The norm squared of the gradient, $|\nabla\psi|^2$, represents the sum of the squares of the partial derivatives with respect to spatial coordinates. This is supposed to represent the application of the kinetic energy operator:

$$\hat{T} = -\frac{\hbar^2}{2m}\nabla^2. \tag{2.3}$$

The integration is done by parts under taking the boundary terms to be 0. In the code, this formulation of the calculation of the kinetic energy was adopted because the code is already calculating the gradient of psi at every step to solve the Schrödinger equation. This pre-computation of the gradient can then be used here to reduce the memory demand of the code. This means that the gradient is evaluated in a Fourier space using the identity:

$$\partial_i \psi(\vec{x}) \longleftrightarrow i k_i \tilde{\psi}(\vec{k}),$$

where $\tilde{\psi}(\vec{k})$ is the Fourier transform of the wave function and k_i is the wave number in the i-th spatial direction. The kinetic energy is then obtained by summing the contributions from all spatial components and wave functions. This sum is multiplied by appropriate constants to reflect the total kinetic energy of the system.

Potential Energy

Following the calculation of kinetic energy, the method `_compute_potential_energy` calculates the potential energy of the system. The code accomplishes this by integrating the product of the external potential $V(\vec{x})$ and the total particle density $\rho(\vec{x})$ over space. This corresponds to the standard quantum mechanical expression for potential energy:

$$W = \int V(\vec{x}) \cdot \rho(\vec{x})\, d^n x, \tag{2.4}$$

where $\rho(\vec{x}) = \mathrm{Tr}[\psi^\dagger(\vec{x})\psi(\vec{x})]$ is the total density of the wave function components at each point in space. In the current implementation, the only kind of potential taken into account in the calculation stems from the gravitational potential between solitons. Computation which would include other external potential $V_{\mathrm{ext}}(\vec{x})$ is not yet implemented, effectively assuming $V_{\mathrm{ext}}(\vec{x}) = 1$[3].

The integration is again done by summing over all the densities on the grid and multiplying this sum by the volume element dx. The volume element is given by the product of the grid spacings along each dimension.

Once both kinetic and potential energies are computed, the total energy E is calculated as their sum:

$$E = K + W,$$

and the ratio $W/|E|$ is also computed for later analysis[4]. These energy components are then stored, along with the current simulation time.

Evolution

After computing the energies, the evolutionary step itself is performed. This is done using the helper `_perform_evolution_step` method, which decides what order of evolutionary method should be used, these were mentioned in Section 1.7.3. By default, the order of simulation is chosen to be 2, so that is the one I will focus on here. This helper method calls on one of the evolutionary methods depending on the order chosen. These look similar to the one for order 2 shown in the Listing 7.

This method once again uses helper methods to perform the kick and drift steps on all available wave functions. The only difference from the algorithm described in Subsection 1.7.3 is that, as can be seen in Equation 1.57, the kick steps carry a prefactor of $\frac{1}{2}$ in the exponential. Since we apply a full kick-drift-kick cycle in one step, followed by another in the next, it is possible to start the simulation with a half-kick, then perform a full drift, and from then on alternate between "doubled" kicks and drifts. This technique reduces the number of operations required per step by only applying the $\frac{1}{2}$ prefactor during the first and last kicks of the simulation, where it is necessary.

What actually happens in the `_kick_all_wave_functions` and `_drift_all_wave_functions` will be covered within the Section dedicated to `Propagator_Class`.

[3] One of the many things the author aims to improve down the line.

[4] Later analysis as in the next work.

```python
def _evolve_order_2(self, wave_functions,total_density ,is_first,
↪  is_last,save_step):
        """Second-order split-step evolution."""

        # Kick step
        self._kick_all_wave_functions(wave_functions, total_density, is_first,
        ↪  is_last)
        # Drift step
        self._drift_all_wave_functions(wave_functions)

        # Final kick for last step
        if is_last:
            total_density = self._compute_total_density(wave_functions)
            self._kick_all_wave_functions(wave_functions, total_density, is_first,
            ↪  is_last)

        return wave_functions
```

Listing 7: Method within `Evolution_Class` which handles the order 2 split step method

Saving Data

Other than the evolution, this class also handles the saving functionality within the code. Every number of steps selected by the user while calling the `Simulation_Class.evolve(save_every)` method, the code takes a snapshot of what is happening within the simulation. Furthermore the code saves the computed energy values and maximal values within the simulation at a given step. This data is then accessible within the *resources/data* directory.

The last interesting functionality of `Evolution_Class` is the computation of the radial density profile. Aim of this is to reproduce a similar plot to Figure 1.1, which could be fitted with the NFW profile shown in Equation 1.37. Outcomes of this functionality will be provided in the Chapter 4 dedicated to results.

2.2.6 Propagator Class

The `Propagator` class is a supporting class responsible for computing the exponential propagators used in the split step evolution of the wave functions shown in Equation 1.57.
These propagators are used within the methods `_kick_all_wave_functions` and `_drift_all_wave_functions`, mentioned in the previous subsection, which loop over all provided wave functions and apply either the kinetic or potential propagator to perform one full evolution step.

To compute these propagators, the methods shown in Listings 9 and 8 are called. At each time step, the system evolves according to the Schrödinger–Poisson system, with spatial derivatives handled in Fourier space to solve the differential equations.

The potential propagator is composed of two parts, a gravitational component and an optional static external potential. The gravitational part is obtained by solving the Poisson equation from the `total_density` and simulates gravitational attraction between the solitons.
If self-interactions are enabled, this part also calculates an additional potential derived from the density simulating the physics of SIFDM. The optional static component allows for the

inclusion of external potentials such as a quadratic potential.

Both kinetic and potential propagators take the general exponential form:

$$e^{-i\frac{t}{2\tilde{\hbar}}\cdot\Phi} \quad \text{or} \quad e^{-i\frac{t}{2\tilde{\hbar}}\cdot k^2},$$

where Φ represents the total potential (gravitational and possibly self-interaction), and k^2 corresponds to the squared wave number in Fourier space. These exponential factors emerge directly from the time evolution operator applied to the respective parts of the Hamiltonian, as was discussed in Section 1.7.

The gravitational part is computed by the `compute_gravity_propagator` method, shown in Listing 9. It first checks whether gravity is enabled, and if so, solves the Poisson equation from the density field. If self-interactions are also active, they add the corresponding self-interaction potential. The total potential is then used to construct the exponential propagator.

The kinetic propagator, implemented in Listing 8, is constructed from the Fourier representation of the Laplacian operator. Specifically, the squared wave number vector k^2 is computed and used to solve the differential equation. This is then used to construct the exponential for the kinetic propagator.

```
def compute_kinetic_propagator(self,time_factor=1):
        # Calculate k_squared_sum for the Laplacian operator
        k_squared_sum = cp.zeros_like(self.k_space[0], dtype=cp.float32)
        for k in self.k_space:
            k_squared_sum += k ** 2

        self.kinetic_propagator = cp.exp(((-1j * ((self.h*time_factor) / 2) *
        ↪  k_squared_sum )*(self.h_bar_tilde)), dtype=cp.complex64)
        return self.kinetic_propagator
```

Listing 8: Computation of kinetic propagator by solving the Schrödinger equation. Kinetic propagator is used by `Evolution_Class` in Equation 1.57 to evolve the system.

```python
def compute_gravity_propagator(self, psi, density,first_step=False,
    last_step=False,time_factor=1):
        if not self.simulation.use_gravity:
            return cp.ones_like(psi, dtype=cp.complex64)

        a_s = (self.simulation.a_s *
        units.cm).to(f"{self.simulation.dUnits}").value

        if not self.simulation.use_self_int:
            self_int_potential = cp.ones_like(psi, dtype=cp.complex64)
        elif self.simulation.use_self_int:
            self_int_potential = self.get_self_int_potential(density, psi, a_s)

        # Solve Poisson equation for gravitational potential
        gravity_potential = self.solve_poisson(density)
        self.gravity_potential = gravity_potential + self_int_potential

        if first_step or last_step:
            self.gravity_propagator = cp.exp((-1j * ((self.h*time_factor) / 2) *
            self.gravity_potential )/(self.simulation.h_bar_tilde),
            dtype=cp.complex64)
        else:
            self.gravity_propagator = cp.exp((-1j  * (self.h*time_factor) *
            self.gravity_potential)/(self.simulation.h_bar_tilde),
            dtype=cp.complex64)


        return self.gravity_propagator
```

Listing 9: Computation of potential propagator by solving the Poisson equation. Potential propagator is used by `Evolution_Class` in Equation 1.57 to evolve the system.

Chapter 3

Code Validation

To verify the correct functionality of the simulation framework, in detail described in Chapter 2, a number of tests along its development were conducted. These included comparisons of code output with theoretical predictions and tests for numerical stability. In this chapter, I shall discuss how each test was conducted and its results.

3.1 Quantum Harmonic Oscillator

First test was done to verify the correct functionality of the Schrödinger equation solver using the Fourier method explained in Subsection 1.7.1. This test aimed to reproduce the results that can be obtained from the analytical solution of the quantum linear harmonic oscillator in a quadratic potential using the simulation framework. This was done by first computing the analytical solution.

3.1.1 Analytical Solution of Quantum Harmonic Oscillator

[1] The one-dimensional time-independent Schrödinger equation for a particle of mass m in a harmonic potential

$$V(x) = \frac{1}{2}m\omega^2 x^2 \tag{3.1}$$

is given by

$$-\frac{\hbar^2}{2m}\frac{d^2\psi_n(x)}{dx^2} + \frac{1}{2}m\omega^2 x^2 \psi_n(x) = E_n \psi_n(x), \tag{3.2}$$

where ω is the angular frequency of the oscillator, and $n \in \mathbb{N}_0$ is the quantum number indexing the eigenstates.

The normalized eigenfunctions $\psi_n(x)$ are:

$$\psi_n(x) = \left(\frac{1}{\sqrt{2^n n!}}\right)\left(\frac{m\omega}{\pi\hbar}\right)^{1/4} e^{-\frac{m\omega x^2}{2\hbar}} H_n\left(\sqrt{\frac{m\omega}{\hbar}}x\right), \tag{3.3}$$

and the corresponding energy eigenvalues are:

$$E_n = \hbar\omega\left(n + \frac{1}{2}\right). \tag{3.4}$$

[1] Reader may once again want to refer to their favourite quantum mechanics textbook for more information on this topic.

The Hermite polynomials $H_n(\xi)$ were obtained from the Python library called *scipy*. Description of the LHO model here is shown for a one-dimensional case, but can be trivially extended to three dimensions [70].

3.1.2 Test Results

In this validation test, two ground states corresponding to $\psi_0(x)$ from Equation 3.3 were initialized and evolved. One was evolved using the simulation framework, and the other using the analytical solution via multiplication by the time-dependent exponential:

$$\psi_0(x,t) = \psi_0(x)\, e^{-iE_0 t/\hbar}. \tag{3.5}$$

To verify the correctness of the implementation, the real and imaginary parts of both the analytical and simulated wave functions were compared at multiple time steps. As shown in Figure 3.1, the numerical results (dashed lines) produced by the simulation are in excellent agreement with the analytical solution (shown with full lines), confirming the correct implementation of the Fourier-based time evolution method.

