UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE FÍSICA DE SÃO CARLOS

Noel Araujo Moreira

CoupledDipoles.jl: a Julia package for cold atoms

São Carlos

2023

Noel Araujo Moreira

CoupledDipoles.jl: a Julia package for cold atoms

Thesis presented to the Graduate Program in Physics at the Instituto de Física de São Carlos da Universidade de São Paulo, to obtain the degree of Doctor in Science.

Concentration area: Computational Physics

Advisor: Prof. Dr. Romain Pierre Marcel Bachelard

Corrected version
(Original version available on the Program Unit)

São Carlos
2023

Moreira, Noel Araujo
   CoupledDipoles.jl: a Julia package for cold atoms / Noel Araujo Moreira; advisor Romain Pierre Marcel Bachelard - corrected version -- São Carlos 2023.
   116 p.

   Thesis (Doctorate - Graduate Program in Computational Physics) -- Instituto de Física de São Carlos, Universidade de São Paulo - Brasil , 2023.

   1. Cold atoms. 2. Coupled dipoles. 3. Julia language. I. Bachelard, Romain Pierre Marcel, advisor. II. Title.

To all the days and nights I've dedicated solely to studying.

“All suffering is caused by ignorance.”
Dalai Lama XIV

ABSTRACT

MOREIRA, N.A. **CoupledDipoles.jl**: a Julia package for cold atoms. 2023. 116p. Thesis (Doctor in Science) - Instituto de Física de São Carlos, Universidade de São Paulo, São Carlos, 2023.

Modern physics includes theoretical, experimental, and numerical approaches that often overlap. One such field exemplifying this integration is Cold Atoms, which has witnessed a surge in intriguing discoveries over the past few decades. Not surprisingly, its numerical aspects, such as convergences tolerances, or fastest algorithms, have been largely absent from the literature, with the prevailing notion that equations can be effortlessly solved using conventional techniques. This perception, however, cannot align with reality. Computer simulations in this field have boundaries that are not documented and can affect physical outcomes, but pinpointing them is challenging. We introduce `CoupledDipoles.jl`, a specialized Julia Package designed for simulating interacting cold atoms through various mathematical models. Our package offers a flexible infrastructure that allows for different models (e.g. 2D models, where the effective physics is constraint into a plane) to be incorporated and, in addition, brings guarantees that its core methods have optimal performance. By addressing this unconventional gap in the literature, we aim to shed light on the numerical methods in the field, often overlooked, providing a valuable resource for both newcomers seeking an entry point and experts aiming to enhance their productivity in this domain.

Keywords: Cold atoms. Coupled dipoles. Julia language.

RESUMO

MOREIRA, N.A. **CoupledDipoles.jl**: um pacote Julia para átomos frios. 2023. 116p. Tese (Doutorado em Ciências) - Instituto de Física de São Carlos, Universidade de São Paulo, São Carlos, 2023.

A física moderna inclui abordagens teóricas, experimentais e numéricas que frequentemente se sobrepõem. Um campo que exemplifica essa integração é o dos Átomos Frios, visto que testemunhou um aumento de descobertas intrigantes nas últimas décadas. Não é surpreendente que seus aspectos numéricos, como tolerâncias de convergência ou algoritmos mais rápidos, tenham estado em grande parte ausentes na literatura, com a noção predominante de que equações podem ser facilmente resolvidas usando técnicas convencionais. No entanto, essa percepção não condiz com a realidade. As simulações computacionais nesse campo têm limites que não estão documentados e podem afetar os resultados físicos, mas identificá-los é desafiador. Apresentamos o `CoupledDipoles.jl`, um pacote escrito Julia especializado projetado para simular átomos frios interagindo por meio de vários modelos matemáticos. Nosso pacote oferece uma infraestrutura flexível que permite a incorporação de diferentes modelos (por exemplo, modelos 2D, nos quais a física efetiva é limitada a um plano) e, além disso, garante que seus métodos principais tenham desempenho ótimo. Ao abordar essa lacuna não convencional na literatura, pretendemos esclarecer os métodos numéricos no campo, frequentemente negligenciados, fornecendo um recurso valioso tanto para iniciantes que buscam um ponto de entrada quanto para especialistas que desejam aprimorar sua produtividade nesse domínio.

Palavras-chave: Atomos frios. Dipolos acoplados. Linguagem Julia.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

The phenomenon of spontaneous emission is a cornerstone of quantum optics, governing the quantum mechanical behavior of emitters. It is a fundamental process in which an excited quantum system, such as an atom, undergoes a transition to a lower energy state, releasing a photon.[1] The rate at which spontaneous emission occurs, often quantified through emission probabilities, holds critical importance in a wide array of quantum technologies, ranging from atomic clocks to single-photon sources and quantum memories.[2]

Indeed, in the context of quantum technologies the traditional free-space spontaneous emission probabilities undergo profound alterations when a quantum emitter is situated within an external medium, such as a resonant cavity[3,4] or in proximity to other atoms. This medium-mediated interaction results in the modification of the electromagnetic states accessible to the emitter. It is here that the famed Purcell effect[5] comes into play. The Purcell effect, named after physicist Edward M. Purcell, characterizes how the density of electromagnetic states in a surrounding medium influences spontaneous emission. In essence, the presence of this external environment introduces additional pathways for energy exchange between the emitter and its surroundings. Consequently, this interaction leads to discernible changes in the quantum system's radiative and non-radiative decay rates. Indeed, interference phenomena arise due to the multitude of potential emission channels, influencing the emission dynamics.

Not only cavities but also the presence of other atoms defy predictions made by standard electrodynamics due to collective effects. Notably, collective optical responses deviate significantly from the averaged atom-atom interactions dsecribed in continuous media (dielectric description, for example), and superradiance, introduced by Dicke[6] in 1954, is one of the most well-known effects in this category. It corresponds to the modification of the emission from a two-level atom due to the presence of neighboring atoms. The foundational work by Dicke has laid the groundwork for extensive research in the field of light emission and scattering from ensembles of two-level systems.

To address the challenge of properly modelling the collective emission of light by atomic systems, the Coupled Dipole Equations (CDE) were derived, which accounts for the fact that the atoms interact with common modes of the quantized electromagnetic fields, creating an effective dipole-dipole interaction,[7] from which scattered light intensity is then derived and compared to experiments. For instance, F. Robicheaux has explored line-broadening[8] and polarization[9] in atomic clouds, with recent work on superradiance.[10,11] F. Pinheiro and L. Dal Negro have leveraged the Coupled Dipole Equations for research on structures like Aperiodic Vogel Spirals[12] and hyper uniform structures.[13] R. Kaiser and

W. Guerin have lead investigations into super- and subradiance.[14–16] Finally, S. Skipetrov has notably delved into the disappearance of Anderson localization and its recovery.[17,18]

Studies in this field are usually published in Physics Review A (PRA) or Physics Review Letters (PRL). The PRA guide for authors states that papers *must stand on their own; it must be understandable and convincing without the Supplemental Material.*[19] In limiting the space for publication, it is understandable that certain aspects of the research will not be explicitly mentioned. Consequently, because computer codes are not typically published or shared in physics, the exact numerical methodologies used in these journals have received little attention. On the other hand, communities specializing in numerical methods, such as the Journal of Computational Physics (CompPhy) and the Journal of Open Source Software (JOSS), have specific policies that require authors to share their codes to ensure reproducible results.[20,21] It is important to note that these journals focus primarily on the numeric techniques employed, rather than the specific outcomes in the domain of physics research.

This thesis seeks to address a significant gap in the field, the glaring absence of optimizations on the numerical aspects of Coupled Dipole Equations and their extensions. QuTip[22] and QuantumOptics.jl[23] are packages focused only on simulation of quantum systems and are limited to a small number of atoms. Even supplementary materials fall short of elucidating these complex challenges comprehensively, Robicheaux and Sutherland[9] is one of the few exceptions worth mentioning, which proposes a dedicated iterative method to solve linear systems of equations; Yet, with further testing, we conclude that it is a tool limited to some specific parameters regime. In response, we develop a library of meticulously crafted models, optimized to deliver exceptional speed and controlled accuracy for linear and quantum-truncated regimes. Our primary aim is to empower researchers, enabling them to work more efficiently within this domain, especially for newcomers in the field, who need an entry-point with tested and validated resources. The resulting package was written in Julia language given its unique characteristic of being as fast (when properly optimized) as C or Fortran, and its friendly syntax akin to Python or Matlab.

The speed aspect influences directly productivity on a personal computer, and, if scientists migrate to Cloud Computing to develop their research on CDE, it will further reduce the computational cost. For this reason, we had two core objectives driving our methodology. First, to identify the different simulations utilized in this area, and write a systematic documentation of equations in a standardized notation to enhance clarity and reproducibility. Second, to develop specialized methods tailored to meet the unique demands of these simulations. To ensure the optimization process is thorough and effective, we continually scrutinize and compare various computational tools, relying on profiling and benchmarking as our guiding principles.

Also, we will present the strategies employed to enhance the speed and accuracy of these equations, offering detailed insights into the decision-making process that influenced our results. Importantly, our approach is geared towards improving existing fundamental equations rather than introducing entirely new algorithms. While we aim to provide a valuable reference and benchmarking platform, we do not intend to replicate the existing literature. Last, we will not consider outcomes stemming from packages that deliver results at a slower pace.

This thesis elucidates the theoretical foundations, focusing on the origins of fundamental equations in Chapter 2. Chapter 3 offers insight into the inner mechanisms of our package, outlining its core principles. In Chapter 4, we present a comprehensive array of optimization techniques designed to improve efficiency. Shifting our focus to practical applications, Chapter 5 reveals new physical results in the field of subradiance and localization. Lastly, in Chapter 6, we provide practical resources, including real code implementations and valuable tips for the seamless daily utilization of our package.

2 THEORETICAL BACKGROUND

Our goal is to explore how light interacts with a medium at a macroscopic scale, while taking into account the quantum behavior of the atomic emitters. To do this, we need to use a microscopic model that describes how a collection of atoms interacts with the electromagnetic field. A common example of this phenomenon is induced polarization, which is often discussed in electromagnetism textbooks. When a material is exposed to an external electromagnetic field, it exerts a force on the charged particles inside the material. These particles then oscillate, creating dipole moments within the material. The dipoles generate electric fields that modify the propagation of the total field by interfering with the incident field. To fully understand the atom-light interaction, we have to use quantum mechanics, not just classical electromagnetism. Quantum mechanics explains how atoms and electromagnetic fields interact to produce spectral lines, spontaneous decay, and atom saturation. These effects arise from the specific quantum properties of atoms, such as discrete energy levels, transitions, and stochastic processes like spontaneous emission.

Lehmberg[7] treats the problem starting from a light-matter Hamiltonian, with atoms interacting with quantized modes of the electromagnetic field, and after some approximations, traces out the degrees of freedom of the light, thus reaching a Coupled Dipole Model. We will present a different mathematical approach, using density matrix, to derive the same model, which is a powerful approach to studying ultra-cold physics in both experimental and theoretical settings.

2.1 Model

Consider N identical atoms arranged at different positions $\mathbf{r}_j$. Each atom is treated as having two states: lower state $|g_j\rangle$ and upper state $|e_j\rangle$. The atoms are characterized by a transition frequency ω_a, a decay rate Γ, and a lifetime $\tau = 1/\Gamma$. The system is illuminated by a monochromatic laser field E_L with amplitude E_0, wave vector $\mathbf{k}_0 = \hat{z}$, and frequency ω, leading to a pump-atom detuning $\Delta = \omega_0 - \omega_a$. Atoms are indirectly linked to each other by the electromagnetic field, causing an effective interaction between their electric dipoles.[24]

Figure 1 illustrates a homogeneous spherical distribution of atoms driven by a Gaussian laser field. The shape of the atomic cloud can affect some results, but model used is not restricted to a specific geometry.

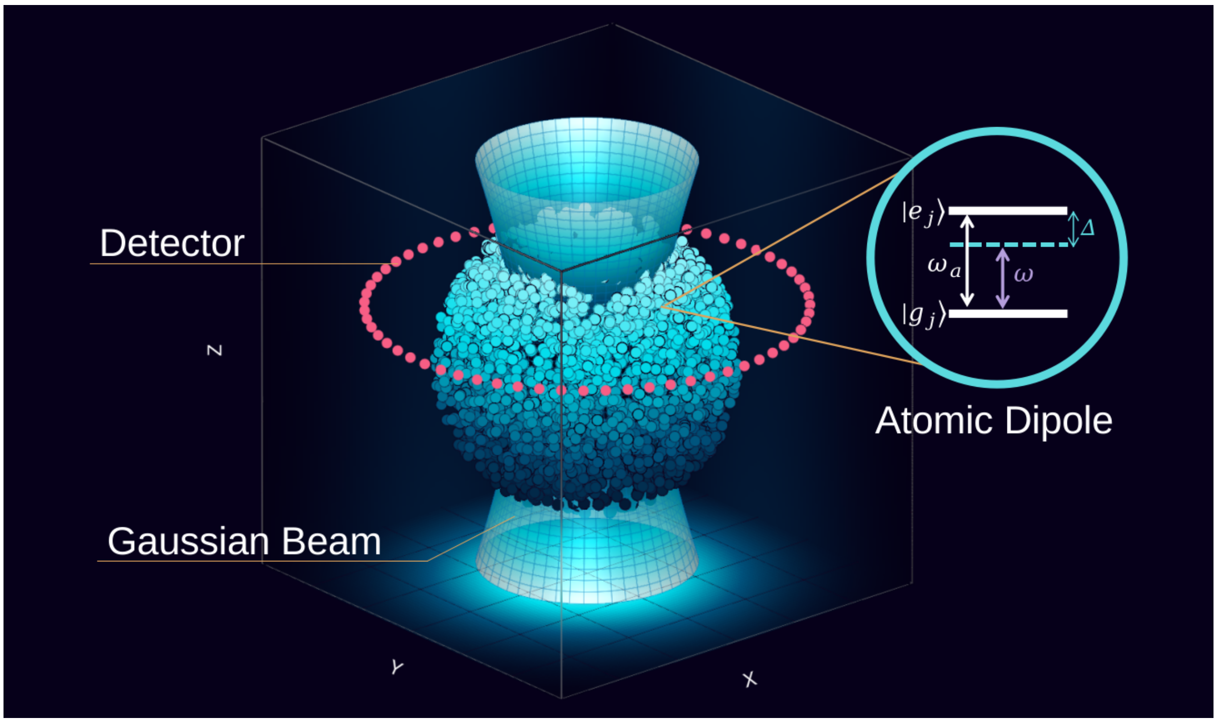


Figure 1 – Cloud of N atoms, each containing $|g_j\rangle$ and $|e_j\rangle$ states, driven by a Gaussian beam of frequency ω. The observable of the system is the scattered intensity, which is measured at some detectors, represented by the ring of points around the cloud.

Source: By the author.

2.1.1 Effective Hamiltonian

Each atom follows the algebra of spin-1/2 angular momentum. To switch between states $|e\rangle$ and $|g\rangle$, jump operators $\hat{\sigma}^- = |g\rangle\langle e|$ and $\hat{\sigma}^+ = |e\rangle\langle g|$ are used. Our study relies on these operators because they play a key role in dipole moment formation.[25] Let us first introduce Bienaimé *et al.*[26]'s Effective Hamiltonian, which simplifies the problem by tracing out the modes of the electromagnetic field[7] and single-excitation approximation.[7]

The Effective Hamiltonian is given by:

$$\mathcal{H}(t) = \sum_{j=1}^{N} \overbrace{-\hbar\frac{\Delta_j}{2}\hat{\sigma}_j^z + \hbar\frac{\Omega_j(t)}{2}(\hat{\sigma}_j^+ + \hat{\sigma}_j^-)}^{H_j(t)} + \sum_{j,m\neq j}^{N} \hbar\Delta_{jm}\hat{\sigma}_j^+\hat{\sigma}_m^-, \tag{2.1}$$

where H_j refers to the single-atom Hamiltonian for atom j. The first term represents the detuning between the frequency of the driving laser and of the transition of atom j, with $\hat{\sigma}^z$ representing the Pauli matrix (population inversion). The second term describes the interaction between the atom and the laser field (drive), where $\hat{\sigma}^+$ and $\hat{\sigma}^-$ are the raising and lowering operators, respectively. The drive of this interaction is modulated by the time-dependent Rabi frequency $\Omega_j(t)$, which governs the absorption and emission of photons from the laser by the atom. The last term accounts for the dipole-dipole interaction between the j-th and m-th atoms, with Δ_{jm} representing the interaction strength.

The time evolution of an open system is governed by the master equation:

$$\frac{d\rho}{dt} = -\frac{i}{\hbar}[\mathcal{H}(t), \rho(t)] + \mathcal{L}[\rho(t)], \tag{2.2}$$

where ρ is the density matrix and $\mathcal{L}$ is the Lindblad operator describing the dissipative dynamics of the system

$$\mathcal{L}[\rho(t)] = \frac{1}{2}\sum_{j,m} \Gamma_{jm}[2\hat{\sigma}_j^- \rho(t)\hat{\sigma}_m^+ - \rho(t)\hat{\sigma}_j^+\hat{\sigma}_m^- - \hat{\sigma}_j^+\hat{\sigma}_m^-\rho(t)]. \tag{2.3}$$

The Lindblad operator is decomposed in a series of collapse operators, with both single-atom and coupled decay terms. These represent the interaction between the system and its environment. Analytical solutions for Equation 2.2 typically exist only for systems consisting of one and two atoms. For larger systems, one typically uses packages such as QuTiP[22] or QuantumOptics.jl,[23] or approximated methods. Norambuena, Tancara and Coto[27] provide a comprehensive overview of numerical techniques in this field. It is worth noting that numerical approaches rely on matrices and vectors to represent states and operators. For example, considering three dipoles labeled as $\hat{\hat{\sigma}}_1$, $\hat{\hat{\sigma}}_2$, and $\hat{\hat{\sigma}}_3$, their fictitious Hamiltonian can be expressed:

$$\hat{H} = \hat{\sigma}_1^- + \hat{\sigma}_2^- + \hat{\sigma}_3^- . \tag{2.4}$$

In the case of a two-level system, the Pauli matrices are a practical representation. However, when dealing with a tensor product of spins, the explicit representation of the tensor product is required. Therefore, Equation 2.4 can be rewritten as:

$$\hat{H} = [\hat{\sigma}_1^- \otimes \mathbf{1}_2 \otimes \mathbf{1}_3] + [\mathbf{1}_1 \otimes \hat{\sigma}_2^- \otimes \mathbf{1}_3] + [\mathbf{1}_1 \otimes \mathbf{1}_2 \otimes \hat{\sigma}_3^-]. \tag{2.5}$$

where $\mathbf{1}_n$ represents the identity matrix for the n-th spin. The size of the Pauli matrices being 2×2, the tensor product operation for N spins yields 2^N elements. Norambuena, Tancara and Coto[27] also estimate the memory requirement for storing a dense matrix as $M = (2^N)^2 \times (8 \text{ bytes}) \times (10^{-9} \text{ Gb/bytes})$. To only represent Equation 2.4, without any dynamics (this is usually not an interesting scenario), and considering a computer with 16Gb of RAM, $N = 14$ or $N = 15$ would represent the upper limit for full quantum mechanical simulations, without employing specialized techniques. In practice, one's interest lies in the quantum state evolving over time, which requires a greater allocation of resources. As a result, in practice, the upper limit is always smaller than predicted.

To overcome the limitations posed by the number of atoms, one can simplify the mathematical framework while retaining relevant information. In particular, if the atomic dipoles linked to the scattered electric field are the only properties of interest, monitoring the full quantum state of the system may not be necessary. In the following section, we will derive the expectation values of the atomic states using the master equation approach.

2.2 Scalar Coupled Dipoles Equations

Alan Costa dos Santos gave valuable insights about the approximations for the equations and derivations in this work. Our collaboration focused on refining and expanding upon the intermediary steps to enhance clarity and facilitate comprehension for the reader.

The solution of Equation 2.2 corresponds to a density matrix at a given time, from which any physical quantity can be computed. Nonetheless, our goal is at first to obtain the equations of motion only for the expectation value of $\langle \hat{\sigma}^{\pm} \rangle$ using the trace with the density matrix, because they provide the electric field emitted by the atom. For each atom j, the expected value of $\langle \hat{\sigma}_j^- \rangle$ will be denoted as β_j:

$$\dot{\beta}_j(t) = \langle \dot{\hat{\sigma}}_j^- \rangle = \mathrm{Tr}\{\dot{\rho}(t)\hat{\sigma}_j^-\} = \mathrm{Tr}\{[\mathcal{H}, \rho(t)]\hat{\sigma}_j^-\} + \mathrm{Tr}\{\mathcal{L}[\rho(t)]\hat{\sigma}_j^-\}, \tag{2.6}$$

where $\mathcal{H}$ and $\mathcal{L}$ denote the unitary and non-unitary part of the dynamics, respectively, defined as[26]

$$\mathcal{H}[\rho(t)] = -\frac{i}{\hbar}\sum_{j=1}^{N}[H_j(t), \rho(t)] - i\sum_{j,m\neq j}^{N}\Delta_{jm}[\hat{\sigma}_j^+\hat{\sigma}_m^-, \rho(t)], \tag{2.7}$$

$$\mathcal{L}[\rho(t)] = \frac{1}{2}\sum_{j,m}\Gamma_{jm}[2\hat{\sigma}_j^-\rho(t)\hat{\sigma}_m^+ - \rho(t)\hat{\sigma}_m^+\hat{\sigma}_j^- - \hat{\sigma}_m^+\hat{\sigma}_j^-\rho(t)], \tag{2.8}$$

with

$$\Delta_{jm} = -\frac{\Gamma}{2}\frac{\cos(k_0|r_j - r_m|)}{k_0|r_j - r_m|}, \tag{2.9}$$

$$\Gamma_{jm} = \Gamma\frac{\sin(k_0|r_j - r_m|)}{k_0|r_j - r_m|}. \tag{2.10}$$

In order to clarify the numerical implementation of Equation 2.6 and to better motivate the approximations which can then be applied to the underlying equations, we dedicate the following subsections to their derivation. The specific form of the interaction, incorporated in Γ_{jm} and Δ_{jm}, will be used only at the end, to make the final equations explicit.

2.2.1 Unitary Part

Our convention is to consistently use j as the primary index for the dynamics of a given atom, while summations describing the interaction are operated on indices l and m. Consequently, only the $\hat{\sigma}_j^-$ term will retain the j label, while in Equation 2.7 and Equation 2.8, there will be a change of index from j to l.

$$\begin{aligned}\mathrm{Tr}\{\mathcal{H}[\rho(t)]\hat{\sigma}_j^-\} &= -\frac{i}{\hbar}\sum_{l=1}^{N}\mathrm{Tr}\{[H_l(t),\rho(t)]\hat{\sigma}_j^-\} - i\sum_{l,m\neq l}^{N}\Delta_{lm}\mathrm{Tr}\{[\hat{\sigma}_l^+\hat{\sigma}_m^-,\rho(t)]\hat{\sigma}_j^-\}\\ &= -\frac{i}{\hbar}\sum_{l=1}^{N}\mathrm{Tr}\{H_l(t)\rho(t)\hat{\sigma}_j^- - \rho(t)H_l(t)\hat{\sigma}_j^-\} - i\sum_{l,m\neq l}^{N}\Delta_{lm}\mathrm{Tr}\{[\hat{\sigma}_l^+\hat{\sigma}_m^-,\rho(t)]\hat{\sigma}_j^-\}.\end{aligned} \tag{2.11}$$

Moving forward, we will omit the explicit dependence on time. Regarding the first term in the above equation, we can express it as follows:

$$\sum_{l=1}^{N}\mathrm{Tr}\{H_l\rho\hat{\sigma}_j^- - \rho H_l\hat{\sigma}_j^-\} = \sum_{l=1}^{N}\mathrm{Tr}\{\rho\hat{\sigma}_j^- H_l - \rho H_l\hat{\sigma}_j^-\} = \mathrm{Tr}\{\rho[\hat{\sigma}_j^-, H_j]\}. \tag{2.12}$$

For the final step, we utilized the commutation relation $[\hat{\sigma}_j^-, H_l] = \delta_{jl}[\hat{\sigma}_j^-, H_l]$. The expression for H_j are the first and second terms on Equation 2.1

$$\begin{aligned}\mathrm{Tr}\{\rho[\hat{\sigma}_j^-, H_j]\} &= \mathrm{Tr}\left\{\rho\left[\hat{\sigma}_j^-, -\hbar\frac{\Delta_j}{2}\hat{\sigma}_j^z + \hbar\frac{\Omega_j}{2}(\hat{\sigma}_j^+ + \hat{\sigma}_j^-)\right]\right\}\\ &= -\frac{\hbar}{2}\Delta_j\mathrm{Tr}\{\rho[\hat{\sigma}_j^-,\hat{\sigma}_j^z]\} + \frac{\hbar}{2}\Omega_j\mathrm{Tr}\{\rho[\hat{\sigma}_j^-,(\hat{\sigma}_j^+ + \hat{\sigma}_j^-)]\}\\ &= -\frac{\hbar}{2}\Delta_j\mathrm{Tr}\{\rho[\hat{\sigma}_j^-,\hat{\sigma}_j^z]\} + \frac{\hbar}{2}\Omega_j\mathrm{Tr}\{\rho[\hat{\sigma}_j^-,\hat{\sigma}_j^+]\}.\end{aligned} \tag{2.13}$$

In the following steps, we will perform various operations involving Pauli matrices for indices j and m. The following relations are used:

$$[\hat{\sigma}_j^\pm,\hat{\sigma}_m^\pm] = 0;\ \{\hat{\sigma}_j^\pm,\hat{\sigma}_m^\pm\} = 2\hat{\sigma}_j^\pm\hat{\sigma}_m^\pm,$$

$$[\hat{\sigma}_j^\pm,\hat{\sigma}_m^\mp] = \pm\delta_{mj}\hat{\sigma}_j^z;\ \{\hat{\sigma}_j^\pm,\hat{\sigma}_m^\pm\} = 2(1-\delta_{mj})\hat{\sigma}_m^\pm\hat{\sigma}_j^\pm,$$

and lastly, $2\hat{\sigma}_j^\pm\hat{\sigma}_j^\mp - 1 = \pm\hat{\sigma}_j^z$, one may proof that

$$[\hat{\sigma}_j^z,\hat{\sigma}_m^\pm] = \pm 2\delta_{mj}\hat{\sigma}_j^\pm;\ \{\hat{\sigma}_j^z,\hat{\sigma}_m^\pm\} = 2[\hat{\sigma}_m^z\hat{\sigma}_j^\pm \mp \delta_{mj}]\hat{\sigma}_j^\pm.$$

These relations allow us to rewrite Equation 2.13 as follows:

$$\begin{aligned}\mathrm{Tr}\{\rho[\hat{\sigma}_l^-, H_l]\} &= -\frac{\hbar}{2}\Delta_j\mathrm{Tr}\{\rho[\hat{\sigma}_j^-,\hat{\sigma}_j^z]\} + \frac{\hbar}{2}\Omega_j\mathrm{Tr}\{\rho[\hat{\sigma}_j^-,\hat{\sigma}_j^+]\}\\ &= -\frac{\hbar}{2}\Delta_l\mathrm{Tr}\{\rho(2\hat{\sigma}_j^-)\} + \frac{\hbar}{2}\Omega_l\mathrm{Tr}\{\rho(-\hat{\sigma}_j^z)\}\\ &= -\hbar\Delta_l\mathrm{Tr}\{\rho\hat{\sigma}_j^-\} - \frac{\hbar}{2}\Omega_j\mathrm{Tr}\{\rho\hat{\sigma}_j^z\}.\end{aligned} \tag{2.14}$$

To complete the derivation of the unitary dynamics, we still need to calculate the second term in Equation 2.11. However, since we have already demonstrated the general procedure, we can now proceed more efficiently by omitting the justification of each step:

$$\sum_{l,m\neq l}^{N}\Delta_{lm}\mathrm{Tr}\{[\hat{\sigma}_l^+\hat{\sigma}_m^-,\rho]\hat{\sigma}_j^-\} = \sum_{l,m\neq l}^{N}\Delta_{lm}[\mathrm{Tr}\{\hat{\sigma}_l^+\hat{\sigma}_m^-\rho\hat{\sigma}_j^-\} - \mathrm{Tr}\{\rho\hat{\sigma}_l^+\hat{\sigma}_m^-\hat{\sigma}_j^-\}]$$

$$
\begin{aligned}
&= \sum_{l,m\neq l}^{N} \Delta_{lm}[\mathrm{Tr}\{\rho\hat{\sigma}_j^-\hat{\sigma}_l^+\hat{\sigma}_m^-\} - \mathrm{Tr}\{\rho\hat{\sigma}_l^+\hat{\sigma}_j^-\hat{\sigma}_m^-\}] \\
&= \sum_{l,m\neq l}^{N} \Delta_{lm}\mathrm{Tr}\{\rho[\hat{\sigma}_j^-,\hat{\sigma}_l^+]\hat{\sigma}_m^-\} \\
&= \sum_{l,m\neq l}^{N} \Delta_{lm}\mathrm{Tr}\{\rho(-\delta_{lj}\hat{\sigma}_j^z)\hat{\sigma}_m^-\} \\
&= -\sum_{m\neq j}^{N} \Delta_{jm}\mathrm{Tr}\{\rho\hat{\sigma}_j^z\hat{\sigma}_m^-\}. \qquad (2.15)
\end{aligned}
$$

In summary, the unitary component of the evolution of $\hat{\sigma}_j^-$ is given by:

$$
\begin{aligned}
\mathrm{Tr}\{\mathcal{H}[\rho(t)]\hat{\sigma}_j^-\} &= -\frac{i}{\hbar}\left(-\hbar\Delta_j\mathrm{Tr}\{\rho\hat{\sigma}_j^-\} - \frac{\hbar}{2}\Omega_j\mathrm{Tr}\{\rho\hat{\sigma}_j^z\}\right) - i\left(-\sum_{j,m\neq j}^{N}\Delta_{jm}\mathrm{Tr}\{\rho\hat{\sigma}_j^z\hat{\sigma}_m^-\}\right) \\
&= i\left(\Delta_j\mathrm{Tr}\{\rho\hat{\sigma}_j^-\} + \frac{1}{2}\Omega_j\mathrm{Tr}\{\rho\hat{\sigma}_j^z\}\right) + i\sum_{m\neq j}^{N}\Delta_{jm}\mathrm{Tr}\{\rho\hat{\sigma}_j^z\hat{\sigma}_m^-\}. \qquad (2.16)
\end{aligned}
$$

2.2.2 Lindbladian component

The dissipative part of the system evolution is obtained by plugging the Equation 2.8 into Equation 2.6. Once again, the expressions can be considerably simplified by using the relations between Pauli matrices mentioned previously.

$$
\begin{aligned}
\mathrm{Tr}\{\mathcal{L}[\rho]\hat{\sigma}_j^-\} &= \frac{1}{2}\sum_{l,m\neq l}\Gamma_{lm}[2\cdot\mathrm{Tr}\{\hat{\sigma}_l^-\rho\hat{\sigma}_m^+\hat{\sigma}_j^-\} - \mathrm{Tr}\{\rho\hat{\sigma}_m^+\hat{\sigma}_l^-\hat{\sigma}_j^-\} - \mathrm{Tr}\{\hat{\sigma}_m^+\hat{\sigma}_l^-\rho\hat{\sigma}_j^-\}] \\
&= \frac{1}{2}\sum_{l,m\neq l}\Gamma_{lm}[2\cdot\mathrm{Tr}\{\rho\hat{\sigma}_m^+\hat{\sigma}_j^-\hat{\sigma}_l^-\} - \mathrm{Tr}\{\rho\hat{\sigma}_m^+\hat{\sigma}_l^-\hat{\sigma}_j^-\} - \mathrm{Tr}\{\hat{\sigma}_m^+\hat{\sigma}_l^-\rho\hat{\sigma}_j^-\}] \\
&= \frac{1}{2}\sum_{l,m\neq l}\Gamma_{lm}[\mathrm{Tr}\{\rho\hat{\sigma}_m^+\hat{\sigma}_j^-\hat{\sigma}_l^-\} + \mathrm{Tr}\{\rho\hat{\sigma}_m^+\hat{\sigma}_j^-\hat{\sigma}_l^-\} - \mathrm{Tr}\{\rho\hat{\sigma}_m^+\hat{\sigma}_j^-\hat{\sigma}_l^-\} - \mathrm{Tr}\{\hat{\sigma}_m^+\hat{\sigma}_j^-\rho\hat{\sigma}_l^-\}] \\
&= \frac{1}{2}\sum_{l,m\neq l}\Gamma_{lm}[\mathrm{Tr}\{\hat{\sigma}_l^-\rho\hat{\sigma}_m^+\hat{\sigma}_j^-\} - \mathrm{Tr}\{\hat{\sigma}_l^-\rho\hat{\sigma}_m^+\hat{\sigma}_j^-\} + \mathrm{Tr}\{\rho\hat{\sigma}_m^+\hat{\sigma}_j^-\hat{\sigma}_l^-\} - \mathrm{Tr}\{\rho\hat{\sigma}_m^+\hat{\sigma}_j^-\hat{\sigma}_l^-\}] \\
&= \frac{1}{2}\sum_{l,m\neq l}\Gamma_{lm}[\mathrm{Tr}\{\hat{\sigma}_l^-\rho[\hat{\sigma}_m^+,\hat{\sigma}_j^-]\} + \mathrm{Tr}\{\rho\hat{\sigma}_m^-[\hat{\sigma}_l^-,\hat{\sigma}_j^-]\}] \\
&= \frac{1}{2}\sum_{l,m\neq l}\Gamma_{lm}[\mathrm{Tr}\{\hat{\sigma}_l^-\rho\delta_{jm}\hat{\sigma}_m^z\}] \\
&= \frac{1}{2}\sum_{l}\Gamma_{lj}[\mathrm{Tr}\{\hat{\sigma}_l^-\rho\hat{\sigma}_j^z\}] \\
&= \frac{\Gamma_{jj}}{2}\mathrm{Tr}\{\hat{\sigma}_j^-\rho\hat{\sigma}_j^z\} + \frac{1}{2}\sum_{l\neq j}\Gamma_{lj}[\mathrm{Tr}\{\hat{\sigma}_l^-\rho\hat{\sigma}_j^z\}] \\
&= \frac{\Gamma_{jj}}{2}\mathrm{Tr}\{\rho(-\hat{\sigma}_j^-)\} + \frac{1}{2}\sum_{l\neq j}\Gamma_{lj}[\mathrm{Tr}\{\rho\hat{\sigma}_l^-\hat{\sigma}_j^z\}]
\end{aligned}
$$

$$= -\frac{\Gamma_{jj}}{2}\mathrm{Tr}\{\rho\hat{\sigma}_j^-\} + \frac{1}{2}\sum_{j\neq l}\Gamma_{jl}[\mathrm{Tr}\{\rho\hat{\sigma}_l^-\hat{\sigma}_j^z\}]. \tag{2.17}$$

2.2.3 Gathering Results

By joining Equation 2.16 and Equation 2.17, we get an explicit form of Equation 2.6. For convenience, we retain the primary index into j, and change all secondary indices as m, so we get

$$\begin{aligned}\langle\dot{\hat{\sigma}}_j^-\rangle &= i\left(\Delta_j\mathrm{Tr}\{\rho\hat{\sigma}_j^-\} + \frac{1}{2}\Omega_j\mathrm{Tr}\{\rho\hat{\sigma}_j^z\}\right) + i\sum_{m\neq j}^{N}\Delta_{jm}\mathrm{Tr}\{\rho\hat{\sigma}_j^z\hat{\sigma}_m^-\} - \frac{\Gamma_{jj}}{2}\mathrm{Tr}\{\rho\hat{\sigma}_j^-\} \\ &+ \frac{1}{2}\sum_{m\neq j}\Gamma_{jm}[\mathrm{Tr}\{\rho\hat{\sigma}_m^-\hat{\sigma}_j^z\} \qquad (2.18)\\ &= \left(i\Delta_j - \frac{\Gamma_{jj}}{2}\right)\mathrm{Tr}\{\rho\hat{\sigma}_j^-\} + i\frac{\Omega_j}{2}\mathrm{Tr}\{\rho\hat{\sigma}_j^z\} + i\sum_{m\neq j}^{N}\left[\Delta_{jm} - i\frac{\Gamma_{jm}}{2}\right]\mathrm{Tr}\{\rho\hat{\sigma}_j^z\hat{\sigma}_m^-\}. \qquad (2.19)\end{aligned}$$

Now, we will introduce the already mentioned notation $\langle\dot{\hat{\sigma}}_j^-\rangle = \beta_j$. Additionally, we will make the approximations that the laser pump has weak saturation, resulting in the excited population $(1+\hat{\sigma}_j^z)/2 = \hat{\sigma}_j^+\hat{\sigma}_j^-$ being close to zero, i.e., $\hat{\sigma}_j^z \approx -1$. Consequently, $\mathrm{Tr}\{\rho\hat{\sigma}_l^z\hat{\sigma}_m^-\}$ can be substituted by $-\mathrm{Tr}\{\rho\hat{\sigma}_m^-\}$, resulting in

$$\begin{aligned}\dot{\beta}_j(t) &= \left(i\Delta_j - \frac{\Gamma_{jj}}{2}\right)\beta_j + i\frac{\Omega_j}{2}(-1) + i\sum_{m\neq j}^{N}\left[\Delta_{jm} - i\frac{\Gamma_{jm}}{2}\right](-1)\mathrm{Tr}\{\rho\hat{\sigma}_m^-\} \\ &= \left(i\Delta_j - \frac{\Gamma_{jj}}{2}\right)\beta_l - i\frac{\Omega_j}{2} - i\sum_{m\neq j}^{N}\left[\Delta_{jm} - i\frac{\Gamma_{jm}}{2}\right]\beta_m. \qquad (2.20)\end{aligned}$$

Substituting $\Gamma_{jj} = \Gamma$, $\Delta_j = \Delta$ and using Equation 2.9

$$\begin{aligned}\frac{d\beta_j(t)}{dt} &= \left(i\Delta - \frac{\Gamma}{2}\right)\beta_j(t) - i\frac{\Omega_j(t)}{2} - i\sum_{m\neq j}^{N}\left[\frac{-\Gamma}{2}\frac{\cos(k_0r_{jm})}{k_0r_{jm}} - \frac{i}{2}\Gamma\frac{\sin(k_0r_{jm})}{k_0r_{jm}}\right]\beta_m(t). \\ &= \left(i\Delta - \frac{\Gamma}{2}\right)\beta_j(t) - i\frac{\Omega_j(t)}{2} + i\frac{\Gamma}{2}\sum_{m\neq j}^{N}\left[\frac{\cos(k_0r_{jm})}{k_0r_{jm}} + i\frac{\sin(k_0r_{jm})}{k_0r_{jm}}\right]\beta_m(t),\end{aligned}$$

We obtain the Coupled-Dipole Equation

$$\frac{d\beta_j(t)}{dt} = \left(i\Delta - \frac{\Gamma}{2}\right)\beta_j(t) - i\frac{\Omega_j(t)}{2} + i\frac{\Gamma}{2}\sum_{m\neq j}^{N}\frac{e^{ik_0r_{jm}}}{k_0r_{jm}}\beta_m(t). \tag{2.21}$$

We note that Svidzinsky, Chang and Scully[28] obtained Equation 2.21 from pure classical arguments, meaning that Equation 2.21 neglects any quantum fluctuations interpretation between the atoms. The impact of such simplifications was studied by Williamson and Ruostekoski.[29] Due to its linear nature, Equation 2.21 is more suitable for simulation in a matrix format

$$\frac{d\vec{\beta}}{dt} = G\vec{\beta} + \vec{\Omega} \tag{2.22}$$

with

$$G_{jm} = \left(-\frac{\Gamma}{2} + i\Delta\right)\mathbb{1}_{jm} + i\frac{\Gamma}{2}\sum_{j}^{N}\sum_{m\neq j}^{N}\frac{e^{ik_0 r_{jm}}}{k_0 r_{jm}}, \tag{2.23}$$

$$\vec{\Omega}(t) = -i\frac{\Omega_j(t)}{2}. \tag{2.24}$$

We will refer to the Coupled-Dipole Model and its equations $\mathrm{Tr}\{\rho\hat{\sigma}_j^-\}$ as `Scalar` Model. To consider polarization along the direction $\mu = [x, y, z]$, the `Vectorial` Model uses $3N$ coupled-dipole equations for the system[30]

$$\frac{d\beta_j^{\mu}(t)}{dt} = \left(i\Delta - \frac{\Gamma}{2}\right)\beta_j^{\mu}(t) - i\frac{\Omega_j^{\mu}(t)}{2} + i\frac{\Gamma}{2}\sum_{m\neq j}^{N} G_{\mu,\eta}(\mathbf{r}_{jm})\beta_m^{\eta}(t). \tag{2.25}$$

where the interaction matrix $G_{\mu,\eta}(\mathbf{r})$ has two usual representations given by

$$G_{\mu,\eta}(\mathbf{r}) = \frac{3}{2}\frac{e^{ik_0 r}}{k_0 r}\left[(\delta_{\mu,\eta} - \hat{r}_\mu\hat{r}_\eta) + (\delta_{\mu,\eta} - 3\hat{r}_\mu\hat{r}_\eta)\left(\frac{i}{k_0 r} - \frac{1}{(k_0 r)^2}\right)\right], \tag{2.26}$$

$$\overleftrightarrow{\mathrm{G}}(\mathbf{r}) = \frac{3}{2}\frac{e^{ik_0 r}}{k_0 r}\left[\left(1 + \frac{i}{k_0 r} - \frac{1}{(k_0 r)^2}\right)\overleftrightarrow{\mathrm{I}} + \left(-1 - \frac{3i}{k_0 r} + \frac{3}{(k_0 r)^2}\right)\frac{\mathbf{r}\otimes\mathbf{r}}{r^2}\right], \tag{2.27}$$

with unit vector $\hat{\mathbf{r}} = \mathbf{r}/r = [\hat{r}_x, \hat{r}_y, \hat{r}_z]$. The Identify Operator $\overleftrightarrow{\mathrm{I}}$ has the same dimensions of the tensor product $\mathbf{r}\otimes\mathbf{r}$. Also, the laser field, Ω_j^{μ}, has a constraint that its polarization and direction of propagation must be orthogonal.

