

# Quantum Mechanics: Electromagnetically-Induced transparency

---

## Abstract

This experiment simulates a Lambda three-level atomic system in order to examine the phenomenon of electromagnetically induced transparency (EIT). The tasks involve calculating the Hamiltonian, density matrix, and numerically solving the dynamics of the system. Research includes factors influencing transitions, the impact of probe fields on atomic populations, and simulations using different intensities of pump fields and activation of probe fields. The study offers valuable insights into the phenomena of electromagnetically induced transparency (EIT) and their relationships with field strengths and activation timings. This contributes to a better understanding of the interaction between light and atoms in three-level atomic systems.

## 1 INTRODUCTION & THEORETICAL BACKGROUND

Quantum mechanics arose due to the limitations of classical physics in describing phenomena occurring at the atomic and subatomic scales. While classical mechanics performed a great job of explaining the macroscopic universe, it was unable to explain the unusual behaviour of particles such as electrons and photons. This motivated a new theory and approach, quantum mechanics, to resolve inconsistencies in describing phenomena such as black-body radiation, the photoelectric phenomenon, and the atomic spectra (Mann, 2022).

Max Planck's pivotal proposal in 1900 introduced a revolutionary concept: that light, with frequency  $\nu$ , is emitted in discrete packets of energy, quantized as integral multiples of  $E = h\nu = \hbar\omega$ , where  $h$  is the Planck's constant,  $\hbar$  is the reduced Planck's constant, and  $\nu$  and  $\omega$  are the frequency and angular frequency respectively (Morin, 2008). This groundbreaking idea laid the groundwork for quantum theory, challenging the very foundations of classical physics.

Quantum mechanics diverges from classical mechanics' deterministic laws, embracing a framework grounded in probability and statistics. Moreover, it uncovered the wave-particle duality, introducing concepts like wave functions, superposition, and entanglement while establishing fundamental principles such as the Heisenberg uncertainty principle.

The evolution of the atomic model began with Dalton's idea of tiny, unbreakable spheres and continued through Thomson's discovery of electrons and Rutherford's nuclear model. Bohr proposed that electrons revolve around the nucleus in fixed orbits (Gallagher & Ingram, 2015), but the subsequent quantum mechanical model developed by Schrödinger and Heisenberg embraced a different concept. Instead of specific orbits, they described where electrons might be found using probability within regions called orbitals.

In this new model, electrons are viewed as waves, making their behaviour a matter of probability. By employing the Schrödinger equation, various wave functions are derived to describe electron behaviour and how these waves change over time. Consequently, these functions help calculate the likelihood of finding electrons in different positions. These probabilities give rise to atomic orbitals, illustrating where an electron is likely to be found (Romforn, 2023).

Additionally, this model relies on four quantum numbers:  $n$ ,  $l$ ,  $m_l$ , and  $m_s$  for a wave function. The principal quantum number  $n$  defines the orbital's size and energy, while the azimuthal quantum number  $l$  represents the orbital's angular momentum. The magnetic quantum number  $m_l$  describes the electron's orientation within a sub-level, and the spin quantum number  $m_s$  signifies the electron's spin (either "spin-up" or "spin-down") (Silberberg et al., 2006). Furthermore,  $m_s$  must adhere to the Pauli exclusion principle, indicating that within the same orbital, two electrons must have opposite spins.

Lasers function based on the principle of stimulated emission, wherein excited atoms emit photons that induce identical emissions in a gain medium. This leads to a population inversion, where there are a greater number of atoms in an excited state compared to those in a lower-energy state (Svelto, Hanna, et al., 2010) Inside an optical resonator (set of reflecting mirrors), these photons undergo amplification and reflection, resulting in the generation of coherent and monochromatic light. The essential elements consist of the gain medium and resonator, which are vital for producing a concentrated, high-energy beam. The foundation was established by Einstein's hypothesis of stimulated emission. Laser applications encompass a wide range of disciplines such as medicine, telecommunications, production, and others, exerting influence on various industries due to their distinct qualities and accurate light attributes.