Because this solution remains stationary up to a global phase, the modulus $|\psi(x,t)|$ should not change in time. For that, the probability density $|\psi(x,t)|^2$ was plotted and is shown in Figure 3.2 for both the numerical (blue line) and analytical (pink line) solutions. Since it's not changing and the two solutions are the same, the blue line is not visible on the plot due to the lines overlapping. Next, the plot shows the time evolution of the spatial averages of real and imaginary parts of the wave functions. The light green and dark blue curves represent the real and imaginary parts of the numerically evolved wave function, respectively. These are compared with the corresponding analytical values, plotted as light green and dark green scatter points. As expected for the harmonic oscillator, the real and imaginary parts oscillate as a sine function and remain in phase throughout the simulation.

3.2 Stability Testing

The next tests focused on verifying the stability of the code. Thanks to Erick Munive Villa, Ph.D, I have access to data representing a spherically symmetric soliton obtained using the procedure described in Section 2.1.1, which is static under self-gravity, generated by the SP system.

To test the numerical stability of the simulation framework, we tracked the maximal value of the density on the grid during time evolution. This quantity serves as an indicator of numerical errors, as the soliton profile is expected to remain stationary in the ideal case. In practice, due to discretization effects and numerical noise, small oscillations of the peak density around its equilibrium value are expected. These arise primarily from the finite resolution of the simulation grid. As the spatial resolution increases, the discretization should become more accurate. The amplitude of these oscillations should decrease with an increase in resolution, reflecting convergence toward the continuous limit.

3.2.1 Test Results

In Figure 3.3, one can observe the time evolution of the normalized maximal value of the density on the simulation grid. The values are normalized to their initial value so that all plots begin at 1. A clear trend can be seen where the amplitude of oscillations in the maximal density decreases as the spatial resolution N increases. This behaviour is consistent with the expectation that numerical artifacts diminish with finer discretization. The simulations were

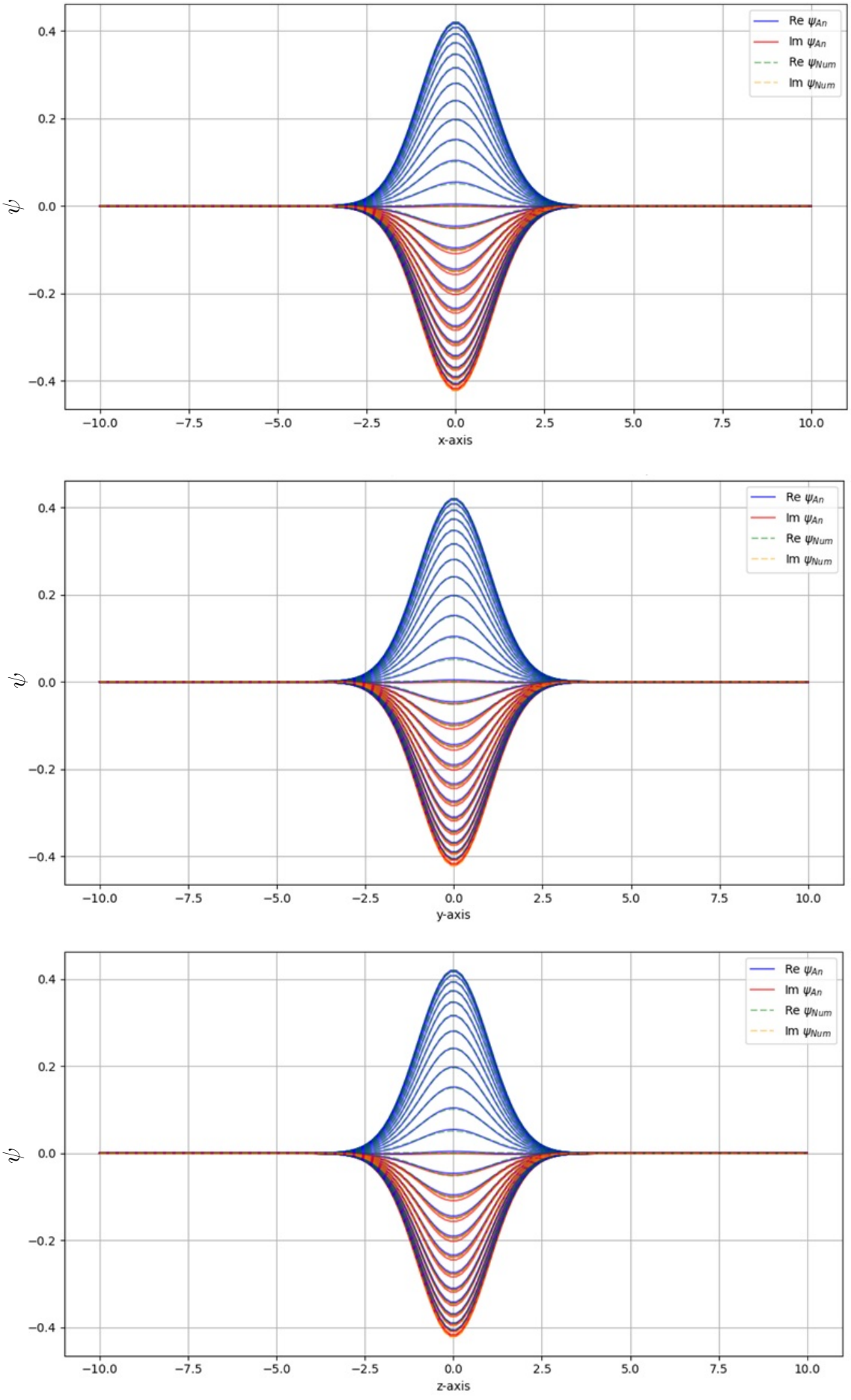

Figure 3.1: Comparison of analytical (full lines) and numerical solutions (dashed lines) of the ground-state wave function of the quantum harmonic oscillator at several time steps in three spatial directions. Both the real and imaginary parts are shown.

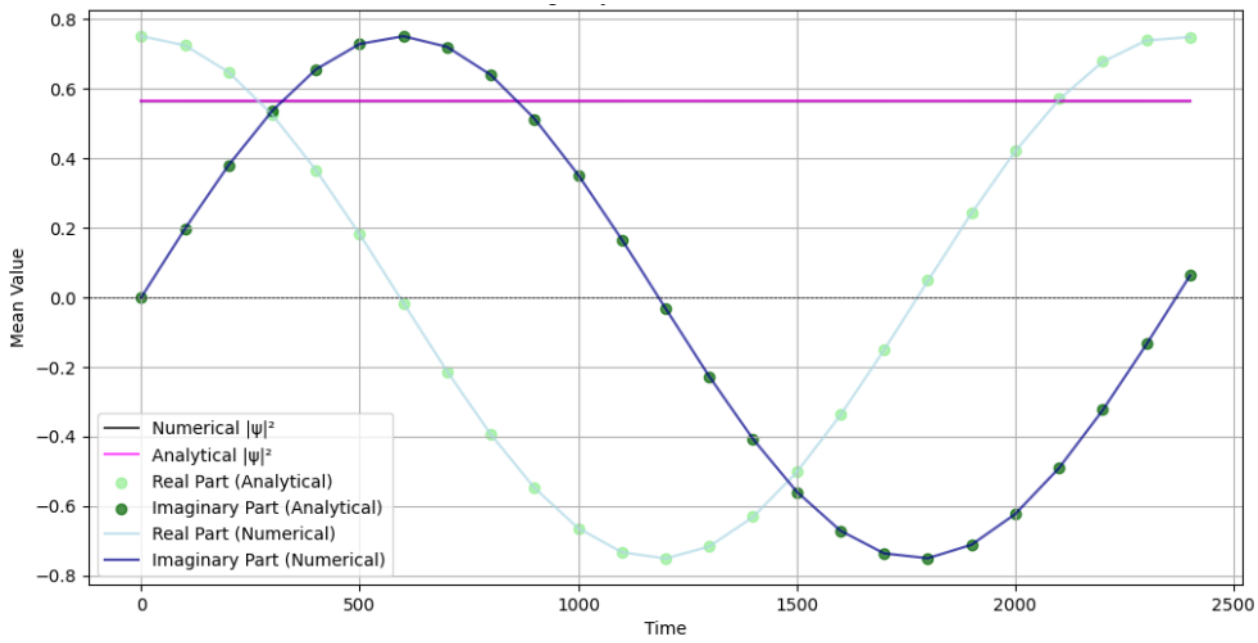

Figure 3.2: Time evolution (shown in number of steps) of the mean values of the real and imaginary parts of the ground-state wave function $\psi_0(x,t)$, with analytical results shown as scatter points. The mean density $|\psi(x,t)|^2$ is shown for both the simulation (solid blue line) and the analytical calculation (solid pink line).

performed with a box size of 20 kpc and a total evolution time of 10 Gyr. Although the time step constraint from Equation 1.66 was not satisfied at the highest resolutions, the simulations remained stable.

The stability of the soliton can also be observed in Figure 3.4, which shows the wave-function probability density $|\psi(x,y,z)|^2$ of a soliton located at (0,0,0) at different times denoted by t in Gyr. The plots display a cross-section through the soliton at $z = 0$ and show that, although the density profile exhibits small oscillations, its overall spatial structure is preserved throughout the simulation, indicating numerical stability over time.

3.3 Movement Testing

To test that the momentum is correctly added to the wave vectors and they behave as expected, two tests were used. Simple one, which consisted of setting a soliton to position (x,y,z), giving it momentum of $(5,0,0)$, and checking if in one unit of time the soliton moved by 5 units of distance. The test yielded the expected results, which will not be included as they are trivial and add no new information, and will be left to the reader's imagination. More interesting tests included testing of the SP system and whether it is solved correctly. This test aimed to reproduce an orbital motion around a massive point-like source of gravity.

3.3.1 Rotational Motion Around a Point like Source

The test was carried out by implementing an external static gravitational source placed at the centre of the simulation grid, and placing a soliton at a radial distance r from the source. A perpendicular velocity was then given to the soliton, replicating a circular orbit around the central mass.

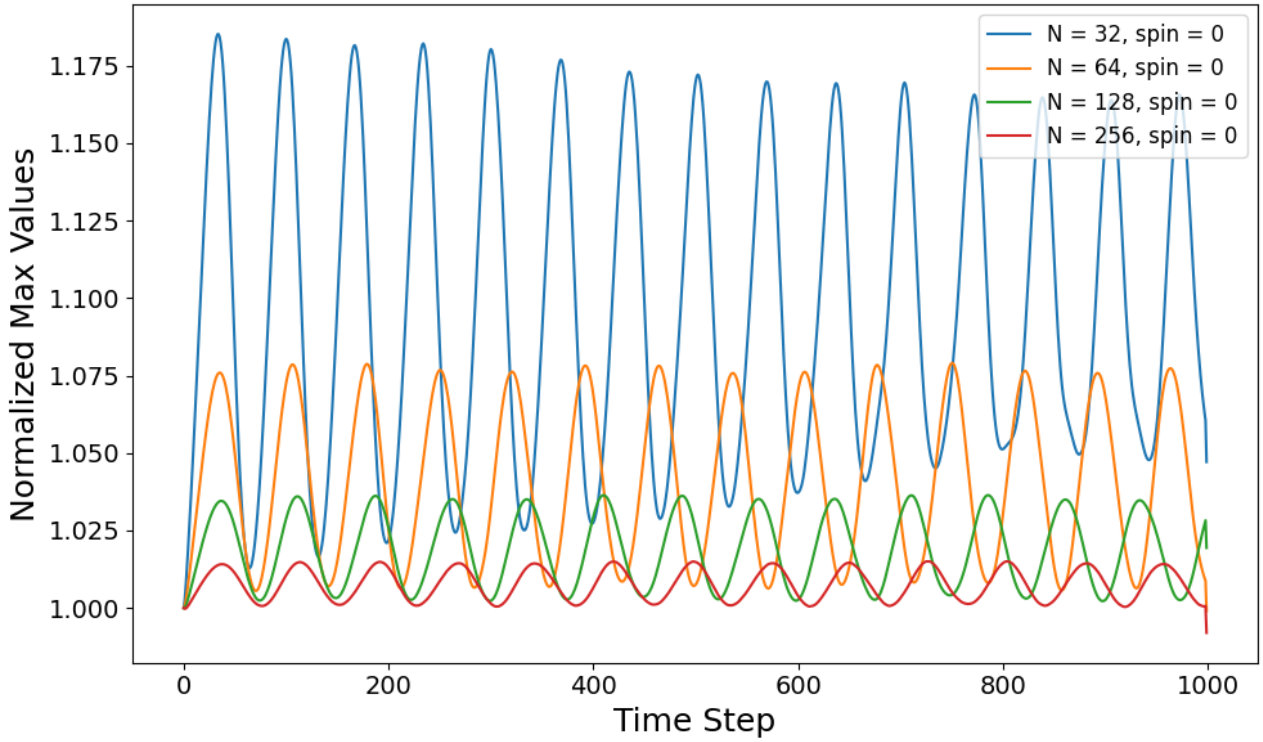

Figure 3.3: Time evolution of the normalized maximal density values on the simulation grid for different resolutions: $N = 32$ (blue), $N = 64$ (orange), $N = 128$ (green), and $N = 256$ (red). All simulations used a spin-0 soliton.