2.3 Observable

One physical quantity of interest is the radiated electric field at a given position in space, denoted as $\hat{E}(\mathbf{r}, t)$. This field allows us to calculate the measurable light intensity, $I = \langle \hat{E}\hat{E}^{+}\rangle >$, which, in the single-photon approximation, is obtained from the sets of β_j. The Effective Hamiltonian in Equation 2.1 is a simplification, where the quantized electromagnetic fields have been traced out from the dynamics, yet it can be used to recover the field E. See the works of Bienaime *et al.*[31] and Castro[30] for details. According to their findings, the electric field E can be expressed as the sum of individual fields scattered by the N atoms located at positions $\mathbf{r}_j$.

$$E(\mathbf{r}, t) = i\frac{\Gamma}{2}\sum_{j}^{N}\frac{e^{ik_0|\mathbf{r}-\mathbf{r}_j|}}{k_0|\mathbf{r}-\mathbf{r}_j|}\beta_j(t). \tag{2.28}$$

The intensity can be calculated from Equation 2.28 as follows:

$$\begin{aligned} I(\mathbf{r}, t) &= |E(\mathbf{r}, t)|^2 \\ &= \left| i\frac{\Gamma}{2}\sum_{j}^{N}\frac{e^{ik_0|\mathbf{r}-\mathbf{r}_j|}}{k_0|\mathbf{r}-\mathbf{r}_j|}\beta_j(t)\right|^2 \end{aligned} \tag{2.29}$$

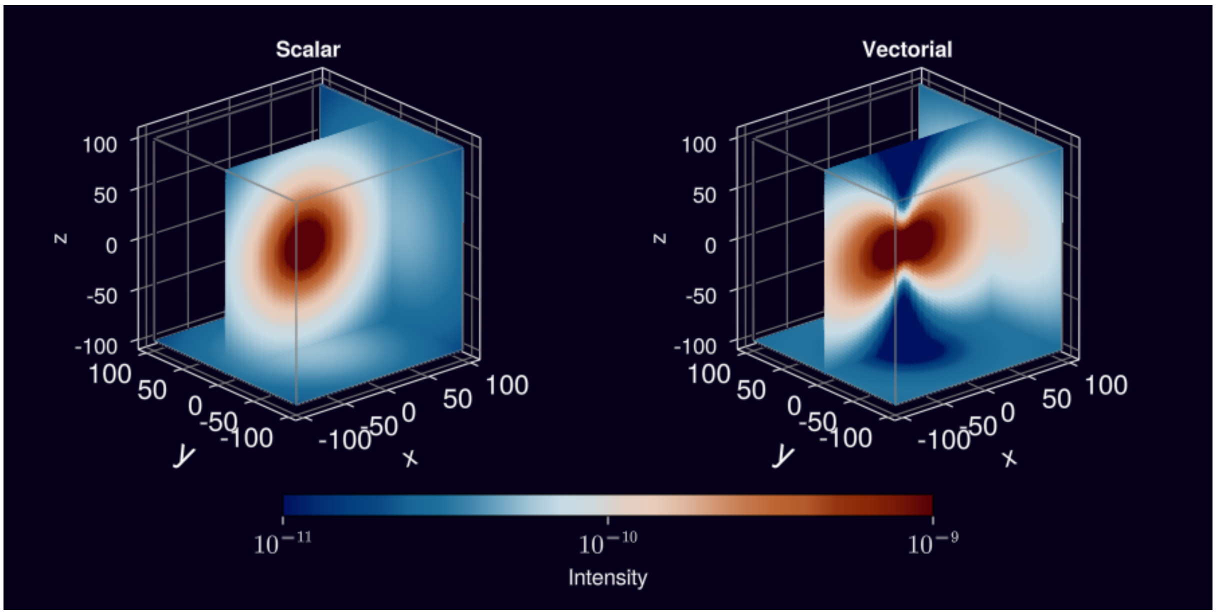


Figure 2 – Comparison of `Scalar` and `Vectorial` radiated fields. The former shows the angular dependence of radiation expected from a dipole emission.

Source: By the author.

$$= \frac{\Gamma^2}{4k_0^2}\sum_j^N\sum_m^N \frac{e^{ik_0(|\mathbf{r}-\mathbf{r}_j|-|\mathbf{r}-\mathbf{r}_m|)}}{|\mathbf{r}-\mathbf{r}_j||\mathbf{r}-\mathbf{r}_m|}\beta_j^*(t)\beta_m(t). \tag{2.30}$$

Note that evaluating the double sum in Equation 2.30 requires $O(N^2)$ operations, whereas computing Equation 2.29 only requires $O(N)$. Thus, using Equation 2.30 is a slower approach, and should be avoided due to its computational complexity.

For `Vectorial` Model the equations are similar, they just take into consideration each component[30]

$$E_\mu(\mathbf{r},t) = i\frac{\Gamma}{2}\sum_j\sum_\eta G_{\mu,\eta}(\mathbf{r}-\mathbf{r}_j)\beta_j^\eta(t), \tag{2.31}$$

$$I(\mathbf{r},t) = |\mathbf{E}|^2 = \sum_\mu |E_\mu|^2. \tag{2.32}$$

2.4 Higher Order Correlation

The Coupled Dipole Equation in Equation (2.21) assumes a weak external pump, so a single photon can be assumed to be present in the system. However, as the saturation parameter increases, the linear regime approximation no longer holds, and the simplification $\langle\hat{\sigma}^z\rangle \approx -1$ is no longer valid. In such cases, the first step would be to compute the dynamics for $\mathrm{Tr}\{\rho\hat{\sigma}_l^z\hat{\sigma}_m^-\}$ using a similar process as the one used for $\mathrm{Tr}\{\rho\hat{\sigma}_j^-\}$. The result also will depend on another joint expectation value, in this case, $\langle\hat{\sigma}_j^z\rangle$ will depend on $\langle\hat{\sigma}_j^+\hat{\sigma}_m^-\rangle$.

We denote $\langle\hat{\sigma}^z\rangle$ as a single-atom expectation value since it depends on a single operator. The two-atom expectation has a joint expectation value of two operators, such as $\langle\hat{\sigma}_j^+\hat{\sigma}_m^-\rangle$. If one derives the dynamical equations for $\langle\hat{\sigma}_j^+\hat{\sigma}_m^-\rangle$, one would find that it also depends on a three-body correlation, and so on. This procedure produces a hierarchy of coupled linear differential equations[32] similar to the scheme

$$\begin{aligned}
\frac{d\langle 1\rangle}{dt} &= \langle 1\rangle + \langle 2\rangle,\\
\frac{d\langle 2\rangle}{dt} &= \langle 1\rangle + \langle 2\rangle + \langle 3\rangle,\\
&\vdots\\
\frac{d\langle n\rangle}{dt} &= \langle 1\rangle + \langle 2\rangle \ldots + \langle n\rangle + \langle n+1\rangle,\\
&\vdots
\end{aligned}$$

where $\langle n\rangle$ here refers to n-body correlation. The Coupled Dipoles Equation considered only the single-atom expectation value and neglected higher-order terms. To achieve this, the terms of order $\langle n \geq 2\rangle$ were assumed to be negligible, and the hierarchy was truncated by setting $\langle n = 2\rangle = 0$. This truncation technique is known as the Cumulant Expansion.[33] In summary, the method allows one to compute cumulants of an operator, $\langle A\rangle_c$, using its expectation values, creating what are called Ursell functions[34] in statistical physics:

$$\langle A\rangle_c = \langle A\rangle, \tag{2.33}$$

$$\langle AB\rangle_c = \langle AB\rangle - \langle A\rangle\langle B\rangle, \tag{2.34}$$

$$\langle ABC\rangle_c = \langle ABC\rangle - \langle AB\rangle\langle C\rangle - \langle AC\rangle\langle B\rangle - \langle BC\rangle\langle A\rangle + 2\langle A\rangle\langle B\rangle\langle C\rangle. \tag{2.35}$$

$$\vdots$$

The truncation procedure involves selecting a maximum order, denoted as $\langle n+1\rangle_c$, and setting it to zero. This truncation effectively eliminates terms of higher order, resulting in a combination of lower-order expectation values:

$$\langle AB\rangle \approx \langle A\rangle\langle B\rangle, \tag{2.36}$$

$$\langle ABC\rangle \approx \langle AB\rangle\langle C\rangle + \langle AC\rangle\langle B\rangle + \langle BC\rangle\langle A\rangle - 2\langle A\rangle\langle B\rangle\langle C\rangle, \tag{2.37}$$

$$\vdots$$

We stress that Equations (2.36) and Equation 2.37 are different approximations. For example, if one uses Equation (2.36), one automatically sets Equation (2.37) to zero - next subsections exemplifies this point. Also, when considering higher degrees of correlation, the system's complexity increases as the number of coupled equations grows. However, it is not always evident how these additional equations affect the system's overall behavior. In a study by Sánchez-Barquilla, Silva and Feist,[35] it was found that incorporating even higher correlations did not enhance the accuracy of the results in certain scenarios, but produced an unstable system of equations. Consequently, there is a need to strike a balance between computational complexity and the potential gain in accuracy when dealing with higher degrees of correlation. Nevertheless, a dedicated study needs to be performed for each system and regime.

Furthermore, the computation of these expressions can be error-prone, and there have been recent developments in performing symbolic computations to mitigate this issue.[36]

2.4.1 Mean Field Equations

We referred to the Coupled-Dipole Model, in Equation 2.21, as `Scalar` Model, and its natural extension, where one computes $\langle\hat{\sigma}_j^z\rangle = \mathrm{Tr}\{\rho\hat{\sigma}_j^z\}$ and $\langle\hat{\sigma}_j^-\hat{\sigma}_m^z\rangle \approx \langle\hat{\sigma}_j^-\rangle\langle\hat{\sigma}_m^z\rangle$, will be referred as `MeanField` Model. The set of $2N$ equations for the latter reads[37,38]

$$W_j = \frac{\Omega_j}{2} + i\frac{\Gamma}{2}\sum_{m\neq j}^{N}\frac{e^{ik_0|\mathbf{r}_j-\mathbf{r}_m|}}{k_0|\mathbf{r}_j-\mathbf{r}_m|}\langle\hat{\sigma}_m^-\rangle,$$

$$\frac{d\langle\hat{\sigma}_j^-\rangle}{dt} = \left(i\Delta - \frac{\Gamma}{2}\right)\langle\hat{\sigma}_j^-\rangle + iW_j\langle\hat{\sigma}_j^z\rangle, \tag{2.38}$$

$$\frac{d\langle\hat{\sigma}_j^z\rangle}{dt} = -\Gamma(1+\langle\hat{\sigma}_j^z\rangle) - 4\Im(\langle\hat{\sigma}_j^-\rangle W_j^*), \tag{2.39}$$

where the symbol '$\Im(x)$' represents the imaginary part of x.

The Electric Field is carried out using $\langle\hat{\sigma}_j^-\rangle$ from Equation 2.38 into (2.28). However, the intensity demands $\langle\hat{\sigma}_j^z\rangle$, and since $\langle\hat{\sigma}_j^z\rangle = 2\langle\hat{\sigma}_j^+\rangle\langle\hat{\sigma}_j^-\rangle - 1$, we can make a substitution for $j = m$:

$$I(\mathbf{r}) = \frac{\Gamma^2}{4k_0^2}\left[\sum_j^N\sum_{m\neq j}^N\frac{e^{ik_0(|\mathbf{r}-\mathbf{r}_j|-|\mathbf{r}-\mathbf{r}_m|)}}{|\mathbf{r}-\mathbf{r}_j||\mathbf{r}-\mathbf{r}_m|}\langle\hat{\sigma}_j^+\rangle\langle\hat{\sigma}_j^-\rangle + \sum_j^N\frac{1+\langle\hat{\sigma}_j^z\rangle}{2|\mathbf{r}-\mathbf{r}_j|^2}\right]. \tag{2.40}$$

We prefer to re-write Equation 2.40 into Equation 2.29 format, which allows also allows to define the *Coherent* and *Incoherent* components of the light as

$$I = I_{\text{coh}} + I_{\text{inc}} = \left|i\frac{\Gamma}{2}\sum_j^N\frac{e^{ik_0|\mathbf{r}-\mathbf{r}_j|}}{k_0|\mathbf{r}-\mathbf{r}_j|}\langle\hat{\sigma}_j^-\rangle\right|^2 + \frac{\Gamma^2}{4k_0^2}\left[\sum_j^N -\frac{|\langle\hat{\sigma}_j^-\rangle|^2}{|\mathbf{r}-\mathbf{r}_j|^2} + \frac{1+\langle\hat{\sigma}_j^z\rangle}{2|\mathbf{r}-\mathbf{r}_j|^2}\right]. \tag{2.41}$$

2.4.2 Quantum Pair Correlation Equations

Turning our attention to the two-atom correlators, we show how the complexity of the equations grows as the order of the correlator increases. These equations used in this project were computed by Nicolla Umberto Cesare during a prior collaboration.

We refer to the set of a set of $2N + 4N^2$ equations as `PairCorrelation` Model. Using $j \neq m$ for all equations, $G_{jm} = -ie^{-ik_0|\mathbf{r}_{jm}|}/k_0|\mathbf{r}_{jm}|$, $\Gamma_{jm} = \Re(G_{jm})$, $\Omega_j^- = \Omega_j$, and

$\Omega_j^+ = (\Omega_j^-)^*$ the equations are given as follows:

$$\begin{aligned}
\frac{d}{dt}\langle\sigma_j^-\rangle &= \left(i\Delta - \frac{\Gamma}{2}\right)\langle\sigma_j^-\rangle + \frac{i\Omega_j^-}{2}\langle\sigma_j^z\rangle + \frac{\Gamma}{2}\sum_{m\neq j}^{N} G_{jm}\langle\sigma_j^z\sigma_m^-\rangle, && (2.42)\\
\frac{d}{dt}\langle\sigma_j^z\rangle &= i\left(\Omega_j^+\langle\sigma_j^-\rangle - \text{c.c.}\right) - \Gamma(1+\langle\sigma_j^z\rangle) - \Gamma\sum_{m\neq j}^{N}[G_{jm}\langle\sigma_j^+\sigma_m^-\rangle + \text{c.c.}], && (2.43)\\
\frac{d}{dt}\langle\sigma_j^z\sigma_m^-\rangle &= (i\Delta - 3\Gamma/2)\langle\sigma_j^z\sigma_m^-\rangle - \Gamma\langle\sigma_m^-\rangle + i\Omega_j^+\langle\sigma_j^-\sigma_m^-\rangle - i\Omega_j^-\langle\sigma_j^+\sigma_m^-\rangle + i\frac{\Omega_m^-}{2}\langle\sigma_j^z\sigma_m^z\rangle\\
&- \Gamma\sum_{k\neq\{j,m\}}\left[G_{jk}\langle\sigma_j^+\sigma_m^-\sigma_k^-\rangle + G_{jk}^\dagger\langle\sigma_k^+\sigma_m^-\sigma_j^-\rangle\right] + \frac{\Gamma}{2}\sum_{k\neq\{j,m\}} G_{mk}\langle\sigma_m^z\sigma_j^z\sigma_k^-\rangle\\
&- \Gamma\Gamma_{jm}\langle\sigma_j^-\sigma_m^z\rangle - \frac{\Gamma}{2}G_{jm}^\dagger\langle\sigma_j^-\rangle, && (2.44)\\
\frac{d}{dt}\langle\sigma_j^+\sigma_m^-\rangle &= -\Gamma\langle\sigma_j^+\sigma_m^-\rangle - \frac{i}{2}\left(\Omega_j^+\langle\sigma_j^z\sigma_m^-\rangle - \Omega_m^-\langle\sigma_j^+\sigma_m^z\rangle\right)\\
&+ \frac{\Gamma}{2}\sum_{k\neq\{j,m\}}\left[G_{jk}^\dagger\langle\sigma_k^+\sigma_m^-\sigma_j^z\rangle + G_{mk}\langle\sigma_j^+\sigma_k^-\sigma_m^z\rangle\right]\\
&+ \frac{\Gamma}{4}(G_{jm}\langle\sigma_m^z\rangle + G_{jm}^\dagger\langle\sigma_j^z\rangle) + \frac{\Gamma}{2}\Gamma_{jm}\langle\sigma_j^z\sigma_m^z\rangle, && (2.45)\\
\frac{d}{dt}\langle\sigma_j^-\sigma_m^-\rangle &= (2i\Delta - \Gamma)\langle\sigma_j^-\sigma_m^-\rangle + \frac{i}{2}\left(\Omega_j^-\langle\sigma_j^z\sigma_m^-\rangle + \Omega_m^-\langle\sigma_m^z\sigma_j^-\rangle\right)\\
&+ \frac{\Gamma}{2}\sum_{k\neq\{j,m\}}\left[G_{jk}\langle\sigma_j^z\sigma_m^-\sigma_k^-\rangle + G_{mk}\langle\sigma_m^z\sigma_j^-\sigma_k^-\rangle\right], && (2.46)\\
\frac{d}{dt}\langle\sigma_j^z\sigma_m^z\rangle &= -\Gamma(\langle\sigma_j^z\rangle + \langle\sigma_m^z\rangle + 2\langle\sigma_j^z\sigma_m^z\rangle) + i\left(\Omega_j^+\langle\sigma_m^z\sigma_j^-\rangle + \Omega_m^+\langle\sigma_j^z\sigma_m^-\rangle + c.c.\right)\\
&- \Gamma\sum_{k\neq\{j,m\}}\left[G_{jk}\langle\sigma_j^+\sigma_m^z\sigma_k^-\rangle + G_{mk}\langle\sigma_j^z\sigma_j^+\sigma_k^-\rangle + c.c.\right]\\
&+ 2\Gamma\Gamma_{jm}\left(\langle\sigma_j^+\sigma_m^-\rangle + c.c.\right). && (2.47)
\end{aligned}$$

All correlations of three operators are then simplified using Equation 2.37, and the evolution of some correlators is not defined explicitly as differential equations since they are available from simple properties:

$$\begin{aligned}
\langle\sigma_j^-\sigma_m^z\rangle &= \langle\sigma_m^z\sigma_j^-\rangle, && (2.48)\\
\langle\sigma_j^+\sigma_m^z\rangle &= \langle\sigma_m^z\sigma_j^-\rangle^*, && (2.49)\\
\langle\sigma_z^+\sigma_m^+\rangle &= \langle\sigma_m^+\sigma_j^z\rangle = \langle\sigma_j^-\sigma_m^z\rangle^*. && (2.50)
\end{aligned}$$

The electric field can be calculated similarly to the Mean Field approach by using the values of $\langle\sigma_j^-\rangle$ obtained from Equation 2.42 and inserting them into Equation 2.28. On the other hand, the intensity field requires the use of Equation 2.30 along with Equation 2.43 and Equation 2.45, since two-body correlators are no longer factorized

$$I(\mathbf{r},t) = \frac{\Gamma^2}{4k_0^2}\left[\sum_j^N\sum_{m\neq j}^N \frac{e^{ik_0(|\mathbf{r}-\mathbf{r}_j|-|\mathbf{r}-\mathbf{r}_m|)}}{|\mathbf{r}-\mathbf{r}_j||\mathbf{r}-\mathbf{r}_m|}\langle\sigma_j^+\sigma_m^-\rangle + \sum_j^N \frac{1+\langle\sigma_j^z\rangle}{2|\mathbf{r}-\mathbf{r}_j|^2}\right]. \qquad (2.51)$$

We note that since the intensity is no longer the square of the electric field, as it was in the single-photon regime, the computation of the intensity is now of complexity $O(N^2)$. Since Equation 2.51 has larger memory requirements, due to the number of equations, the number of atoms is typically $N < 200$, and there are no benefits to optimizations with parallelism, for example. A last comment about the implementation is that $j \neq m$ implies that the matrix G_{jm} has a diagonal with zeros.

3 PACKAGE ARCHITECTURE

3.1 Why Julia

The choice of programming language is extremely important in scientific and computational programming. Due to its practicality, Julia has gained popularity among different projects, such as Fluid Mechanics Modelling,[39] High Energy Physics,[40] Dynamical Systems,[41] Differentiable Programming.[42] It intends to solve the industry's known problem of using different programming languages on a project. Usually, a typical project starts with user-friendly languages like Python or R and then switches to languages like C/C++ or Fortran for better performance. Julia makes it easy for developers to move from creating prototypes to deploying them using one language framework.

Compilers excel in optimizing fundamental data types, like Float64, which adhere to the IEEE 754 standard.[43] Julia uses a frond-end to LLVM compiler[44] and compiles codes during its execution, a process known as JIT (Just-in-time) Compilation. Julias secret key element for speed is a combination of JIT compilation with a strong type system.[45] In a nutshell, a data type system ensures that a function and any function calls within it maintain stable types throughout their execution. Let us look at the multiplication operator `*` in the expression `c = a*b`. It is a type-stable function. In Julia, `*` can mean different things depending on the input types, but the compiler can choose the right method for `*` since it knows the types of `a` and `b` beforehand. By using the cascade effect, the compiler can propagate type information and optimize the code like C or Fortran. Developers can check the compilation stages at any time and verify the machine code produced using `@code_llvm` or `@code_native`.

Julia also offers meta-programming and code inspection to give developers more flexibility. The `ModellingToolkit.jl` package is an example of how useful this feature can be. It simplifies equations before compiling code, resulting in substantial performance improvements. When `MATLAB Simulink` was converted to Julia code, it resulted in a speedup of over 15,000x for Modeling Spacecraft Separation Dynamics, according to NASA.[46]

The installation process is easy and saves developers the trouble of managing dependencies and setting up toolchains.[47] This is especially helpful for Windows users. Also, Julia makes it easy-to-use Unicode and LaTeX characters for math projects. These features help scientists communicate better by enabling them to translate algorithms and equations directly into Julia's code.

Last, Julia is renowned for implementing the Multiple Dispatch programming paradigm. Within this paradigm, the functions behaviors are determined based on two or more argument types. This feature makes it easier for different packages to collaborate, even if they were not originally meant to. The code example in Figure 3 demonstrates how to apply these principles to determine the volume of different atomic cloud shapes.

Establishing data types involves creating both **Abstract types** and **Concrete structures**.[48] Two concrete structures, namely `Cube` and `Sphere`, are derived from the abstract type `AtomicCloud`. Abstract types cannot be instantiated, they serve as the basis for a conceptual hierarchy, while Concrete Types are created with `struct`s keyword and organize data of each object. Cubes and spheres are defined by their side length and radius, respectively, as shown in the code.

The `<:` operator is employed to denote subtype relationships between different types. It serves as a tool for establishing inheritance and helps to structure the type hierarchy. For instance, `A <: B` means `A` is a subtype of `B` and objects of type `A` can be used wherever objects of type `B` are expected.

The `::` operator is used to specify the expected type of variable, expression, or function return value. By appending `x::T` to an expression or variable `x`, it asserts that `x` should be of type `T`. The function `eltype` is commonly used to help programmers specify type expectations and debug more easily. It makes Julias code clearer and safer and allows querying the element type of an object or abstract type.

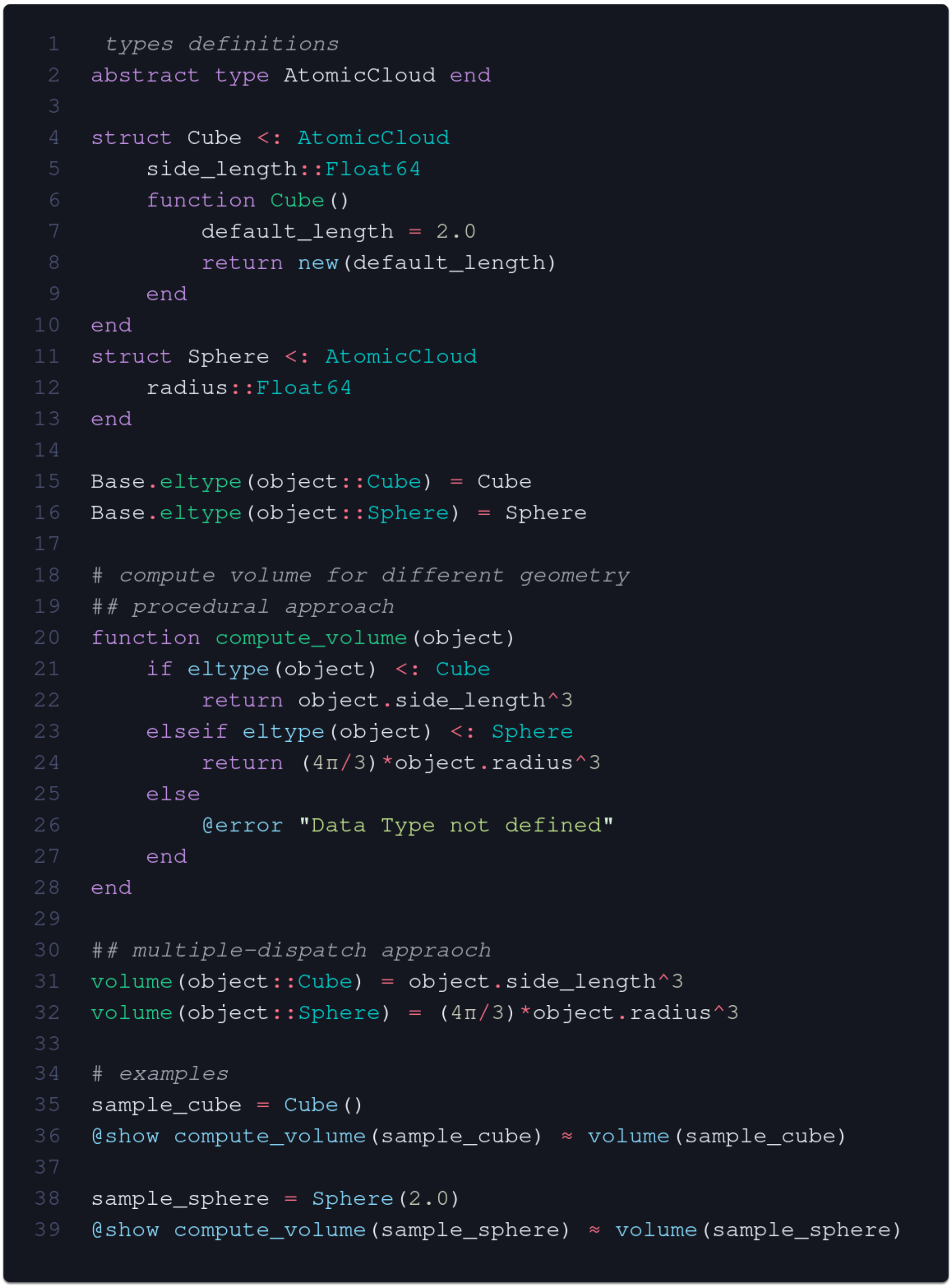
The function `compute_volume` uses a traditional programming method that checks types and uses branching to decide what operation to perform. Unlike single dispatch, multiple dispatch selects the appropriate method based on the object type given as an argument. This approach streamlines the code by defining specialized methods for each data type.

However, using the procedural approach in the `compute_volume` function becomes more problematic with an increase in the number of atomic cloud types. The `compute_-volume` function would have to be recreated if a new data structure, like `Cylinder`, is introduced. In more intricate scenarios, this could potentially disrupt existing functionalities, making maintenance challenging.

3.2 Constructors

Given the significance of data types in Julia, it becomes evident that we need to create our own types to leverage the language. The data types within this project can be categorized into three main groups. `Dimensions` for atomic geometry, `Pump` for laser types, and `Physics` for types and equations. The diagram in Figure 4 summarizes the abstract types as white-filled boxes. The diagram highlights concrete types denoted by yellow-filled boxes.

The project's main aim is to ensure that 3D simulations are fully functional. Some parts of the code for 2D types are awaiting for future improvements - these are stated



```julia
 types definitions
abstract type AtomicCloud end

struct Cube <: AtomicCloud
    side_length::Float64
    function Cube()
        default_length = 2.0
        return new(default_length)
    end
end
struct Sphere <: AtomicCloud
    radius::Float64
end

Base.eltype(object::Cube) = Cube
Base.eltype(object::Sphere) = Sphere

# compute volume for different geometry
## procedural approach
function compute_volume(object)
    if eltype(object) <: Cube
        return object.side_length^3
    elseif eltype(object) <: Sphere
        return (4π/3)*object.radius^3
    else
        @error "Data Type not defined"
    end
end

## multiple-dispatch appraoch
volume(object::Cube) = object.side_length^3
volume(object::Sphere) = (4π/3)*object.radius^3

# examples
sample_cube = Cube()
@show compute_volume(sample_cube) ≈ volume(sample_cube)

sample_sphere = Sphere(2.0)
@show compute_volume(sample_sphere) ≈ volume(sample_sphere)
```

Figure 3 – Comparison of Standard Procedural style of programming against Multiple-Dispatch approach.

Source: By the author.

in grey and serve as a testament to the projects potential for further development and improvement.

One needs to use `Atom`, `Laser`, `LinearOptics`, or `NonLinearOptics` constructors[49] as starting points for a simulation, see Figure 6. Constructors are important for setting up simulations, as the library requires either a `LinearOptics` or `NonLinearOptics` data type to work well. Figure 5 shows their respective data type definition, similar to object properties in object-oriented programming. Note that in Julia, the constructor is the only method that can be defined inside the `struct` scope. Other methods that use it are defined outside of it. This was shown in the code example for the `Cube` data type, in Figure 3.

Both users and developers of the package can gain knowledge from the information presented in figures such as Figure 4 and Figure 5. Users can access valuable hidden information using these diagrams, and propose new `kernelFunctions` for their needs, for example. For developers, these visual representations form the basis upon which the library can be further expanded. The current data types in the library can be easily integrated in new ones.

The process of defining various concrete types is shown in the code snippet in Figure 6. In order to create atomic distributions, the `Atom` constructor must be employed. Its first argument corresponds to the data type, while the subsequent arguments vary depending on the geometric shape. For the user's convenience, there are functions available to calculate the dimensions of the cloud if the density is already known. These functions are called `cube_inputs`, `sphere_inputs`, and `cylinder_inputs`.

3.3 Functions

We optimize some core functions, which implement equations derived in the Theoretical Background section. Table 1 refers to the base functions related to the computation of the evolution of the atomic state over time.

Table 1 – Dynamics related functions

	`LinearOptics`		`NonLinearOptics`	
	`Scalar`	`Vectorial`	`MeanField`	`PairCorrelation`
`default_initial_condition`	✓	✓	✓	✓
`time_evolution`	✓	✓	✓	✓
`steady_state`	✓	✓	✓$^{(a)}$	✓$^{(a)}$

Source: By the author.