A relatively simple laser system is comprised of three levels: a ground state, an excited state, and a metastable state. Atoms are commonly stimulated from their lowest energy level, known as the ground state, to a higher energy state called the excited state. However, they rapidly transition back to a relatively long-lived level known as the metastable state. Nevertheless, the metastable state, characterised by a long lifetime duration, requires a greater amount of time for atoms to undergo decay and return to the ground state. The transition from state 2 to 1 is known as the lasing transition. Imposing additional limitations on this system leads to the formation of a "new" system known as a  $(\Lambda)$  lambda 3 level system. In this system, transitions are possible between states 1 and 3, 2 and 3, but not between states 1 and 2. Prior to undergoing transitions, they are required to obey the selection rules, which are:  $\Delta l = \pm 1$  and  $\Delta m_l = 0$  or  $\pm 1$  (Sakurai & Commins, 1995).

Electromagnetically Induced Transparency (EIT) is a phenomenon that occurs in a three-level system, specifically a  $(\Lambda)$  lambda system, when it is exposed to a strong control field. Typically, atoms that are stimulated to reach an excited state rapidly transition to a metastable state. Nevertheless, if a potent control field is synchronised with the transition from the excited to the metastable states, it modifies the atomic behaviour in a manner that causes the excited state to divide into further levels. Under these conditions, the atoms exhibit transparency to a probing field inserted between the ground state and the excited state, preventing the atoms from moving from the ground state to the excited state (Finkelstein et al., 2023).

The objective of the experiment is to reproduce a  $(\Lambda)$  lambda three-level laser configuration and confirm the existence of electromagnetically induced transparency (EIT) in the presence of a strong electromagnetic field. This entails the examination of fluctuations in state populations when using different pump and probe fields. The objective of simulating this system is to replicate the circumstances in which EIT occurs and demonstrate the alterations in the population distribution among various atomic states based on the intensity and interaction of the pump and probe fields.

## 2 MATERIALS & COMPUTATIONAL TOOLS

The simulation employed Python, utilizing SymPy for symbolic mathematics to specify crucial system components such as the Hamiltonian and density matrices. NumPy was responsible for performing numerical computations, while SciPy's `odeint` function was utilized for solving differential equations. Matplotlib was employed to visualize the population dynamics over time.

## 3 METHODOLOGY

### 3.1 Task 1

The Hamiltonian governing the system was provided as,

$$\hat{H} = \hat{H}_0 + \hat{H}_{\text{pump}} + \hat{H}_{\text{probe}} \quad (1)$$

where,

$$\begin{aligned}\hat{H}_0 &= \hbar\omega_1|1\rangle\langle 1| + \hbar\omega_2|2\rangle\langle 2| + \hbar\omega_3|3\rangle\langle 3|, \\ \hat{H}_{\text{pump}} &= \hbar\varepsilon_{\text{pump}}e^{-i\omega_{\text{pump}}t}|2\rangle\langle 3| + \hbar\varepsilon_{\text{pump}}e^{i\omega_{\text{pump}}t}|3\rangle\langle 2|, \\ \hat{H}_{\text{probe}} &= \hbar\varepsilon_{\text{probe}}e^{-i\omega_{\text{probe}}t}|1\rangle\langle 3| + \hbar\varepsilon_{\text{probe}}e^{i\omega_{\text{probe}}t}|3\rangle\langle 1|.\end{aligned}$$

Where the states 1, 2 and 3 are denoted as  $|j\rangle$  (for  $j = 1, 2, 3$ ) with the respective frequency  $\omega_j$ . The strengths of the pump and probe fields, denoted as  $\varepsilon_{\text{pump}}$  and  $\varepsilon_{\text{probe}}$ . Moreover, the frequencies  $\omega_{\text{pump}}, \omega_{\text{probe}}$  are defined as:  $\omega_{\text{pump}} = \omega_{23} + \Delta_{\text{pump}}$  and  $\omega_{\text{probe}} = \omega_{13} + \Delta_{\text{probe}}$ , where the difference frequencies are represented as  $\omega_{jk} = \omega_k - \omega_j$  (for  $j, k = 1, 2, 3$ ).  $\Delta_{\text{pump}}, \Delta_{\text{probe}}$  denote the difference between the frequency of the pump field/probe field and resonant frequency of the state interacting with (detuning). In this case pump field is between state 2 and 3, while the probe field is between state 1 and 3.