The gravitational potential of a point-like mass M located at the origin is given by:

$$V(r) = -\frac{GM}{r}, \tag{3.6}$$

where G is the gravitational constant and r is the radial distance from the mass.

For a stable circular orbit with radius r, the required orbital velocity is determined from the balance of centripetal and gravitational force:

$$v = \sqrt{\frac{GM}{r}}. \tag{3.7}$$

The corresponding orbital period is then:

$$T = \frac{2\pi r}{v} = 2\pi\sqrt{\frac{r^3}{GM}}. \tag{3.8}$$

We consider a central mass $M = 10^6\ M_\odot$, where $M_\odot$ is the solar mass. Taking the gravitational constant in astrophysical units $G = 4.4985 \times 10^{-6}\ \text{kpc}^3\,\text{M}_\odot{}^{-1}\,\text{Gyr}^{-2}$, and placing the soliton at distance $r = 5$ kpc, we plug into Equation 3.7:

$$v = \sqrt{\frac{4.4985 \times 10^{-6} \times 10^6}{5}} \approx 0.9485\ \text{kpc/Gyr}.$$

Using this velocity in Equation 3.8, the resulting orbital period is:

$$T \approx \frac{2\pi \cdot 5}{0.9485} \approx 33.1\ \text{Gyr}.$$

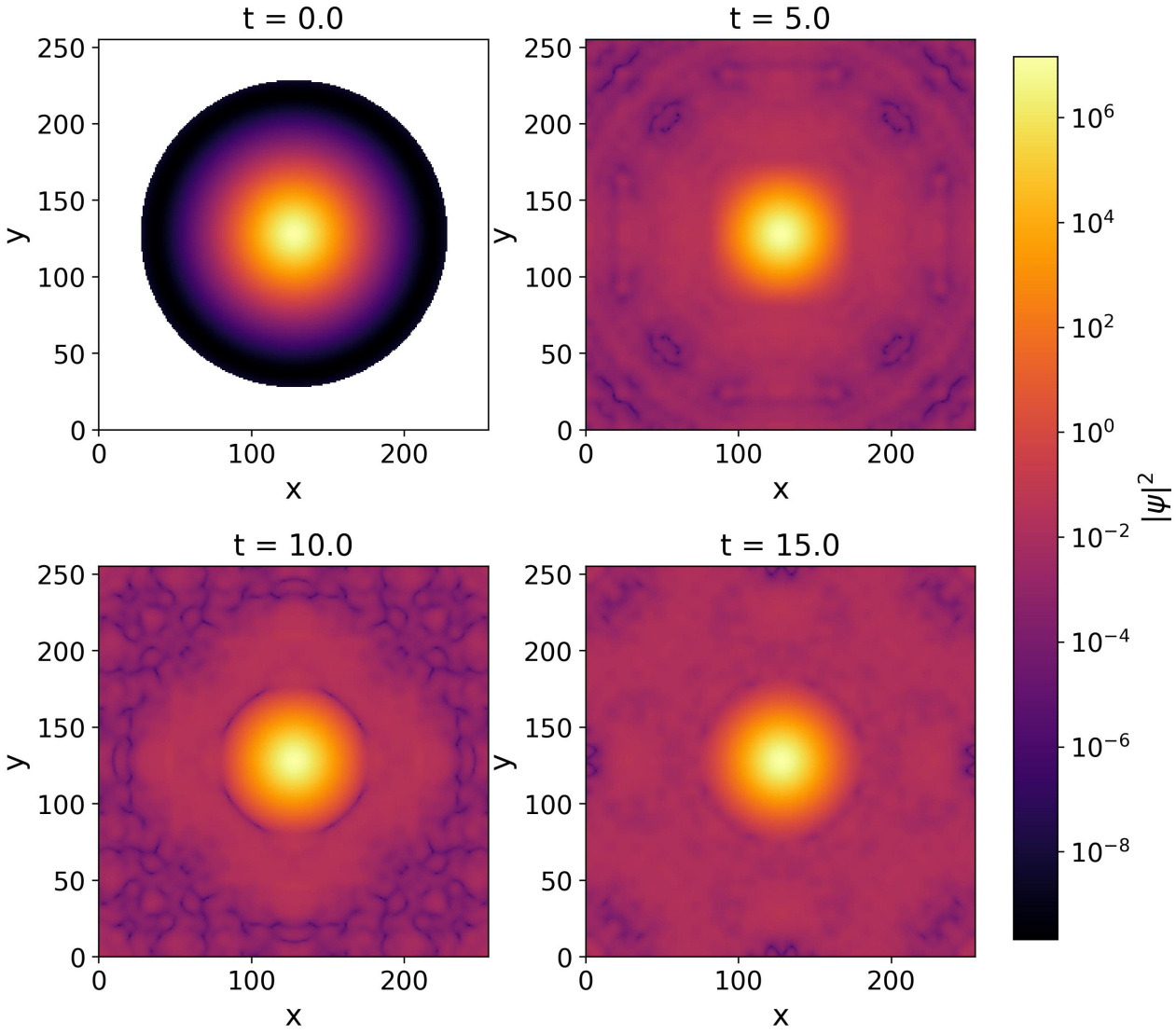

Figure 3.4: Cross-section plots of the wave-function probability density $|\psi(x, y, z)|^2$ in the x–y plane at different times for a simulation with `N = 256` during the evolution of a static soliton. While the soliton exhibits small oscillations in its density profile, its overall structure is preserved over 15 Gyr of evolution (15000 time steps), indicating numerical stability of the solution.

3.3.2 Test Results

This velocity was given to the soliton in the simulation, and its trajectory was tracked to confirm that it follows a circular orbit. The results of this test can be seen on Figure 3.5.

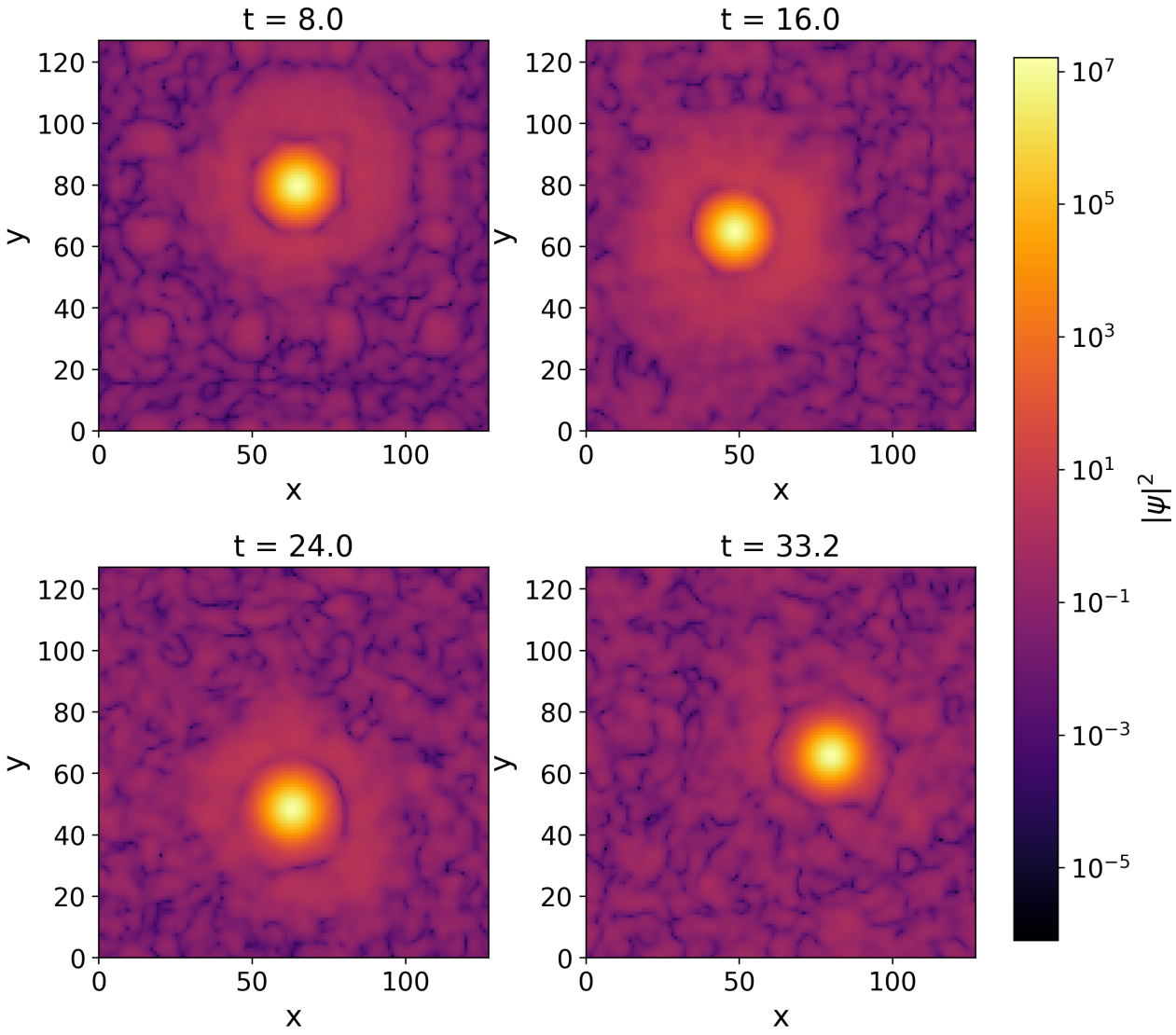

Figure 3.5: Circular motion of a soliton initialized at point $(5, 0, 0)$ around a point like source located at $(0, 0, 0)$ with mass $10^6\ M_\odot$. The sequence of snapshots shows the soliton's probability density at various time steps during the orbital motion.

For better visualization, a plot showing the evolution of the location of the maximum of the density is shown in Figure 3.6. As expected, with given `momenta = [0,0.9485,0]`, the soliton moves in the x-y plane and reaches the same point after approximately 33.1 Gyrs[2].