The $^{(a)}$ mark on the Table 1 shows that users might need to change settings in `MeanField` and `PairCorrelation` to get a solution with `steady_state` function. In such cases, we recommend fine-tuning adjustments by the user, as there is no known analytical solution available for benchmarking. Finding semi-analytical methods for the steady state

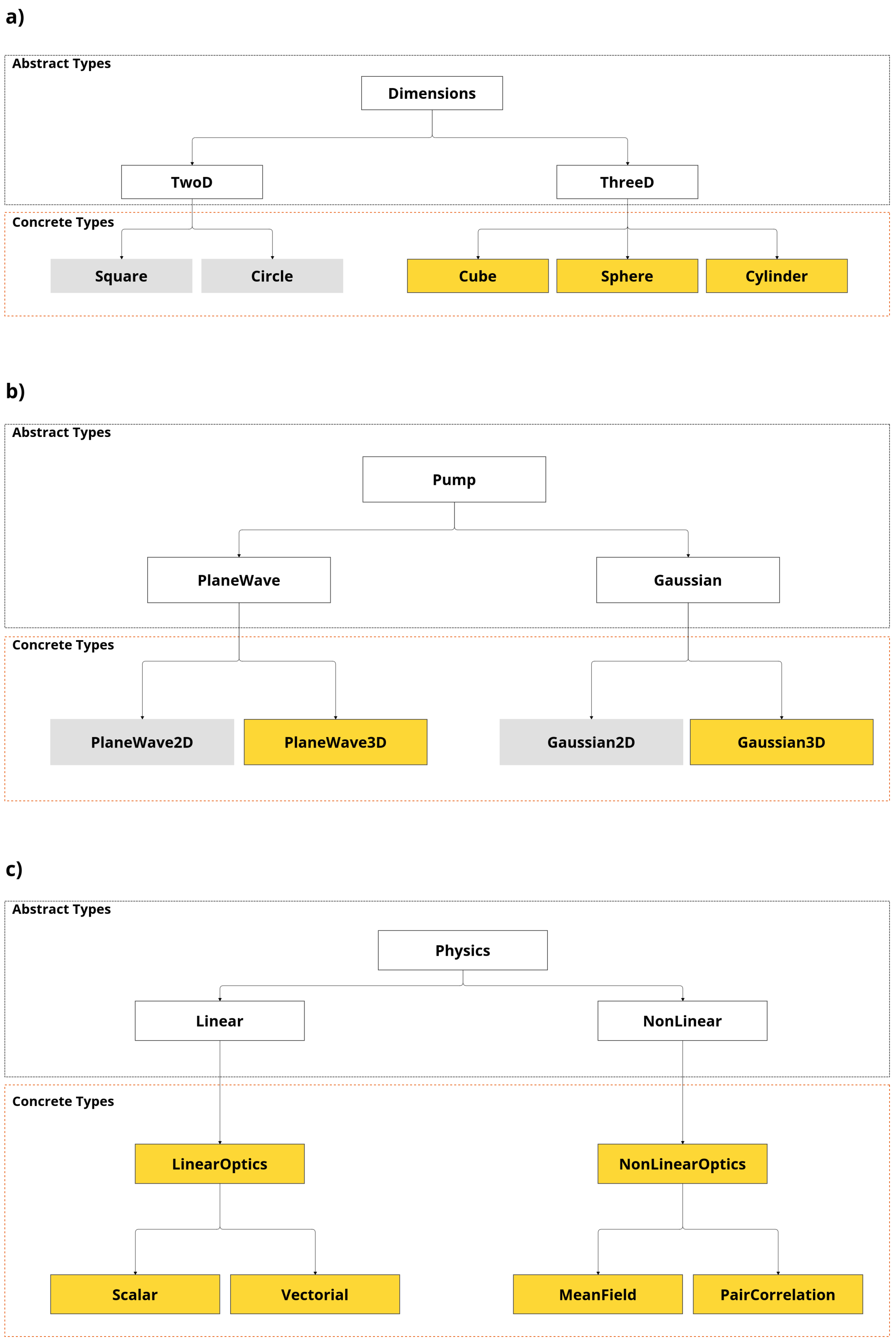
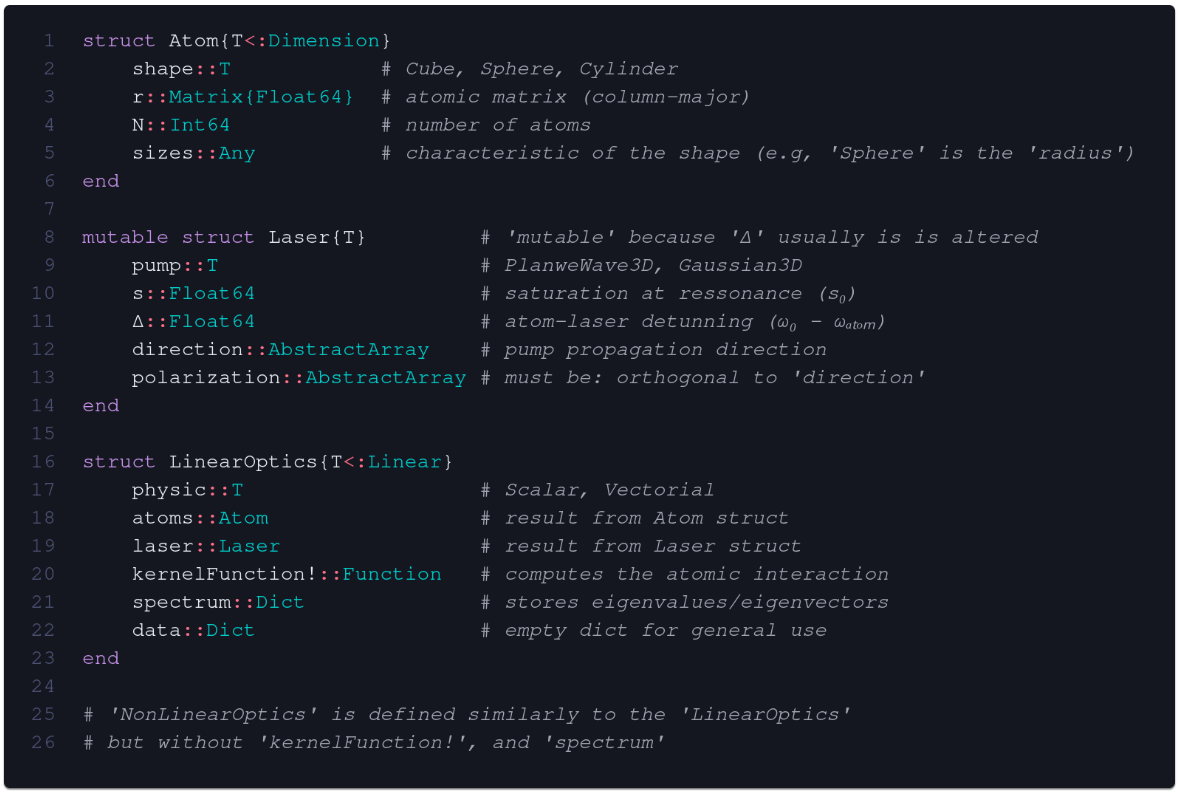


Figure 4 – Abstract types are in white, and all concrete data types are available in yellow-filled boxes.

Source: By the author.



```julia
struct Atom{T<:Dimension}
    shape::T            # Cube, Sphere, Cylinder
    r::Matrix{Float64}  # atomic matrix (column-major)
    N::Int64            # number of atoms
    sizes::Any          # characteristic of the shape (e.g, 'Sphere' is the 'radius')
end

mutable struct Laser{T}         # 'mutable' because 'Δ' usually is is altered
    pump::T                     # PlanweWave3D, Gaussian3D
    s::Float64                  # saturation at ressonance (s₀)
    Δ::Float64                  # atom-laser detunning (ω₀ − ωₐₜₒₘ)
    direction::AbstractArray    # pump propagation direction
    polarization::AbstractArray # must be: orthogonal to 'direction'
end

struct LinearOptics{T<:Linear}
    physic::T                   # Scalar, Vectorial
    atoms::Atom                 # result from Atom struct
    laser::Laser                # result from Laser struct
    kernelFunction!::Function   # computes the atomic interaction
    spectrum::Dict              # stores eigenvalues/eigenvectors
    data::Dict                  # empty dict for general use
end

# 'NonLinearOptics' is defined similarly to the 'LinearOptics'
# but without 'kernelFunction!', and 'spectrum'
```

Figure 5 – Definition of the base package constructors.

Source: By the author.

remains an open question, and our suggestion is to compute Adomian Decomposition,[50] which would require some dedicated effort with symbolic and metaprogramming.

More information on `time_evolution` and `steady_state` will be presented shortly. The default initial conditions vectors u_0 for each model are:

$$u(t=0)_{\texttt{Scalar}} = \left[\vec{0}_N\right], \tag{3.1}$$

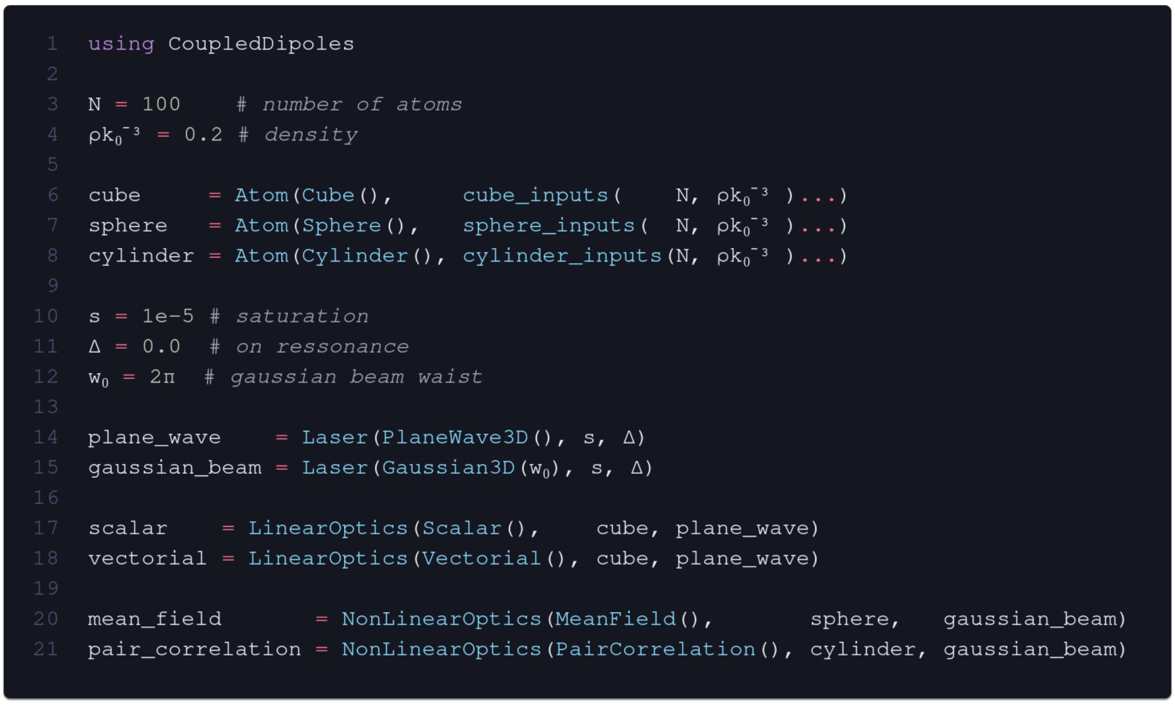
$$u(t=0)_{\texttt{Vectorial}} = \left[\vec{0}_N \mid \vec{0}_N \mid \vec{0}_N\right], \tag{3.2}$$

$$u(t=0)_{\texttt{MeanField}} = \left[\vec{0}_N \mid -\vec{1}_N\right], \tag{3.3}$$

$$u(t=0)_{\texttt{PairCorrelation}} = \left[\vec{0}_N \mid -\vec{1}_N \mid \vec{0}_{N^2} \mid \vec{0}_{N^2} \mid \vec{0}_{N^2} \mid \vec{1}_{N^2}(\text{zero on diagonal})\right]. \tag{3.4}$$

The symbol $\vec{0}_N$ represents a vector comprising zeros with a length of N (number of atoms). Similarly, the same logic applies to $\vec{1}_N$ and $\vec{0}_{N^2}$. All initial conditions use `ComplexF64` - complex numbers with `Float64` precision. Using lower precision was not investigated, and it is not recommended unless users understand its risk and/or trade-offs.

Table 2 summarizes all the functions pertaining to the scattering properties of the system. The `scattered_field` and `scattered_intensity` match the previous chapter definitions. One key implementation decision that significantly impacted these functions is regarding the memory layout of the data. *We opted to represent all fields as matrices*



```julia
using CoupledDipoles

N = 100     # number of atoms
ρk₀⁻³ = 0.2 # density

cube     = Atom(Cube(),     cube_inputs(    N, ρk₀⁻³ )...)
sphere   = Atom(Sphere(),   sphere_inputs(  N, ρk₀⁻³ )...)
cylinder = Atom(Cylinder(), cylinder_inputs(N, ρk₀⁻³ )...)

s = 1e-5 # saturation
Δ = 0.0  # on ressonance
w₀ = 2π  # gaussian beam waist

plane_wave    = Laser(PlaneWave3D(), s, Δ)
gaussian_beam = Laser(Gaussian3D(w₀), s, Δ)

scalar    = LinearOptics(Scalar(),    cube, plane_wave)
vectorial = LinearOptics(Vectorial(), cube, plane_wave)

mean_field       = NonLinearOptics(MeanField(),       sphere,   gaussian_beam)
pair_correlation = NonLinearOptics(PairCorrelation(), cylinder, gaussian_beam)
```

Figure 6 – Code example to initialize a simulation with specific geometry, laser, and physical model.

Source: By the author.

rather than vectors. Standardizing all fields as matrices simplifies the maintenance of the package's source code and ensures consistency across different physical models. It also makes adding new features easier. While this choice may require some attention from users when accessing elements, it was a necessary decision. The reason behind this choice is that the `Vectorial` output includes polarization, and the desired output format is a matrix. The intensity is still represented as a vector in all physical models.

Table 2 – All scattering-related functions available.

	`LinearOptics`		`NonLinearOptics`	
	`Scalar`	`Vectorial`	`MeanField`	`PairCorrelation`
`scattered_electric_field`	✓	✓$^{(b)}$	✓	✓
`scattered_intensity`	✓	✓$^{(b)}$	✓	✓
`laser_field`	✓	✓	✓	✓
`laser_intensity`	✓	✓	✓	✓
`laser_and_scattered_electric_field`	✓	✓	✓	✓
`laser_and_scattered_intensity`	✓	✓	✓	✓
`transmission`	✓	✓$^{(b)}$	✓	✓
`scattered_power`	✓	✗	✓	✗
`get_intensity_over_an_angle`	✓	✓$^{(c)}$	✓	✓

Source: By the author.

Also, regarding the `Vectorial` Model, comment $^{(b)}$ on the table, indicated a word of caution on the `Vectorial` electric field scattering and its correspondent intensity. Our `Vectorial` code works well for most cases, but it does not match the literature in a particular scenario that involves light statistics, a point that will be discussed in a dedicated chapter. This is a challenge due to the relative lack of information on this topic, and the other tests give the result predicted by the literature. For mark $^{(c)}$, `get_intensity_over_an_angle` does not have an analytical solution. It is explained in a separate section.

Figure 7 illustrates the interrelationship among some functions in Table 2. The full hierarchy of dependencies is more intricate, and you can refer to the Appendix section for a comprehensive overview. Once the `scattered_electric_field` function is set up, other complex functions become accessible. When implementing new physical models, it is important to start with this function.

As a final observation, each function incorporates Julias native multi-threading execution mechanisms at their respective code bottlenecks. To improve performance, users can use parallelism with packages like `Distributed.jl` or `MPI.jl`. This recommendation stems from the fact that each process will operate within an independent memory space. Although multi-threading can be done with these functions, it may negatively impact the code's performance.

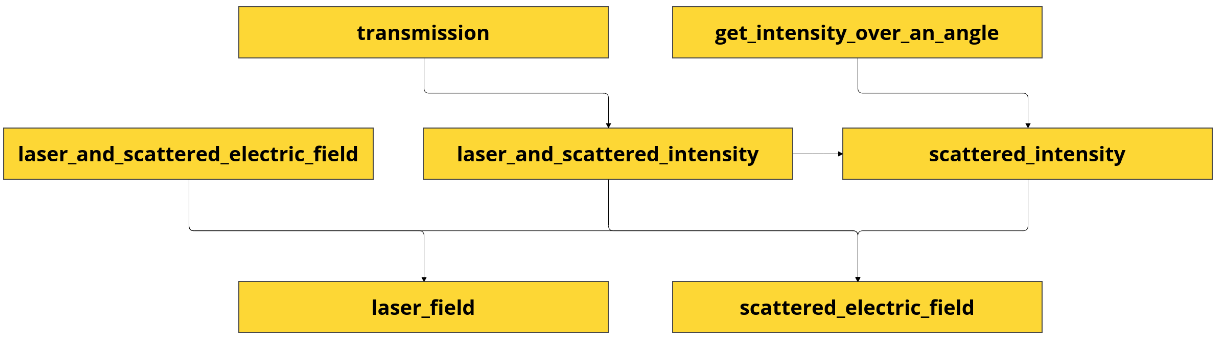

transmission
get_intensity_over_an_angle
laser_and_scattered_electric_field
laser_and_scattered_intensity
scattered_intensity
laser_field
scattered_electric_field

Figure 7 – Overview of some scattering related function and their connections.

Source: By the author.

4 OPTIMIZATIONS

Many tests were realized with different packages and techniques to make sure all the codes ran smoothly in different situations. As a result, the source code has various intricacies that may be challenging to understand. This may raise several questions if the effort of running many tests was needed since simpler options were available. For example, storing matrices in column-major[51] format was adopted because it makes accessing the memory with higher cache hits[52] - increases the likelihood of data being readily available in the cache, which results in faster data access. Our commitment to profiling and investigation drove us to optimize functions with multiple tools. In the following discussion, we detail this optimization process, while also sharing the corresponding theoretical background.

4.1 Laser Direction

The `Laser` function currently accepts `PlaneWave3D` according to textbooks definition, that is, given a measurement position, or `sensor`, at position $\mathbf{r}$, and a plane wave with wave vector $\mathbf{k} = k_0\hat{k}$, the electric field is

$$E(\mathbf{r};\mathbf{k}) = \vec{E}_0 e^{i\mathbf{k}\cdot\mathbf{r}}. \tag{4.1}$$

The `Gaussian3D` laser geometry, which describes a Gaussian beam, reads[53]

$$\vec{E}(\rho, z) = \vec{E}_0 \frac{w_0}{w(z)} e^{-\frac{\rho^2}{w^2(z)}} e^{i[k_0 z - \eta(z) + k\rho^2/2R(z)]}, \tag{4.2}$$

with

$$\begin{aligned}
\rho &= x^2 + y^2,\\
w(z) &= w_0\sqrt{1 + z^2/z_0^2},\\
R(z) &= z(1 + z_0^2/z^2),\\
\eta(z) &= \arctan(z/z_0),\\
z_0 &= k_0 w_0^2/2.
\end{aligned}$$

w_0 is the waist radius and a parameter that can be controlled on the package. The other functions have geometric interpretation, which we do not use, because even though Equation 4.2 is correct, it has an artifact of the cylindrical representation, that is, it is undefined at $z = 0$, $R(0) = 0\cdot(1+\infty)$. To avoid such a scenario, we use the Cartesian representation, also adopted by Novotny[53]

$$\vec{E}(x, y, z) = \frac{\vec{E}_0 e^{ik_0 z}}{(1 + 2iz/k_0 w_0^2)} e^{-\frac{x^2+y^2}{w_0^2}\frac{1}{1+2iz/k_0 w_0^2}}. \tag{4.3}$$

Equation 4.3 describes a laser beam in the z-direction and we need a laser that can propagate in any direction. A solution is to rotate all values of (x, y, z) through a linear transformation. Then, we can apply Equation 4.2 to the new variables. This solution depends on creating and multiplying rotation matrices, which allocates memory. The size of the rotation matrices is small, but since is a recurrent operation, by profiling we identified that it was producing some stress on the garbage collector to free this memory. Our alternative was to have a code without memory allocations modifying the beam equation, but the explanation needs some geometry argumentation. First, we recognize the exponential, $e^{ik_0 z}$, is a plane wave propagating at z-direction, and its generalization to any direction is Equation 4.1.

Regarding the second exponential on Equation 4.3, the argument x^2+y^2 represents the distance from a point $\mathbf{r} = (x, y, z)$ to the direction of propagation, $\hat{z}$. To extend this notion for an arbitrary propagation direction, we rely on geometric guidance from Figure 8 to find the distance $|\vec{AB}|$. By projecting $\mathbf{r}$ onto the direction vector $\mathbf{k}$ we get

$$\mathbf{p} = (\mathbf{r} \cdot \hat{\mathbf{k}})\hat{\mathbf{k}} = \left(\mathbf{r} \cdot \frac{\mathbf{k}}{|\mathbf{k}|}\right) \frac{\mathbf{k}}{|\mathbf{k}|}$$

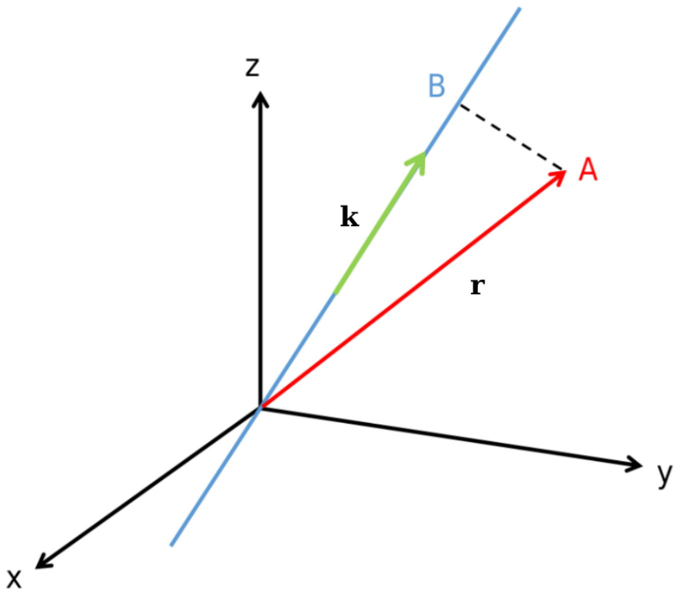


Figure 8 – A is an arbitrary point in space, and we are interested in the minimum distance from A to the line generated from vector $\mathbf{k}$.

Source: Adapted from MATHEMATICS STACK EXCHANGE.[54]

The length of the new vector $\vec{AB}$ comes from Pythagorean's Theorem:

$$\begin{aligned} |AB|^2 &= |\mathbf{r}|^2 - |\mathbf{p}|^2 \\ &= |\mathbf{r}|^2 - \left(\mathbf{r} \cdot \frac{\mathbf{k}}{|\mathbf{k}|}\right) \frac{\mathbf{k}}{|\mathbf{k}|} \\ &= |\mathbf{r}|^2 - \frac{|\mathbf{r} \cdot \mathbf{k}|^2}{|\mathbf{k}|^2} \end{aligned} \tag{4.4}$$

Finally Equation 4.3 is refactored as equation for the `Gaussian3D` laser

$$\vec{E}(\mathbf{r};\mathbf{k}) = \frac{\vec{E}_0 e^{i\mathbf{k}\cdot\mathbf{r}}}{(1+2i\mathbf{k}\cdot\mathbf{r}/k_0w_0^2)} e^{-\frac{|\mathbf{r}|^2 - \frac{|\mathbf{r}\cdot\mathbf{k}|^2}{|\mathbf{k}|^2}}{w_0^2}\frac{1}{1+2i\mathbf{k}\cdot\mathbf{r}/k_0w_0^2}}.$$

4.2 Parallelism

Parallelism can be explored from different perspectives, from data communication patterns, to multi-stage distributed algorithms. We highlight some of our findings that have an impact on the source code.

4.2.1 Memory Access

The efficient handling of matrices is often a critical performance bottleneck. Efficient memory access plays a pivotal role in optimizing matrix operations, as it significantly impacts the execution time of algorithms. Before delving into parallel memory access, let us summarize how elements of matrices are stored in the memory. In most programming languages, matrices are stored in a row-major or column-major order. In row-major order, elements of a row are stored sequentially in memory, followed by the next rows elements. Conversely, Julia is column-major, and matrices store elements of their columns sequentially. This storage scheme ensures that neighboring elements of a matrix are stored contiguously in memory.

Sequential memory access is a well-established technique for optimizing matrix operations. When a program sequentially accesses elements of a matrix, it benefits from spatial locality. Memory locations close to each other are likely to be accessed together. Matrices are faster to retrieve from memory when accessed sequentially in alignment with memory storage order.

4.2.2 SIMD

Single Instruction Multiple Data, SIMD, parallelism applies one fine-grained instruction to multiple data elements simultaneously to improve computational speed by using data-level parallelism.[55] In Julia, we can benefit from the memory layout to explore vectorized operations for data-level parallelism. The `LoopVectorization.jl`[56] package handles Single SIMD instructions, but we did not see advantages in using them because the bottleneck in simulations was the time evolution or eigendecomposition operations, and also, some operations lack complex number support.

4.2.3 Multi-threading

Multi-threading is a technique for running various tasks simultaneously on one computer or node in a cluster. In this paradigm, multiple threads, which are lightweight

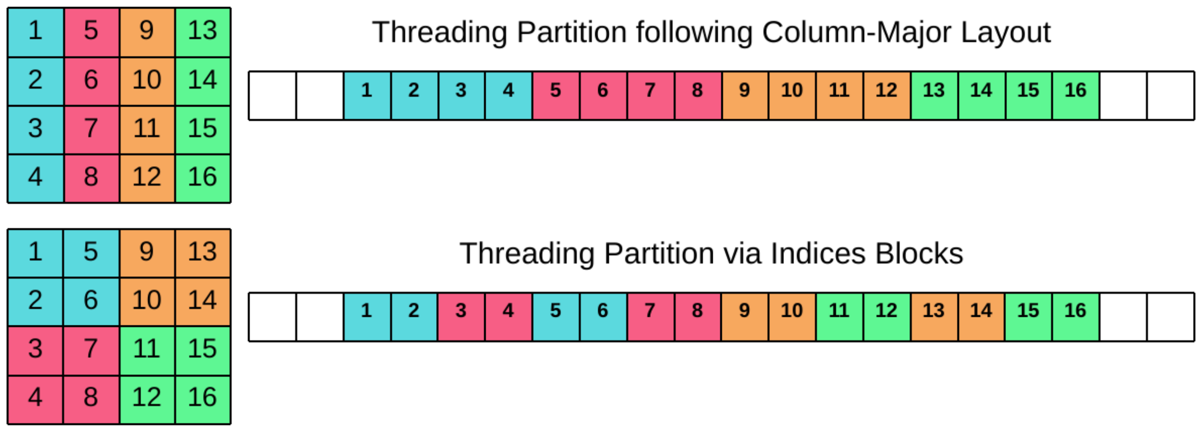

Threading Partition following Column-Major Layout
Threading Partition via Indices Blocks
1 5 9 13
2 6 10 14
3 7 11 15
4 8 12 16
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Figure 9 – Matrices were accessed via their column positions because is a natural choice to read data with spatial locality. As such, multi-threading occurred through columns of data, and not with blocks of indices.

Source: By the author.

units of execution, share the same memory space. Threads can be considered as subtasks within a larger program, and they can communicate and synchronize with one another. It is well-suited for tasks that require shared memory, where threads can easily exchange data and coordinate their efforts.

In Julia, multi-threading is facilitated by the native `Threads.@threads` macro. It has two main benefits. First, it speeds up the composing of the interaction matrices — as pairwise distances are easily computed, then further operations that depend on them, are embarrassingly parallel. We found that the best scenario for domain partition was to assign row segments into different threads, as represented on Figure 9, where each color indicates the thread responsible for processing part of the matrix.

The second scenario includes calculating any observable (e.g. electric fields) at different positions. Computing different observations simultaneously was faster than using a parallel merging operation (e.g. parallel `mapreduce`) of a single element.

4.2.4 Distributed

Distributed parallelism operates across multiple computational units, often distributed geographically. Each unit, referred to as a node, typically has its own memory and computational resources. In this model, tasks are divided into smaller portions, and each portion is executed on a separate node. Nodes use a network to communicate and coordinate, and they can scale horizontally for resource-intensive tasks. Sharing data in this parallelism is harder because it needs direct communication through a network.

`Distributed.jl` uses SSH to connect remote workers to a master-worker setup and split tasks among workers. No internal component of the `CoupledDipole.jl` package depends on this type of parallelism, only multi-threading, and it is up to the User to make the best decision on how to load balance their tasks. To reduce memory communication, one should use `DistributedArrays.jl` package and balance node loads by shuffling input parameters randomly.

4.3 Far-Field Radiation

By observing the electric field's intensity, we can compare experimental and theoretical outcomes. The field's exact expression is model-dependent. Our equations, inspired by Samoylova *et al.*[57] and Castro,[30] were introduced in the background chapter. If we measure the field at position $k_0\mathbf{r} = r\hat{n}$ far from the system dimensions, r_j, $r \gg r_j$, we can simplify the exponential in the scalar model of scattered light as

$$\frac{e^{ik_0|\mathbf{r}-\mathbf{r}_j|}}{k_0|\mathbf{r}-\mathbf{r}_j|} \approx \frac{e^{ik_0 r}e^{-ik_0\hat{n}\cdot\mathbf{r}_j}}{k_0 r}. \tag{4.5}$$

We call this estimation the *Far Field Approximation*, in which near-field terms are thus neglected — the equations without approximations are referenced as *Near Field Equations.*

The electric field is identical for `Scalar`, `MeanField` and `PairCorrelation` within the far-field approximation.

$$E(\mathbf{r},t)_{scalar} \approx i\frac{\Gamma}{2}\frac{e^{ik_0 r}}{k_0 r}\sum_j e^{-ik_0\hat{n}\cdot\mathbf{r}_j}\beta_j(t) \tag{4.6}$$

whereas `Vectorial` expression is

$$E_\mu(\mathbf{r},t) \approx i\frac{3\Gamma}{4}\frac{e^{ik_0 r}}{k_0 r}\sum_j\sum_\eta(\delta_{\mu,\eta} - \hat{n}_\mu\hat{n}^*_\eta)e^{-ik_0\hat{n}\cdot\mathbf{r}_j}\beta^\eta_j(t). \tag{4.7}$$

The intensity is slightly different in each model, for the scalar

$$I(\mathbf{r})_{\texttt{Scalar}} = \left| i\frac{\Gamma}{2}\frac{e^{ik_0 r}}{k_0 r}\sum_j e^{-ik_0\hat{n}\cdot\mathbf{r}_j}\beta_j(t)\right|^2 \tag{4.8}$$

$$I(\mathbf{r})_{\texttt{MeanField}} = I(\mathbf{r})_{\texttt{Scalar}} + \frac{\Gamma^2}{4k_0^2r^2}\left[\sum_j -|\langle\sigma_j^-\rangle|^2 + \frac{1+\langle\sigma_j^z\rangle}{2}\right] \tag{4.9}$$

$$I(\mathbf{r})_{\texttt{PairCorrelation}} = \frac{\Gamma^2}{4k_0^2r^2}\left[\sum_j^N\sum_{m\neq j}^N e^{ik_0\hat{n}\cdot(\mathbf{r}_j-\mathbf{r}_m)}\langle\sigma_j^+\sigma_m^-\rangle + \sum_j^N\frac{1+\langle\sigma_j^z\rangle}{2}\right] \tag{4.10}$$

In simulations involving intensity, results are often normalized, such as with sub-super radiance,[58, 59] since its dynamical behavior which its relevant rather than its absolute value. As a result, multiplicative factors become irrelevant in practical applications. Despite that, the question about the value of r is fundamentally relevant for the validity of the far-field approximation, and we want a precise number since it is used in the electric field definition. In this section, we then discuss the limitation of the far-field approximation in the scalar approximation.

4.3.1 Where exactly is the *far field*?

Here, we argue that choosing $r = 50r_j^2$ is a suitable definition to quantify the far-field approximation. We begin by recalling the far field approximation, which involves calculating $|\mathbf{r}-\mathbf{r}_j|^2$ (the unit k_0 is omitted, since they are already factored in Equation 4.5 to Equation 4.10).

$$\begin{aligned}|\mathbf{r}-\mathbf{r}_j|^2 = (\mathbf{r}-\mathbf{r}_j)\cdot(\mathbf{r}-\mathbf{r}_j) &= \mathbf{r}\cdot\mathbf{r} - 2\mathbf{r}\cdot\mathbf{r}_j + \mathbf{r}_j\cdot\mathbf{r}_j \\ &= r^2 - 2\mathbf{r}\cdot\mathbf{r} + r_j^2 \\ &= r^2\left(1 - \frac{2\mathbf{r}\cdot\mathbf{r}_j}{r^2} + \frac{r_j^2}{r^2}\right).\end{aligned}$$

Next, we take the square root and use first order approximation $(1+\varepsilon)^{1/2} \approx 1+\varepsilon/2$ to write

$$\begin{aligned}|\mathbf{r}-\mathbf{r}_j| = \sqrt{|\mathbf{r}-\mathbf{r}_j|^2} &= r\left(1 - \frac{2\mathbf{r}\cdot\mathbf{r}_j}{r^2} + \frac{r_j^2}{r^2}\right)^{1/2} \\ &\approx r\left(1 - \frac{\mathbf{r}\cdot\mathbf{r}_j}{r^2} + \frac{1}{2}\frac{r_j^2}{r^2}\right) \\ &= r - \frac{\mathbf{r}\cdot\mathbf{r}_j}{r} + \frac{1}{2}\frac{r_j^2}{r} \\ &= r - \frac{\mathbf{r}}{r}\cdot\mathbf{r}_j + \frac{1}{2}\frac{r_j^2}{r} \\ &= r - \hat{n}\cdot\mathbf{r}_j + \frac{1}{2}\frac{r_j^2}{r}.\end{aligned}$$

To obtain the far field approximation, the third term should be negligible

$$\frac{1}{2}\frac{r_j^2}{r} \ll k_0. \tag{4.11}$$

Equation 4.11 is correct but does not give a value for r. To address this, we propose two reasonable conditions:

1. r should be a function of the atomic system size, which we will call r_j by abuse of notation: $r(r_j) = s\cdot r_j^2$, where s is a scaling factor to be determined.
2. The relative error should be within an arbitrary tolerance, and we set it to be 0.1%.

Applying these conditions into Equation 4.11 ($r_j^2/(2sr(r_j) = 0.001$) and solving for the scaling factor ($s = 500/k_0$) results in our numerical far field condition:

$$r(k_0 r_j) = 500r_j^2. \tag{4.12}$$

For consistency test, we need numerical results from Equation 4.5. We created a random cube of size $k_0 L = 10$, and sensors at the surface of a sphere of radius $r = s \times L$, then, we computed the relative error of the exact exponential and its approximated expression in Equation 4.5.

The error was too dispersed to present a reliable data of averages and standard deviations. Figure 10 shows the probability distribution of all configurations at certain scaling factors. All curves have a similar distribution with their mean near the 0.1% target error, but this is not a meaningful piece of information, since its variance is large. Notably, the $s = 50r$ was the scaling with the lowest maximum errors, therefore, it was chosen as our package parameter.

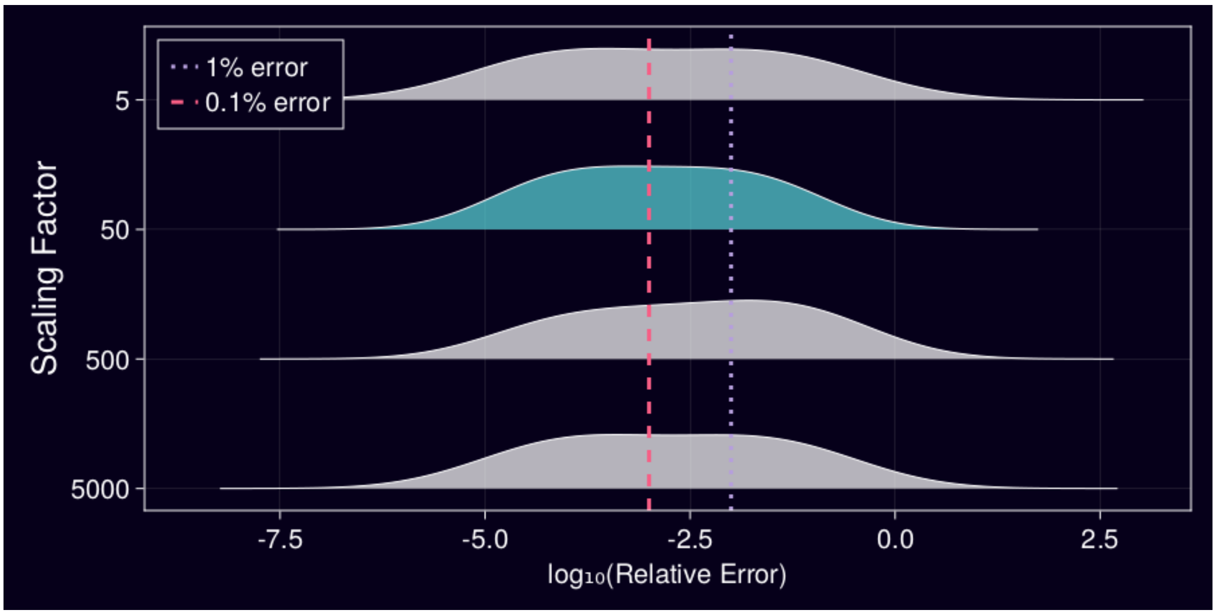


Figure 10 – Distribution of relative errors for scaling values proportional to atomic system size. Our target error for the far field approximation (dashed line) is valid for the average of any distribution, therefore, we opted to use $s = 50r$ since it has the lowest variance.

Source: By the author.

4.4 Integration over ϕ

In some scenarios (typically when the system has, statistically, a rotational symmetry), we are interested in the average intensity at a certain polar angle θ and not in a specific direction (θ, ϕ). Since the electric field was already defined previously, one can convert the measurement point into spherical coordinate $\mathbf{r} = r[\sin(\theta)\cos(\phi), \sin(\theta)\sin(\phi), \cos(\theta)]$ and eliminate the azimuthal angle dependence, ϕ, with an integration

$$I(\theta) = \int_0^{2\pi} |E(\theta, \phi)|^2 d\phi. \tag{4.13}$$

In practice the integral is evaluated with some quadrature algorithm, where the range ϕ is broken into chunks of size $\Delta\phi$ in such a manner that the resulting sum is equivalent, up to a certain precision, to the continuous integral, that is, Equation 4.13 becomes

$$I(\theta) \approx \sum_{\Delta\phi \in [0,2\pi]} |E(\theta, \mathbf{\Delta}\phi)|^2. \tag{4.14}$$

Note that for any quadrature method, its accuracy impacts the execution time to evaluate it, and we seek possible trade-offs between accuracy and performance. To measure the accuracy in a concrete example we will evaluate the Equation 4.13 with the `Scalar` model and Far Field approximation, which is a simplification of Equation 2.30:

$$I(\mathbf{r}) = \frac{\Gamma^2}{4k_0^2 r^2} \sum_{j=1}^{N} \sum_{m=1}^{N} \beta_j^* \beta_m e^{ik_0 \hat{m} \cdot \mathbf{r}_{jm}}$$

The dot product of the exponential reads

$$\begin{aligned} \hat{n} \cdot \vec{r}_{jm} &= [\sin(\theta)\cos(\phi), \sin(\theta)\sin(\phi), \cos(\theta)] \cdot [x_{jm}, y_{jm}, z_{jm}] \\ &= \sin(\theta)[\cos(\phi)x_{jm} + \sin(\phi)y_{jm}] + \cos(\theta)z_{jm}, \end{aligned}$$

with $\vec{r}_{jm} = \vec{r}_j - \vec{r}_m$.Consequently,

$$I(\theta, \phi) = \frac{\Gamma^2}{4k_0^2 r^2} \sum_{j=1}^{N} \sum_{m=1}^{N} \beta_j^* \beta_m e^{ik_0 \cos(\theta) z_{jm}} e^{ik_0 \sin(\theta)[\cos(\phi)x_{jm} + \sin(\phi)y_{jm}]}, \tag{4.15}$$

and one integrates over ϕ:

$$I(\theta) = \int_0^{2\pi} I(\theta, \phi) d\phi = \frac{\Gamma^2}{4k_0^2 r^2} \sum_{j=1}^{N} \sum_{m=1}^{N} \beta_j^* \beta_m e^{ik_0 \cos(\theta) z_{jm}} \int_0^{2\pi} e^{ik_0 \sin(\theta)[\cos(\phi)x_{jm} + \sin(\phi)y_{jm}]} d\phi.$$

By utilizing the expression $e^{ix} = \cos(x) + i\sin(x)$ and with the help of a symbolic calculator,

$$\begin{aligned} &\int_0^{2\pi} \cos(k_0 \sin(\theta)[\cos(\phi)x_{jm} + \sin(\phi)y_{jm}]) d\phi = 2\pi J_0 \left(|k_0 \sin(\theta)| \sqrt{x_{jm}^2 + y_{jm}^2} \right) \\ &\int_0^{2\pi} \sin(k_0 \sin(\theta)[\cos(\phi)x_{jm} + \sin(\phi)y_{jm}]) d\phi = 0, \end{aligned}$$

where $J_0(x)$ is the Bessel function of the first kind and zeroth order. Finally, $I(\theta)$ reads

$$I(\theta) = 2\pi \frac{\Gamma^2}{4k_0^2 r^2} \sum_{j=1}^{N} \sum_{m=1}^{N} \beta_n^* \beta_m e^{ik_0 \cos(\theta) z_{jm}} J_0 \left(|k_0 \sin(\theta)| \sqrt{x_{jm}^2 + y_{jm}^2} \right) \tag{4.16}$$

Note that Equation 4.16 has complexity is $O(N^2)$ and it should not used in practice. Instead, Equation 4.16 is used to benchmark against different tolerances while using Equation 4.14, whose complexity is linear, $O(N)$.