First we set following :  $\Omega_1 = \omega_1 - \Delta_{\text{probe}}$ ,  $\Omega_2 = \omega_2 - \Delta_{\text{pump}}$ , and  $\Omega_3 = \omega_3$ .

then the following substitution was done on the Hamiltonian, to remove any oscillations and time dependent terms.

$$\begin{aligned}|j_i\rangle &\rightarrow e^{-i\Omega_j t}|j_i\rangle \quad (\text{for } j = 1, 2, 3), \\ \hat{H} &\rightarrow \hat{H} - \sum_j \hbar\Omega_j |j_i\rangle\langle j_i|.\end{aligned}$$

### 3.2 Task 2

The final form of the Hamiltonian obtained in task 1 was determined and simplified. Then noting that the atom has three levels one could represent such Hamiltonian in a 3 dimensional system, i.e. a matrix of size  $3 \times 3$ .

This can easily be done by observing the outer product of the states, which indicate the row and column of such matrix. For example the state:

$$\hbar\varepsilon_{\text{pump}}|2\rangle\langle 3| \quad (2)$$

represent the element,  $\hbar\varepsilon_{\text{pump}}$  in the matrix corresponding to row 2 and column 3. This was done for all states to obtain a 3-dimensional matrix representing the Hamiltonian:

$$\hat{H} = \begin{pmatrix} \hbar\Delta_{\text{probe}} & 0 & \hbar\varepsilon_{\text{probe}} \\ 0 & \hbar\Delta_{\text{pump}} & \hbar\varepsilon_{\text{pump}} \\ \hbar\varepsilon_{\text{probe}} & \hbar\varepsilon_{\text{pump}} & 0 \end{pmatrix}$$

Moreover the density matrix was defined as:

$$\rho = \begin{pmatrix} \rho_{11} & \rho_{12} & \rho_{13} \\ \rho_{21} & \rho_{22} & \rho_{23} \\ \rho_{31} & \rho_{32} & \rho_{33} \end{pmatrix}$$

### 3.3 Task 3

The following differential equation governing this system is given by ,

$$\dot{\rho} = \frac{i}{\hbar} [\rho, \hat{H}] + \mathcal{L}[\rho], \quad (3)$$

where

$$\mathcal{L}[\rho] = \gamma \sum_{j=1,2} |j\rangle\langle 3|\rho|3\rangle\langle j| - \frac{1}{2}|3\rangle\langle 3|\rho + \rho|3\rangle\langle 3| \quad (4)$$

Initially, we split the differential equation into two parts, the real part and the imaginary part. Then we solved each part separately but simultaneously. This was done by using the `odeint` function in Python, where `odeint` is a function that numerically solves ordinary differential equations. After solving and obtaining the elements of the density matrix, the diagonal elements were extracted, since they consist of the varying population in states  $|1\rangle$ ,  $|2\rangle$  and  $|3\rangle$ . This was repeated multiple times, but each time changing the value and behaviour of  $\varepsilon_{\text{pump}}$  and  $\varepsilon_{\text{probe}}$

After deriving the elements of the density matrix, the subsequent step involved plotting graphs showcasing the population against time in seconds. This was executed across diverse strengths of the probe and pump fields, along with different initial population states. Additionally, two specific tests were conducted: one where the probe field was abruptly activated and another where its strength was gradually increased. It's essential to note that the probe field strength consistently remained significantly lower than that of the pump field.