3.4 Energy Conservation

The final set of tests focused on verifying energy conservation within the simulation. For this purpose, a simulation was performed with 25 solitons on a grid of size $N = 100$. This number was chosen to be consistent with the setup used in [60], where it was shown that simulations

[2]Any discrepancies between the result and the theoretical expectation are due to rounding and finite resolution effects.

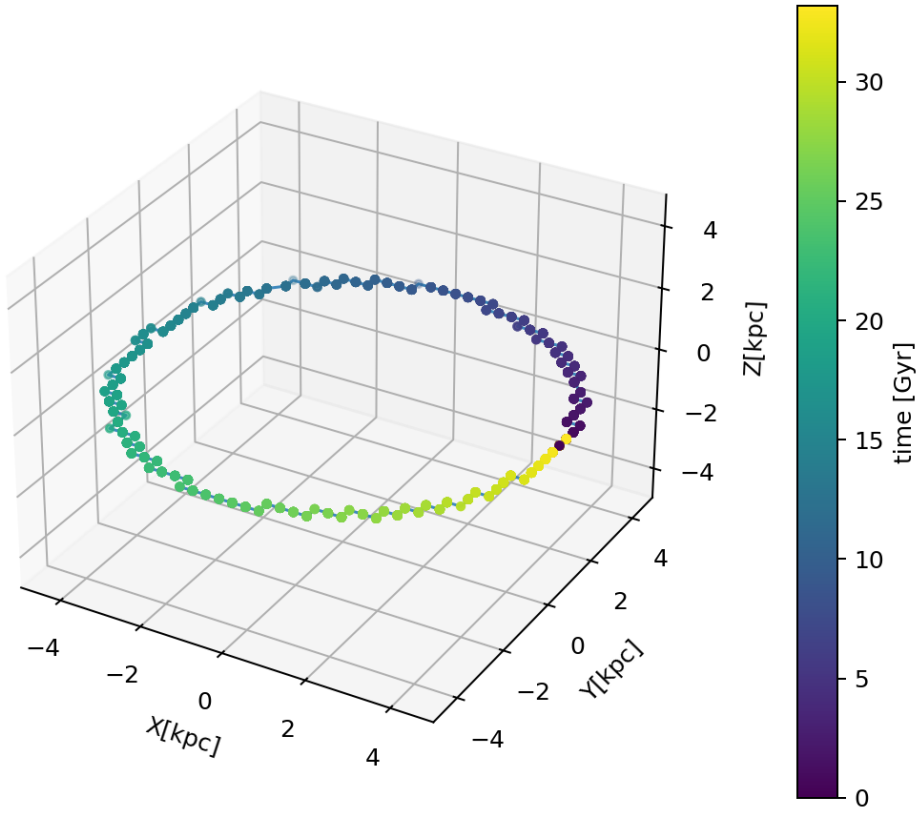

Figure 3.6: Plot showing the trajectory of a soliton initialized with `momenta = [0,0.9485,0]` at point $(5, 0, 0)$ in time [Gyr]. The simulation was set up in a way so that the soliton performs a circular orbit around a massive object located at point (0,0,0).

with fewer than 5 initial solitons require simulations of longer times than 30 Gyr, making them computationally expensive. On the other hand, using more than approximately 120 solitons exceeds the spatial resolution and box size that we are using in this work. This sets the allowed range for the number of solitons to $5 \leq N_{\text{sol}} \leq 120$. Our choice of 25 solitons, therefore, represents a practical compromise between having a sufficiently large number of solitons and keeping the computational cost manageable.

For these tests, the total energy of the system was recorded over time by using the processes described in Subsection 2.2.5 for its computation. The evolution of the total energy was then visualized by plotting the normalized deviation of the total energy from its initial value. Specifically, the plots are showing:

$$\frac{\Delta E}{E_0} = \frac{E_t - E_0}{E_0}, \tag{3.9}$$

on their y-axis. Where E_t is the total energy at a time step t, and E_0 is the initial total energy at $t = 0$.

3.4.1 Test Results

The primary goal of this analysis was to determine whether the total energy is conserved during the simulation, and if so, to what degree. A secondary objective was then to identify which simulation parameters have the greatest impact on the accuracy of energy conservation. Based on the results shown in [60], I expected spatial resolution to be the dominant factor. As shown in Figure 3.7, increasing the grid resolution from $N = 128$ to $N = 512$ significantly reduces the amplitude of energy fluctuations.

However, resolution did not prove to be the most critical factor. The most crucial parameter turned out to be the time step `h`, as can be seen in Figure 3.8. Here, simulations with identical setups but different time steps show that smaller values of `h` result in better energy conservation. Even when both values selected for `h` shown in Figure 3.8 satisfy the Δt constraint from Equation 1.66, there is a clear change in stability favouring smaller time steps. While yes, reducing the time step by a factor of 10 drastically improves the results, it also increases the total number of steps by the same factor. Therefore, it is almost unusable in practice due to high computational cost.

Fortunately, this issue can be bypassed by using higher-order split step methods, which were described in Section 1.7.3. Employing a fourth-order split step method allows for improved energy conservation without reducing the time step, therefore keeping the same number of iterations. This is shown in Figure 3.9, where we can see the energy deviations for simulations with resolutions `N = 128` and `N = 256` with different time steps: `h = 0.001` (blue, using order two method) and `h = 0.01` (green, using fourth-order evolution). In practice, using the fourth-order method achieves a comparable reduction in energy deviation as increasing the resolution from $N = 256$ to $N = 512$, or reducing the time step by an order of magnitude while remaining computationally affordable.

However, it is worth noting that the energy deviation in the fourth-order run (orange) exhibits a monotonic upward split-step time. This is unexpected because, while higher-order methods improve local accuracy, they do not guarantee exact global energy conservation, particularly when using nonlinear or self-gravitating systems as we are. The split step Fourier method assumes a factorization of the time-evolution operator:

$$e^{-iH\Delta t} \approx \prod_j e^{-iT_j\Delta t} e^{-iV_j\Delta t},$$

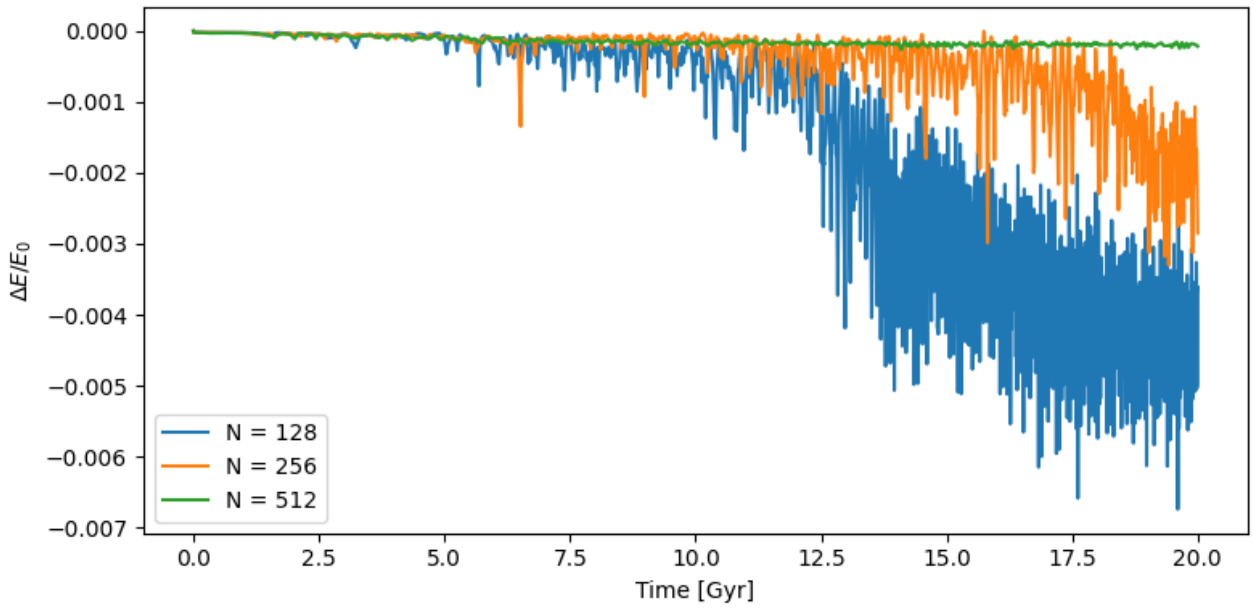

Figure 3.7: Normalized energy deviation $\Delta E/E_0$ for simulations with grid resolutions `N = {128,256,512}`. The setup for all simulations included a box size of 100, 25 solitons with spin 1, and a time step of `h = 0.001`.

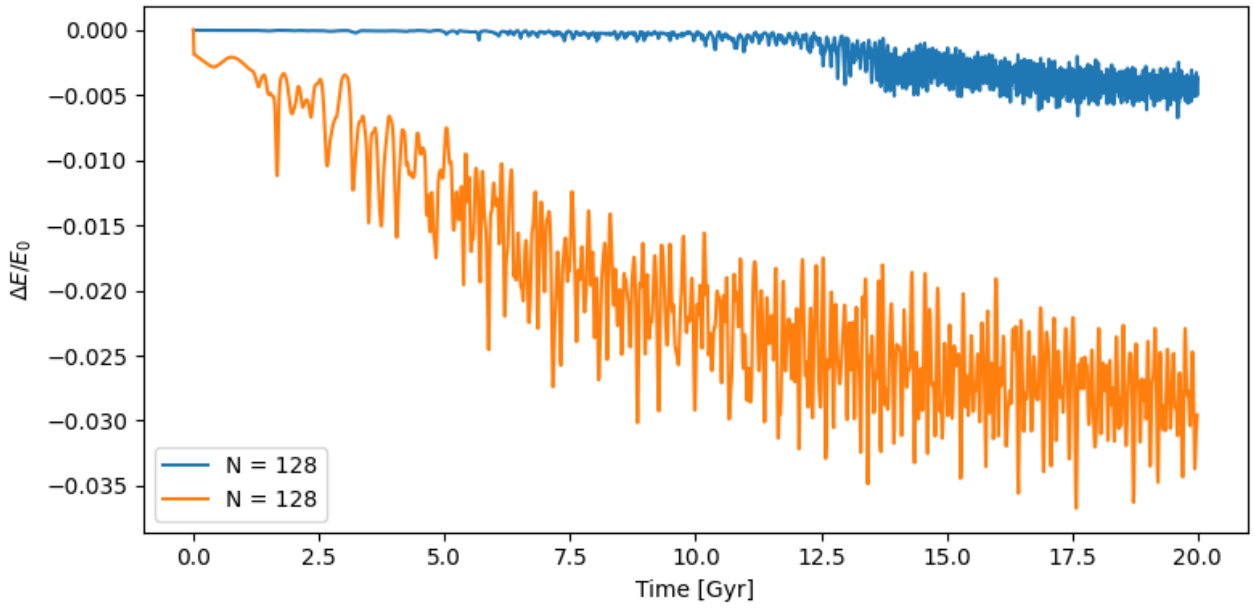

Figure 3.8: Normalized energy deviation $\Delta E/E_0$ for simulations with resolution `N = 128` and different time steps `h = 0.001` (blue) and `h = 0.01` (orange). The setup for both simulations included a box size of 100 and 25 solitons with spin 1.