For numerical integration, we utilize the Cubature-Quadrature algorithm proposed by Genz and Malik,[60] which is available in the `HCubature.jl` package.[61] This algorithm allows us to adjust the tolerance for convergence, and Figure 11 explores this feature. We observe the difference between the exact value from Equation 4.16 and the numerical approximation from Equation 4.14 as the tolerance is increased.

However, Figure 11 only provides a partial view because it represents a single angle θ. When testing different cloud configurations and angles, we find that the relative error exhibits outliers with orders of magnitude larger than the average. Figure 12 illustrates one example of an outlier highlighted within a green box. To address this issue, we estimate the smallest tolerance, based on Figure 11, that is both smaller than 10^{-7} (to avoid outliers) and results in the largest execution time difference.

Thus, we choose the implementation of `get_intensity_over_an_angle` to have the first value where the execution time exhibits a significant jump, marked with a dotted line in Figure 11, at $tol = 10^{-7.4}$. This choice was made since these parameters were representative of typical simulations. If one wants to change this parameter, use the variable `tol`, or use the analytical solution Equation 4.16, set `exact_solution=true`.

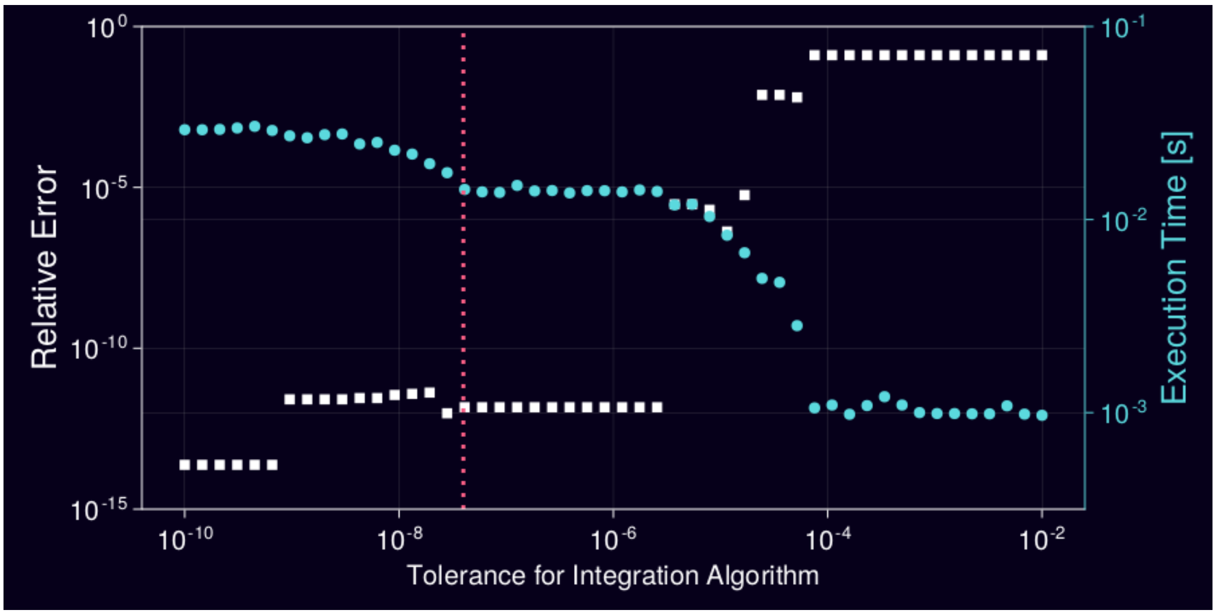


Figure 11 – The relative error and execution times have jumps, showing the stability of the algorithm. Simulation with $N = 1200$, $\rho k_0^{-1} = 0.1$, $\Delta = 0$, $\theta = 50°$ - other values of N presented similar results.

Source: By the author.

4.5 Time Evolution

We will delve into the complexities of time evolution equations. Our goal is to determine the most efficient numerical method for solving such equations, which depends

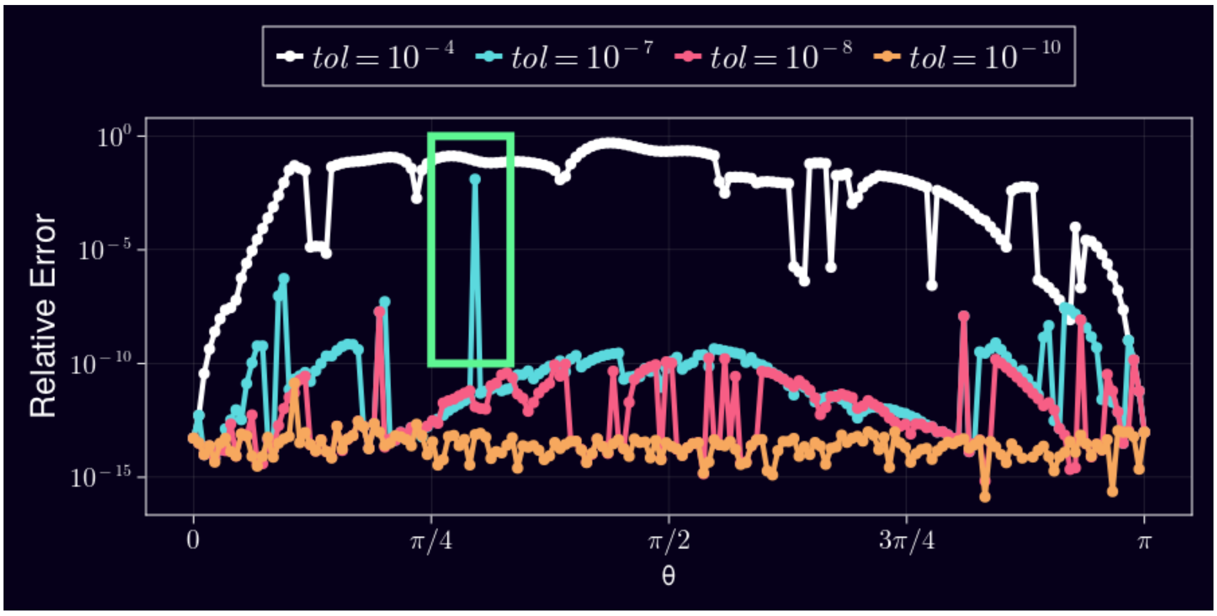


Figure 12 – In this particular configuration, and $tol = 10^{-7}$ there is one θ with a relative error of order 10^{-1}, a clear outlier result that cannot be overlooked.

Source: By the author.

on various system parameters.

4.5.1 Formal Solution of Linear Systems

Given a square matrix $G \in \mathbb{C}$ and vectors $\vec{\beta}, \vec{\Omega} \in \mathbb{C}$ that form an Ordinary Differential Equation of the form

$$\frac{d\vec{\beta}}{dt} = G\vec{\beta} + \vec{\Omega}. \tag{4.17}$$

We aim for a formal solution to serve as a comparison of errors and running time against numerical methods. Equation 4.17 is a standard problem in mathematics, and its solution is well known (check Appendix A for an overview of the proof).

$$\vec{\beta}(t) = (\psi e^{t\lambda}\psi^{-1})\vec{\beta}(0) + \psi e^{t\lambda}\lambda^{-1}(e^{-t\lambda} - I)\psi^{-1}\vec{\Omega}, \tag{4.18}$$

where the eigenvalues and eigenvectors of the matrix G are denoted by λ and ψ, respectively. We do not use our own eigendecomposition algorithm, we rely on Intel oneAPI Math Kernel Library (MKL) instead. By reducing non-symmetric matrices to Hessenberg form and applying Schur factorization, the library can find eigenvalues. However, the library's documentation does not provide details about the specific implementation.[62] We measured eigendecomposition computational cost to be $O(N^{2.7})$ but the numerical complexity of Equation 4.18 depends on hardware-specific details; see Appendix A for examples.

Let us emphasize that using Equation 4.18 is not advisable most times; it is mainly meant for comparison with numerical methods. For time evolution problems, when the laser is switched on (so called switch-on dynamics), no issues have been encountered. Whereas, when the laser is switched off (decay dynamics), numerical errors affect the outcomes. In Figure 13, we show the time evolution after reaching a steady state. The curve `ODE Solver (default)` uses an Ordinary Differential Equation Solver to solve Equation 4.17 (more details in the next section). Since this curve does not show any divergence, one might consider it as acceptable.

Missing values in the `Formal Solution + Float64` curve are due to limits of double precision arithmetic. These limitations become apparent as t becomes larger because of the extremely large or small values generated by $e^{\pm\lambda t}$. Changing the tolerance to `abstol=1e-15` can lessen the error, but still, it can not match the formal solution's accuracy.

The issue occurs for long time scales, that would be hard to measure, since there is essentially zero scattered light, and perhaps only full quantum problems would benefit to explore such a regime. Nevertheless, we still need to deliver a solution to the identified issue. A natural approach would to use Quadruple Precision (`Float128`), but such solution slows down all basic mathematical operations, turning the execution prohibitively slow. The optimal solution lies in recognizing that in the switch-off scenario ($\vec{\Omega} = \vec{0}$), the issue of numerical precision can be resolved by simplifying Equation 4.18 into

$$\vec{\beta}(t) = (\psi e^{t\lambda}\psi^{-1})\vec{\beta}(0). \tag{4.19}$$

Equation 4.19 is important because it gives identical results as `Float128`, but uses `Float64` precision. Regarding execution times, Figure 14 displays a ratio of execution time of Equation 4.19 (using diagonalization) and Equation 4.17 (using an ODE solver), there is a cutoff ratio of 1.1 to make easier to visualize the line where the ratio is equal to 1. Values above 1 (in red) show that ODE is faster, which is where typical simulations are expected to be run. Meanwhile, the region where diagonalization is faster (in blue) only happens for longer time periods, $t_{max} \geq 400\Gamma^{-1}$. To optimize our results, if $t_{max} \leq 200\Gamma^{-1}$, we use an ODE solver since Equation 4.19 is quicker than the formal solution.

4.5.2 ODE Solvers

As mentioned, computing Equation 4.18 is not fast compared to numerical methods for solving the dynamics, but determining the fastest numerical method for our problem remains an open question. The `DifferentialEquations.jl` package[63, 64] which offers a variety of solvers for different categories of problems, allowing one to test various methods with minimal effort. For example, the code to solve Equation 4.17 is shown in Figure 15. At lines 3-8, the function `ode_equation` is a direct transcription of Equation 4.17. Once

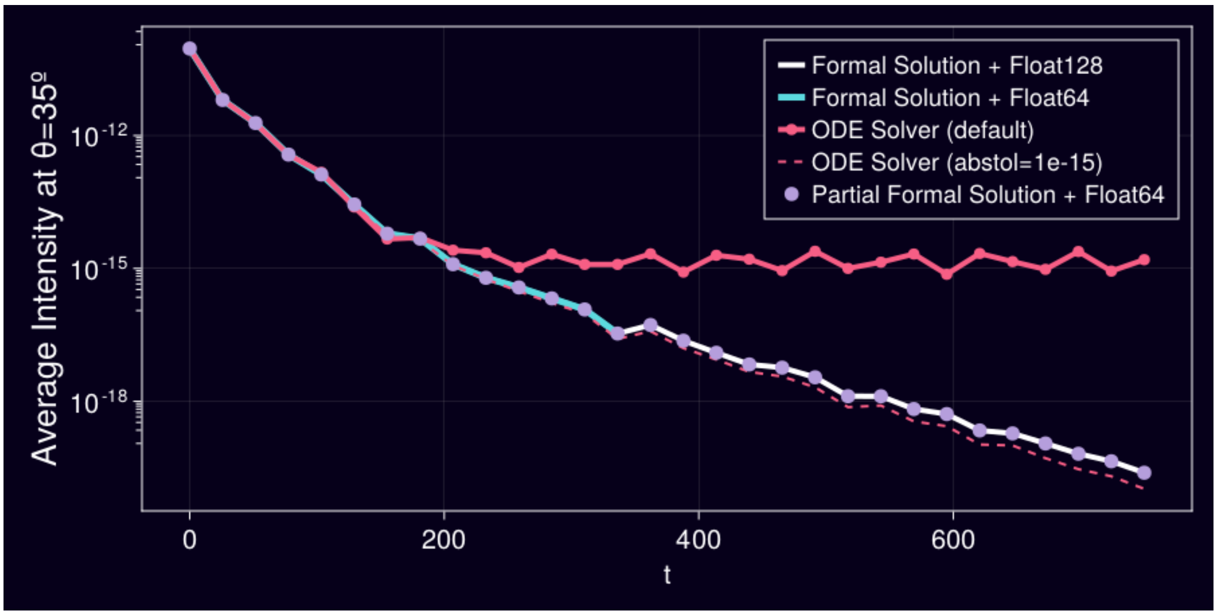


Figure 13 – For time evolution over longer times, numerical errors have a direct impact on the results. This is relevant to switch-off decay dynamics for a system starting from a steady state. `Sphere` with $N = 150, k_0R = 10$ pumped by Plane Wave with $s = 10^{-5}, \Delta = 0$.

Source: By the author.

all parameters are defined, the ODE solver can be changed at the last line and, in this particular example, we used the Runge-Kutta 4th order solver with the command `RK4()`.

To identify the fastest ODE solver, we assessed the speedup by comparing the execution time of the ODE solver, denoted as t_N, with the Formal Solution, denoted as t_F, presented in Equation 4.18. This speedup is expressed as follows

$$\text{Speedup} = \frac{t_F}{t_N}. \tag{4.20}$$

We want to emphasize that the results presented are strongly dependent on the computer hardware, including the CPU model and RAM frequency, so they should not be accepted at face value. However, the general order of magnitude for the speeds found for these results remains valid.

For benchmarking purposes, we chose a value of $N = 1500$ atoms and computed the time evolution within the time interval ($t \in [0, 75]\Gamma^{-1}$). We utilized adaptive time steps (`adaptive = true`) for all available non-stiff solvers,[64] excluding solvers that proved excessively slow or did not operate with complex numbers. In the following, we highlight the top 10 ODE solvers based on their performance, but the full list can be found in the Appendix section.

For the low-density regime, Figure 16 (a) shows that `VCABM3` is the most efficient solver. This method is a multistep solver based on the Adams-Moulton formula.[65] When

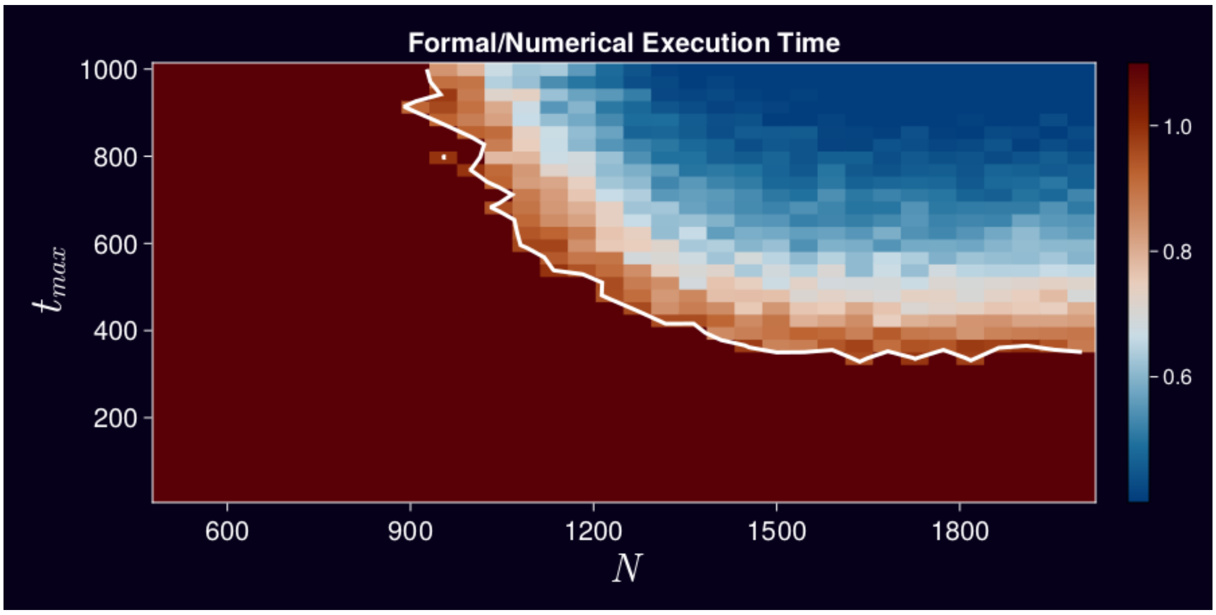


Figure 14 – Execution Time Ratio to compute the time evolution of switch-off dynamics for different numbers of particles and maximum evolution time - starting from steady state, and performing time evolution until time t_{max}. The line indicates where the ratio is 1, that is, both methods have similar speeds. The system consists of a `Sphere` with a constant density of $\rho k_0^{-3} = 0.2$ and a radius determined by N.

Source: By the author.

the atomic density is high, simulations take longer (strong interactions regime), causing a slowdown, as seen in Figure 16 (b). The fastest solution was given by a specialized solver called `RDPK3SpFSAL35`, which was initially developed for compressible fluid mechanics simulations.[66] Since the relative difference between `RDPK3SpFSAL35` and `VCABM3` is small on the high-density regime, but not for the dilute one, we set up the `VCABM3` as the default ODE solver for `LinearOptics`.

We cannot provide a speedup comparison for our `MeanField` equations since they do not have a formal solution, so we rely only on the execution time, t_N. The fastest solver for both dilute and high densities in Figure 16 is `VCABM`. Thus, it is now our preferred ODE solver for `NonLinearOptics`.

An observation is that, despite the assumption that `MeanField` is equivalent to `Scalar` with lower saturation, there is a noticeable difference, for example, in the scattered intensity, see Figure 18, with saturation as low as $s = 10^{-5}$. The solution is to set `abstol=1e-10` on the `MeanField` time evolution function.

```julia
using DifferentialEquations, CoupledDipoles

function ode_equation(β, parameters, t)
    G, Ω = parameters

    dβ = G * β + Ω
    return dβ
end

N, ρ = 1500, 0.2 # number of atoms, and sphere density
atoms = Atom(Sphere(), sphere_inputs(N, ρ)...)

s, Δ = 1e-5, 1.0 # laser saturation and detuning
laser = Laser(PlaneWave3D(), s, Δ)

simulation = LinearOptics(Scalar(), atoms, laser)
G = interaction_matrix(simulation)
Ω = laser_field(laser, atoms.r)
parameters = G, Ω

initial_condition = zeros(ComplexF64, N) # all atoms on the ground state
tspan = (0.0, 75.0) # minium and maximum time point
saveat = range(tspan[1], tspan[2], length = 30) # time points of interest
prob = ODEProblem(ode_equation, initial_condition, tspan, parameters)
elapsed_time = @elapsed solve(prob, RK4(), dt = 1e-9, adaptive = true, saveat = saveat)
```

Figure 15 – Minimal working example to measure the execution time of an ODE problem using standard Runke-Kutta of 4th order, `RK4()`. Our benchmarks focused on changing the ODE Solver since the other parameters are representative of typical simulations.

Source: By the author.

4.5.3 Steady State

4.5.3.1 `CUDA` implementation

The steady state, denoted as $\vec{\beta}_{ss}$ in Equation 4.17, has a formal solution presented as follows:

$$\vec{\beta}_{ss} = -G^{-1}\vec{\Omega}, \tag{4.21}$$

$$= -G\backslash\vec{\Omega}. \tag{4.22}$$

While various implementations of Equation 4.22 are possible, the most widely used algorithm involves performing LU decomposition on matrix G. This process results in a lower triangular matrix (L) and an upper triangular matrix (U), which are then used for forward and backward substitutions. For larger matrices, it is recommended to use iterative methods, such as Jacobi and Gauss-Seidel. These methods involve less data communication between computational blocks. We have not encountered a scenario where caching (saving) the LU decomposition or conducting simulations with a larger number of atoms that do not fit within a single computer memory would be advantageous.

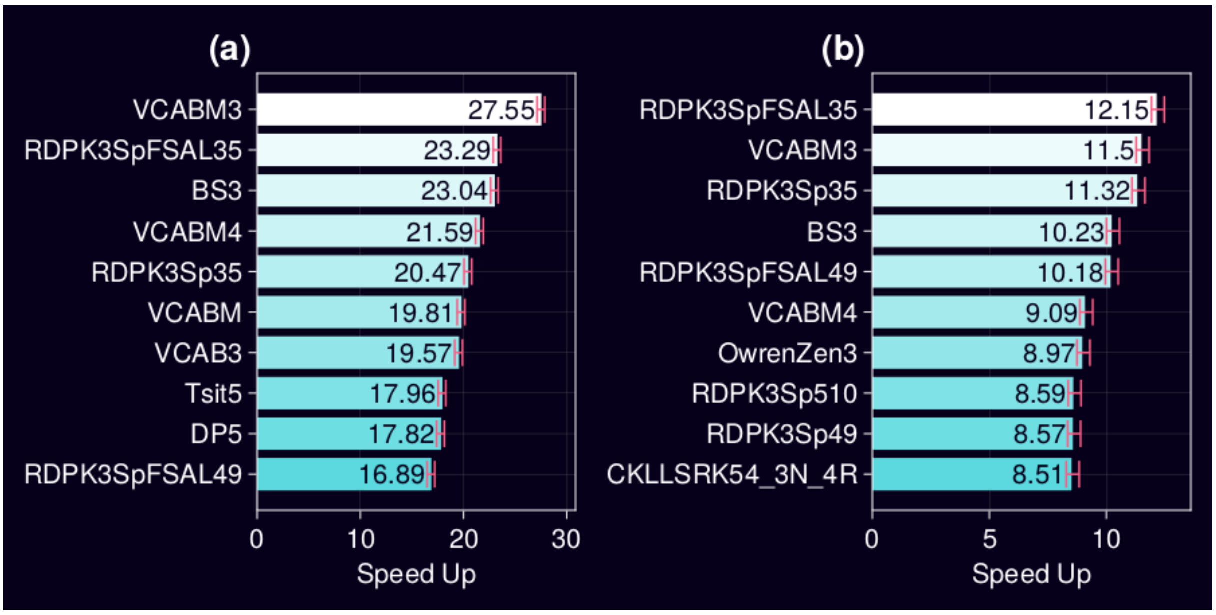


Figure 16 – Speedup of the top 10 ODE solvers against the formal solution. Laser was switch-on, on the time interval $t \in [0, 75]\Gamma^{-1}$, averaged over 50 realizations. In panel (a), the density is $\rho k_0^{-3} = 0.02$, and in panel (b), $\rho k_0^{-3} = 0.2$. Both figures had $N = 1500$.

Source: By the author.

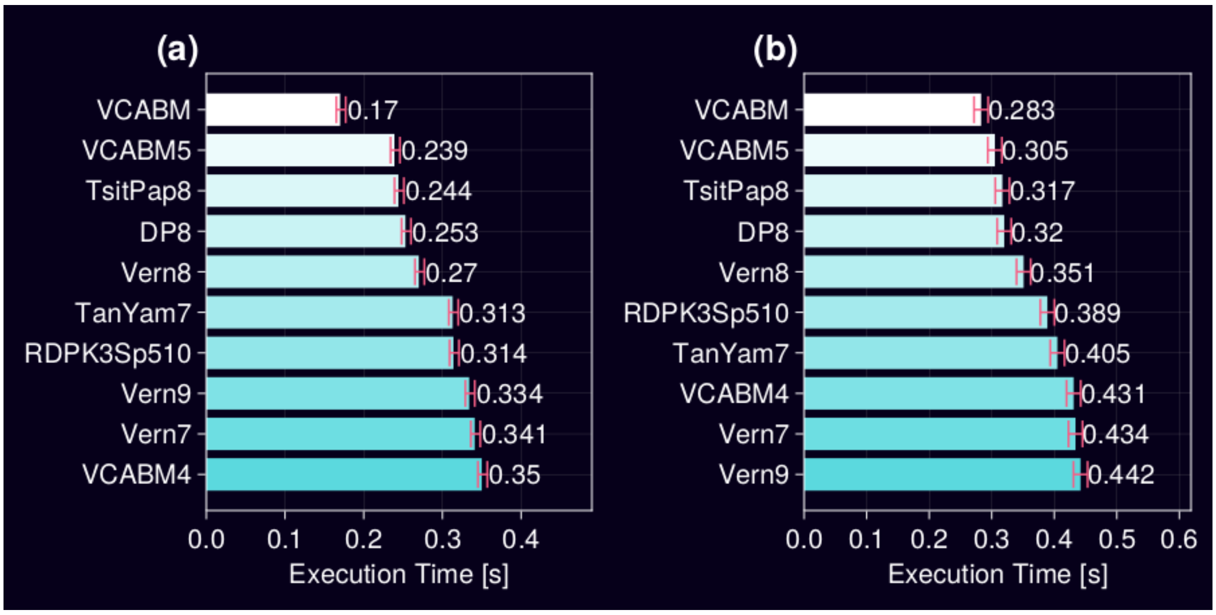


Figure 17 – Execution time to compute time evolution using `MeanField` model on the low and high-density regimes. The parameters are the same as previous speedup figures.

Source: By the author.

We are investigating the advantages of using Graphics Processing Units (GPUs),

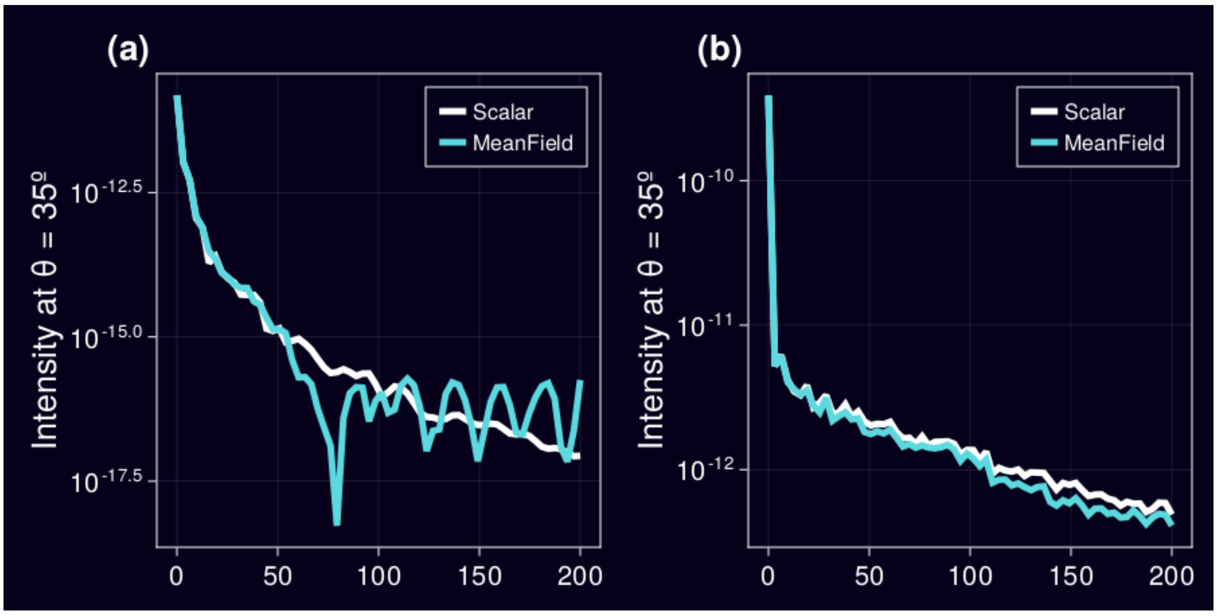


Figure 18 – Decay dynamics for a `Cube` with $N = 500$ shows the difference of `Scalar` and `MeanField` models with low saturation, $s = 10^{-5}$. The relative error highlights the finding. Panel (a) correspond to density $\rho k_0^{-3} = 0.02$, and panel (b) to $\rho k_0^{-3} = 0.2$.

Source: By the author.

particularly NVIDIA GPUs with the CUDA library.[67,68] To conduct our experiments, we used cloud resources provided by the website vast.ai.[69] This platform allowed us to test the solution on a variety of GPU models. Figure 19 displays the models and their average execution times, for a fixed configuration. The machine configurations to which each GPU was attached, as well as the details of the benchmark code, are outlined in Appendix D.

According to our findings, there is a strong correlation between the GPU's theoretical GFLOPs (Giga Floating Point Operations Per Second), and the time it takes to compute $\vec{\beta}_{ss}$, as displayed in Figure 19(b). GPUs from the top-tier category (commonly labeled as `xx90`, typically designed for high-end gaming applications, and server-grade GPUs denoted as `x100`) demonstrate significantly faster execution times compared to lower-end counterparts. These results are relevant and useful for deciding on future hardware investments. For example, getting two computers with 3090 GPU might cost less and perform the same as a single computer with 4090 GPU.

4.5.3.2 `MeanField`

The nonlinear `MeanField` equations require numerical optimization methods to estimate their steady-state solutions. Within the Julia ecosystem, the main packages for handling nonlinear systems are `NLSolver.jl`[70] and `SIAMFANLEquations.jl`.[71] However, the latter package has certain limitations for operations involving complex numbers. As

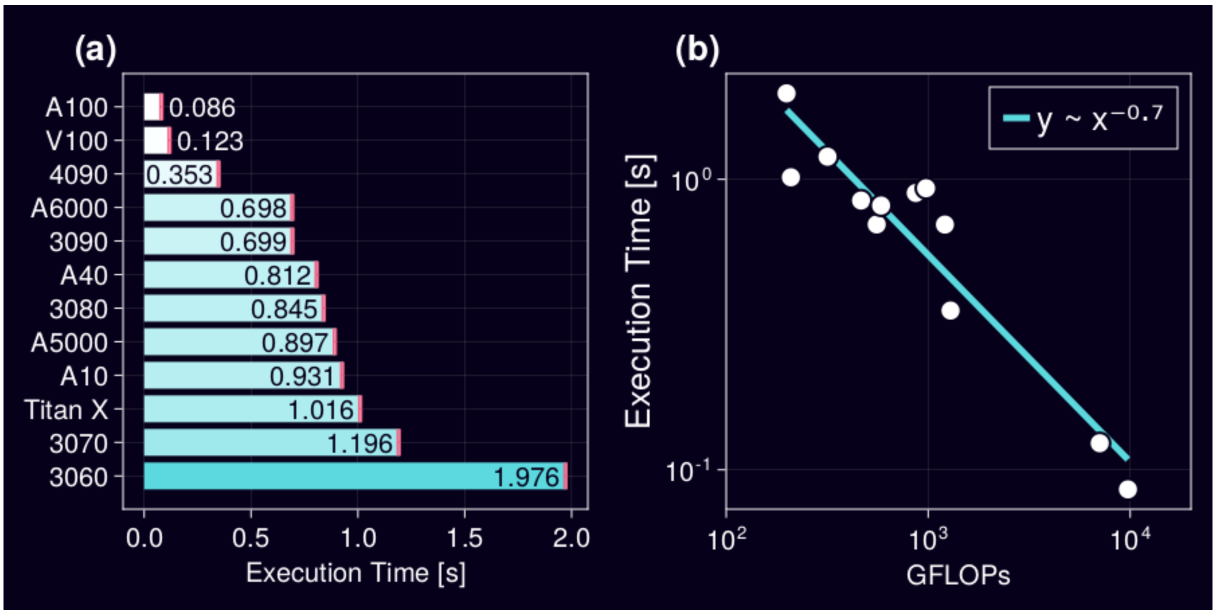


Figure 19 – Execution time to solve $x = G\backslash b$ for a system with $N = 5000$ atoms using `CUDA`.

Source: By the author.

a result, it was not considered in our analysis.

`NLSolver.jl` has 3 methods to solve nonlinear equations: `Trust region`, `Newton with linesearch`, and `Anderson acceleration`. We compared three methods in Figure 20 and found that the `Anderson acceleration` approach is faster in solving various problems. We have identified that performing the `Anderson acceleration` method can be enhanced by selecting an optimal initial condition. Specifically, we use the time-evolved state results from $t \in [0, 250]\Gamma^{-1}$ as the initial condition for the `NLSolver.jl` package.

The `NLSolve.jl` package does not consistently converge. Small system configurations with less than 100 atoms are more likely to experience non-convergence, although the reasons are unclear. Also, saturation values that are too high or too low can cause problems. We have experienced issues when saturation is above $s \approx 10^{-1}$ or below $s \approx 10^{-6}$.

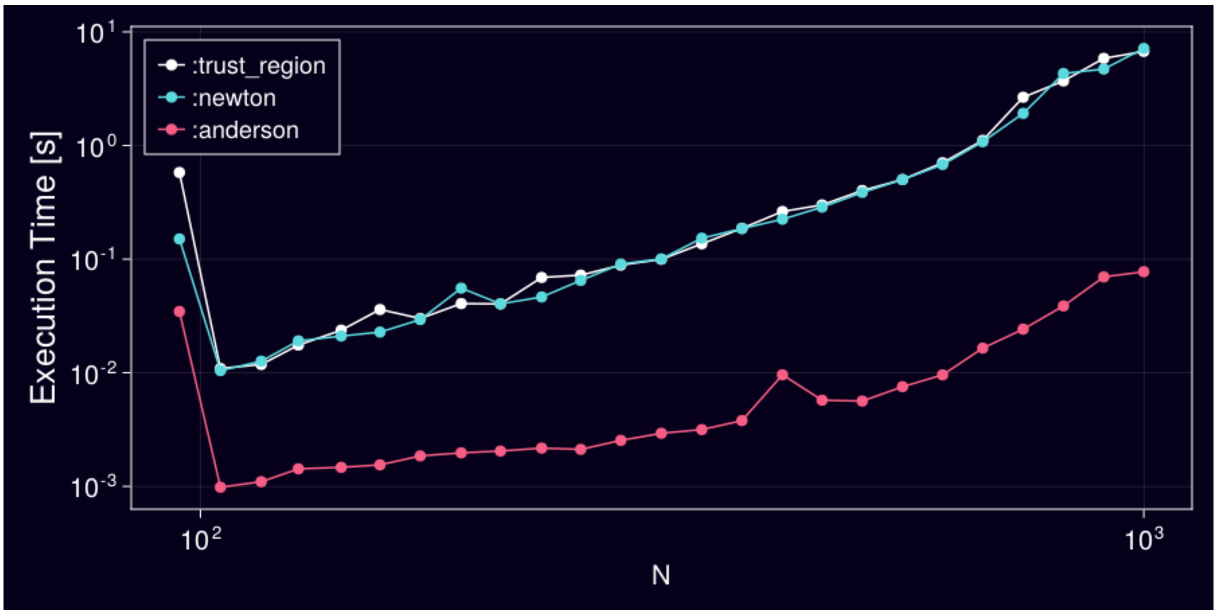


Figure 20 – Execution Time to find the steady state of `MeanField` equations using non-linear methods.

Source: By the author.

5 LIGHT STATISTICS AND LOCALIZATION

We now move to the physical cases to which the numerical schemes discussed previously were applied to, considering light scattering in disordered media problems.