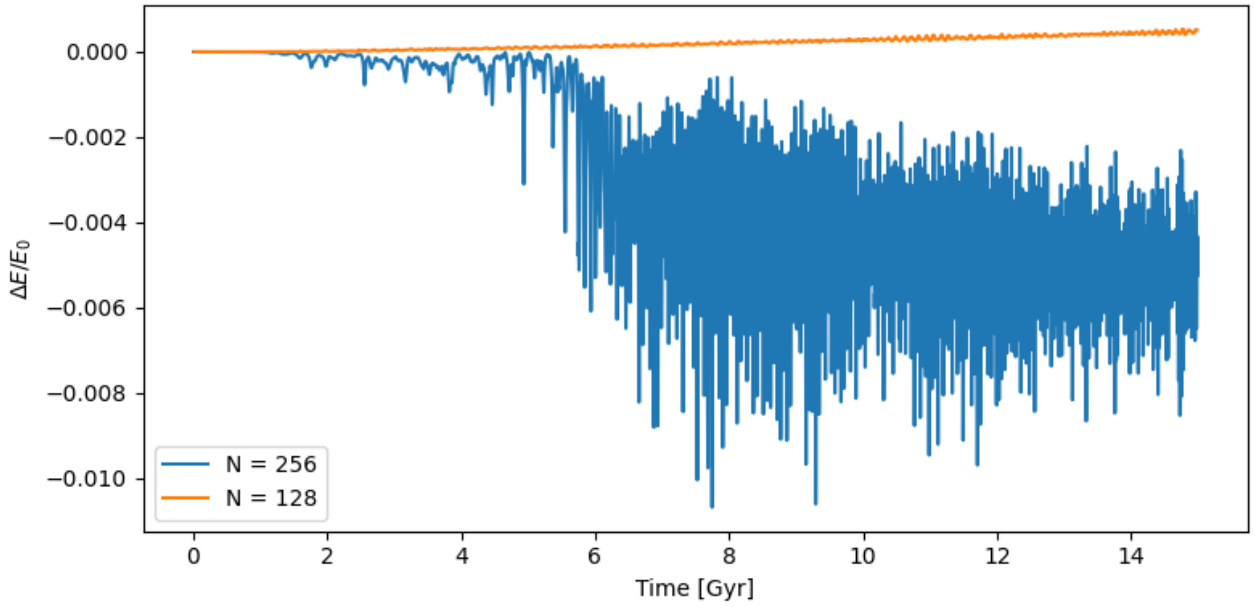

Figure 3.9: Normalized energy deviation $\Delta E/E_0$ for simulations with resolution `N = 256` and second-order evolution (blue), and resolution `N = 128` using fourth-order evolution (orange). Time steps were `h = 0.001` for second-order runs and `h = 0.01` for the fourth-order run. All simulations used a box size of 100 and 25 solitons with spin 1.

where T_j and V_j represent kinetic and potential contributions to the propagator. For systems governed by the SP system Equation 1.12, the potential depends non-linearly on the evolving wave function, introducing non-commutativity between kinetic and potential operators. This leads to the operator splitting being only an approximation. Furthermore, the error in simulations is associative, meaning once it begins, it is propagated into the next steps and grows over each iteration. This upward trend can be reduced by making the time step smaller or increasing the grid resolution.

Further plots showing the influence of soliton spin on energy conservation within the simulation framework will be shown in Chapter 4.

Chapter 4

Results

In this chapter, I will show and discuss some of the results obtained by the simulation framework shown in this work. Since this part of my work focused mainly on building the framework, there are not many results to speak of, and future prospects and plans will be described in Chapter 5. For now, I will show parts of the analysis similar to the one done in [60]. All data shown in this chapter were obtained from a simulation with a fixed pseudo–random seed corresponding to `np.random.seed(1234)`.

4.1 Dark Matter Density Profiles

In Figure 4.1, we show the spherically averaged density profiles obtained from simulations of a merger of 25 solitons in a box of size 100 for spins $s = 0$, 1, 2, 3 on a grid with resolution $N = 256$.

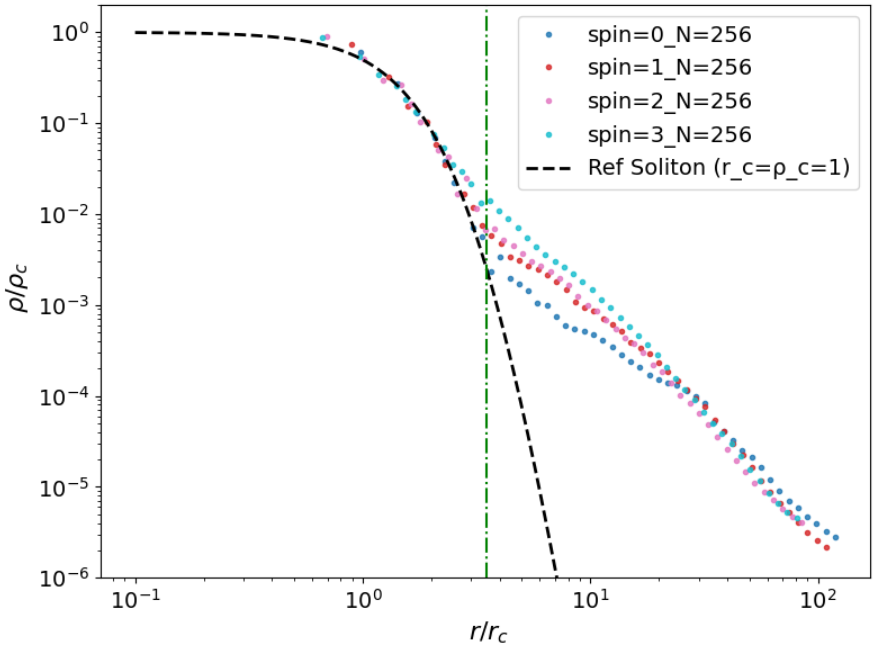

Figure 4.1: Spherically averaged density profiles from simulations of a 25–soliton merger in a box of size 100 for spins $s = 0$, 1, 2, 3 at $t = 40$ Gyr. The profiles are normalized to the central density ρ_c and the radius is scaled by the core radius r_c obtained from fitting Equation 1.38. The dashed black line shows the reference soliton profile with $r_c = \rho_c = 1$, while the vertical dash–dotted green line marks the transition radius $r_\epsilon \simeq 3.5$ where the fit switches from the solitonic core to the NFW-like outer halo.

These profiles were obtained by fitting the final density field at $t = 40$ Gyr with Equation 1.38. The profiles are normalized to their respective central densities ρ_c and plotted as a function of radius r. A vertical dash–dotted green line marks the transition radius $r_\epsilon \simeq 3.5$, where the fit changes from the solitonic profile to the NFW-like outer halo (Equation 1.37). Simulation results reproduce the expected behaviour, we observe a flat-density core extending up to $r \sim r_c$, followed by a power-law decline in the outer halo. The inner parts of all four spin cases align closely with the reference soliton solution, confirming that our split-step evolution correctly preserves the stationary core structure over long integration times.

Beyond the transition radius, the simulated profiles fall off more slowly than the pure soliton solution, which is to be expected from the contribution of surrounding halo material. This deviation is most visible at $r/r_c \gtrsim 5$, where interference patterns and residual wave turbulence lead to small-scale oscillations in $\rho(r)$.

The outer slope of the simulated profiles shows variations with spin, and the overall trends are compatible with the results presented in [60]. In our case, higher-spin runs deviate from the reference soliton solution at slightly smaller radii. The $s = 0$ profile follows the reference soliton most closely in both the core and transition region, with the steepest and smoothest decline among all cases, which could be caused by the absence of polarization mode coupling. In contrast, the $s = 3$ and $s = 2$ profiles show a slightly shallower drop just beyond r_c, which may indicate enhanced mode mixing between polarization components in the higher-spin cases.

For a more robust comparison with [60], a so-called "systematic ensemble" study should be performed, in which the simulations would be repeated multiple times with a fixed pseudo–random seed to keep the initial large-scale configuration consistent while varying the soliton spin. Averaging over these runs would reduce statistical fluctuations and make spin-dependent differences more apparent, but due to time constraints, this is left for future work.

The agreement between our simulated profiles and the reference soliton in the core region is even clearer when looking at the best-fit curves shown in Figure 4.2. Here we plot the fitted profiles for each spin case, with the corresponding values of r_ϵ, r_c and r_s shown in the legend. We do not include fit errors in this work, as the small sample size of our simulations leads to large statistical uncertainties, and showing only the parameter-fit errors could give a misleading impression of accuracy.

Looking at the fitted curves, the overall picture is the same as in Figure 4.1: a very good match to the reference soliton in the core region, and then differences in the outer slope between the spins and the reference soliton. The $s = 0$ profile still shows the steepest and cleanest drop, while the higher-spin cases are slightly shallower just beyond r_c. Using the fits smooths out the small oscillations in the data and makes these differences easier to see.

4.2 Core Radius Growth

In Figure 4.3 we show the time evolution of the dimensionless core radius parameter $\lambda_\rho(t)$ for spins $s = 0$ (blue), 1 (orange), 2 (green), 3 (red), extracted from our simulations of 25 solitons merger with grid resolution $N = 256$. Here λ_ρ is defined following [60] as the ratio between the initial and central densities at time step t:

$$\lambda_\rho(t) = \left(\frac{\rho_c^f}{\rho_c^i} \right)^{\frac{1}{4}}, \tag{4.1}$$

and can be interpreted as a measure of the core size growth during the merger process. This is because in the soliton profile $\rho_c \propto r_c^{-4}$ (see Equation 1.26), so a decrease in the central density ρ_c implies an increase in the core radius r_c. Since $\lambda_\rho(t)$ is defined as in Equation 4.1,

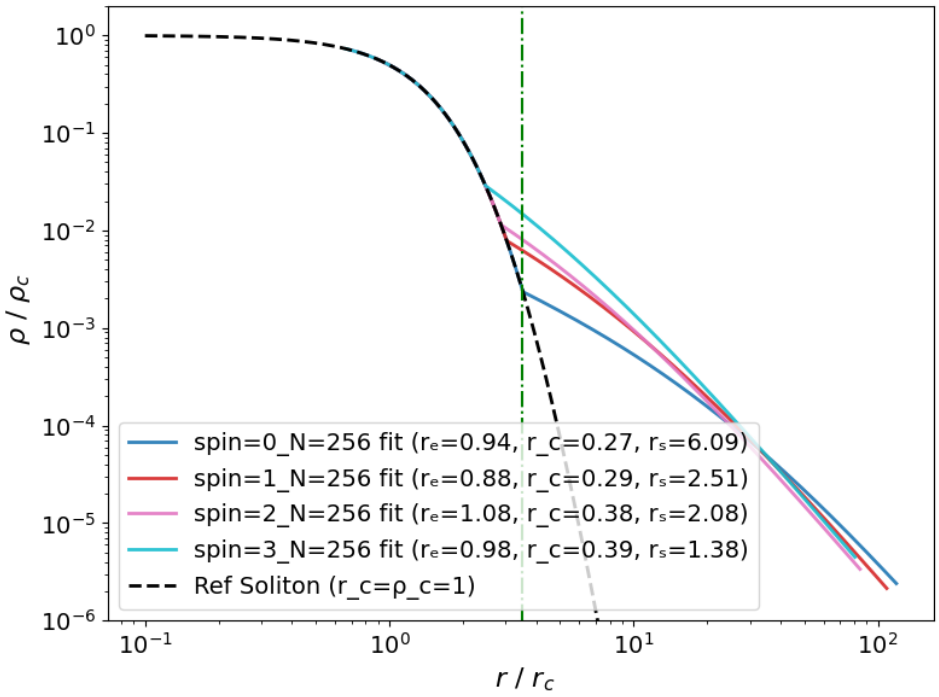

Figure 4.2: Spherically averaged density profiles with the best fits from simulations of a 25 soliton merger in a box of size 100 for spins $s = 0, 1, 2, 3$ at $t = 30$ Gyr. The profiles are normalized to the central density ρ_c and the radius is scaled by the core radius r_c obtained from fitting Equation 1.38. The dashed black line shows the reference soliton profile with $r_c = \rho_c = 1$, while the vertical dash–dotted green line marks the transition radius $r_\epsilon \simeq 3.5$ where the fit switches from the solitonic core to the NFW-like outer halo.

values larger than unity correspond to a lower central density and therefore a physically larger solitonic core. The dots represent the raw values obtained at each saved time step, while the solid black lines are exponential fits of the form

$$\lambda_\rho(t) = \lambda_\infty - (\lambda_\infty - 1)\, e^{-t/\tau} \tag{4.2}$$

which provide an estimate of the asymptotic value λ_∞ for each spin case. These λ_∞ values are indicated both in the legend and by the dashed horizontal lines.