5.1 Anderson Localization

Anderson Localization is a fascinating phenomenon that occurs when a wave travels through a disordered medium and experiences scattering events. As a result of interference effects, the wave becomes exponentially confined to certain regions of space. This phenomenon was first proposed by Anderson in 1958[72] as a mechanism for the metal-insulator transition for electrons. Over time, Anderson Localization has been recognized as a general wave phenomenon and has been observed in various settings, including electronic systems,[73] photonic structures,[74] matter waves,[75] and cold atom experiments.[76] To better understand this concept, one can refer to Schrödinger's equation, which provides a concrete mathematical framework for studying wave behavior

$$\hat{H}|\phi\rangle = \left(-\frac{1}{2}\nabla^2 + U\right)|\phi\rangle = E|\phi\rangle, \tag{5.1}$$

where U is a random potential. One then proceeds to study the system eigenspectrum. To create a matrix representation of Equation 5.1, first, we discretize the space into N regular bins. For convenience, we consider a 1D space. To each position, we associate a potential $U_j = U_0 R_j$ $(j = 1..N)$, where R_j is drawn from a uniform distribution. Inspired by Kutz[77] and Landau,[78] the Schrodinger equation is rewritten using the matrix representation:

$$H = \left(-\frac{1}{2}\nabla^2 + U\right) = \frac{1}{\delta x^2}\begin{pmatrix} -2 & 1 & 0 & 0 & \ldots & 0 \\ 1 & -2 & 1 & 0 & \ldots & 0 \\ 0 & 1 & -2 & 1 & \ldots & 0 \\ & & \vdots & & & \\ & \ldots & & 1 & -2 & 1 \\ 0 & \ldots & & 0 & 1 & -2 \end{pmatrix} + U_0 \begin{pmatrix} R_1 & 0 & 0 & 0 & \ldots & 0 \\ 0 & R_2 & 0 & 0 & \ldots & 0 \\ 0 & 0 & R_3 & 0 & \ldots & 0 \\ & & \vdots & & & \\ & \ldots & & 0 & R_{N-1} & 0 \\ 0 & \ldots & & 0 & 0 & R_N \end{pmatrix}. \tag{5.2}$$

δx^2 is the size of the space bins. The Hamiltonian produced in Equation 5.2 is then decomposed as $H = \Psi E \Psi^{-1}$, that is, eigenvalues are stored as diagonal elements of matrix E, and their associated eigenstates are the columns of the matrix $|\Psi\rangle$

$$E = \begin{pmatrix} E_1 & 0 & 0 & \dots & 0 \\ 0 & E_2 & 0 & \dots & 0 \\ & & \vdots & & \\ & \dots & & E_{N-1} & 0 \\ 0 & \dots & & 0 & E_N \end{pmatrix}, \Psi = \begin{pmatrix} \Psi_{11} & \Psi_{12} & \Psi_{13} & \dots & \Psi_{1N} \\ \Psi_{21} & \Psi_{22} & \Psi_{23} & \dots & \Psi_{2N} \\ \Psi_{31} & \Psi_{32} & \Psi_{33} & \dots & \Psi_{3N} \\ & & \vdots & & \\ \Psi_{N-11} & \dots & & \Psi_{N-1N-1} & \Psi_{N-1N} \\ \Psi_{N1} & \dots & & \Psi_{NN-1} & \Psi_{NN} \end{pmatrix}. \quad (5.3)$$

The eigenvectors Ψ_{jn} are concatenated as columns of a matrix $\Psi = [\Psi_{j1}|\Psi_{j2}|...|\Psi_{jN}]$, and Figure 21 presents the *Spatial Profile*, $|\Psi(x)|^2$, of the 3 eigenstates with lowest eigenvalues, with and without random potential. For a free particle in a box ($U = 0$ and hard wall), the numerical solution tends to the known periodic solutions $\Psi(x) = \sqrt{2/L}\sin(n\pi x/L)$ of the square box potential, panel (a). Differently, for a random potential, $|\Psi(x)|^2$ exhibits an exponential decay, see panel (b), that is, the modes are said to be *localized*.

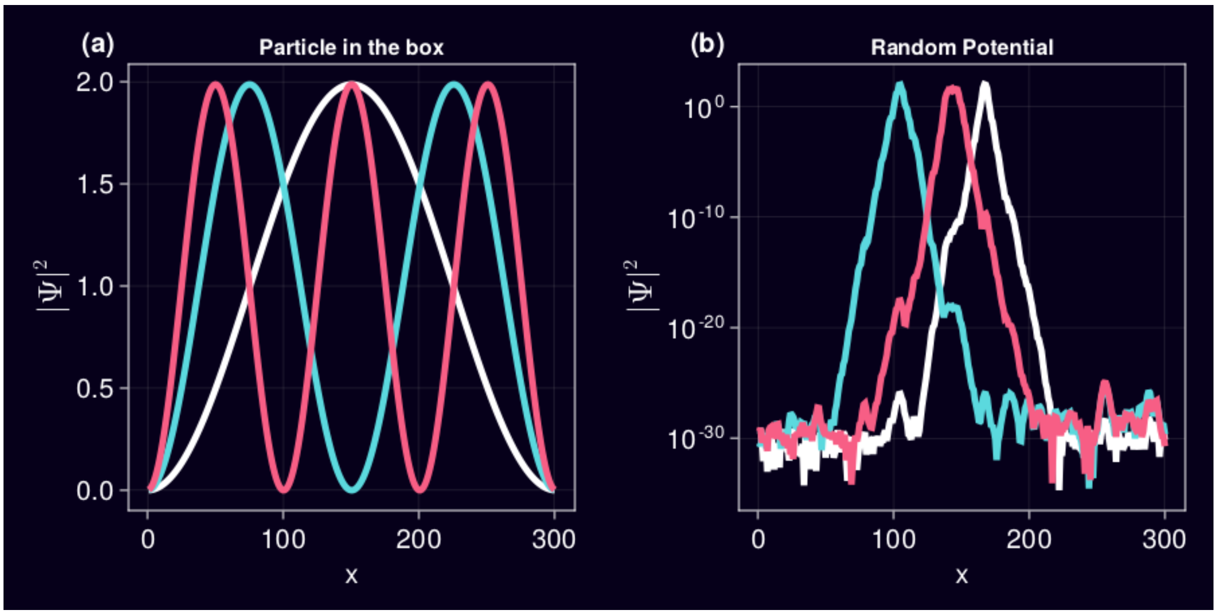


Figure 21 – Eigenstates from Schrodinger's equation in 1D with $U_0 = 10^5$. In panel (a), where no disorder is present, the eigenstates present a wave-like behavior filling the entire box, therefore, these are named *extended* modes. In panel (b), in the presence of disorder, the eigenstates are confined in some regions of space, exhibiting an exponential decay (until numerical precision is reached, and only numerical noise remains).

Source: By the author.

This exponential decay is characterized by the localization length ξ, corresponding to $|\Psi(r)|^2 = Ae^{-r/\xi}$. Some experiments allow for direct inspection of the wave function,[79–81] and monitoring its narrowing/spreading from which the localization length of the scattering mode can be extracted (the localization length for the system is defined, strictly speaking, only in the thermodynamic limit). To increase the accuracy of the fitting procedure, it is convenient to "linearize" the equation as $\ln(|\Psi(r)|^2) = \ln(A) - r/\xi$, before performing a linear fit as a function of r, for example using a linear least square fitting procedure.

The spatial extension of Ψ can be further characterized by introducing the Inverse Participation Ratio (IPR), a number between zero and one, which quantifies the number of atoms contributing effectively to an eigenmode, ultimately giving an estimation of the mode size. For a mode n, the IPR_n is given by[82]

$$\text{IPR}_n = \frac{\sum_j^N |\Psi_{nj}|^4}{\left(\sum_j^N |\Psi_{nj}|^2\right)^2}. \tag{5.4}$$

In particular, IPR reaches its maximum value, 1, when the mode is completely localized on a single point x in space. Oppositely, the IPR reaches value $1/N$, when all spatial positions contribute equally to the mode ($\Psi_{nj} = 1/\sqrt{N}$), which corresponds to an extended mode. It is worth mentioning that other signatures of localization include the statistics of eigenenergies,[83] and the fractal dimension of eigenstates.[84] To compute IPRs one should use the function `get_IPRs`, and its inverse with `get_PRs`.

The IPR is a quantity that requires access to the eigenstates of the system, but the eigenstates are often not accessible experimentally, see a case where eigenstates are inspected on Semeghini *et al.*[85] To overcome this limitation, a more commonly used approach is to study the macroscopic transport properties of the system, particularly by measuring the transmission, T, as a function of the sample thickness, L. In the diffuse regime, the transmission is expected to follow Ohm's Law, with $T \propto 1/L$.[86] As the system undergoes disorder-induced changes,[87,88] the transmission scaling transitions through $T \propto 1/L^2$ before reaching the strongly localized regime with $T \propto \exp(-L/\xi)$,[87,88] ξ is the localization length. The experimental observation of this change in transmission scaling was used by Wiersma *et al.*[89] to claim the observation of three-dimensional (3D) localization of light. However, these findings were later challenged by Beek *et al.*,[90] who provided evidence suggesting that the exponential transmission could be attributed to weak light absorption rather than localization effects.

A different experimental approach, pioneered by Chabanov, Stoytchev and Genack,[91] involves measuring the fluctuations in transmission. Recently, Cottier *et al.*[92] tested this method using both `Scalar` and `Vectorial` models, and they successfully observed clear signatures of localization. In this chapter, we aim to replicate and explore these results and signatures. Specifically, we will investigate the impact of atom saturation using the `MeanField` model, which has not been previously considered. Unlike earlier works that focused solely on the single-photon regime, where only the atomic dipole moments are relevant, we will also consider the influence of the atomic population. As a preliminary step, we will demonstrate how Anderson localization can be observed in the Coupled Dipole Model.

5.2 Localized Scattering Modes using `Scalar` Model

Differently from Schrodinger's equation discussed previously, the Coupled-Dipole Equation does not represent directly a wave equation, so the meaning of its eigenenergies and eigenstates still needs to be defined. We diagonalize the Green's matrix representing the atomic interaction which, for 3D `Scalar` case, was defined in Equation 2.23, which reads

$$\mathbf{G}_{jm}^{3D-scalar} = \left(-\frac{\Gamma}{2} + i\Delta\right)\mathbb{1}_{jm} + i\frac{\Gamma}{2}\sum_{j}^{N}\sum_{m\neq j}^{N}\frac{e^{ik_0 r_{jm}}}{k_0 r_{jm}}. \tag{5.5}$$

The function `interaction_matrix` computes the matrix for `Scalar` and `Vectorial` models. We highlight that the density $\rho = N/V$ is the driving force for localization in 3D clouds, that is, systems with a density below some threshold ($\rho k_0^{-3} < 0.1$) do not exhibit localization,[17] even in the large N limit. Indeed, the increasing density corresponds to stronger scattering properties, and thus to an increase in disorder for the propagating waves. The eigendecomposition of Equation 5.5,

$$\mathbf{G}_{jm}^{3D-scalar} = \Psi\lambda\Psi^{-1}, \tag{5.6}$$

produces eigenvalues and eigenvectors similar to Equation 5.3, but the eigenvalues λ_n are complex — because the interaction matrix is not Hermitian. Their real part corresponds to the decay rate of each mode, $\Gamma_n = -\Re(\lambda_n)$, and the imaginary part provides the energy shift from the single atom transition, $\Delta_n = \Im(\lambda_n)$ — since $\Delta_n = \omega_0 - \omega_n$, $\omega_n = \omega_0 - \Delta_n$. These computations are performed by the function `get_spectrum`. The spatial profile, produced by the eigenvectors, can be interpreted as the excitation of the j-th atom, on the n-th mode. Let us now illustrate this point by presenting the profiles of a localized and an extended mode.

Let us imagine atoms randomly distributed inside a spherical cloud, with density chosen to produce localized eigenmodes. One could select a mode n $|\Psi_{jn}|^2$, and plot the value of each atom j according to some color scheme. However, visualizations inside volumes are intrinsically more complicated to interpret, and one has to rely on some creativity. In Figure 22(a)-(c) shows a spherical cloud where each marker size is proportional to $|\Psi_{jn}|^2$, the meaning of `LOC, SUB, SUPER` will be defined shortly, yet, this spatial visualization is not insightful.

A better approach is to define the distance of the atoms from the mode *center of mass*, $\mathbf{r}_{cm}^n$, using $|\Psi_{jn}|^2$ as weights

$$\mathbf{r}_{cm}^n = \frac{\sum_j \mathbf{r}_j|\Psi_{nj}|^2}{\sum_j |\Psi_{nj}|^2}. \tag{5.7}$$

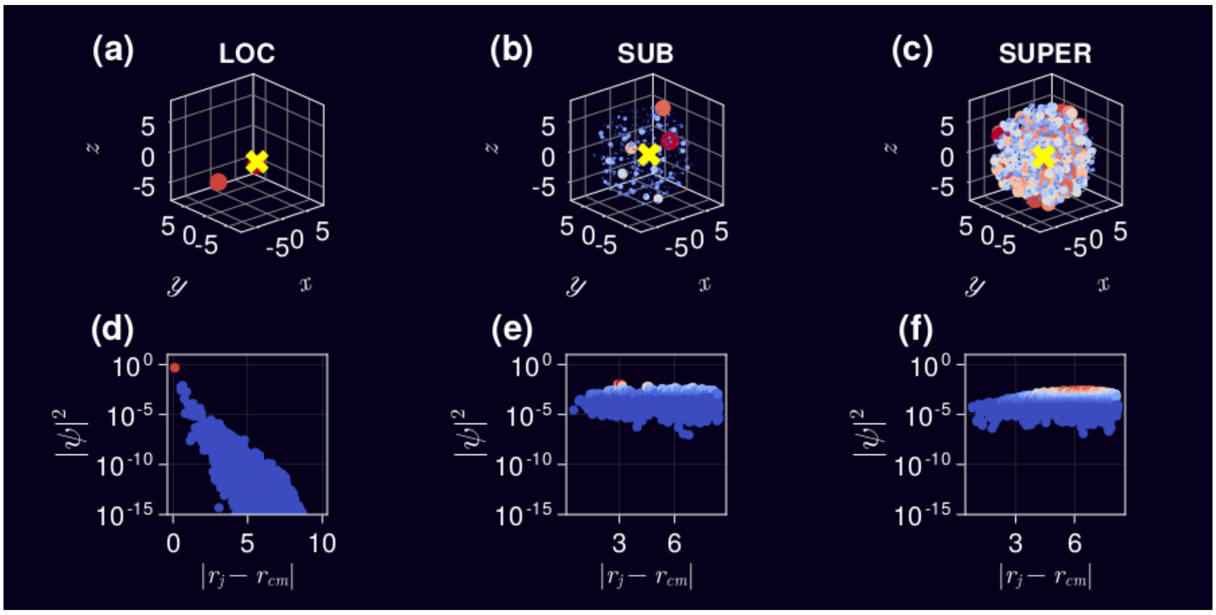


Figure 22 – A sphere with $N = 2000$ atoms (density $\rho k_0^{-3} = 1.0$) is excited by a plane wave in the z direction with detuning $\Delta\Gamma^{-1} = 1.0$. The marker size in plots (a)-(c) represents $|\Psi_{jn}|^2$, which spans orders of magnitude, causing some points to be invisible on the plot. The yellow cross marker indicates the center of mass positions of each mode. In plots (d)-(f), profiles of the mode excitation are shown as a function of distance from the center of mass. For more details, refer to the main text.

Source: By the author.

Then, we compute a new spatial profile, based upon the eigenvectors and the distance between each atom $\mathbf{r}_j$ from this center of mass. The results are presented in Figure 22(d)-(f). Note that the localized mode exhibits an exponential decay, from which we extract its localization length ξ of the scattering mode.

The distribution of eigenvalues in the complex plane exhibits interesting characteristics. We classify modes with $\Gamma_n > \Gamma$ as **superradiant** (or `SUPER` for short) because they decay faster than the natural single-atom decay rate. On the other hand, modes with $\Gamma_n < \Gamma$ are referred to as **subradiant** (or `SUB`), and among them, some exhibit decay rates Γ_n many orders of magnitude smaller than the single-atom decay rate, as illustrated in Equation 5.2. These modes are termed **localized** (or `LOC`) since they usually correspond to exponentially localized modes. However, distinguishing between `SUB` and `LOC` modes relies on other metrics, such as the Inverse Participation Ratio. The classification of each mode contains a degree of arbitrariness, and Appendix B provides further details on our current implementation. It is sufficient to mention that the method `classify_modes` is responsible for this task.

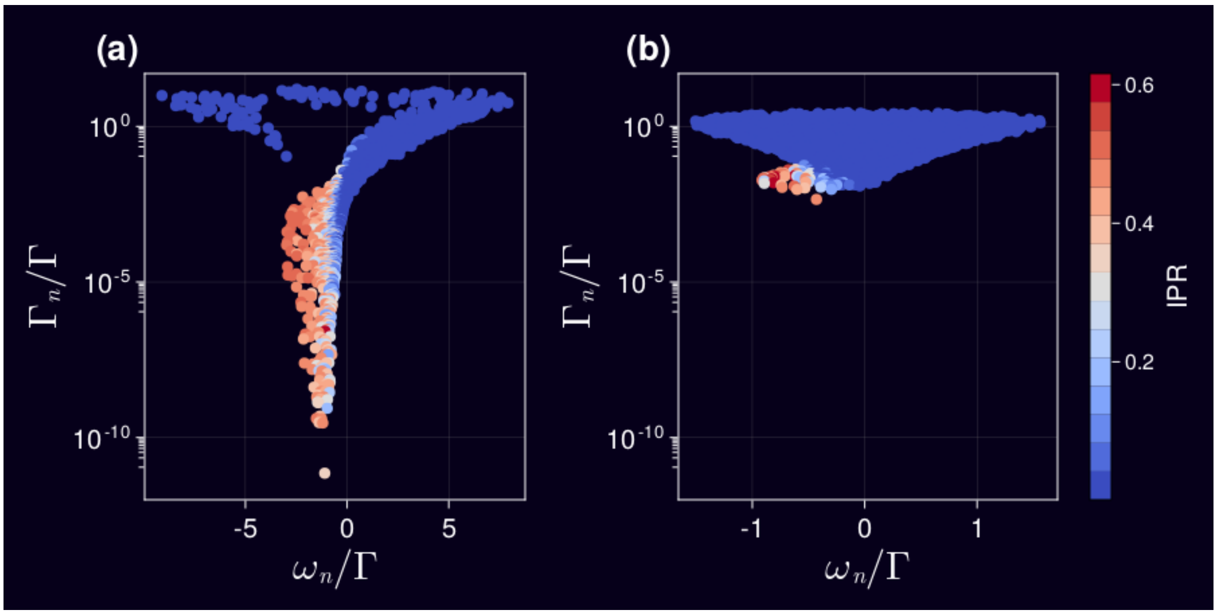


Figure 23 – Examples of eigenmodes on the Complex Plane: (a) $\rho k_0^{-3} = 0.5$, modes with $\Gamma_n/\Gamma < 10^{-4}$ also present higher IPRs, indicating localization. (b) $\rho k_0^{-3} = 0.02$, modes densely accumulated in a small region.

Source: By the author.

5.3 Scattered Light as Localization Signature

The localization signatures we have discussed thus far rely on the properties of eigenmodes. However, many experimental setups lack direct access to the eigenvalue statistics shown in Figure 22. To address this limitation, Cottier *et al.*[92] proposed an alternative approach by studying the statistics of scattered light, which is generally a more accessible observable. Their investigation revealed that the intensity fluctuations exhibit an increase precisely in the parameter regime where localized scattering modes appear.

Figure 24 presents the scattered intensity I for high and low densities. Their spatial profiles appear different, but more precise indicators are needed.

Cottier *et al.*[92] quantified that at low density, the speckle pattern follows the *Rayleigh distribution*, which reads

$$P(I) = \frac{1}{\langle I \rangle} e^{-I/\langle I \rangle}, \tag{5.8}$$

where the average $\langle\rangle$ is here taken on the polar angle. In fact, Equation 5.8 is an *exponential probability distribution* for the intensity has been described by Goodman.[93] However, in the localized regime, with higher densities, these statistics are no longer valid. To characterize the intensity fluctuations, the intensity variance, σ^2, can be monitored. In particular, the variance for the distribution Equation 5.8 is $\sigma^2 = \langle I \rangle^2$, since Equation 5.8 is normalized, and one gets a unitary variance, $\sigma^2 = 1$.

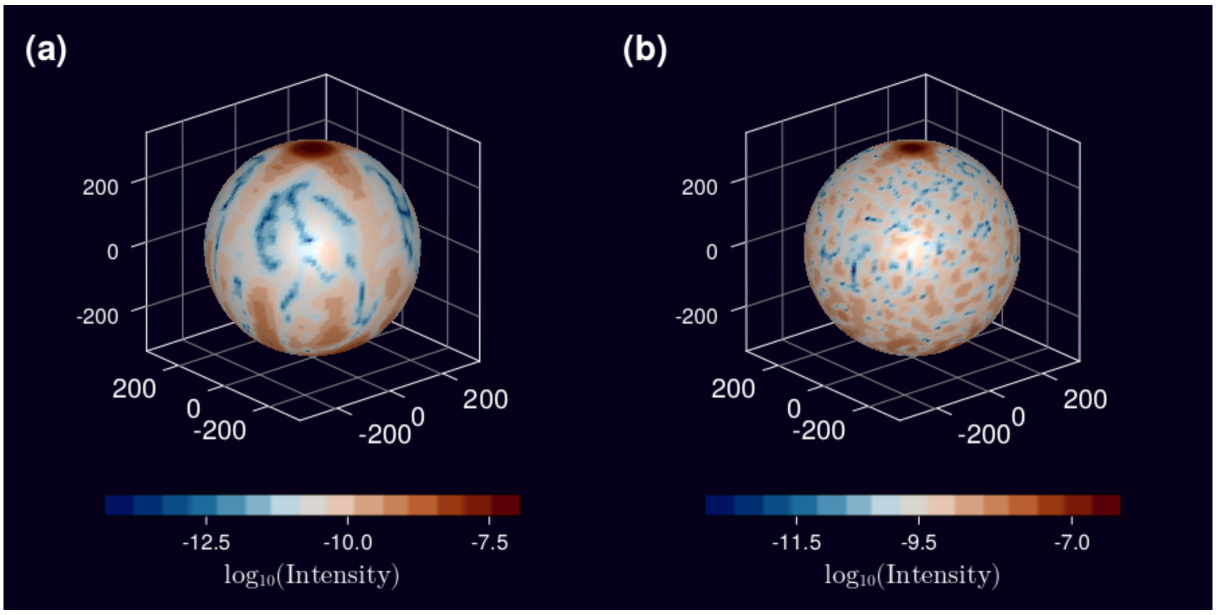


Figure 24 – To visualize the scattering in all directions, we construct a spherical shell around our atomic cloud using 15,000 grid points. At each position on the shell, we measure the intensity and combine them to create a surface representation. Both spheres enclose a cubic cloud of $N = 3000$ atoms. In (a), the density is $\rho k_0^{-3} = 0.1$, while in (b), the density is $\rho k_0^{-3} = 0.02$.

Source: By the author.

The procedure to obtain sufficient statistics and compute the variance σ^2 can be summarized in 7 steps:

1. Create one atomic realization;
2. Compute the steady-state solution
3. Measure the light intensity at certain points in space;
4. Save the intensities into an Array;
5. Repeat steps 1-4 for a given number of realizations;
6. Compute the average from all the data, and divide each intensity by it;
7. Produce a `scatter` plot histogram of the data to facilitate interpretation.

Our analysis results, shown in Figure 25, are consistent with Cottier *et al.*.[92] The white dots (low density) exhibit a variance of $\sigma^2 = 1.042$, very close to unity. In contrast, in the localized regime (high density), the variance is $\sigma^2 = 4.96$. We acknowledge that the precise variance value depends on various parameters, including laser waist and angle of measurement. The following section will delve into exploring some of these aspects.

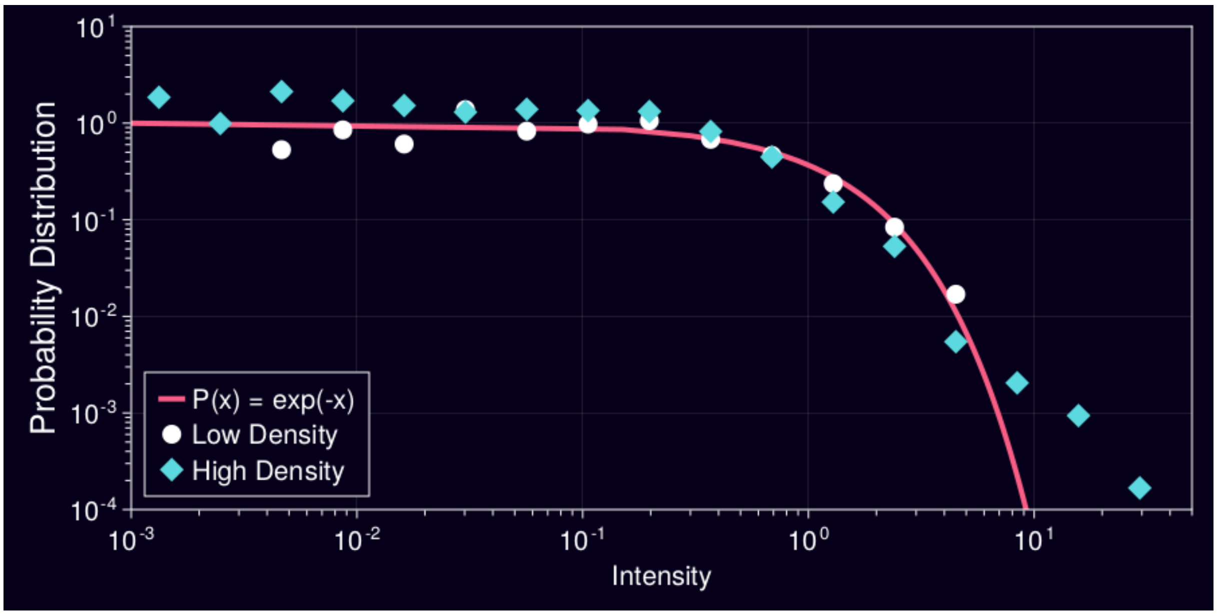


Figure 25 – Scattered intensity probability distribution for two density regimes: white dots represent Low Density with $\sigma^2 = 1.042$, and blue diamond-markers represent High Density with $\sigma^2 = 4.954$. Each curve has been obtained from 10 realizations of a `Cube` with a fixed side of $k_0L = 32.4$ (Low Density with $N = 684$ and High Density with $N = 6066$), excited by a laser with $w_0 = k_0L/4$ and $\Delta\Gamma^{-1} = 1$. The scattered intensity was measured in a ring of 64 points at an angle of $\theta = 5\pi/12$.

Source: By the author.

5.4 Fluctuations of the variance

Figure 25 provides an illustration of the increase in intensity fluctuations in the localized regime. However, it offers only a partial view as it pertains to a specific measurement position and atomic configuration. A more comprehensive analysis, like Figure 26, demonstrates that the variance varies substantially for different angles and atomic distributions. The vertical line in Figure 26 indicates $\theta = 5\pi/12$, the angle used in Figure 25. Nonetheless, Figure 26 represents a behavior of the fluctuations associated with a certain set of parameters. For example, Figure 27 highlights a case where the same atomic configuration yields different variance results when changing the laser waist and system size ratio.

We conducted a study of variance across different angles for a fixed spherical cloud and excited by three lasers with different laser waists. As observed in Figure 27, the variance increases, consistent with our previous findings, only for laser waist half the system size. Nonetheless, when the laser waist is comparable to the system size (indicated by the blue and pink curves), the variance remains consistently below 1.

Figure 28 complements Figure 26 by considering the number of repetitions to

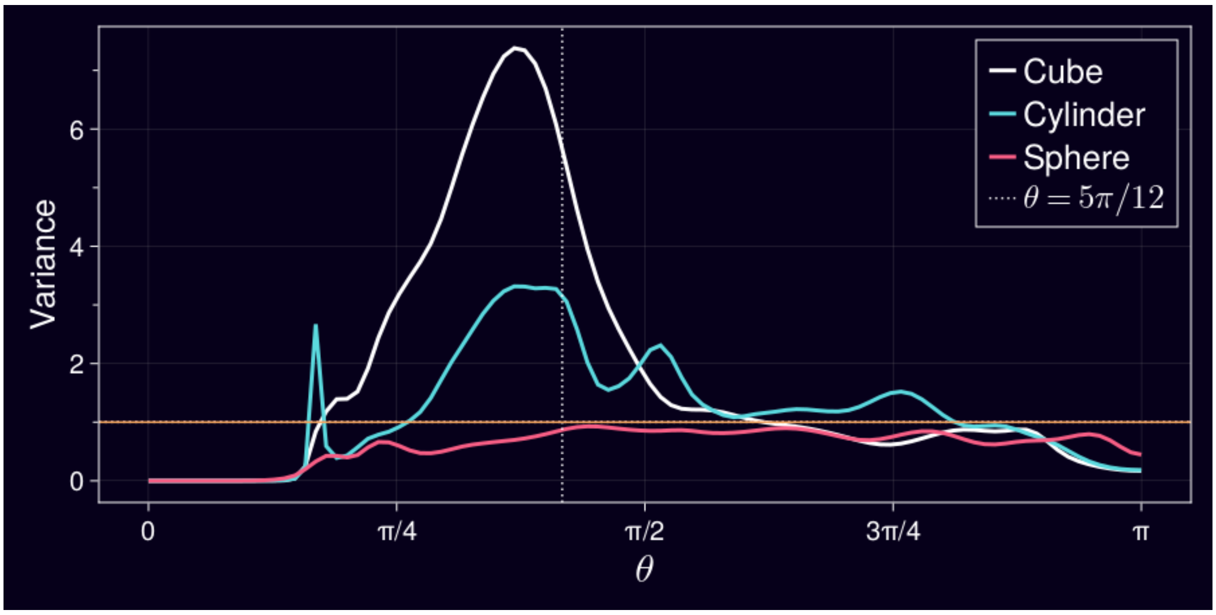


Figure 26 – Variance dependence at observation angles and cloud shape - with $N = 6066$ and $\rho k_0^{-3} = 0.17$. The Cubical shape has the largest variances, while the Spherical does not even reach $\sigma^2 = 1$ in most angles.

Source: By the author.

produce each data point. The results for all atomic distributions with 320 realizations converge to values close to 3200, but their absolute values still do not match. Since the absolute values of the variance are context-dependent, we had to make some *ad hoc* choices and trade-offs. We note that even with 16-32 realizations, we can observe the increase of fluctuations characteristic of the localization regime. Therefore, we perform simulations within this range to reduce the execution time for our analysis.

5.5 Localization Signature with `MeanField` Model

`Scalar` model is valid for linear optics regime, that is, very low laser saturation. Using `MeanField` model allows us to overcome such limitation, and may account for eventual nonlinearities in the experiments, due to the finite pump strength. Considering the published literature, it is unclear if such nonlinearities would overshadow the localization signature or not. To answer such question, Figure 29 compares `Scalar` and `MeanField` models using different saturation. We found that increasing the saturation decreases the variance, and only the lowest saturation values display variance above unity, leading us to conclude that, for the system sizes that we consider, any future experiments should have a maximum saturation level of $s_0 = 10^{-4}$ to capture the localization signature, with a recommended saturation regime of $s_0 = 10^{-5}$.

Before delving into the study of the incoherent scattering component, it is crucial to check its significance on the localization signatures, or if Figure 29 results were

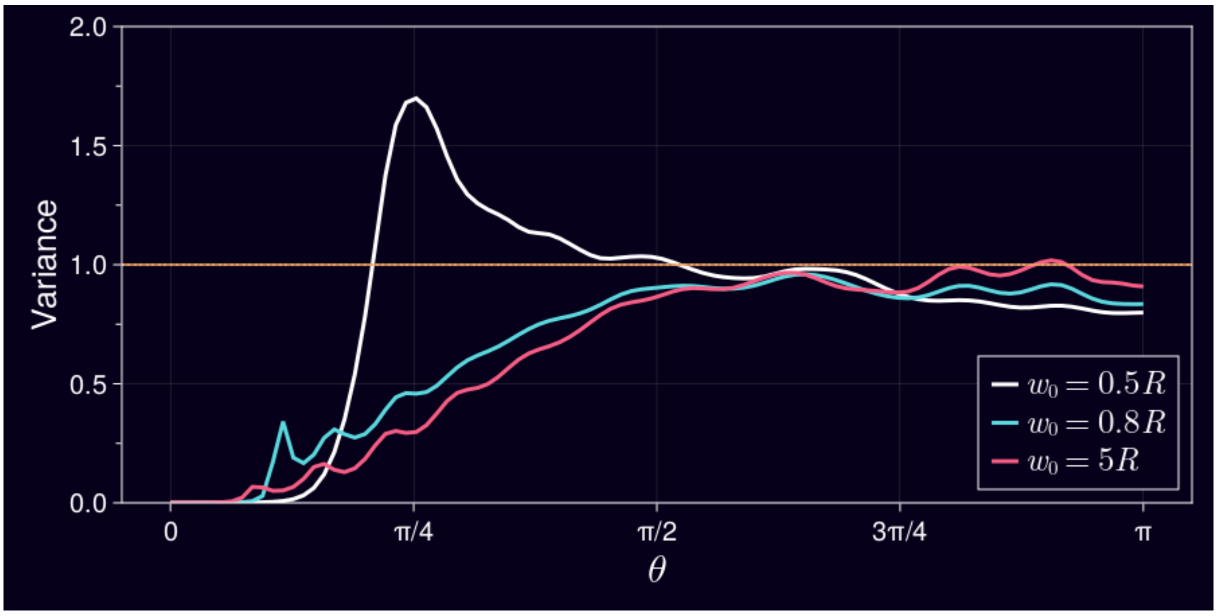


Figure 27 – We investigated the variance across different angles for spherical clouds (averaging over 200 realizations) with a density of $\rho k_0^{-3} = 0.1$ and a radius of $k_0 R = 4.5\pi$. The clouds were excited with three different laser configurations, with detuning $\Delta\Gamma^{-1} = 0.3$ and strength $s_0 = 10^{-6}$, but each with a different waist. The variance exhibits distinct values depending on the laser parameters, which poses limitations on obtaining quantitative results.

Source: By the author.

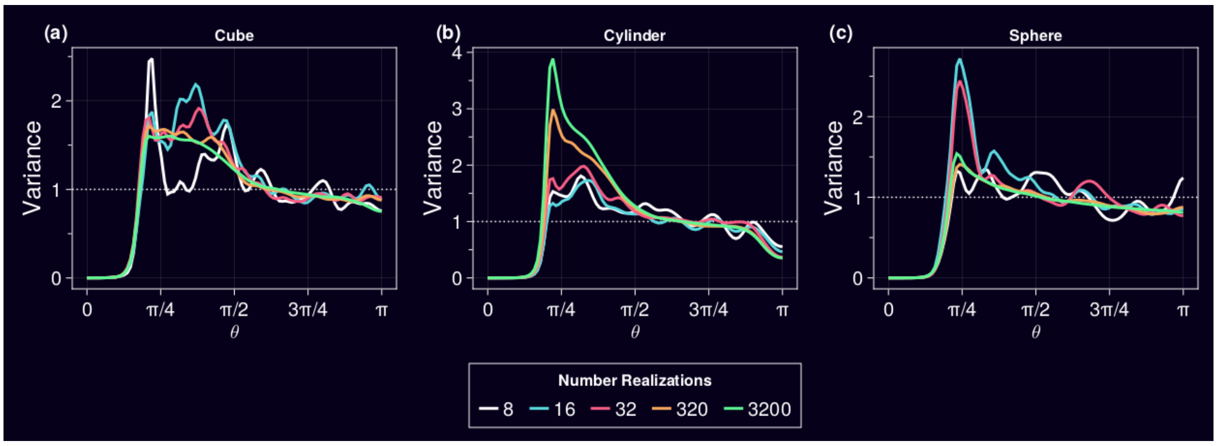


Figure 28 – For all cloud shapes with $N = 2000$ and density $\rho k_0^{-3} = 0.1$. The variance peaks for the `Cylinder` and `Sphere` occur in a region around $\pi/4$. On the other hand, the `Cube` exhibits a flat region of angles with similar variance values from $\pi/4$ to $\pi/2$.

Source: By the author.

isolated cases. To investigate this point, we created a variance map, $\sigma^2(\rho, \Delta)$, displayed in Figure 30, which shows the variance as a function of density and detuning. Our findings demonstrate that $\sigma^2 < 1$ occurs in the same region where Cottier *et al.*[92] observed

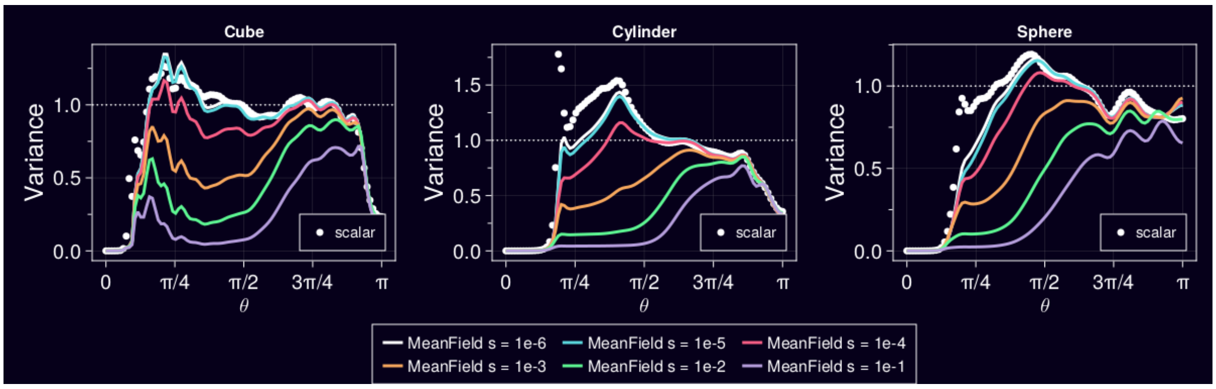


Figure 29 – Variance changes for different saturation levels shows the variance peak being lost. Simulations had fixed density $N = 2000, \rho k_0^{-3} = 0.1$ and 32 realizations. The laser had $w_0 = R/2$ and $\Delta\Gamma^{-1} = 0.3$.