For all spins, λ_ρ increases rapidly at the beginning of the simulation and then gradually approaches a plateau, consistent with the core halo relaxation seen in previous works such as [60]. The $s = 0$ case reaches an asymptotic value of $\lambda_\infty \approx 3.41$, which is slightly larger than the $s = 1$ and $s = 2$ cases, giving $\lambda_\infty \approx 3.25$ and $\lambda_\infty \approx 3.00$, respectively. The $s = 3$ case shows the smallest growth, with $\lambda_\infty \approx 2.54$. These values are in line with the trends reported in [60], where the maximal values of λ_ρ are observed to stabilize at lower levels for higher spins.

In our simulations, the $s = 0$ solution appears to continue growing for a longer time before approaching its plateau, while higher-spin cases reach their asymptotic values earlier. Once again, the errors on these parameters are not included here due to the large statistical uncertainties caused by the limited sample size.

The scatter around the fits is mainly due to interference patterns and residual oscillations in the density field after major merger events. Part of these fluctuations can also be attributed to the oscillations seen in Figure 3.3, which arise from discretization effects and numerical noise.

Although the overall behaviour matches the qualitative trends in [60], the differences between spins in our runs are slightly smaller, probably due to the limited statistical representation. Repeating this analysis with more simulation runs, as discussed earlier for the density profiles,

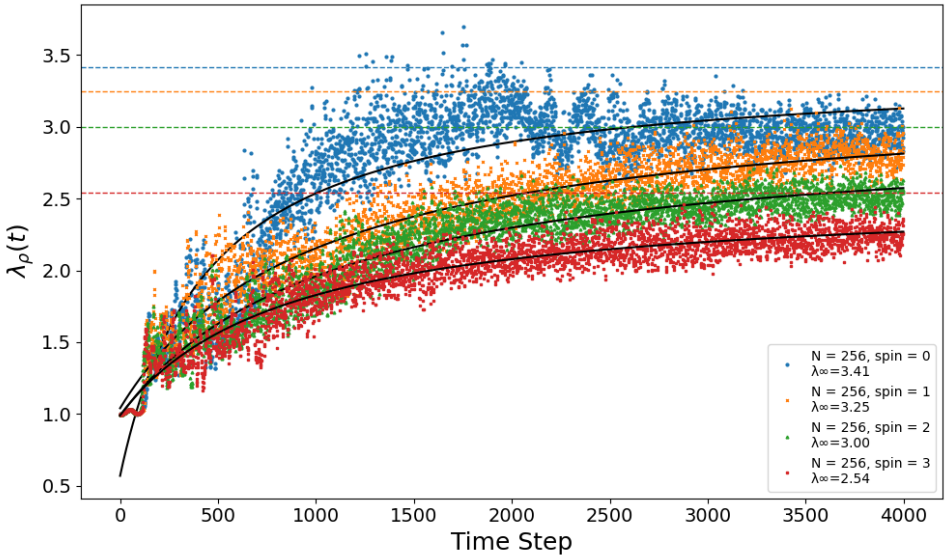

Figure 4.3: Dependence of the dimensionless core radius parameter $\lambda_\rho(t)$ on time from simulations of a 25–soliton merger in a box of size 100 for spins $s = 0$ (blue), 1 (orange), 2 (green), 3 (red) at resolution $N = 256$. Here $\lambda_\rho(t)$, defined in Equation 4.1, is the ratio between the initial and central densities at time step t, so values above unity correspond to a growth of the solitonic core radius during the merger process. Horizontal dashed lines indicate the asymptotic values λ_∞ for each spin.

might yield more reliable averages and help determine whether the observed spin dependence of λ_∞ is a robust physical effect or partly statistical.

4.3 Energy Conservation

In Figure 4.4 we plot the relative energy change $\Delta E/E_0$ for simulation of a merger of 25 solitons with spins $s = 0$, 1, 2, 3 at resolution $N = 256$. Ideally, for a perfectly energy-conserving numerical scheme, this quantity would remain close to zero throughout the simulation.

The $s = 1$ (in orange) case shows the best energy conservation, with only a small monotonic drift over the 40 Gyr evolution. The $s = 2$ (in green) and $s = 3$ (in red) runs maintain an approximately constant energy offset after the initial relaxation, although they exhibit more short-timescale fluctuations, especially the $s = 3$ case, where the larger oscillations likely arise from stronger mode coupling and interference between the different components. In contrast, the $s = 0$ (in blue) case displays a steady and significant loss of total energy over time, which could indicate a systematic numerical error such as low resolution or large size of the time step. This could lead to an accumulation of errors in the split-step integration, resulting in a loss of total energy within the system.

The spin-0 case appears most affected by error propagation, likely due to its denser and narrower core structure and the fact that its entire mass–energy content is carried by a single component wave function. This can lead to very high local densities approaching the resolution limit, making the simulation is more sensitive to discretization errors than in the higher-spin cases, where the wave function is split into multiple components and such effects are reduced.

Ideally, energy should remain constant, but numerical effects such as finite resolution and finite time-step size cause deviations. The case $s = 1$ shows the best conservation, while $s = 0$ exhibits a gradual energy loss, likely due to resolution limits. Increasing resolution or reducing the time step could improve conservation of the total energy within the simulation.

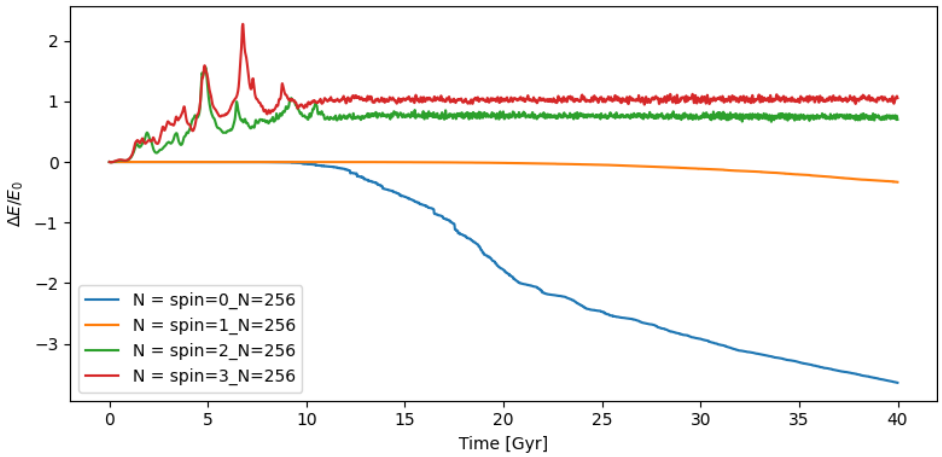

Figure 4.4: Relative energy change $\Delta E/E_0$ as a function of time for simulations of a 25 soliton merger with spins $s = 0$ (blue), $s = 1$ (orange), $s = 2$ (green) and $s = 3$ (red) at resolution $N = 256$.

4.4 Discussion

The density profiles, core growth, and energy conservation results are consistent with the trends observed in [60]. All spins show solitonic cores matching the reference profile, with higher-spin cases having slightly shallower outer slopes and the spin-0 case most closely following the reference solitonic solution. Core growth analysis confirms that higher spins stabilize at lower λ_∞, while spin-0 continues to evolve for longer. Energy conservation is best for higher-spin runs, which maintain a stable offset in $\Delta E/E_0$, whereas spin-0 shows a steady loss, likely due to the resolution sensitivity of its dense, single-component core.

From this limited dataset, our observations suggest that higher-spin systems may exhibit greater structural and energetic stability, while single-component configurations appear to remain dynamically active for longer. These findings, if combined with observational data, could potentially provide some insights into the nature of dark matter in our universe. However, conclusive results that could support such claims with confidence would require a more comprehensive study involving systematic ensemble runs with a more complex analysis of the studied systems. Such analysis will be the subject of the continuation of this work, together with implementation of improvements to the simulation framework, which will be discussed in Chapter 5

Chapter 5

Future Plans

This simulation framework provides a solid foundation for further investigation of axion-like particle dark matter. The results obtained so far, such as the generation and fitting of density profiles, validate the correctness of the implementation and offer several directions for further analysis.

5.1 Tracking the Virial Ratio

An additional diagnostic to verify the correct functionality of the code is to track the virial ratio

$$\frac{2K}{|W|}, \tag{5.1}$$

where K is the total kinetic energy and W is the gravitational potential energy. However, in these simulations, where quantum pressure appears, the usual virial condition $2K/|W| \approx 1$ is no longer satisfied. Instead, the kinetic energy can be decomposed into two components:

$$K = K_v + K_q, \tag{5.2}$$

where K_v corresponds to kinetic energy associated with the phase gradient (interpreted as fluid velocity) and K_q is the so called quantum kinetic energy (or quantum pressure term) adopted from [61].

This decomposition arises naturally from the Madelung representation of the wave function talked about in Section 1.7,

$$\psi(\mathbf{x}, t) = \sqrt{\rho(\mathbf{x}, t)}\, e^{iS(\mathbf{x},t)}, \tag{5.3}$$

where $\rho = |\psi|^2$ and $\mathbf{v} = \nabla S$ is the velocity field. Plugging these relations into the expression for the kinetic energy yields the following relations:

$$K = \frac{1}{2}\int |\nabla\psi|^2\, d^3x \tag{5.4}$$

$$= \frac{1}{2}\int \left[(\nabla\sqrt{\rho})^2 + \rho(\nabla S)^2\right]\, d^3x \tag{5.5}$$

$$= K_q + K_v. \tag{5.6}$$

The two terms can be interpreted as:

- **Gradient Energy caused by quantum pressure (K_q)**

$$K_q = \frac{1}{2}\int |\nabla\sqrt{\rho}|^2\, d^3x, \tag{5.7}$$

  which arises purely from the inhomogeneity of the density field, even when $\nabla S = 0$.

- **Classical kinetic energy (K_v):**

$$K_v = \frac{1}{2}\int \rho(\nabla S)^2\, d^3x, \tag{5.8}$$

  which is zero for static or ground-state solitons. In simulations, we should therefore see $K_v \to 0$ in the solitonic cores. The kinetic term tending toward 0 is a strong indicator of convergence to equilibrium and, therefore a good working simulation.