Source: By the author.

fluctuation larger than 1. The affected region expands with increasing saturation, but it remains limited. This suggests that regions with large detuning are not influenced, indicating that the incoherent component mainly affects the localization region, at least for the parameters considered (which correspond to a rather weak saturation parameter).

To understand the decrease in variance and its implications, we first study the probability distribution of intensity at various saturation levels. Figure 31 illustrates our expectations for low densities, where $\sigma^2 \approx 1$ holds true for any saturation level. However, at high densities, we observe that as saturation increases, the probability distribution functions tend to have higher values, contrary to the expected behavior for independent scatterers, where light intensity should approach zero ($I \approx 0$), due to destructive interference. Therefore, the presence of an incoherent component from `MeanField` model had a strong influence on intensity fluctuations.

To illustrate even further the direct link between the decrease in variance and the population $\langle\sigma^z\rangle$, we compare the scattered intensity with and without incoherent component, I_{inc}. For convenience, the intensity in the far field is re-written here as

$$I = I_{\text{coh}} + I_{\text{inc}} = \left| i\frac{\Gamma}{2}\frac{e^{ik_0R}}{k_0R}\sum_j \langle\sigma_j^-\rangle e^{-ik_0\hat{n}\cdot\vec{r}j}\right|^2 + \frac{\Gamma^2}{(2k_0R)^2}\left[\sum_j -|\langle\sigma_j^-\rangle|^2 + \frac{1+\langle\sigma_j^z\rangle}{2}\right]. \quad (5.9)$$

Figure 32 shows that if one focuses solely on the coherent part of the intensity yields similar patterns to those obtained from the `Scalar` Model. This suggests that the population, represented by the incoherent part, plays a crucial role in shaping the scattering characteristics observed in the system, and is detrimental to the observation of Anderson localization of light.

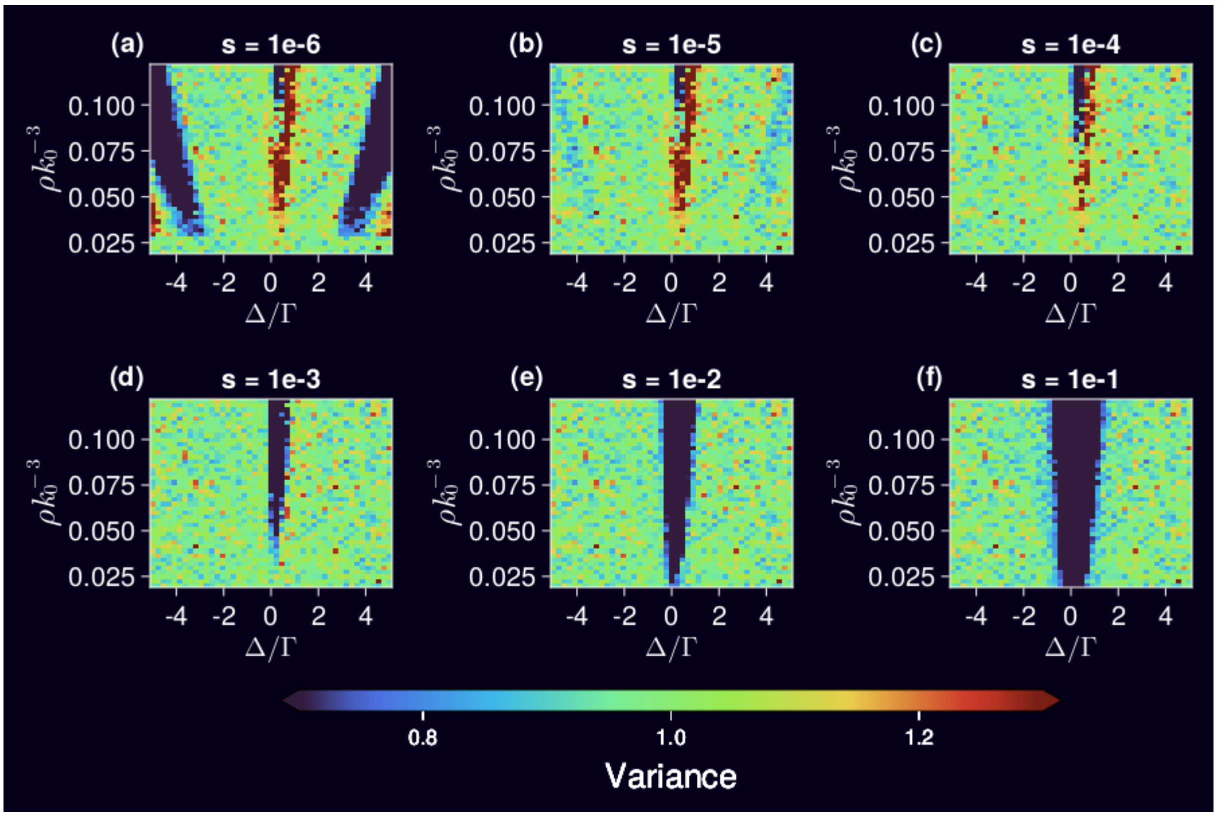


Figure 30 – Map of the intensity variance, as a function of the atomic density and laser detuning. We used a cylinder with a fixed radius ($k_0R = 3\lambda_0$) and height ($k_0H = 6\lambda_0$). The number of atoms and realizations varied depending on the density. For smaller densities, we used 32 realizations and adjusted them to 16 for larger atomic clouds. The laser waist was fixed at $w_0 = 0.5k_0R$.

Source: By the author.

5.6 Coherent and Incoherent Powers

The saturation parameter used, up to 0.1, suggests that the incoherent component should be rather weak, at least in the case of independent-atom scattering. Let us now study more quantitatively the effect of the excited population and of incoherent scattering on the intensity statistics, from the independent scatterer prediction.

We aim to explore the incoherent part verifying if it follows at least the expectations of the single-atom theory. Steck[94] pointed out how the photon scattering rate R_{sc}, which is the radiated power divided by the photon energy $\hbar\omega$, can be split into coherent and incoherent parts, namely,

$$R_{sc}^{\text{coh}} = \frac{\Gamma}{2}\frac{s}{(1+s)^2}, \tag{5.10}$$

$$R_{sc}^{\text{inc}} = \frac{\Gamma}{2}\frac{s^2}{(1+s)^2}. \tag{5.11}$$

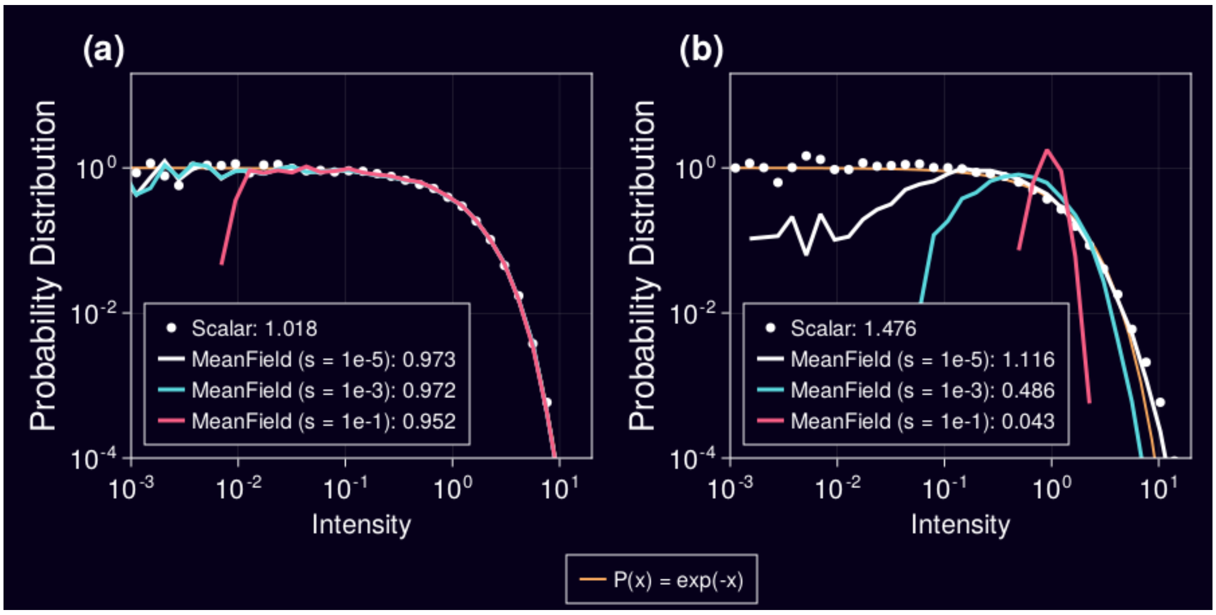


Figure 31 – Probability Distribution for low (a) and moderate (b) densities, comparing `Scalar` and `Meanfield` models and their respective variance is shown on the legend. Simulations had $N = 1000$ atoms, $\rho k_0^{-3} = 0.01$ low density, and $\rho k_0^{-3} = 0.1$ for moderate density, over 320 realizations of Cylinders of fixed radius $k_0 R = 4.5\pi$; the Gaussian beam had $w_0 = R/2$ and $\Delta\Gamma^{-1} = 0.3$.

Source: By the author.

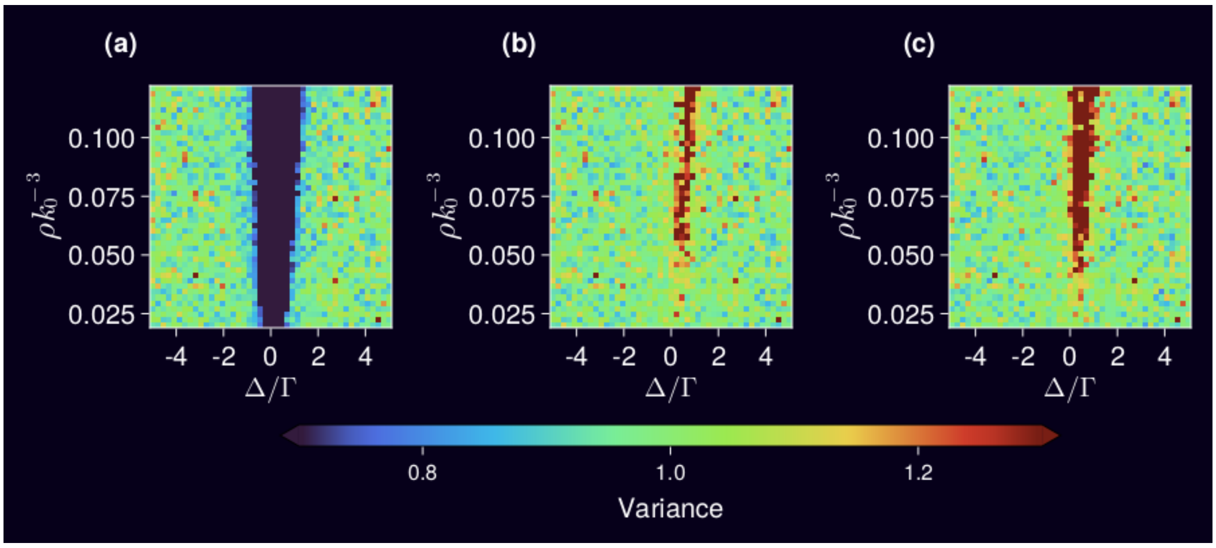


Figure 32 – (a) Previous results as a reference, using the `MeanField` model with a saturation parameter value of $s = 0.1$. (b) The region of small variances vanishes when considering only the coherent part of the intensity. (c) The scattering behavior produced by I_{coh} qualitatively reproduces the results obtained from the `Scalar` Model.

Source: By the author.

Our goal is to verify if these relations still hold for `MeanField`, or if the interactions lead to very different weights of these respective scattering contributions. In practice, one needs to find the coherent and incoherent powers, P^{coh} and P^{inc}, and up for some constant, we expect Equation 5.10 and Equation 5.11 to have the format

$$P_{sc}^{\text{coh}} \times \frac{2}{\Gamma}\frac{(1+s)^2}{s} = 1, \tag{5.12}$$

$$P_{sc}^{\text{inc}} \times \frac{2}{\Gamma}\frac{(1+s)^2}{s^2} = 1. \tag{5.13}$$

The validity of Equation 5.12 and Equation 5.13 are limited to a single atom. In systems with N independent atoms, we expect all atoms to emit the same power, requiring a normalization by $1/N$ — this is true only in optically dilute clouds, where the interaction between atoms is negligible. Furthermore, each atom may represent a different saturation due to their positions (the local intensity changes in space), requiring the calculation of an average saturation parameter $\langle s_j \rangle$ across all atoms. In summary, as a quantifier of the saturation in the interacting cloud, we aim to verify how much the following quantities, valid for independent scatterers, deviate from the unity

$$\frac{P_{sc}^{\text{coh}}}{0.5\Gamma N \langle s_j/(1+s_j)^2 \rangle} \overset{?}{=} 1, \tag{5.14}$$

$$\frac{P_{sc}^{\text{inc}}}{0.5\Gamma N \langle s_j^2/(1+s_j)^2 \rangle} \overset{?}{=} 1 \tag{5.15}$$

The saturation in `CoupledDipoles` incorporates the local Rabi frequency of the laser $\Omega(\mathbf{r}_j)$ within it. When considering a Gaussian laser, and given s_0 as the local saturation parameter at resonance, the expression for the saturation at different detunings is as follows:

$$\begin{aligned} s(\Delta) &= \frac{2\Omega(\vec{r})^2/\Gamma^2}{1+(2\Delta/\Gamma)^2} \\ &= \left(\Gamma\sqrt{\frac{s_0}{2}}\frac{e^{ikz}e^{-\frac{x^2+y^2}{w_0^2}\frac{1}{1+2iz/kw_0^2}}}{(1+2iz/kw_0^2)}\right)^2 \frac{2/\Gamma^2}{1+(2\Delta/\Gamma)^2} \\ &= \frac{s_0}{1+(2\Delta/\Gamma)^2}\left(\frac{e^{ikz}e^{-\frac{x^2+y^2}{w_0^2}\frac{1}{1+2iz/kw_0^2}}}{(1+2iz/kw_0^2)}.\right)^2. \end{aligned} \tag{5.16}$$

To evaluate the Coherent and Incoherent Powers, one integrates the intensity Equation 5.9 overall space

$$\text{Power} = \int I(\hat{\mathbf{r}})d\hat{\mathbf{r}} = \int_0^{\pi}\int_0^{2\pi} I(\hat{\mathbf{r}})r^2\sin\theta dr d\theta d\phi. \tag{5.17}$$

The coherent part of the local radiated power has been computed by Araújo, Guerin and Kaiser,[95] and with our formulas and definitions, there are only two modifications to adapt it to the `MeanField`. The first is to include multiplicative constants, and the second to decrease the number of operation in half

$$
\begin{aligned}
P_{coh} &= \frac{\Gamma^2}{(2k_0R)^2} 4\pi \sum_{j,m} \frac{\sin(k_0|\mathbf{r}_j - \mathbf{r}_m|)}{k_0|\mathbf{r}_j - \mathbf{r}_m|} \langle \sigma_j^- \rangle \langle \sigma_m^+ \rangle \\
&= \frac{4\pi\Gamma^2}{(2k_0R)^2} \left[2\Re \left(\sum_{j,m>j} \frac{\sin(|\mathbf{r}_j - \mathbf{r}_m|)}{|\mathbf{r}_j - \mathbf{r}_m|} \langle \sigma_j^- \rangle \langle \sigma_m^+ \rangle \right) + \sum_j |\langle \sigma_j^- \rangle|^2 \right] .
\end{aligned}
\tag{5.18}
$$

The incoherent part does not have any position dependence, and thus, no angular dependence and the integral simplifies as

$$
\begin{aligned}
P_{inc} &= \int_0^\pi \int_0^{2\pi} \frac{\Gamma^2}{(2k_0R)^2} \left[\sum_j -|\langle \sigma_j^- \rangle|^2 + \frac{1 + \langle \sigma_j^z \rangle}{2} \right] r^2 \sin\theta dr d\theta d\phi \\
&= \frac{\Gamma^2}{(2k_0R)^2} \left[\sum_j -|\langle \sigma_j^- \rangle|^2 + \frac{1 + \langle \sigma_j^z \rangle}{2} \right] \int_0^\pi \int_0^{2\pi} r^2 \sin\theta dr d\theta d\phi \\
&= \frac{\Gamma^2}{(2k_0R)^2} 4\pi \left[\sum_j -|\langle \sigma_j^- \rangle|^2 + \frac{1 + \langle \sigma_j^z \rangle}{2} \right] .
\end{aligned}
\tag{5.19}
$$

For a sanity check, we plot Equation 5.12, Equation 5.13 and their ratios at Figure 33. For the case of $N = 1$, all detunings exhibit the expected behavior. Still, for $N = 500$, a peak appears, originating from the interaction between the atoms.

Figure 34 summarizes how the coherent and incoherent parts behave for the same data from Figure 30. The coherent power is influenced by the term $\sin(r_{jm})/r_{jm}$, which approaches unity only for small densities. We found that such a condition is met for densities $\rho k_0^{-3} < 10^{-4}$, which is not within the density range of our interest. Typically, we work in the regime $\rho k_0^{-3} \in [10^{-2}, 10^{-1}]$, therefore, our coherent results will never be close to unity.

We acknowledge that the results obtained from the simulations are unstable and sensitive to the exact value of $\langle \sigma^z \rangle$ used — it can even produce nonphysical negative results (which are not shown on Figure 34), or most commonly, exponential large values. This is a consequence of the `MeanField` approach, which introduces a nonlinearity in the equations (a consequence of factorizing quantum correlations), potentially leading to non physical states, and often to numerical instabilities. While our fastest results utilized the `NLSolver.jl`, to achieve more reliable outcomes, the steady states were computed by the time evolution of the `MeanField` ODEs. Setting higher precision is an option, but it is crucial for future works to find a more stable numerical method for the steady state. The incoherent part of the saturation $s_0 = 10^{-1}$ is the most reliable dataset, and it has similar results to variances maps from Figure 30. Only in the region $\Delta\Gamma^{-1} \in [0, 2]$ do localization-related effects become noticeable, and these are stronger for higher densities.

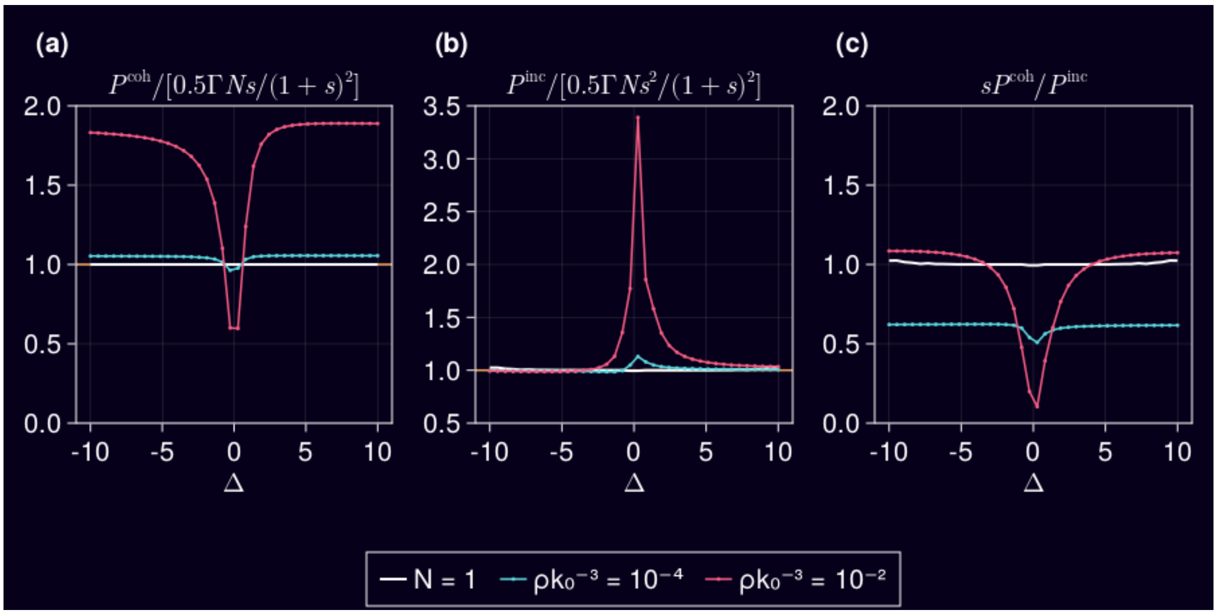


Figure 33 – The curves represent the scattering behavior of a single atom (shown in white) and a cloud of $N = 500$ particles in a cylindrical shape at two different densities. The system was pumped by a laser with a Gaussian Beam profile, where the saturation parameter was $s_0 = 10^{-3}$ and the laser waist was $w_0 = 0.8R$.

Source: By the author.

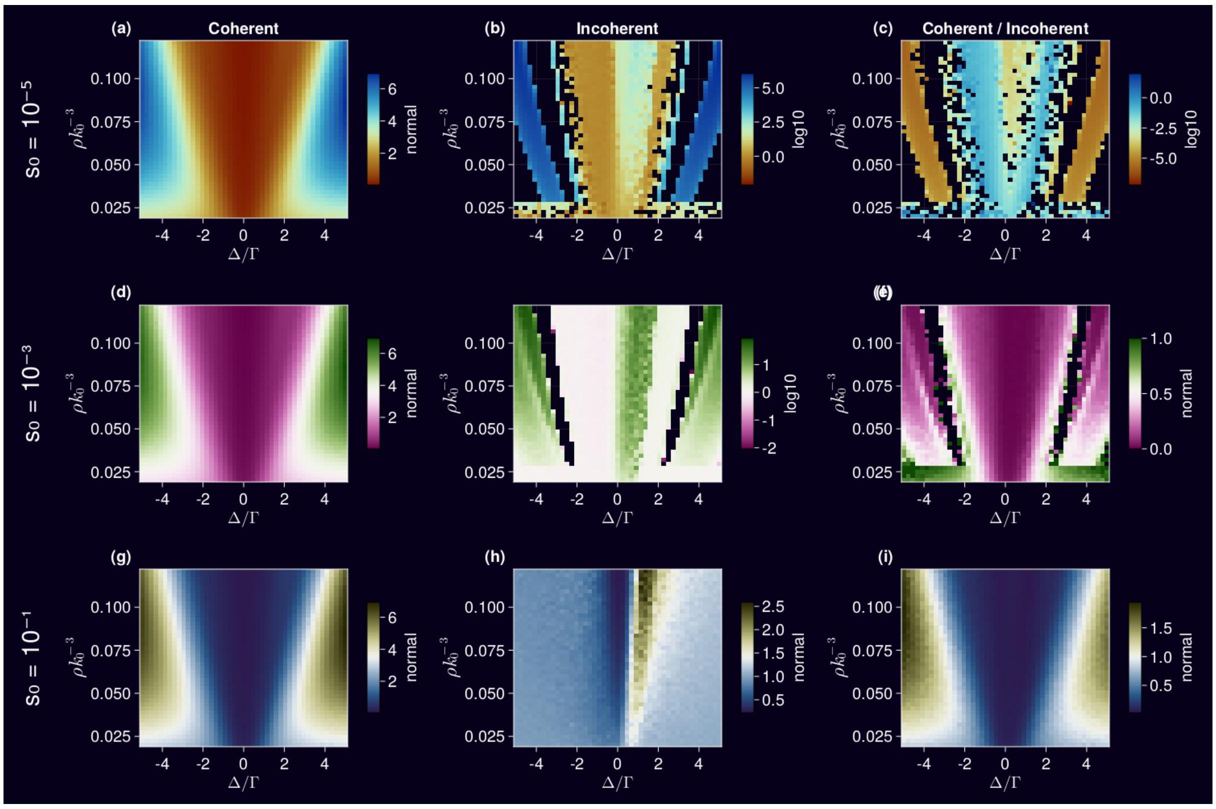


Figure 34 – When using the same data as to compute the variances, the coherent part remains stable across different saturation levels. Nevertheless, as the saturation decreases, numerical precision issues in computing the steady-state have a significant impact. The missing points on the figures (areas in black) correspond to the neglected negative values, as explained in the main text.

Source: By the author.

6 MISCELLANEOUS CODE APPLICATIONS

This chapter is devoted to showing some realistic codes. Keep in mind that the exact syntax of the examples may change in future versions of the package, but the main ideas should stay the same.

6.1 Comparing Models

If you want to explore a physical regime beyond the `MeanField` and the number of atoms is not large, using the `PairCorrelation` model is a great option among the physical models. Our aim is to show how to replicate results akin to those presented by Cipris *et al.*[96] Figure 2(a), which illustrates the disparities between different physical descriptions, and showcase a scenario where both `Scalar` and `MeanField` models were not as reliable as the `PairCorrelation` model to match decay rates experimental data.

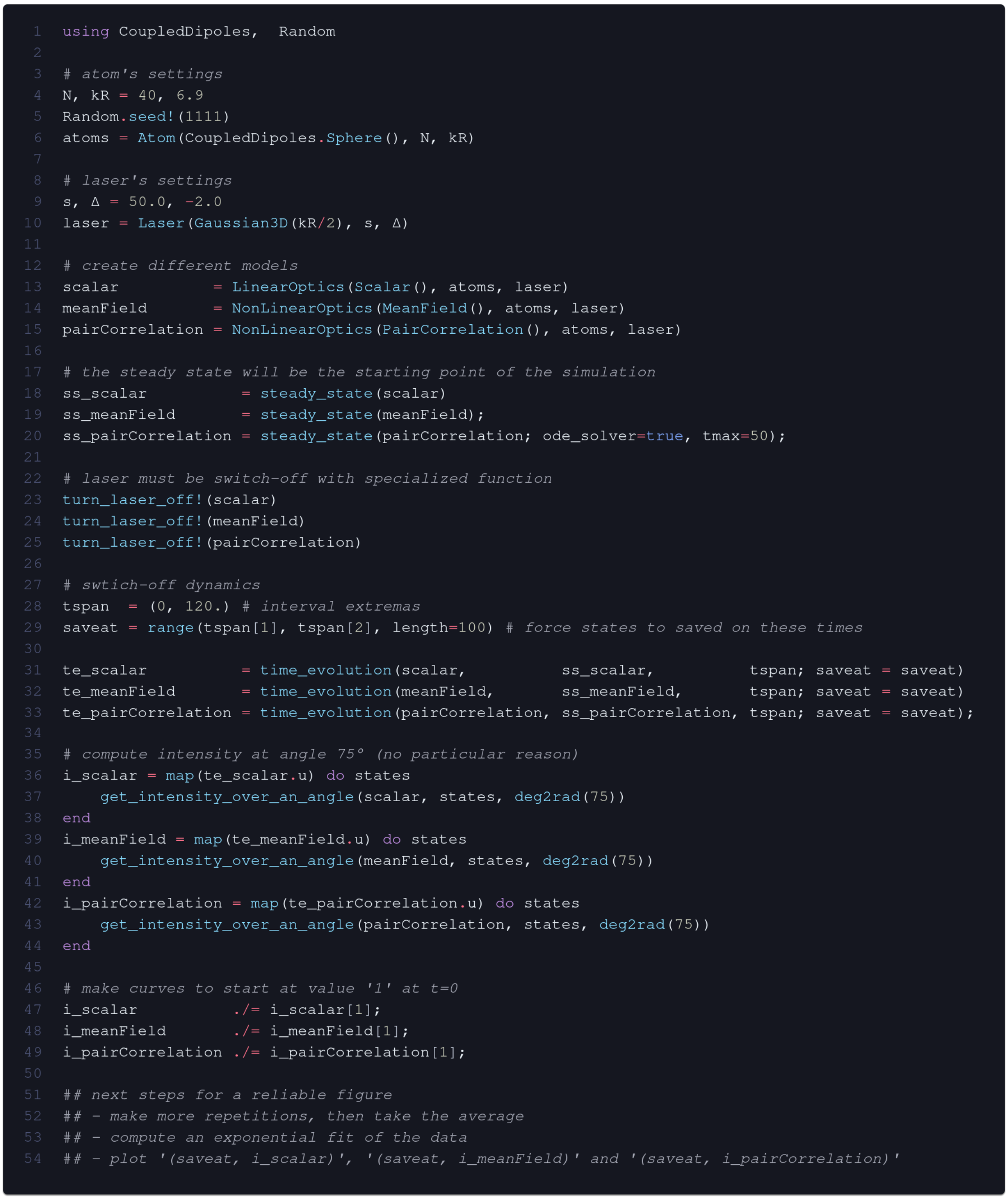
This code snippet at Figure 35 performs a simulation involving a fixed set of atoms and lasers but with different physical models (lines 13-15). The steady state of each model is calculated at lines 18-20 because they will be used as the starting point for the simulation of switch-off dynamics. We added extra parameters to line 20 to ensure accurate results for the `PairCorrelation` model since it is not stable for every atomic configuration and needs careful adjustments to be useful.

Then, the specialized function `turn_laser_off!` is used to switch off the lasers at lines 23-25. This is the only function so far that alters any of our data types. As a result, if users want a simulation with different parameters, for example, different detuning, a new data type instance has to be created.

Next, in lines 31-33, the code simulates the switch-off dynamics over a specified time interval and the final piece of the code computes and normalizes intensity at the chosen angle of 75°. The code is an easy example that merges recurring commands, but it cannot generate plots or compute averages from various outcomes. When all the extra changes are made, the intensity curves should look like the ones in Figure 36. We used this type of figure as the backbone of our Cipris *et al.*[96] publication by analyzing it with an exponential fitting to extract characteristic time lengths, rather than using the raw intensity data.

6.2 Random Phases

Subradiant modes, even if not exponentially localized, find applications as potential quantum memories in atomic systems. However, it is not entirely clear whether subradiance, which is a collective and coherent effect, is the dominant factor, or if another



```julia
using CoupledDipoles,  Random

# atom's settings
N, kR = 40, 6.9
Random.seed!(1111)
atoms = Atom(CoupledDipoles.Sphere(), N, kR)

# laser's settings
s, Δ = 50.0, -2.0
laser = Laser(Gaussian3D(kR/2), s, Δ)

# create different models
scalar          = LinearOptics(Scalar(), atoms, laser)
meanField       = NonLinearOptics(MeanField(), atoms, laser)
pairCorrelation = NonLinearOptics(PairCorrelation(), atoms, laser)

# the steady state will be the starting point of the simulation
ss_scalar          = steady_state(scalar)
ss_meanField       = steady_state(meanField);
ss_pairCorrelation = steady_state(pairCorrelation; ode_solver=true, tmax=50);

# laser must be switch-off with specialized function
turn_laser_off!(scalar)
turn_laser_off!(meanField)
turn_laser_off!(pairCorrelation)

# swtich-off dynamics
tspan  = (0, 120.) # interval extremas
saveat = range(tspan[1], tspan[2], length=100) # force states to saved on these times

te_scalar          = time_evolution(scalar,          ss_scalar,          tspan; saveat = saveat)
te_meanField       = time_evolution(meanField,       ss_meanField,       tspan; saveat = saveat)
te_pairCorrelation = time_evolution(pairCorrelation, ss_pairCorrelation, tspan; saveat = saveat);

# compute intensity at angle 75° (no particular reason)
i_scalar = map(te_scalar.u) do states
    get_intensity_over_an_angle(scalar, states, deg2rad(75))
end
i_meanField = map(te_meanField.u) do states
    get_intensity_over_an_angle(meanField, states, deg2rad(75))
end
i_pairCorrelation = map(te_pairCorrelation.u) do states
    get_intensity_over_an_angle(pairCorrelation, states, deg2rad(75))
end

# make curves to start at value '1' at t=0
i_scalar          ./= i_scalar[1];
i_meanField       ./= i_meanField[1];
i_pairCorrelation ./= i_pairCorrelation[1];

## next steps for a reliable figure
## - make more repetitions, then take the average
## - compute an exponential fit of the data
## - plot '(saveat, i_scalar)', '(saveat, i_meanField)' and '(saveat, i_pairCorrelation)'
```

Figure 35 – Code example to create and compare different physical models. Once different models are created, their analysis follows the same commands.

Source: By the author.

effect, such as radiation trapping with incoherent physics, justifies the long lifetime. As coherent effects depend on phases, a pulse of random values is added to the interaction matrix to distinguish them from incoherent effects such as radiation trapping. Changing the phase does not affect the atomic dynamics and observable if incoherent effects are the

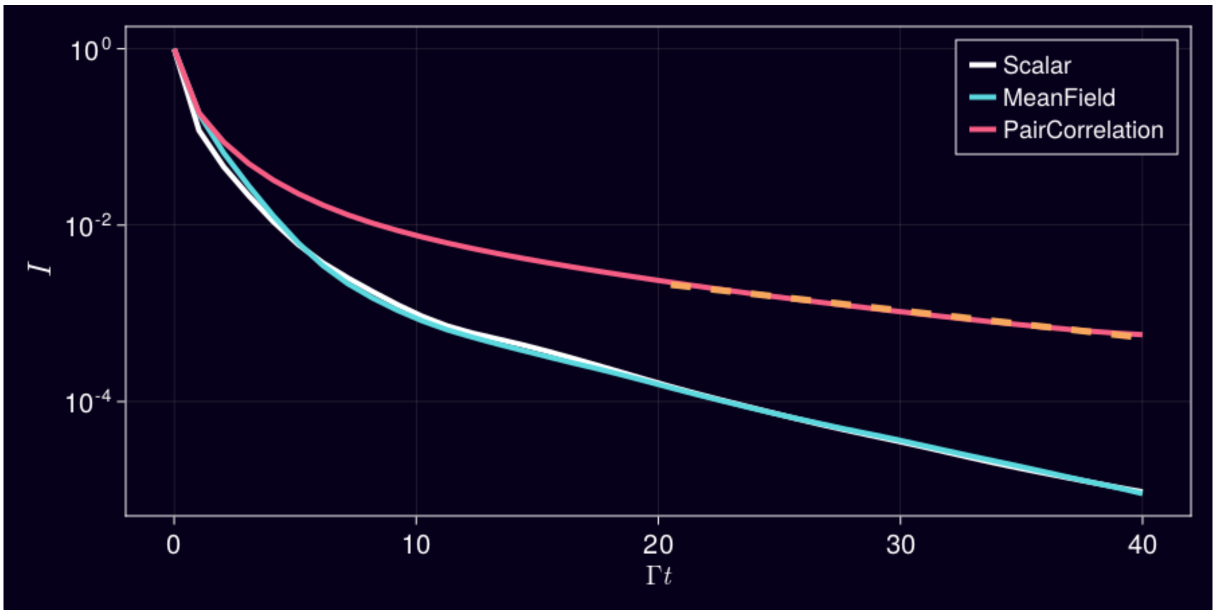


Figure 36 – `Scalar` and `MeanField` models have similar intensity over time, while `PairCorrelation` model could bring new physics.

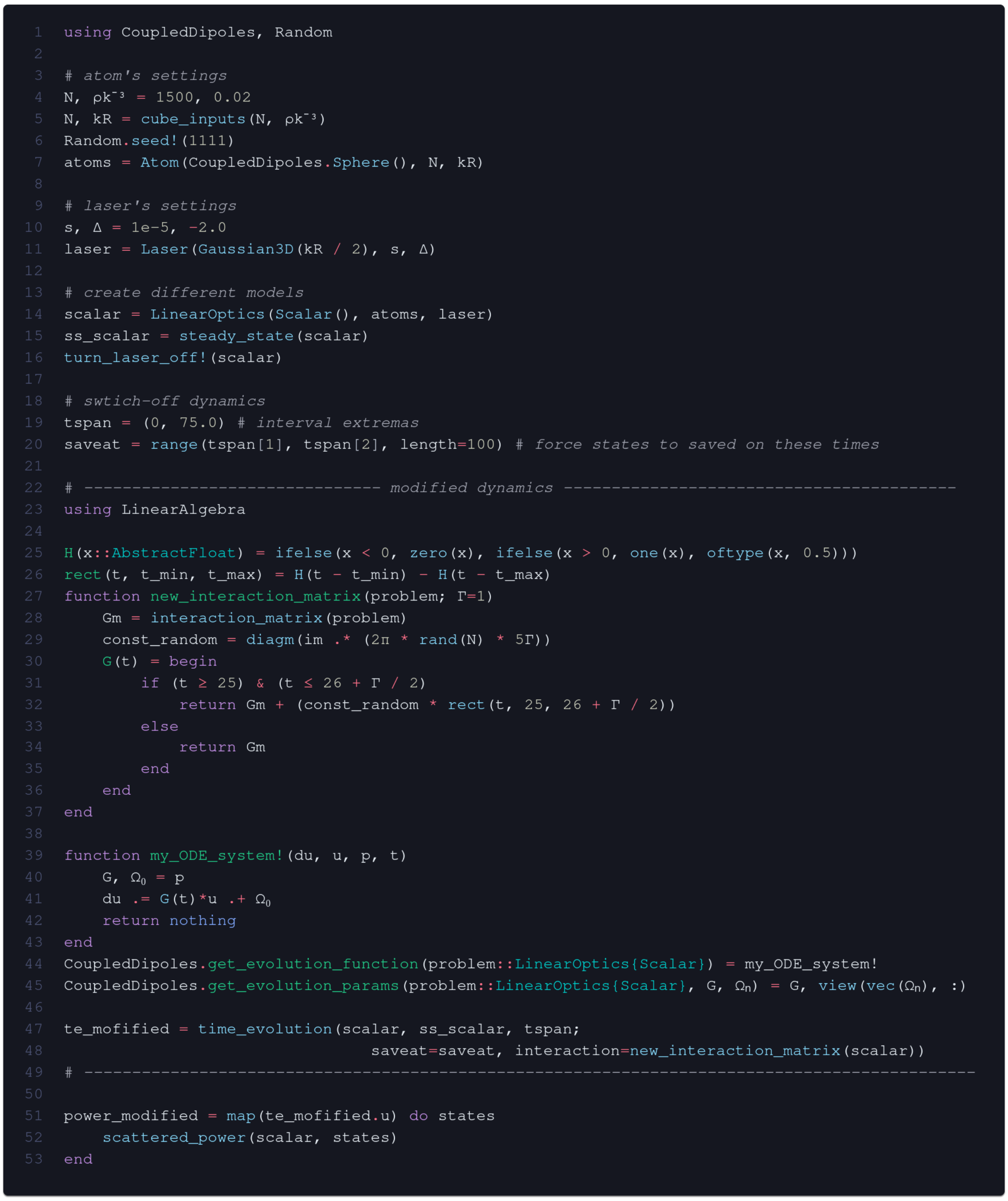
Source: By the author.

main factor. In what follows, we show an implementation of a random phase change (that is, different for each atom) relying on the introduction of an external magnetic field.