To implement this computation into the code, I propose these steps:

1. Extraction of $\rho = |\psi|^2$ and $S = \arg(\psi)$.
2. Computation of gradients using FFT:
   - $\nabla\sqrt{\rho}$ for K_q,
   - ∇S for K_v.
3. Integration over space with the correct volume element:

$$K_q = \frac{1}{2}\sum \left(\nabla\sqrt{\rho}\right)^2 \cdot dV, \quad K_v = \frac{1}{2}\sum \rho\left(\nabla S\right)^2 \cdot dV.$$

Using this it should be possible to verify the virial balance in various regions in the simulation, e.g. core or halo, show if any numerical artifacts appear in the simulations. This would be by having non-zero K_v in soliton cores. Furthermore, it should be possible to define a local virial ratio and map its evolution in time over the simulation grid to visualize the thermalization and collapse of regions. These diagnostics can also help us with broader FDM studies, such as tidal disruption, sub halo formation and gravitational cooling. This would also be very interesting for the study of the self-interaction, which is our next plan.

5.2 Analysis of Self-Interaction Effects

The simulation code already supports self-interactions via an additional term in the SP system, using the formulation introduced in [43]. In this approach, the self-interaction arises naturally in multi-component scalar field models, and leads to an additional term in the equation of motion:

$$i\partial_t\psi = \left[-\frac{1}{2}\nabla^2 + V + \lambda|\psi|^2\right]\psi, \tag{5.9}$$

where λ is the self-interaction coupling related to the scattering length a_s by

$$\lambda = \frac{4\pi a_s}{m}, \tag{5.10}$$

and V includes the gravitational potential from the Poisson equation.

Static solitonic profiles under this type of self-interaction have already been constructed and are accessible (thanks to Erick Munive Villa, Ph.D[1]) within the simulation as a valid `Packet_type`, but any analysis has to be postponed due to the approaching deadline of this work. Future plans using these self-interacting solitons then include:

- simulation of static solitons with attractive ($\lambda < 0$) and repulsive ($\lambda > 0$) interactions,[2]

[1] These were prepared in similar process to the one described in Subsection 1.6.2

[2] This is mainly to test the predictability and consistency of the code, rather than being of primary interest for the study of self-interaction phenomenology.

- tests of soliton stability using energy conservation and tracking of the soliton density[3] Beyond this, the framework could be extended to study phenomena such as the Jeans instability, condensation processes, tidal disruption of subhalos, and more.
- once virial ratio tracking is implemented, a study of the energy contributions in self-interacting systems becomes possible,[4]

According to [43], repulsive self-interactions tend to stabilize the core against gravitational collapse and lead to broader density profiles, while attractive interactions can accelerate collapse and produce more compact solitons. The combined strength of gravity and self-interaction should change the soliton mass–radius relation, so further analysis including comparison between self-interacting and non-self-interacting solitons would be required.

Furthermore, the presence of self-interaction can change the conditions for dynamical stability and modify the formation of interference patterns. These features can also be studied with the simulation framework. In particular, a complete description of higher-spin ULDM models requires the inclusion of self-interactions. For example, in the spin-2 case, the self-interaction term cannot be zero [71]. Moreover, if self-interaction allows mixing between different spin components, mass could be exchanged among the spin-s ULDM components, potentially leading to new properties of the systems.

5.3 Inclusion of Background Expansion

To better simulate cosmological evolution in the code, we could generalize the system to allow for an expanding background by introducing the scale factor $a(t)$ back into the Schrödinger-Poisson equations. This way we would use the SP system in comoving coordinates as shown in Equation 1.45. This would allow for the study of halo formation and soliton dynamics in a cosmological setting, e.g.,

$$\psi(\mathbf{x}, t) \to \psi(\mathbf{x}, \eta), \quad \nabla \to \frac{1}{a(\eta)}\nabla, \tag{5.11}$$

where η is conformal time. Unfortunately, implementation would require rescaling of all variables and appropriate initial conditions, including the generation of an initial Gaussian random field consistent with the primordial power spectrum shown in Figure 1.1, which depends on the mass of the ULDM particle[5].

5.4 Support for Angular Momentum and Vortex Formation

Another feature that would be interesting to add is the inclusion of initial angular momentum. This would allow the study of rotating solitons and vortex structures, which are expected in systems where the phase of the wave function carries a non-trivial topological charge [72]. A simple form of such a wave function is

$$\psi(\mathbf{x}) = f(r)e^{im\phi}, \tag{5.12}$$

[3]This and the previous point would be done in an analysis similar to the one performed in Chapter 3.

[4]This is a useful diagnostic even in the $\lambda = 0$ (non-interacting) case, and serves as a consistency check on the evolution.

[5]This sounds very time consuming and the author does not have a clear plan in mind how to implement this feature.

where $f(r)$ is a radial profile (e.g., soliton), ϕ is the azimuthal angle in cylindrical or spherical coordinates, and $m \in \mathbb{Z}$ is the winding number. The inclusion of angular momentum would provide an option to study new systems and test for example the stability of rotating solitons and their equilibrium profiles, formation of vortices during soliton mergers and maybe monitor the angular momentum transfer in these simulations. Of course analysis comparing rotating solitons to stationary ones could also be done. The total angular momentum of the field can be computed via

$$\mathbf{L} = \int \psi^*(\mathbf{r} \times \hat{\mathbf{p}})\psi \, d^3x, \tag{5.13}$$

where $\hat{\mathbf{p}} = -i\nabla$ is the momentum operator, which in the code could be simplified to:

$$L_z = m \int |\psi(\mathbf{x})|^2 \, d^3x = mM, \tag{5.14}$$

where M would be the total mass of the configuration and m some integer value.

This could be implemented by extending the `Packet` class to support a new `packet_type="vortex"` that would apply a phase $e^{im\phi}$ to the wave function to imprint the angular momentum and then monitor if the conservation of the angular momentum holds. For that, we could use the current

$$\mathbf{j} = \frac{1}{2i} \left(\psi^* \nabla\psi - \psi \nabla\psi^* \right),$$

and compute the moment $\mathbf{r} \times \mathbf{j}$ at each time step. Which could be used for testing the correct implementation of the angular momentum.

5.5 Computation of the Matter Power Spectrum $\Delta^2(k)$

In the longer term, an important goal would be to compute the matter power spectrum in a cosmological context, as shown in Figure 1.1. This would require running full cosmological simulations, which is not straightforward because setting up the correct initial conditions for such simulations is still not fully understood and developed [73]. Part of the work would therefore involve learning how to implement these initial conditions properly. This step would include the computation of the dimensionless matter power spectrum as has been shown on Figure 1.1 using:

$$\Delta^2(k) = \frac{k^3}{2\pi^2} P(k), \tag{5.15}$$

where $P(k)$ is the power spectrum of the density contrast in the Fourier space used in simulations. The goal is to extract $\Delta^2(k)$ at different simulation times by:

- computing the total density field $\rho(\mathbf{x}) = |\psi(\mathbf{x})|^2$,
- applying a Fourier transform to obtain $\tilde{\rho}(\mathbf{k})$,
- computing $P(k) = |\tilde{\rho}(\mathbf{k})|^2$ and binning it spherically.

Most of these steps have already been employed, as has been shown in Chapter 2, so the implementation itself to the simulation framework should be quite straightforward. In addition, specialized tools such as Pylians can be used to enhance the overall analysis.

Chapter 6

Conclusion

This research task presented the development of a modular and extensible simulation framework for studying ultra-light dark matter, specifically in the context of fuzzy and axion-like particle models. The primary goal was to implement and test a numerical solver based on the split step Fourier method for evolving the Schrödinger–Poisson system, which describes the non-relativistic dynamics of bosonic dark matter on astrophysical scales.

To achieve this, I studied dark matter problematic (Chapter 1), designed and implemented computational tools (Chapter 2). The resulting framework allows for initialization of arbitrary-spin **identical** solitonic configurations, their evolution under self-gravity and optional interactions, and extraction of physical observables.

A series of validation tests were carried out (Chapter 3), which included:

1. The Quantum Harmonic Oscillator Test - to verify the correct working of the Fourier solver,
2. Stability Checks - to confirm that with rising resolution, the oscillations caused by numerical errors are decreasing and solitonic solutions are stable in time,
3. Kinematic Tests - to reproduce the predicted circular motion around a point-like mass,
4. Energy Conservation Analysis - to show the stability of the simulation and the impact of spatial resolution, time-step size, and integrator order.

With the validated framework, simulations were carried out (Chapter 4), producing density profiles, soliton core growth plots, and energy conservation statistics even for higher-spin systems. The results are in agreement with previous studies such as [60]. The framework reproduced the expected solitonic core halo structure and revealed spin-dependent differences in outer profile slopes. Furthermore, the conservation of energy within these systems was studied, revealing that higher-spin configurations tend to exhibit greater energetic stability. At the same time, the spin-0 case showed prolonged evolution and stronger sensitivity to numerical parameters.

In addition to delivering these first results, the work identified clear directions for improvement (Chapter 5). These include: direct computation of the matter power spectrum from simulation outputs, implementation of virial ratio diagnostics including quantum and classical kinetic energy contributions, systematic exploration of self-interaction effects, inclusion of cosmological background expansion, and support for initial conditions with angular momentum.

Bibliography

[1] W. Hu, R. Barkana and A. Gruzinov, *Fuzzy cold dark matter: The wave properties of ultralight particles*, *Physical Review Letters* **85** (2000) 1158–1161.

[2] L. Hui, J.P. Ostriker, S. Tremaine and E. Witten, *Ultralight scalars as cosmological dark matter*, *Physical Review D* **95** (2017) .

[3] E.G.M. Ferreira, *Ultra-light dark matter*, *The Astronomy and Astrophysics Review* **29** (2021) .

[4] J.C. Niemeyer, *Small-scale structure of fuzzy and axion-like dark matter*, *Progress in Particle and Nuclear Physics* **113** (2020) 103787.

[5] F. Zwicky, *Republication of: The redshift of extragalactic nebulae*, *General Relativity and Gravitation* **41** (2009) 207.

[6] P.A.R. Ade, N. Aghanim, M. Arnaud, M. Ashdown, J. Aumont, C. Baccigalupi et al., *Planck2015 results: Xiii. cosmological parameters*, *Astronomy &; Astrophysics* **594** (2016) A13.

[7] C. Boehm and R. Schaeffer, *Constraints on dark matter interactions from structure formation: damping lengths*, *Astronomy amp; Astrophysics* **438** (2005) 419–442.

[8] A. Ibarra, M. Reichard and G. Tomar, *Probing dark matter electromagnetic properties in direct detection experiments*, *Journal of Cosmology and Astroparticle Physics* **2025** (2025) 072.

[9] R.C. Cotta, J.L. Hewett, M.-P. Le and T.G. Rizzo, *Bounds on dark matter interactions with electroweak gauge bosons*, *Physical Review D* **88** (2013) .

[10] D.N. Spergel and P.J. Steinhardt, *Observational evidence for self-interacting cold dark matter*, *Physical Review Letters* **84** (2000) 3760–3763.

[11] L. Anderson, Aubourg, S. Bailey, F. Beutler, V. Bhardwaj, M. Blanton et al., *The clustering of galaxies in the sdss-iii baryon oscillation spectroscopic survey: baryon acoustic oscillations in the data releases 10 and 11 galaxy samples*, *Monthly Notices of the Royal Astronomical Society* **441** (2014) 24–62.

[12] P.J.E. Peebles, *Principles of physical cosmology*, Princeton university press (2020).

[13] B. Moore, *Evidence against dissipation-less dark matter from observations of galaxy haloes*, **370** (1994) 629.

[14] N. Aghanim, Y. Akrami, M. Ashdown, J. Aumont, C. Baccigalupi, M. Ballardini et al., *Planck2018 results: Vi. cosmological parameters*, *Astronomy amp; Astrophysics* **641** (2020) A6.

[15] G. Bertone and D. Hooper, *History of dark matter*, *Reviews of Modern Physics* **90** (2018) .

[16] G. Bertone, D. Hooper and J. Silk, *Particle dark matter: evidence, candidates and constraints*, *Physics Reports* **405** (2005) 279–390.

[17] M. Tegmark, M.R. Blanton, M.A. Strauss, F. Hoyle, D. Schlegel, R. Scoccimarro et al., *The three-dimensional power spectrum of galaxies from the sloan digital sky survey*, *The Astrophysical Journal* **606** (2004) 702.

[18] S. Kazantzidis, L. Mayer, C. Mastropietro, J. Diemand, J. Stadel and B. Moore, *Density profiles of cold dark matter substructure: Implications for the missing-satellites problem*, *The Astrophysical Journal* **608** (2004) 663.

[19] M. Boylan-Kolchin, J.S. Bullock and M. Kaplinghat, *Too big to fail? the puzzling darkness of massive milky way subhaloes*, *Monthly Notices of the Royal Astronomical Society: Letters* **415** (2011) L40–L44.

[20] A. Klypin, A.V. Kravtsov, O. Valenzuela and F. Prada, *Where are the missing galactic satellites?*, *The Astrophysical Journal* **522** (1999) 82–92.

[21] A. Pontzen and F. Governato, *Cold dark matter heats up*, *Nature* **506** (2014) 171–178.

[22] F. Governato, A. Zolotov, A. Pontzen, C. Christensen, S.H. Oh, A.M. Brooks et al., *Cuspy no more: how outflows affect the central dark matter and baryon distribution in Λ cold dark matter galaxies: Galaxy cores in Λcdm*, *Monthly Notices of the Royal Astronomical Society* **422** (2012) 1231–1240.

[23] M. Vogelsberger, S. Genel, V. Springel, P. Torrey, D. Sijacki, D. Xu et al., *Introducing the illustris project: simulating the coevolution of dark and visible matter in the universe*, *Monthly Notices of the Royal Astronomical Society* **444** (2014) 1518–1547.

[24] C. O'Hare, *Cosmology of axion dark matter*, in *Proceedings of 1st General Meeting and 1st Training School of the COST Action COSMIC WSIPers — PoS(COSMICWISPers)*, COSMICWISPers, p. 040, Sissa Medialab, Apr., 2024, DOI.

[25] J.S. Bullock and M. Boylan-Kolchin, *Small-scale challenges to the Λcdm paradigm*, *Annual Review of Astronomy and Astrophysics* **55** (2017) 343–387.

[26] D.M. Jacobs, G.D. Starkman and B.W. Lynn, *Macro dark matter*, *Monthly Notices of the Royal Astronomical Society* **450** (2015) 3418–3430.

[27] M. Khlopov, *Direct and indirect probes for composite dark matter*, *Frontiers in Physics* **7** (2019) .

[28] L. Roszkowski, E.M. Sessolo and S. Trojanowski, *Wimp dark matter candidates and searches—current status and future prospects*, *Reports on Progress in Physics* **81** (2018) 066201.

[29] G. Arcadi, M. Dutra, P. Ghosh, M. Lindner, Y. Mambrini, M. Pierre et al., *The waning of the wimp? a review of models, searches, and constraints*, *The European Physical Journal C* **78** (2018) .

[30] R.D. Peccei and H.R. Quinn, CP *conservation in the presence of pseudoparticles*, *Phys. Rev. Lett.* **38** (1977) 1440.

[31] S. Weinberg, *A new light boson?*, *Phys. Rev. Lett.* **40** (1978) 223.

[32] F. Wilczek, *Problem of strong p and t invariance in the presence of instantons*, *Phys. Rev. Lett.* **40** (1978) 279.

[33] L. Di Luzio, M. Giannotti, E. Nardi and L. Visinelli, *The landscape of qcd axion models*, *Physics Reports* **870** (2020) 1–117.

[34] H.-Y. Schive, M.-H. Liao, T.-P. Woo, S.-K. Wong, T. Chiueh, T. Broadhurst et al., *Understanding the core-halo relation of quantum wave dark matter from 3d simulations*, *Physical Review Letters* **113** (2014) .

[35] P.-H. Chavanis, *Mass-radius relation of newtonian self-gravitating bose-einstein condensates with short-range interactions. i. analytical results*, *Physical Review D* **84** (2011) .

[36] E. Calabrese and D.N. Spergel, *Ultra-light dark matter in ultra-faint dwarf galaxies*, *Monthly Notices of the Royal Astronomical Society* **460** (2016) 4397–4402.

[37] M.Y. Khlopov, B.A. Malomed and Y.B. Zeldovich, *Gravitational instability of scalar fields and formation of primordial black holes*, *Monthly Notices of the Royal Astronomical Society* **215** (1985) 575 [https://academic.oup.com/mnras/article-pdf/215/4/575/4082842/mnras215-0575.pdf].

[38] D.J. Marsh, *Axion cosmology*, *Physics Reports* **643** (2016) 1–79.

[39] V. Iršič, M. Viel, M.G. Haehnelt, J.S. Bolton, S. Cristiani, G.D. Becker et al., *New constraints on the free-streaming of warm dark matter from intermediate and small scale lyman-α forest data*, *Phys. Rev. D* **96** (2017) 023522.

[40] K.K. Rogers and H.V. Peiris, *General framework for cosmological dark matter bounds using n-body simulations*, *Phys. Rev. D* **103** (2021) 043526.

[41] N. Banik, J. Bovy, G. Bertone, D. Erkal and T. de Boer, *Novel constraints on the particle nature of dark matter from stellar streams*, *Journal of Cosmology and Astroparticle Physics* **2021** (2021) 043.

[42] J. Sipple, A. Lidz, D. Grin and G. Sun, *Fuzzy dark matter constraints from the hubble frontier fields*, 2025.

[43] M. Jain and M.A. Amin, *i-spin: an integrator for multicomponent schrödinger-poisson systems with self-interactions*, *Journal of Cosmology and Astroparticle Physics* **2023** (2023) 053.

[44] J. Jaeckel and A. Ringwald, *The low-energy frontier of particle physics*, *Annual Review of Nuclear and Particle Science* **60** (2010) 405–437.

[45] G. Davies and L.M. Widrow, *Test-bed simulations of collisionless, self-gravitating systems using the schrödinger method*, 1996.

[46] A. Kunkel, H.Y.J. Chan, H.-Y. Schive, H. Huang and P.-Y. Liao, *A hybrid scheme for fuzzy dark matter simulations combining the schrödinger and hamilton-jacobi-madelung equations*, 2025.

[47] D. Levkov, A. Panin and I. Tkachev, *Relativistic axions from collapsing bose stars*, *Physical Review Letters* **118** (2017) .

[48] E.W. Kolb and I.I. Tkachev, *Axion miniclusters and bose stars*, *Physical Review Letters* **71** (1993) 3051–3054.

[49] E. Braaten, A. Mohapatra and H. Zhang, *Dense axion stars*, *Physical Review Letters* **117** (2016) .

[50] L. Visinelli, S. Baum, J. Redondo, K. Freese and F. Wilczek, *Dilute and dense axion stars*, *Physics Letters B* **777** (2018) 64–72.

[51] F.S. Guzmán and L.A. Ureña-López, *Evolution of the schrödinger-newton system for a self-gravitating scalar field*, *Physical Review D* **69** (2004) .

[52] S.U. Ji and S.J. Sin, *Late-time phase transition and the galactic halo as a bose liquid. ii. the effect of visible matter*, *Physical Review D* **50** (1994) 3655–3659.

[53] F.S. Guzman and L.A. Urena-Lopez, *Gravitational cooling of self-gravitating bose condensates*, *The Astrophysical Journal* **645** (2006) 814–819.

[54] P. Sikivie and Q. Yang, *Bose-einstein condensation of dark matter axions*, *Physical Review Letters* **103** (2009) .

[55] D. Levkov, A. Panin and I. Tkachev, *Gravitational bose-einstein condensation in the kinetic regime*, *Physical Review Letters* **121** (2018) .

[56] B. Eggemeier and J.C. Niemeyer, *Formation and mass growth of axion stars in axion miniclusters*, *Physical Review D* **100** (2019) .

[57] S. Davidson and M. Elmer, *Bose-einstein condensation of the classical axion field in cosmology?*, *Journal of Cosmology and Astroparticle Physics* **2013** (2013) 034–034.

[58] J.F. Navarro, C.S. Frenk and S.D.M. White, *The structure of cold dark matter halos*, *The Astrophysical Journal* **462** (1996) 563.

[59] J.F. Navarro, C.S. Frenk and S.D.M. White, *A universal density profile from hierarchical clustering*, *The Astrophysical Journal* **490** (1997) 493–508.

[60] J.N. López-Sánchez, E. Munive-Villa, C. Skordis and F.R. Urban, *Scaling relations and tidal disruption in spin s ultralight dark matter models*, 2502.03561.

[61] P. Mocz, M. Vogelsberger, V.H. Robles, J. Zavala, M. Boylan-Kolchin, A. Fialkov et al., *Galaxy formation with becdm – i. turbulence and relaxation of idealized haloes*, *Monthly Notices of the Royal Astronomical Society* **471** (2017) 4559–4570.

[62] S. May and V. Springel, *Structure formation in large-volume cosmological simulations of fuzzy dark matter: impact of the non-linear dynamics*, *Monthly Notices of the Royal Astronomical Society* **506** (2021) 2603–2618.

[63] M. Feit, J. Fleck and A. Steiger, *Solution of the schrödinger equation by a spectral method*, *Journal of Computational Physics* **47** (1982) 412.

[64] S. MacNamara and G. Strang, *Operator splitting*, in *Splitting Methods in Communication, Imaging, Science, and Engineering*, R. Glowinski, S.J. Osher and W. Yin, eds., (Cham), pp. 95–114, Springer International Publishing (2016), DOI.

[65] F. Edwards, E. Kendall, S. Hotchkiss and R. Easther, *Pyultralight: a pseudo-spectral solver for ultralight dark matter dynamics*, *Journal of Cosmology and Astroparticle Physics* **2018** (2018) 027–027.

[66] H. Yoshida, *Construction of higher order symplectic integrators*, *Physics Letters A* **150** (1990) 262.

[67] X. Du, B. Schwabe, J.C. Niemeyer and D. Bürger, *Tidal disruption of fuzzy dark matter subhalo cores*, *Physical Review D* **97** (2018) .

[68] M.E. Peskin and D.V. Schroeder, *An Introduction to quantum field theory*, Addison-Wesley, Reading, USA (1995), 10.1201/9780429503559.

[69] R. RUFFINI and S. BONAZZOLA, *Systems of self-gravitating particles in general relativity and the concept of an equation of state*, *Phys. Rev.* **187** (1969) 1767.

[70] D.J. Griffiths and D.F. Schroeter, *Introduction to quantum mechanics*, Cambridge University Press, Cambridge ; New York, NY, third edition ed. (2018).

[71] E. Babichev, L. Marzola, M. Raidal, A. Schmidt-May, F. Urban, H. Veermäe et al., *Heavy spin-2 Dark Matter*, *JCAP* **09** (2016) 016 [1607.03497].

[72] T. Rindler-Daller and P.R. Shapiro, *Angular momentum and vortex formation in bose-einstein-condensed cold dark matter haloes: Angular momentum in bec-cdm haloes*, *Monthly Notices of the Royal Astronomical Society* **422** (2012) 135–161.

[73] J. Veltmaat and J.C. Niemeyer, *Cosmological particle-in-cell simulations with ultralight axion dark matter*, *Physical Review D* **94** (2016) .