The code in Figure 37 is a basic example that builds on the principles outlined in Figure 35. Specifically, it encompasses the switch-off dynamics started from the steady state. Notably, the innovative aspect lies within lines 22-43. Random values are added only to the diagonal of the original interaction matrix between the time interval $\Gamma t \in [25, 26.5]$ by the function `new_interaction_matrix`. The way we do this is by using a Rectangular Function (`rect`) that has been built using two Heaviside step functions (`H`). The meaning of this extra term would represent the influence of an external magnetic field applied over the atomic system for a defined amount of time.

In addition to the interaction matrix, users must also specify a chain of auxiliary functions within the package. For example, the ODE system needs to be rewritten to consider the matrix's time dependence, in lines 39-43. Next, lines 44 and 45 set up internal functions for the `time_evolution` function to use our custom functions. Users have to inspect the source code, and there is no easy solution to select the auxiliary functions that require further changes. Finally, in line 48, we pass the argument `interaction` with the result of `new_interaction_matrix(scalar)`.

The outcome of the code (with average over repetitions) is plotted in Figure 38. A peak is visible in the scattered power and an exact interpretation of the results demands further analysis, studying the energy of the system for example. Researchers can use



```julia
using CoupledDipoles, Random

# atom's settings
N, ρk⁻³ = 1500, 0.02
N, kR = cube_inputs(N, ρk⁻³)
Random.seed!(1111)
atoms = Atom(CoupledDipoles.Sphere(), N, kR)

# laser's settings
s, Δ = 1e-5, -2.0
laser = Laser(Gaussian3D(kR / 2), s, Δ)

# create different models
scalar = LinearOptics(Scalar(), atoms, laser)
ss_scalar = steady_state(scalar)
turn_laser_off!(scalar)

# swtich-off dynamics
tspan = (0, 75.0) # interval extremas
saveat = range(tspan[1], tspan[2], length=100) # force states to saved on these times

# ------------------------------ modified dynamics ----------------------------------------
using LinearAlgebra

H(x::AbstractFloat) = ifelse(x < 0, zero(x), ifelse(x > 0, one(x), oftype(x, 0.5)))
rect(t, t_min, t_max) = H(t - t_min) - H(t - t_max)
function new_interaction_matrix(problem; Γ=1)
    Gm = interaction_matrix(problem)
    const_random = diagm(im .* (2π * rand(N) * 5Γ))
    G(t) = begin
        if (t ≥ 25) & (t ≤ 26 + Γ / 2)
            return Gm + (const_random * rect(t, 25, 26 + Γ / 2))
        else
            return Gm
        end
    end
end

function my_ODE_system!(du, u, p, t)
    G, Ω₀ = p
    du .= G(t)*u .+ Ω₀
    return nothing
end
CoupledDipoles.get_evolution_function(problem::LinearOptics{Scalar}) = my_ODE_system!
CoupledDipoles.get_evolution_params(problem::LinearOptics{Scalar}, G, Ωₙ) = G, view(vec(Ωₙ), :)

te_mofified = time_evolution(scalar, ss_scalar, tspan;
                                saveat=saveat, interaction=new_interaction_matrix(scalar))
# -------------------------------------------------------------------------------------------

power_modified = map(te_mofified.u) do states
    scattered_power(scalar, states)
end
```

Figure 37 – Minimal Working Example of a code where the internals of the package had to be altered to attend to the user's needs.

Source: By the author.

other methods in the package for further analysis with no extra restrictions since the only modification made from the defined work was the interaction matrix.

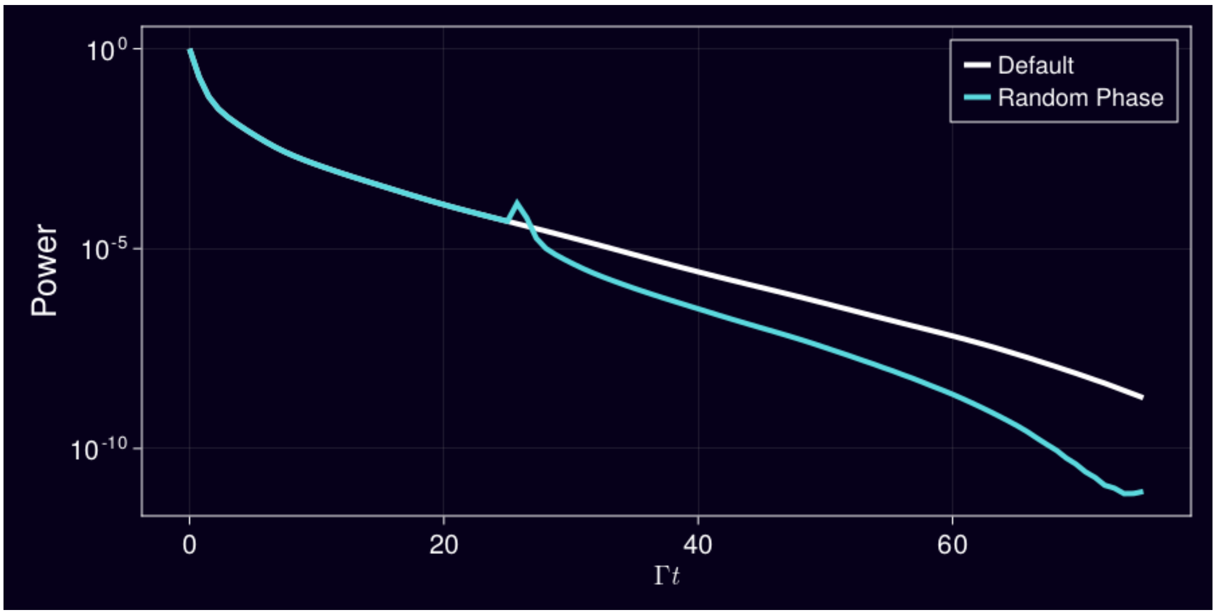


Figure 38 – Scattered power over time using default settings compared with an interaction matrix time dependent.

Source: By the author.

6.3 Parallel Load Balancing

Parallel simulations usually require averaging over independent realizations. As already mentioned in the Optimization Chapter, the benefit of parallelism appears with `Distributed.jl`. This aspect has been avoided in previous discussions, but for instance, the Variance map at Figure 30 is a real-world scenario where parallelism plays an important role.

In a simulation with changes in densities, for a fixed system size, the number of atoms varies accordingly to achieve a certain density. Therefore, parameters with high density (stronger interactions) take more time to be computed. As an extreme solution, a simplistic method would be to distribute different regions evenly among different nodes. As a result, nodes with a smaller number of atoms would finish first, and be idle awaiting the other nodes. The other extreme of task scheduling is to let each node act as a *deamon* or micro-service, awaiting a request and executing on demand. Operational nodes are assured, but data communication demand is high, especially when transferring large matrices. The solution is then to shuffle the data prior to the task division into simple blocks.

The fastest way to achieve this distribution is with the `DistributedArrays.jl` package. The code in Figure 40 is not complete, but it highlights the cumbersome part of identifying local indices for code operation. Note that the `DistributedArrays.jl` package does not call the garbage collector to clean unsed memory, so the User has to

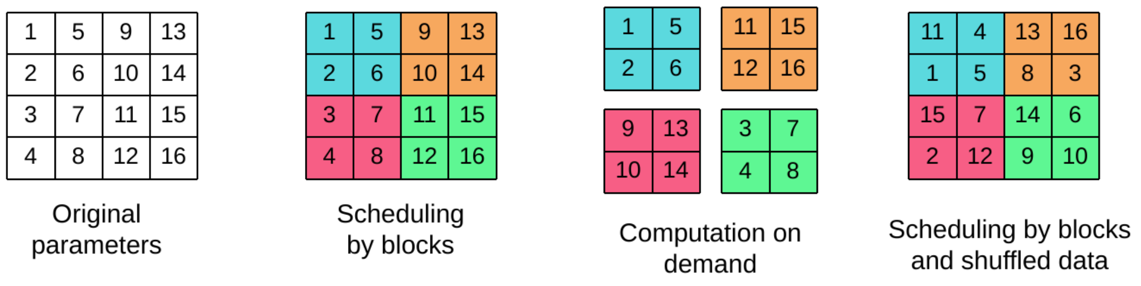
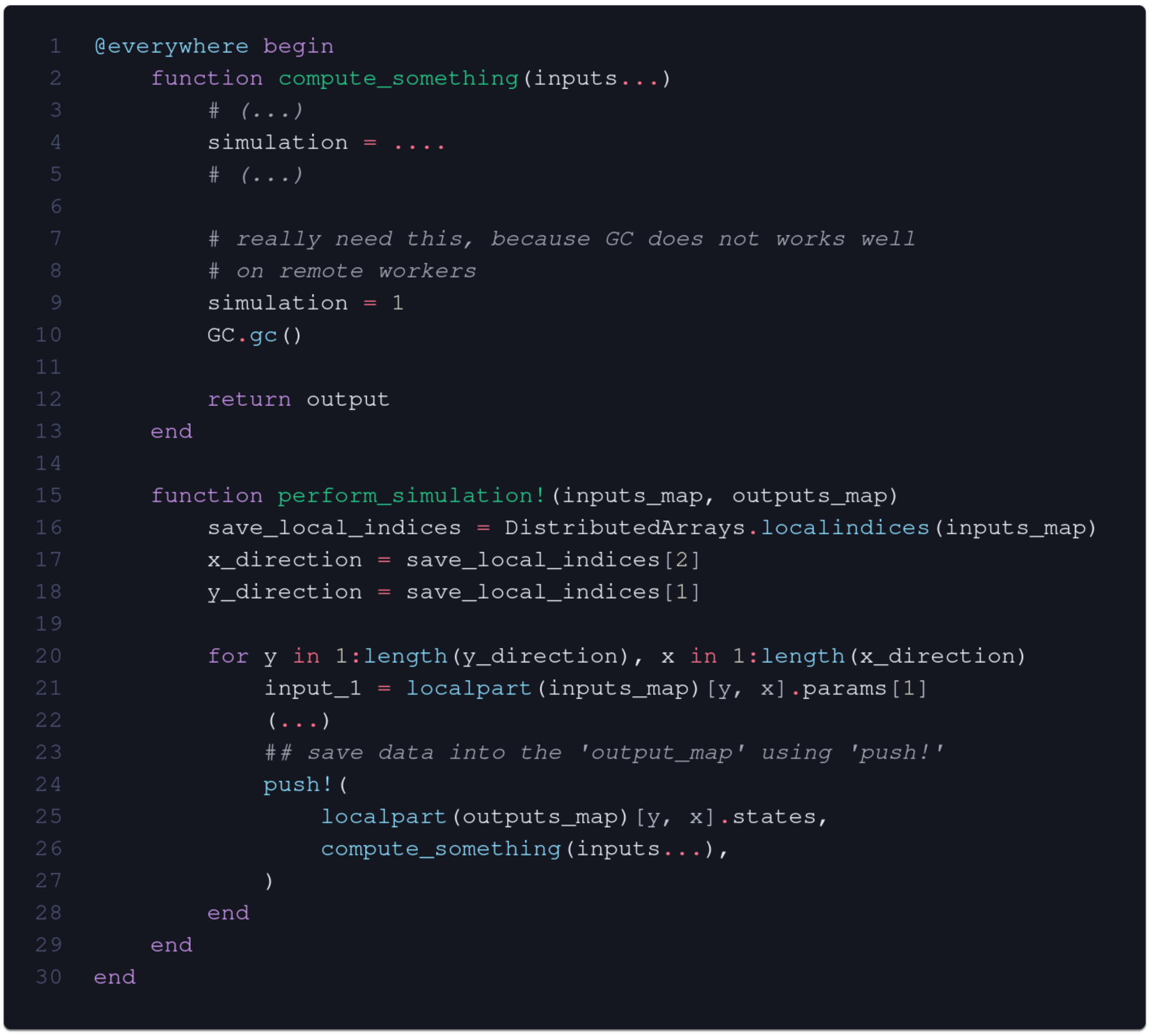


Figure 39 – Task division, when each element of computation takes an uneven time of execution, is a challenge to reduce idle processing. A simple division by blocks, or on-demand computing, is easier to program but has its own drawbacks. Our approach is an intermediary, where data is shuffled randomly at workers, and on average, all workers have an equal load.

Source: By the author.

call it manually with the command `GC.gc(true)`.



```julia
@everywhere begin
    function compute_something(inputs...)
        # (...)
        simulation = ....
        # (...)

        # really need this, because GC does not works well
        # on remote workers
        simulation = 1
        GC.gc()

        return output
    end

    function perform_simulation!(inputs_map, outputs_map)
        save_local_indices = DistributedArrays.localindices(inputs_map)
        x_direction = save_local_indices[2]
        y_direction = save_local_indices[1]

        for y in 1:length(y_direction), x in 1:length(x_direction)
            input_1 = localpart(inputs_map)[y, x].params[1]
            (...)
            ## save data into the 'output_map' using 'push!'
            push!(
                localpart(outputs_map)[y, x].states,
                compute_something(inputs...),
            )
        end
    end
end
```

Figure 40 – It is recommended to have two distributed matrices - one for parameters and one for results. Therefore, when saving the data, reduces data communication.

Source: By the author.

7 CONCLUSION

The investigation of cold atoms necessitates a meticulous and efficient numerical methodology. Our journey began with the goal of understanding numerical intricacies that affect simulating dipole equations in 3D systems with different models. This dedicated effort has culminated in the development of a specialized Julia package, specifically tailored to address the unique computational challenges posed by cold atom simulations.

The core of our endeavor revolved around achieving substantial performance enhancements. A big accomplishment was possible by picking the best computational tools and algorithms. Equally important was understanding the algorithmic intricacies that govern fundamental operations. Semi-analytical techniques helped us make better choices for efficient computation and precise error estimation.

As a testament to the versatility of our developed package, we apply our tools to the study of Anderson Localization, exploring how the statistics of the scattered light may reveal experimental signatures of the localization transition. While the previous results were based on the single-excitation approach, we investigated these statistics using our numerical tools, using the non-linear set of equations (Mean-Field Equations) which takes into account the excited population of the atoms and thus describes the moderate-drive regime. This mean-field approach has allowed us to demonstrate that the presence of multiple excitations in the system will not prevent the localization phenomenon, provided that one monitors the elastically scattered light.

However, it is essential to acknowledge certain inherent limitations within our research. Our focus remained squarely on 3D Systems, with other dimensions of physical phenomena lying beyond our scope. Also, we did not investigate using mixed precision (`Float32`) for quicker simulations, but one may want to consider this aspect in the future. Last, comprehensive documentation is an ongoing process that needs the scientific community's feedback to improve it.

Currently, various numerical results are scattered across different research groups, with the technical implementation often not provided, which can be considered a weakness from a scientific point of view. The ultimate goal of this package is to stimulate more unified behavior in the community, providing the first step of an adaptable package. By providing a centralized resource, we aim to facilitate collaboration across groups and stimulate new works and the exploration of new regimes in the field.

REFERENCES

1 MILONNI, P. W. Why spontaneous emission? **American Journal of Physics**, v. 52, n. 4, p. 340343, 1984.

2 GRYNBERG, G.; ASPECT, A.; FABRE, C. **Introduction to quantum optics**: from the semi-classical approach to quantized light. Cambridge: Cambridge University Press, 2010. ISBN 0521551129.

3 KLEPPNER, D. Inhibited spontaneous emission. **Physical Review Letters**, v. 47, n. 4, p. 233–236, 1981.

4 HULET, R. G.; HILFER, E. S.; KLEPPNER, D. Inhibited spontaneous emission by a Rydberg atom. **Physical Review Letters**, v. 55, n. 20, p. 2137–2140, 1985.

5 PURCELL, E. M. Spontaneous emission probabilities at radio frequencies. **Physical Review**, v. 69, n. 9-10, p. 681, 1946.

6 DICKE, R. H. Coherence in spontaneous radiation processes. **Physical Review**, v. 93, n. 1, p. 99–110, 1954.

7 LEHMBERG, R. H. Radiation from an n-atom system. I. general formalism. **Physical Review A**, v. 2, n. 3, p. 883–888, 1970.

8 SUTHERLAND, R. T.; ROBICHEAUX, F. Coherent forward broadening in cold atom clouds. **Physical Review A**, v. 93, n. 2, p. 023407, 2016.

9 ROBICHEAUX, F.; SUTHERLAND, R. T. Photon scattering from a cold, gaussian atom cloud. **Physical Review A**, v. 101, n. 1, p. 013805, 2020.

10 ROBICHEAUX, F. Theoretical study of early-time superradiance for atom clouds and arrays. **Physical Review A**, v. 104, n. 6, p. 063706, 2021.

11 FERIOLI, G. *et al.* Laser-driven superradiant ensembles of two-level atoms near Dicke regime. **Physical Review Letters**, v. 127, n. 24, p. 243602, 2021.

12 SGRIGNUOLI, F. *et al.* Localization of scattering resonances in aperiodic Vogel spirals. **Physical Review B**, v. 99, n. 10, p. 104202, 2019.

13 SGRIGNUOLI, F.; TORQUATO, S.; NEGRO, L. D. Subdiffusive wave transport and weak localization transition in three-dimensional stealthy hyperuniform disordered systems. **Physical Review B**, v. 105, n. 6, p. 064204, 2022.

14 WEISS, P. *et al.* Subradiance and radiation trapping in cold atoms. **New Journal of Physics**, v. 20, n. 6, p. 063024, 2018.

15 COTTIER, F.; KAISER, R.; BACHELARD, R. Role of disorder in super- and subradiance of cold atomic clouds. **Physical Review A**, v. 98, n. 1, p. 013622, 2018.

16 CIPRIS, A. *et al.* van der Waals dephasing for dicke subradiance in cold atomic clouds. **Physical Review A**, v. 103, n. 3, p. 033714, 2021.

17 SKIPETROV, S.; SOKOLOV, I. Absence of anderson localization of light in a random ensemble of point scatterers. **Physical Review Letters**, v. 112, n. 2, p. 023905, 2014.

18 SKIPETROV, S. Localization transition for light scattering by cold atoms in an external magnetic field. **Physical Review Letters**, v. 121, n. 9, p. 093601, 2018.

19 AMERICAN PHYSICAL SOCIETY. **Information for Authors**. 2019. Available at: https://journals.aps.org/pra/authors. Accessible at: 16 June 2023.

20 ELSEVIER. **Guide for Authors**. 2023. Available at: https://www.elsevier.com/journals/journal-of-computational-physics/0021-9991/guide-for-authors. Accessible at: 15 June 2023.

21 JOURNAL OF OPEN SOURCE SOFTWARE. **JOSS Policies**. 2018. Available at: https://joss.readthedocs.io/en/latest/policies.html#data-sharing-policy. Accessible at: 15 June 2023.

22 JOHANSSON, J.; NATION, P.; NORI, F. QuTiP 2: a python framework for the dynamics of open quantum systems. **Computer Physics Communications**, v. 184, n. 4, p. 1234–1240, 2013.

23 KRÄMER, S. *et al.* Quantumoptics.jl: a julia framework for simulating open quantum systems. **Computer Physics Communications**, v. 227, p. 109–116, 2018.

24 KRÄMER, S.; RITSCH, H. Generalized mean-field approach to simulate the dynamics of large open spin ensembles with long range interactions. **The European Physical Journal D**, v. 69, n. 12, p. 282, 2015.

25 FICEK, Z. M. R. W. **Quantum optics for beginners**. New York: Pan Stanford Pub, 2014.

26 BIENAIMÉ, T. *et al.* Cooperativity in light scattering by cold atoms. **Fortschritte der Physik**, v. 61, n. 2-3, p. 377–392, 2012.

27 NORAMBUENA, A.; TANCARA, D.; COTO, R. Coding closed and open quantum systems in matlab: applications in quantum optics and condensed matter. **European Journal of Physics**, v. 41, n. 4, p. 045404, 2020.

28 SVIDZINSKY, A. A.; CHANG, J.-T.; SCULLY, M. O. Cooperative spontaneous emission of n atoms: many-body eigenstates, the effect of virtual lamb shift processes, and analogy with radiation of n classical oscillators. **Physical Review A**, v. 81, n. 5, p. 053821, 2010.

29 WILLIAMSON, L. A.; RUOSTEKOSKI, J. Optical response of atom chains beyond the limit of low light intensity: the validity of the linear classical oscillator model. **Physical Review Research**, v. 2, n. 2, p. 023273, 2020.

30 CASTRO, L. B. de. **Localisation de la lumiere et effets cooperatifs dans des nuages d'atomes froids**. 2013. 232 p. Ph. D. Thesis (Physics) — Universite de Nice - Sophia Antipolis, Nice, 2013.

31 BIENAIME, T. *et al.* Atom and photon measurement in cooperative scattering by cold atoms. **Journal of Modern Optics**, v. 58, n. 21, p. 1942–1950, 2011.

32 ZENS, M. **Cumulant expansion approach to nonlinear dynamics of inhomogeneous ensembles in cavity QED**. 2016. 60 p. Ph. D. Thesis (Doctor) — Technische Universität Wien, Vienna, 2016.

33 KUBO, R. Generalized cumulant expansion method. **Journal of the Physical Society of Japan**, v. 17, n. 7, p. 1100–1120, 1962.

34 HANDWIKI. **Ursell function**. 2023. Available at: https://handwiki.org/wiki/Ursell_function. Accessible at: 28 June 2023.

35 SÁNCHEZ-BARQUILLA, M.; SILVA, R. E. F.; FEIST, J. Cumulant expansion for the treatment of light–matter interactions in arbitrary material structures. **The Journal of Chemical Physics**, v. 152, n. 3, p. 034108, 2020. DOI: 10.1063/1.5138937.

36 PLANKENSTEINER, D.; HOTTER, C.; RITSCH, H. Quantumcumulants.jl: a Julia framework for generalized mean-field equations in open quantum systems. **Quantum**, v. 6, p. 617, 2022. DOI: 10.22331/q-2022-01-04-617.

37 SANTO, T. S. do E. *et al.* Collective excitation dynamics of a cold atom cloud. **Physical Review A**, v. 101, n. 1, p. 013617, 2020.

38 MÁXIMO, C. E. *et al.* Cooperative spontaneous emission via a renormalization approach: classical versus semiclassical effects. **Physical Review A**, v. 101, n. 2, p. 023829, 2020.

39 RAMADHAN, A. *et al.* Oceananigans.jl: fast and friendly geophysical fluid dynamics on GPUs. **Journal of Open Source Software**, v. 5, n. 53, p. 2018, 2020.

40 ESCHLE, J. *et al.* Potential of the Julia programming language for high energy physics computing. **Computing and Software for Big Science**, v. 7, n. 1, 2023. DOI: 10.1007/s41781-023-00104-x.

41 DATSERIS, G. Dynamicalsystems.jl: a Julia software library for chaos and nonlinear dynamics. **Journal of Open Source Software**, v. 3, n. 23, p. 598, 2018. DOI: 10.21105/joss.00598.

42 INNES, M. Don't unroll adjoint: differentiating SSA-form programs. 2018. Available at: https://arxiv.org/pdf/1810.07951.pdf. Accessible at: 12 May 2023.

43 IEEE Std 754-2019. **IEEE standard for floating-point arithmetic**. p.1-84, July 2019. DOI: 10.1109/IEEESTD.2019.8766229.

44 LATTNER, C. **LLVM**: an infrastructure for multi-stage optimization. 2002. 67 p. Dissertation (Master) — Dept. Computer Science, University of Illinois at Urbana-Champaign, Urbana, Dec 2002.

45 BEZANSON, J. **Why is Julia fast?** 2015. Available at: https://www.youtube.com/watch?v=cjzcYM9YhwA. Accessible at: 5 Sept. 2023.

46 DIEGELMAN, J. **Modeling spacecraft separation dynamics in Julia**. 2021. Available at: https://www.youtube.com/watch?v=tQpqsmwlfY0. Accessible at: 5 Sept. 2023.

47 ELINUX.ORG. **Toolchains**. 2021. Available at: https://elinux.org/Toolchains. Accessible at: 4 Sept. 2023.

48 JULIA. **Types**. 2023. Available at: https://docs.julialang.org/en/v1/manual/types/. Accessible at: 1 Sept. 2023.

49 JULIA. **Constructors**. 2023. Available at: https://docs.julialang.org/en/v1/manual/constructors/. Accessible at: 01 Sept. 2023.

50 ADOMIAN, G. **Solving frontier problems of physics**: the decomposition method. Netherlands: Springer, 1994. 354 p.

51 JULIA. **Performance tips**. 2023. Available at: https://docs.julialang.org/en/v1/manual/performance-tips/. Accessible at: 04 Sept. 2023.

52 MCCOOL, M.; REINDERS, J.; ROBISON, A. **Structured parallel programming patterns for efficient computation**. San Francisco: Morgan Kaufmann, 2012. 432 p.

53 NOVOTNY, B. H. L. **Principles of nano-optics**. Cambridge: Cambridge University Press, 2006.

54 MATHEMATICS STACK EXCHANGE. **Distance between a point and a line in 3D**. Available at: https://math.stackexchange.com/questions/3757271/distance-between-a-point-and-a-line-in-3d. Accessible at: 18 Nov. 2022.

55 SCHMIDT, B. *et al.* **Parallel programming**: concepts and practice. Cambridge: Morgan Kaufmann, 2018. 404 p.

56 JULIASIMD. **LoopVectorization**. Available at: https://github.com/JuliaSIMD/LoopVectorization.jl. Accessible at: 13 Sept. 2023.

57 SAMOYLOVA, M. *et al.* Microscopic theory of photonic band gaps in optical lattices. **Optics Communications**, v. 312, p. 94–98, 2014. DOI: 10.1016/j.optcom.2013.09.016.

58 FOFANOV, Y. A. *et al.* Subradiance in dilute atomic ensembles: role of pairs and multiple scattering. **Physical Review A**, v. 104, p. 023705, 2021.

59 WEISS, P. *et al.* Superradiance as single scattering embedded in an effective medium. **Physical Review A**, v. 103, n. 2, p. 023702, 2021.

60 GENZ, A.; MALIK, A. Remarks on algorithm 006: an adaptive algorithm for numerical integration over an n-dimensional rectangular region. **Journal of Computational and Applied Mathematics**, v. 6, n. 4, p. 295–302, 1980.

61 JULIAMATH. **HCubature**. 2022. Available at: https://github.com/JuliaMath/HCubature.jl. Accessible at: 01 May 2022.

62 INTEL. **Intel® oneAPI Math Kernel Library Developer Reference for C**. Available at: https://www.intel.com/content/www/us/en/contentdetails/728197/intel-oneapi-math-kernel-library-developer-reference-for-c.html. Accessible at: 04 Nov. 2023.

63 RACKAUCKAS, C.; NIE, Q. Differentialequations.jl–a performant and feature-rich ecosystem for solving differential equations in julia. **Journal of Open Research Software**, v. 5, n. 1, 2017. DOI: 10.5334/jors.151.

64 ODE SOLVERS. **Differentialequations.jl**. Available at: https://diffeq.sciml.ai/latest/solvers/ode_solve/. Accessible at: 17 Nov. 2022.

65 CANALE, R. P.; CHAPRA, S. C. **Numerical methods for engineers**. 8th. ed. New York: McGraw-Hill Education, 2021.

66 RANOCHA, H. *et al.* Optimized Runge-Kutta methods with automatic step size control for compressible computational fluid dynamics. **Communications on Applied Mathematics and Computation**, v. 4, n. 4, p. 1191–1228, 2021.

67 NVIDIA DOCUMENTATION HUB. **NVIDIA CUDA**. Available at: https://docs.nvidia.com/cuda/doc/index.html. Accessible at: 17 Mar. 2023.

68 BESARD, T.; FOKET, C.; SUTTER, B. D. Effective extensible programming: unleashing Julia on GPUs. **IEEE Transactions on Parallel and Distributed Systems**, v. 30, n. 4, p. 827–841, 2019.

69 VAST INC. **vast.ai**. Available at: https://vast.ai. Accessible at: 17 Mar. 2023.

70 JULIANLSOLVERS. **NLsolve.jl**. Available at: https://github.com/JuliaNLSolvers/NLsolve.jl. Accessible at: 13 Sept. 2023.

71 KELLEY, C. T. **Solving nonlinear equations with iterative methods**: solvers and examples in Julia. Philadelphia: SIAM, 2022.

72 ANDERSON, P. W. Absence of diffusion in certain random lattices. **Physical Review**, v. 109, n. 5, p. 1492–1505, 1958.

73 RESTA, R.; SORELLA, S. Electron localization in the insulating state. **Physical Review Letters**, v. 82, n. 2, p. 370–373, 1999.

74 DEMUTH, A. *et al.* Quantum light transport in phase-separated Anderson localization fiber. **Communications Physics**, v. 5, n. 1, 2022. DOI: 10.1038/s42005-022-01036-5.

75 BILLY, J. *et al.* Direct observation of Anderson localization of matter waves in a controlled disorder. **Nature**, v. 453, n. 7197, p. 891–894, 2008.

76 WHITE, D. H. *et al.* Observation of two-dimensional Anderson localization of ultracold atoms. **Nature Communications**, v. 11, n. 1, 2020. DOI: 10.1038/s41467-020-18652-w.

77 KUTZ, J. N. **Data-driven modeling and scientific computation**: methods for complex systems and big data. New York: Oxford University Press, 2013. 657 p.

78 LANDAU, R. H. **Computational problems for physics**. Boca Raton: CRC Press, 2018.

79 LAHINI, Y. *et al.* Anderson localization and nonlinearity in one-dimensional disordered photonic lattices. **Physical Review Letters**, v. 100, n. 1, p. 013906, 2008.

80 TZORTZAKAKIS, A. F. *et al.* Transport and spectral features in non-hermitian open systems. **Physical Review Research**, v. 3, n. 1, p. 013208, 2021.

81 KUMAR, B. *et al.* Localized modes revealed in random lasers. **Optica**, v. 8, n. 8, p. 1033–1039, 2021.

82 EDWARDS, J.; THOULESS, D. Numerical studies of localization in disordered systems. **Journal of Physics C:** solid state physics, v. 5, n. 8, p. 807, 1972.

83 WEGNER, F. Inverse participation ratio in 2+ε dimensions. **Zeitschrift for Physik B:** condensed matter and quanta, v. 36, n. 3, p. 209–214, 1980.

84 CHANG, T. M.; BAUER, J. D.; SKINNER, J. L. Critical exponents for Anderson localization. **The Journal of Chemical Physics**, v. 93, n. 12, p. 8973–8982, 1990.

85 SEMEGHINI, G. *et al.* Measurement of the mobility edge for 3d Anderson localization. **Nature Physics**, v. 11, n. 7, p. 554–559, 2015.

86 ABRAHAMS, E. *et al.* Scaling theory of localization: absence of quantum diffusion in two dimensions. **Physical Review Letters**, v. 42, n. 10, p. 673–676, 1979.

87 JOHN, S. Electromagnetic absorption in a disordered medium near a photon mobility edge. **Physical Review Letters**, v. 53, n. 22, p. 2169–2172, 1984.

88 ANDERSON, P. W. The question of classical localization a theory of white paint? **Philosophical Magazine B**, v. 52, n. 3, p. 505–509, 1985.

89 WIERSMA, D. S. *et al.* Localization of light in a disordered medium. **Nature**, v. 390, n. 6661, p. 671–673, 1997.

90 BEEK, T. van der *et al.* Light transport through disordered layers of dense gallium arsenide submicron particles. **Physical Review B**, v. 85, n. 11, p. 115401, 2012.

91 CHABANOV, A. A.; STOYTCHEV, M.; GENACK, A. Z. Statistical signatures of photon localization. **Nature**, v. 404, n. 6780, p. 850–853, 2000.

92 COTTIER, F. *et al.* Microscopic and macroscopic signatures of 3d Anderson localization of light. **Physical Review Letters**, v. 123, n. 8, p. 083401, 2019.

93 GOODMAN, J. W. **Speckle phenomena in optics**. 2nd. ed. Bellingham: SPIE Press, 2020. (Press Monographs).

94 STECK, D. A. **Quantum and atom optics**. 2021. Available at: https://atomoptics.uoregon.edu/ dsteck/teaching/quantum-optics/quantum-optics-notes.pdf. Accessible at: 21 Sept. 2023.

95 ARAÚJO, M. O.; GUERIN, W.; KAISER, R. Decay dynamics in the coupled-dipole model. **Journal of Modern Optics**, v. 65, n. 11, p. 1345–1354, 2017.

96 CIPRIS, A. *et al.* Subradiance with saturated atoms: population enhancement of the long-lived states. **Physical Review Letters**, v. 126, n. 10, p. 103604, 2021.

97 SETHI, S. P.; THOMPSON, G. L. **Optimal control theory**: applications to management science and economics. 2nd. ed. Berlin: Springer, 2000.

98 MOREIRA, N. A.; KAISER, R.; BACHELARD, R. Localization vs. subradiance in three-dimensional scattering of light. **Europhysics Letters**, v. 127, n. 5, p. 54003, 2019.

99 SATMAN, M. H. Fast online detection of outliers using least-trimmed squares regression with non-dominated sorting based initial subsets. **International Journal of Advanced Statistics and Probability**, v. 3, n. 1, p. 53, 2015.

100 HAWKINS, D. M.; OLIVE, D. Applications and algorithms for least trimmed sum of absolute deviations regression. **Computational Statistics and Data Analysis**, v. 32, n. 2, p. 119–134, 1999.

APPENDIX

APPENDIX A – TIME EVOLUTION

A.1 Formal Solution

Following the reasoning of Sethi and Thompson,[97] while emphasizing the parts that are relevant to our project, we here present in detail the derivation of the formal solution of the linear equation discussed in the main text.

Let us start by multiplying both sides of 4.17 by e^{-Gt}, and then combining the terms containing $\vec{\beta}$ into the derivative:

$$\begin{aligned} e^{-Gt}\frac{d\vec{\beta}}{dt} &= e^{-Gt}\vec{\beta} + e^{-Gt}\vec{\Omega} \\ e^{-Gt}\frac{d\vec{\beta}}{dt} - e^{-Gt}\vec{\beta} &= e^{-Gt}\vec{\Omega} \\ \frac{d(\vec{\beta}e^{-Gt})}{dt} &= e^{-Gt}\vec{\Omega}. \end{aligned}$$

Integrating from 0 to t:

$$\begin{aligned} e^{-G\tau}\vec{\beta}(\tau)|_0^t &= \int_0^t e^{-G\tau}\vec{\Omega}d\tau \\ e^{-Gt}\vec{\beta}(t) - e^{-G0}\vec{\beta}(0) &= \int_0^t e^{-\tau G}\vec{\Omega}d\tau \\ e^{-Gt}\vec{\beta}(t) &= \vec{\beta}(0) + \int_0^t e^{-G\tau}\vec{\Omega}d\tau, \end{aligned}$$

leads to the well-know solution

$$\vec{\beta}(t) = e^{Gt}\vec{\beta}(0) + e^{Gt}\int_0^t e^{-G\tau}\vec{\Omega}d\tau. \tag{A.1}$$

As currently written, Equation A.1 still can be improved by using matrix diagonalization. Specifically, if we express G as $\psi\lambda\psi^{-1}$, the analytical solution can be further refined, as also shown in Sethi and Thompson[97]

$$\vec{\beta}(t) = (\psi e^{t\lambda}\psi^{-1})\vec{\beta}(0) + \psi e^{t\lambda}\int_0^t e^{-t\lambda}\psi^{-1}\vec{\Omega}d\tau. \tag{A.2}$$

While Equation A.2 is computationally efficient, integration is still required using a numerical quadrature method. Nevertheless, this integral can be solved analytically, and the proof is as follows. Expanding $e^{-\tau} =$ in power series

$$\int_0^t e^{-t\lambda}\psi^{-1}\vec{\Omega}d\tau = \psi^{-1}\vec{\Omega}\int_0^t \left[I - \lambda\tau + \frac{1}{2!}\lambda^2\tau^2 - \frac{1}{3!}\lambda^3\tau^3 + ...\right]d\tau.$$

Integrating term by term

$$\int_0^t e^{-t\lambda}\psi^{-1}\vec{\Omega}d\tau = \psi^{-1}\vec{\Omega}\left[t - \lambda\frac{t^2}{2} + \frac{1}{2!}\lambda^2\frac{t^3}{3} - \frac{1}{3!}\lambda^3\frac{t^4}{4} + ...\right],$$

and multiplying by λ on both sides to adjust the exponent series to start at 1

$$-\lambda\left(\int_0^t e^{-t\lambda}\psi^{-1}\vec{\Omega}d\tau\right) = \psi^{-1}\vec{\Omega}\left[-\frac{1}{1}(\lambda t)^1 + \frac{1}{2!}(\lambda t)^2 - \frac{1}{3!}(\lambda t)^3 + \frac{1}{4!}(\lambda t)^4 - ...\right].$$

To complete the power series, we include the term $\psi^{-1}\vec{\Omega}(\lambda t)^0$

$$-\lambda\left(\int_0^t e^{-t\lambda}\psi^{-1}\vec{\Omega}d\tau\right) + \psi^{-1}\vec{\Omega} = \psi^{-1}\vec{\Omega}\left[I - \frac{1}{1}(\lambda t)^1 + \frac{1}{2!}(\lambda t)^2 - \frac{1}{3!}(\lambda t)^3 + \frac{1}{4!}(\lambda t)^4 + ...\right].$$

Converting the power series into an exponential form,

$$-\lambda\left(\int_0^t e^{-t\lambda}\psi^{-1}\vec{\Omega}d\tau\right) + \psi^{-1}\vec{\Omega} = \psi^{-1}\vec{\Omega}e^{-\lambda t}.$$

Finally, the integral term can be solved as

$$\int_0^t e^{-t\lambda}\psi^{-1}\vec{\Omega}d\tau = \lambda^{-1}(e^{-t\lambda} - I)\psi^{-1}\vec{\Omega}. \tag{A.3}$$

By substituting equation A.2 into equation A.3, we obtain a numerical solution for equation 4.17, at the cost of performing an eigen decomposition. The resulting expression is as follows:

$$\vec{\beta}(t) = (\psi e^{t\lambda}\psi^{-1})\vec{\beta}(0) + \psi e^{t\lambda}\lambda^{-1}(e^{-t\lambda} - I)\psi^{-1}\vec{\Omega}. \tag{A.4}$$

The empirical numerical scaling discussed in Equation A.4 is not straightforward, as different machines produced varying results ranging from $O(N^{2.7-3.1})$. The dominant operation comes from the eigendecomposition operation, and we found it to have a complexity of $O(N^{2.7})$. Figure 41 shows different algorithm complexity scalings for different machines - using parameters where eigendecomposition is known to be the fastest approach as compared to a numerical ODE solver.

A.2 Solvers Comparison

In the main text, only the top 10 ODE solvers were presented. Here, we show all 36 results that we studied.

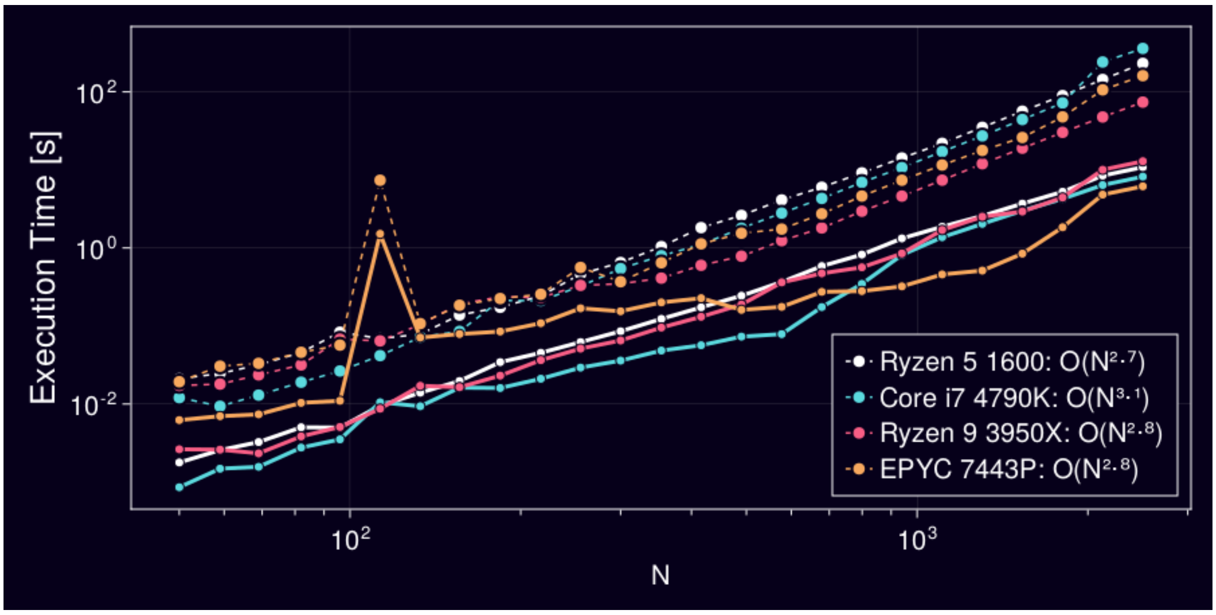


Figure 41 – The dotted lines correspond to the formal solution using eigendecomposition, while the solid lines represent the fastest known numerical ODE solver. The abrupt spike on the EPYC 7443P is indeed a hardware outlier of that particular machine used for benchmarking.

Source: By the author.

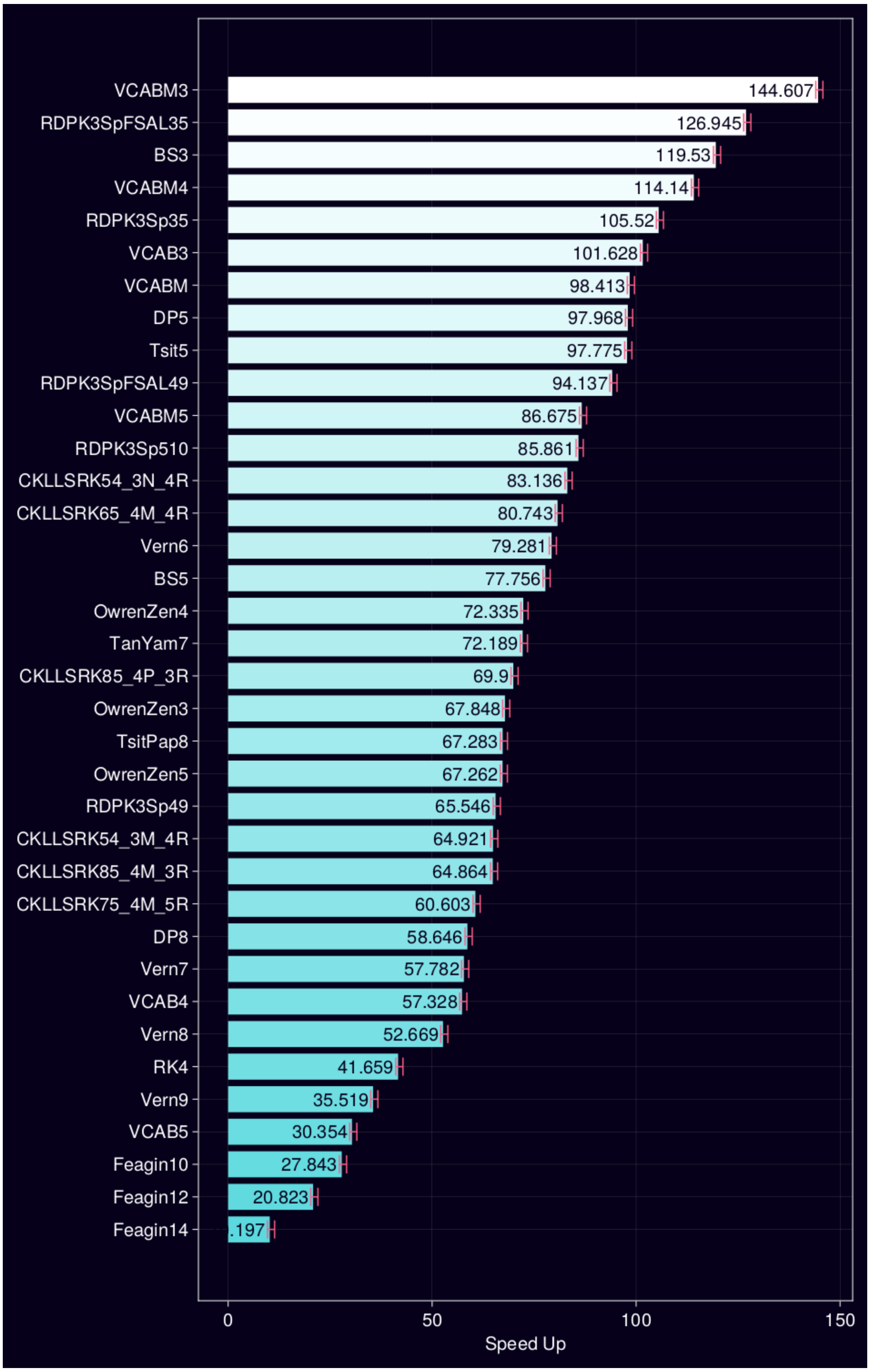


Figure 42 – All speed-ups for dilute density.

Source: By the author.

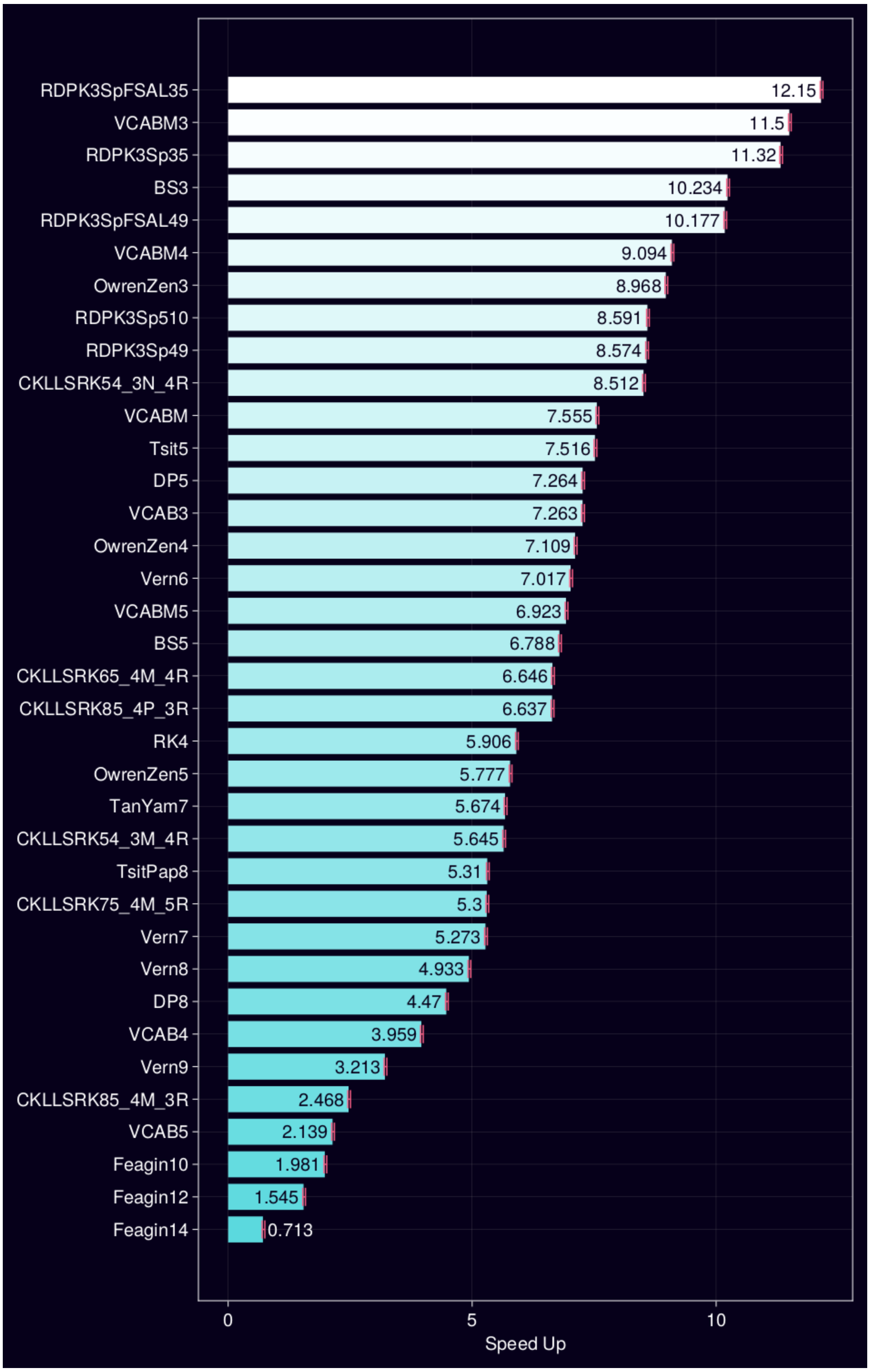


Figure 43 – All speed-ups for high density.

Source: By the author.

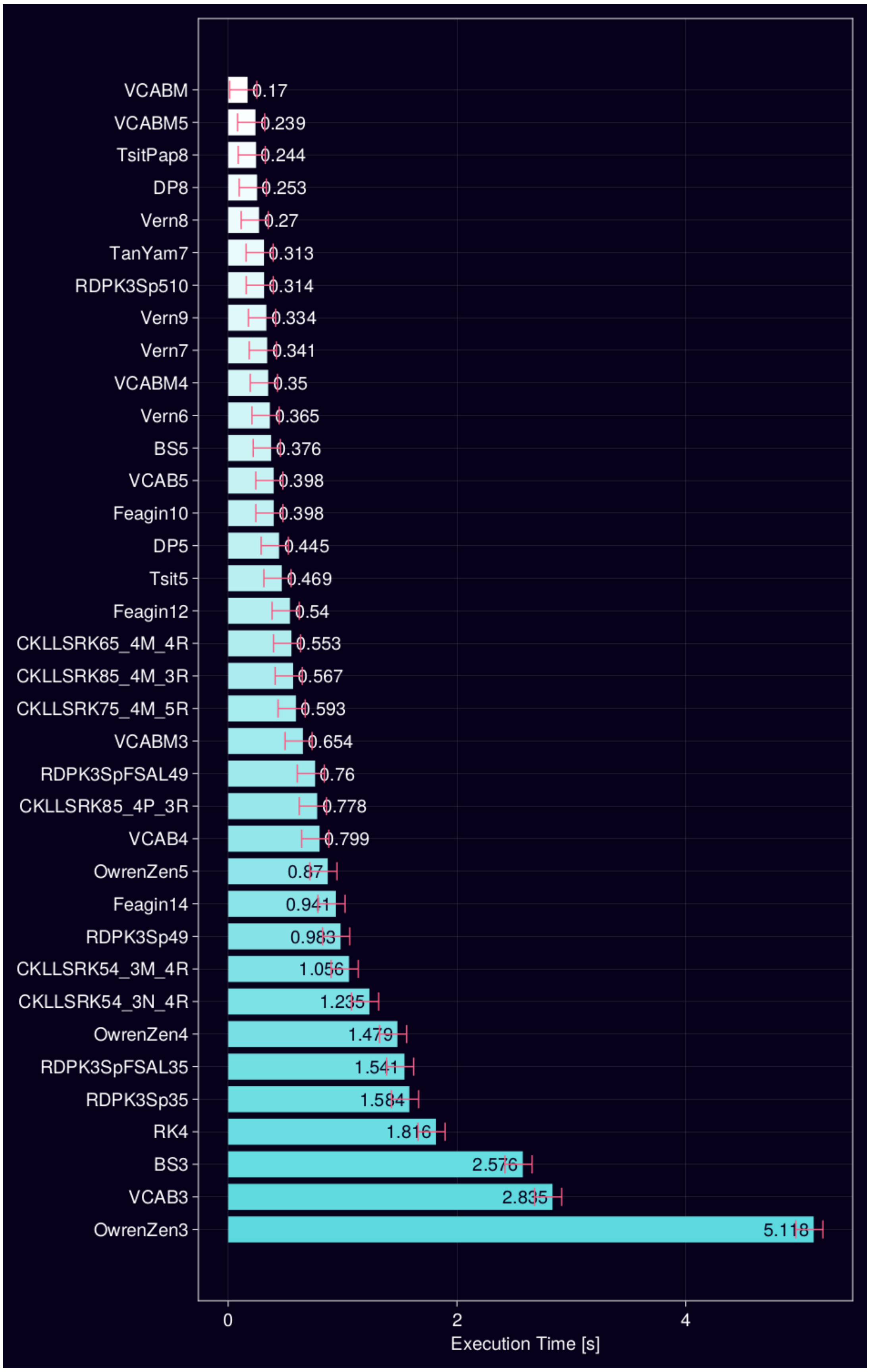


Figure 44 – All execution times for dilute density.

Source: By the author.

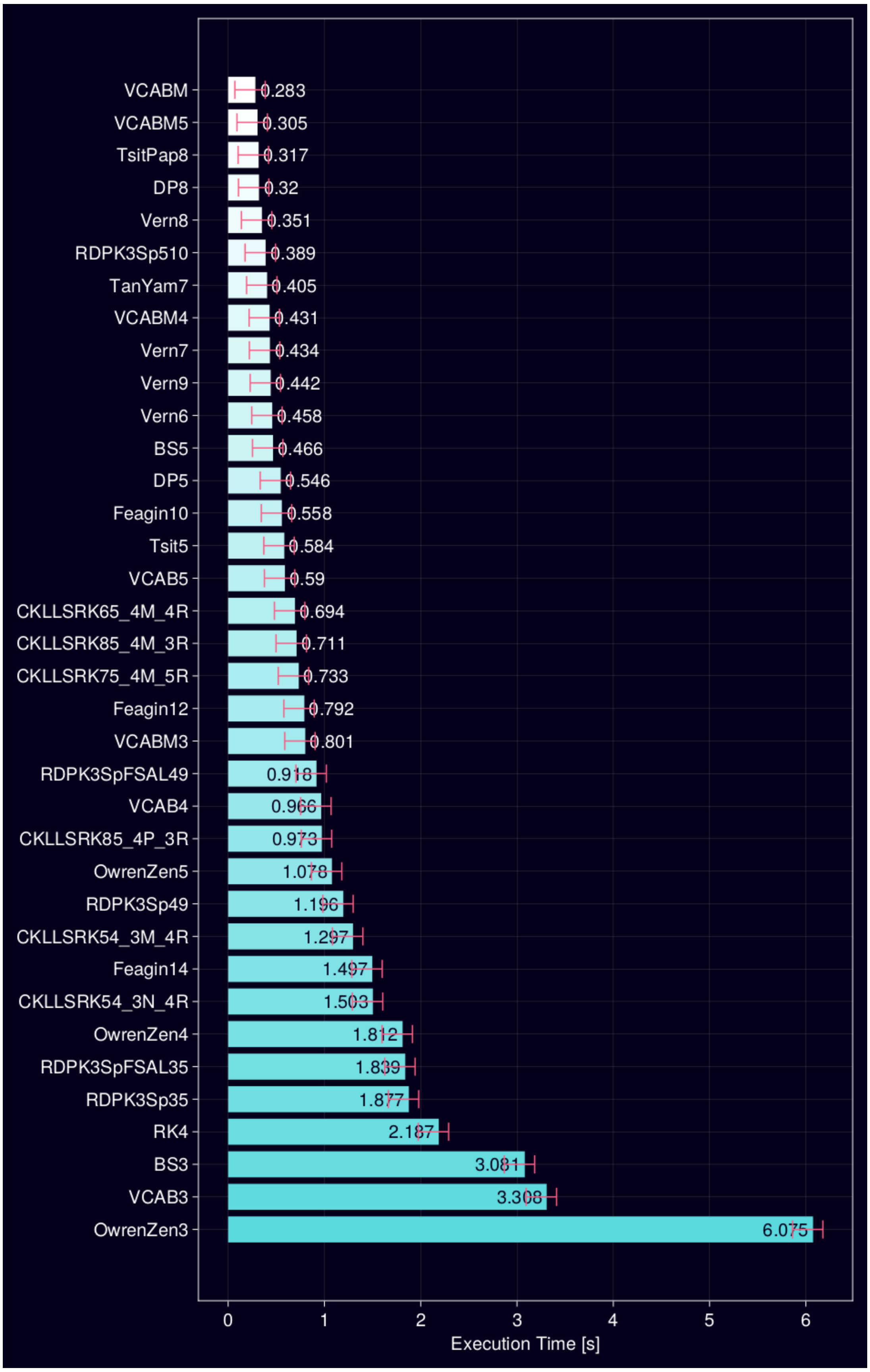


Figure 45 – All execution times for high density.

Source: By the author.

APPENDIX B – LOCALIZATION LENGTH OF SCATTERING MODES

Automatically classifying modes as localized or not is challenging because the common fitting function of the form $y = Ae^{-x/\xi}$ is suitable for only a limited set of spatial profiles, as depicted in Figure 46 (a) and (b). Many profiles require an *ad hoc* pre-processing phase, which is not straightforward even for a trained human. Moreover, automating this pre-processing step is even more complex and not well-defined, as it involves making decisions and adjustments that are not easily programmable.

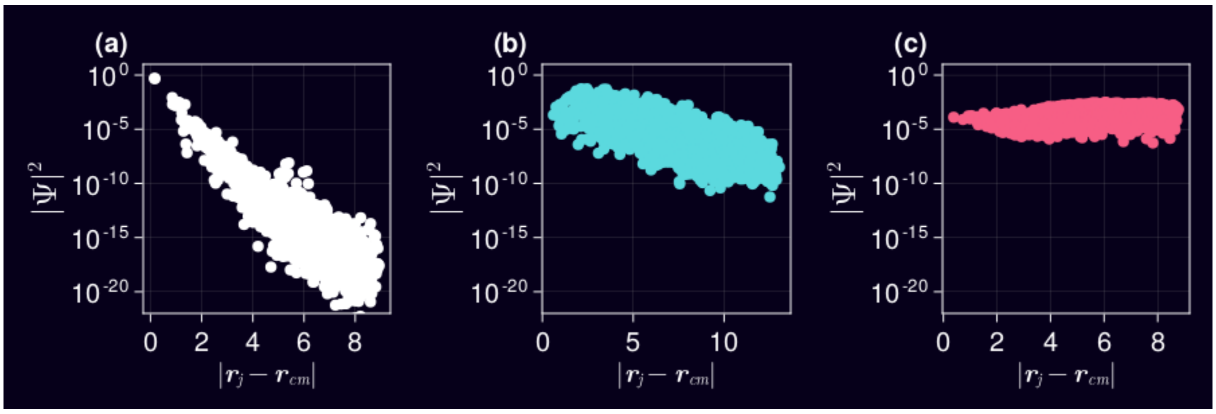


Figure 46 – Different spatial profiles demands different approach for an exponential fitting. A fitting in (a) that takes in account all the values would be misleading, the data has an uneven spread for small values. In (b) this is not the case and a traditional least square fitting would be enough. For comparison, (c) is not a localized mode.

Source: By the author.

Our approach to identify localized modes involves two stages. Firstly, we leverage the eigenvector property that $\sum_j |\Psi_j|^2 = 1$, where we anticipate that atoms near the center of mass would have the highest contribution. Secondly, we employ techniques to identify outliers and eliminate them from the fitting procedure. These steps combined enable us to more effectively pinpoint and characterize localized modes in our analysis.

B.1 Atom's contribution to the mode

Our interpretation of the eigenvectors is that each atom j contributes with $|\Psi_j|^2$ to the n-th mode. In the case of localized modes, certain atoms have a more substantial influence and weight compared to others. To focus our analysis on the most influential atoms and exclude less significant ones, we adopt an approach that selects only the atoms representing a certain cumulative percentage of the overall distribution. Figure 47 shows an example of points selected, in blue, based upon this argument. In practice, this is controlled setting the variable `probability_threshold` in the `get_localization_length` function. Its default value is 0.999 and we chose this specific threshold value carefully to ensure that it does not significantly alter any previously known interpretations of the mode statistics, even though it may not be perfect to distinguish localized modes.

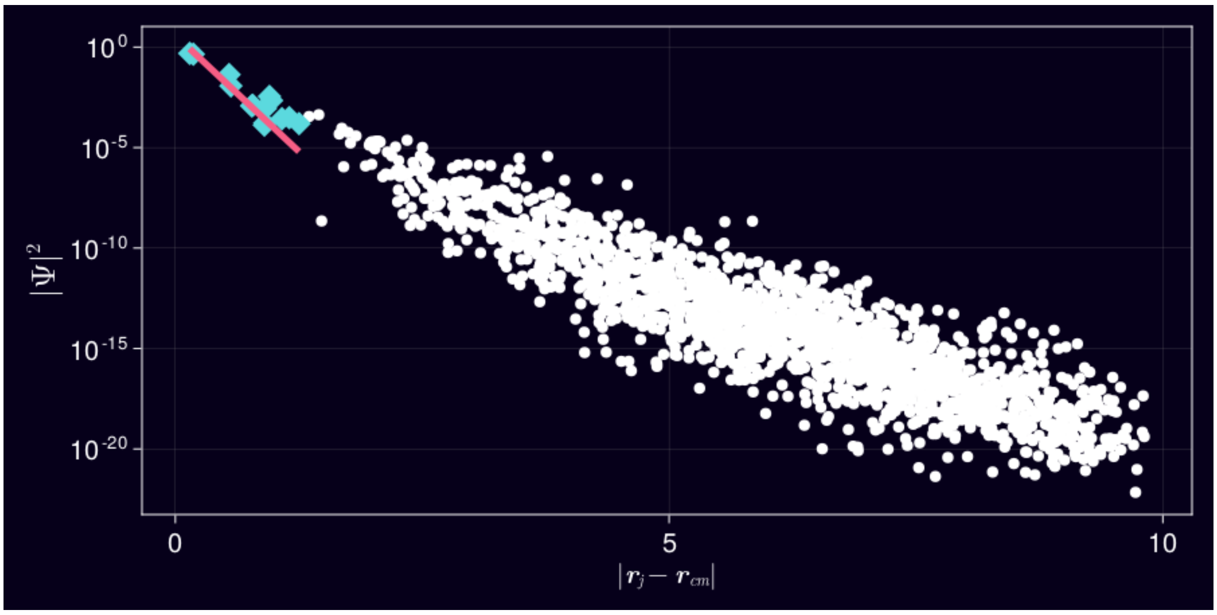


Figure 47 – Localized modes typically have most of their weight concentrated around only a few atoms. In this example, the atoms marked in blue represent the selected ones used to compute the fitting of the mode, the solid line.

Source: By the author.

B.2 Fitting with outliers

After selecting the atoms from the Spatial Profile, we observe that they exhibit irregularities, which can affect the accuracy of the fitting process. To mitigate this issue, we aim to downplay the influence of outliers in the fitting procedure. In a previous work,[98] we utilized a fitting method that minimized the absolute difference of errors rather than the usual squared error. However, for the current analysis, we chose to use the `LinRegOutliers.jl` package, which effectively identifies outliers and excludes them from the analysis. In the end result, describe on the next section, this approach has proven to be more reliable to identify localized modes. Among various benchmarked algorithms, the `satman2015` method[99] demonstrated the lowest error and is therefore set as the default method. The second option, `lta`,[100] while not as precise, offers faster execution times.

B.3 Localized Mode Selection

After completing the fitting procedure, we assess its accuracy using the Coefficient of Determination, R^2, which is given by the equation:

$$R^2 = 1 - \frac{\sum_j (y_j - \langle y \rangle)^2}{\sum (y_j - \tilde{y}_j)^2}, \tag{B.1}$$

where $y_j = |\Psi_j^n|^2$, and $\tilde{y}_j$ represents the estimated value obtained from the fitting process. Based on the value of R^2, we classify modes with $R^2 > 0.5$ as localized. Although this threshold is arbitrary, it is not unreasonable, as demonstrated in Figure 48, with modes on dilute regime, top row, never reaching such value. To change this value, set the variable `fitting_threshold` on the function `classify_modes`.

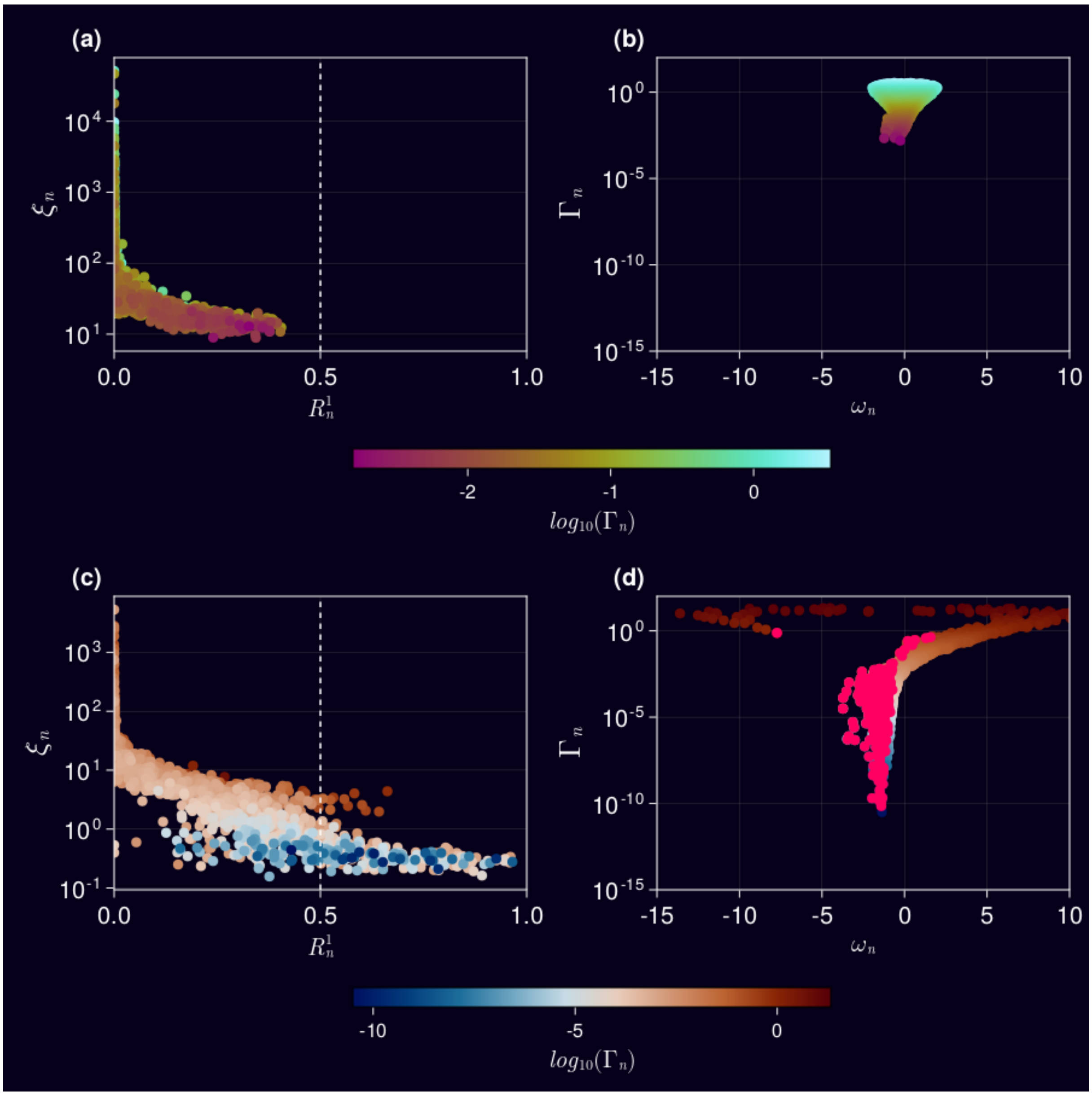


Figure 48 – The data comes from a cylindrical cloud with $N = 2000$ atoms and varying densities. Panels (a) and (b) show low-density cases with $\rho k_0^{-3} = 0.04$, resulting in ξ_n values consistently below $R^2 = 0.5$. Panels (c) and (d) correspond to high-density cases with $\rho k_0^{-3} = 1.0$, showing ξ_n values above the threshold.

Source: By the author.

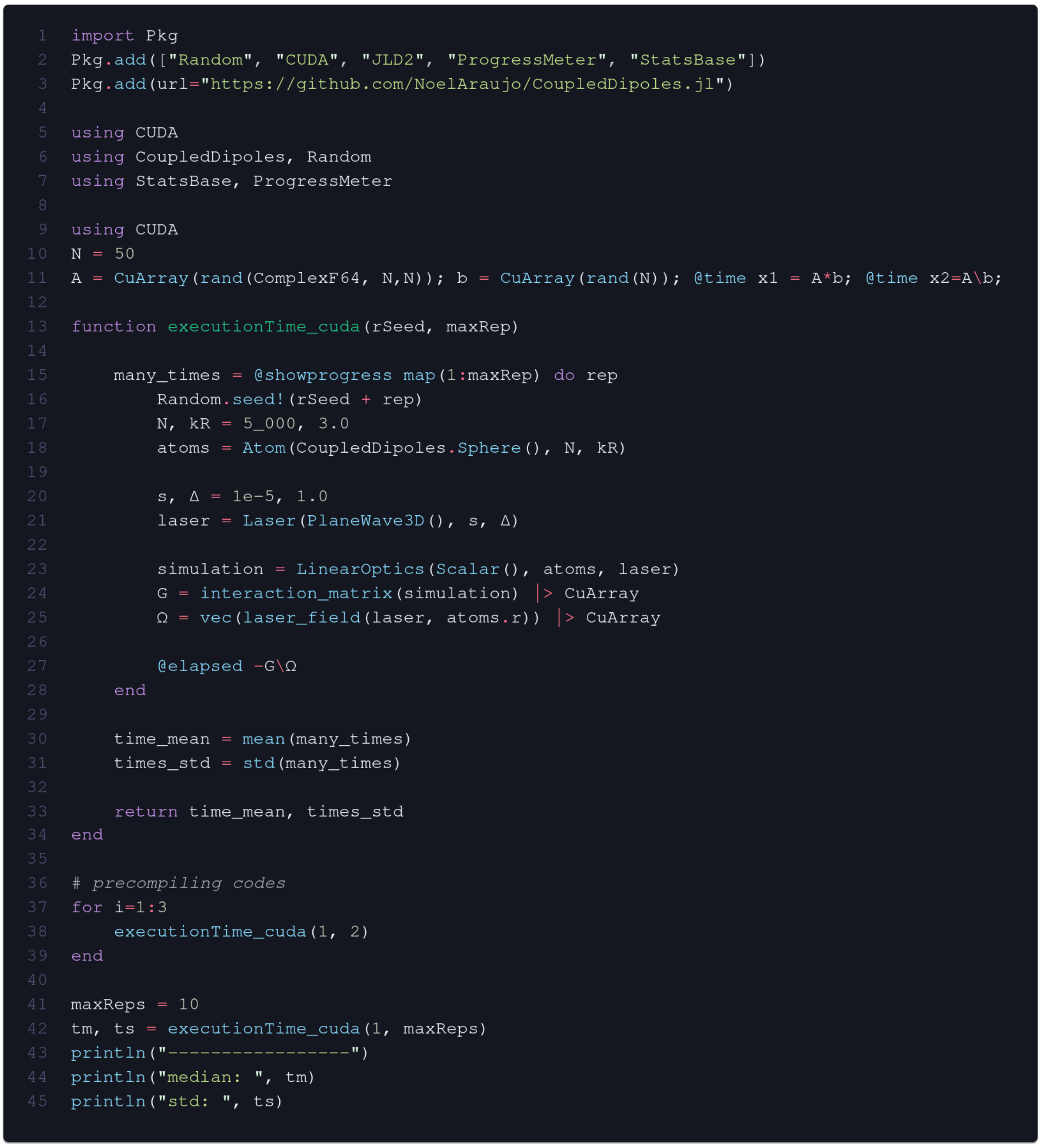
APPENDIX C – GRAPHICAL CARDS UNITS

To make sure the tests can be repeated, we gave reference numbers for the machines we used. You can find them on `vast.ai`. Its important to acknowledge that the machines offered are part of a marketplace, and its possible for the host to withdraw a machine from the network. Our main goal is to set the expectation from the hardware used for future comparison if needed.

Table 3 – GPUs models and their respective host and machine on `vast.ai` market place.

GPU Model	**Host**	**Machine**	**CUDA version**	**CPU Model**
RTX 3060	48810	8888	12.0	Xeon E5-2680 v3
RTX 3070	23445	3510	11.4	Core i9-10940X
RTX 3080	43425	8114	11.6	Ryzen 9 5900X
RTX 3090	46971	8459	12.0	Xeon E5-2698 v3
RTX 4090	9656	8652	12.0	EPYC 7443
A10	32241	7710	12.0	EPYC 7413
A40	23806	3809	11.4	Xeon E5-2650 v4
A100 SXM4	23697	6062	11.6	EPYC 7763
A5000	47223	7471	11.7	EPYC 7252
A6000	47223	7458	11.7	EPYC 7252
V100	32241	7654	11.8	Xeon E5-2698 v4

Source: By the author.



```julia
import Pkg
Pkg.add(["Random", "CUDA", "JLD2", "ProgressMeter", "StatsBase"])
Pkg.add(url="https://github.com/NoelAraujo/CoupledDipoles.jl")

using CUDA
using CoupledDipoles, Random
using StatsBase, ProgressMeter

using CUDA
N = 50
A = CuArray(rand(ComplexF64, N,N)); b = CuArray(rand(N)); @time x1 = A*b; @time x2=A\b;

function executionTime_cuda(rSeed, maxRep)

    many_times = @showprogress map(1:maxRep) do rep
        Random.seed!(rSeed + rep)
        N, kR = 5_000, 3.0
        atoms = Atom(CoupledDipoles.Sphere(), N, kR)

        s, Δ = 1e-5, 1.0
        laser = Laser(PlaneWave3D(), s, Δ)

        simulation = LinearOptics(Scalar(), atoms, laser)
        G = interaction_matrix(simulation) |> CuArray
        Ω = vec(laser_field(laser, atoms.r)) |> CuArray

        @elapsed -G\Ω
    end

    time_mean = mean(many_times)
    times_std = std(many_times)

    return time_mean, times_std
end

# precompiling codes
for i=1:3
    executionTime_cuda(1, 2)
end

maxReps = 10
tm, ts = executionTime_cuda(1, maxReps)
println("-----------------")
println("median: ", tm)
println("std: ", ts)
```

Figure 49 – Benchmark run of all remote nodes to acquire the data presented on the text.

Source: By the author.