## 4 RESULTS

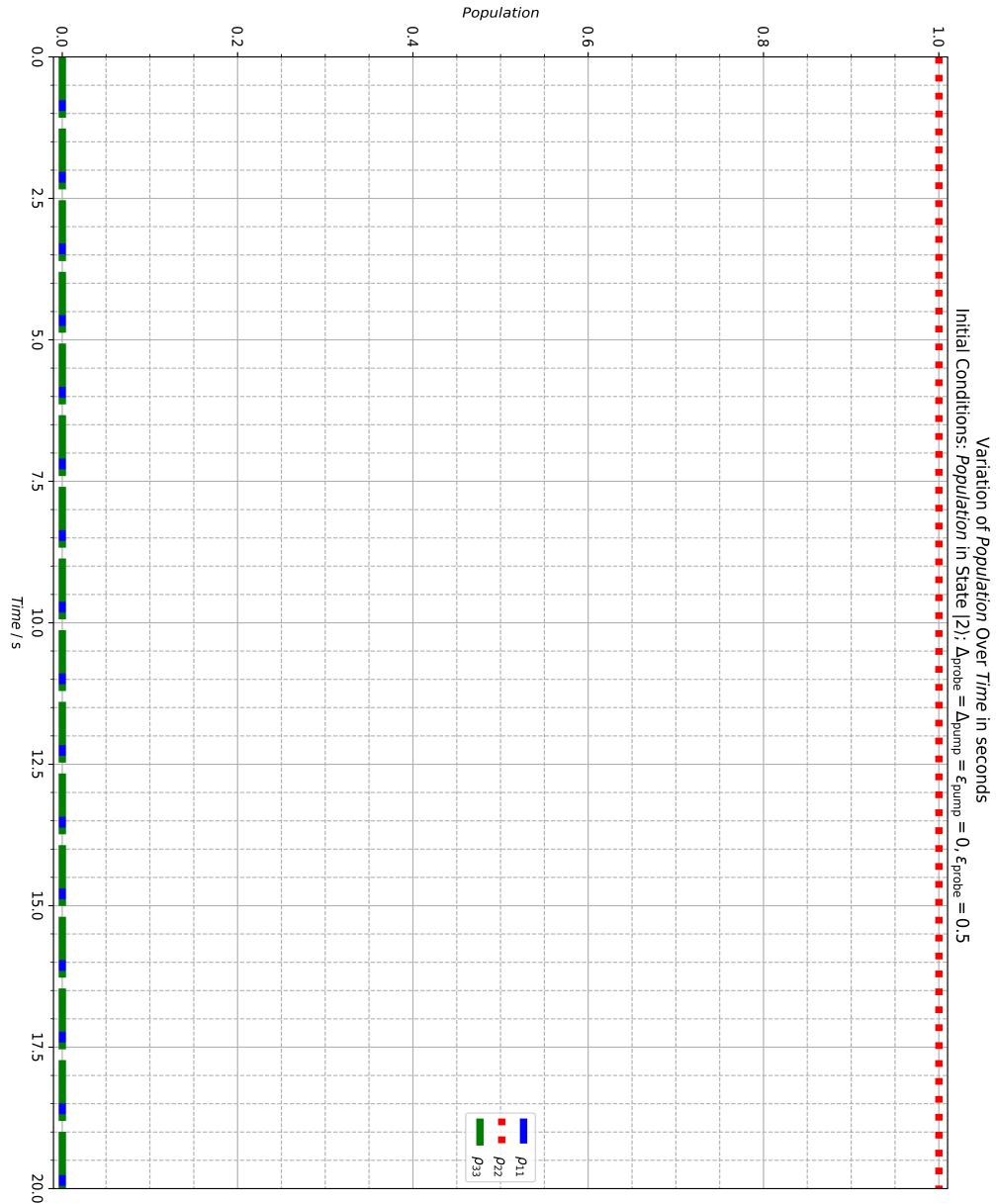
Figure 1 illustrates the population dynamics when initialised in state  $|2\rangle$ , with only the probe field activated. Evidently, the population remains constant as the probe field lacks coherence with the  $|2\rangle \rightarrow |3\rangle$  transition, resulting in no atom transitions.

Moving to Figure 2, it displays the population evolution starting from state  $|1\rangle$ , solely with the probe field active. Here, a transition from state  $|1\rangle$  to state  $|3\rangle$  is observed, followed by decay due to spontaneous emission to state  $|2\rangle$ , where the population stabilises. It stabilises in state  $|2\rangle$  since, there is no pump field active and thus, transition from state  $|2\rangle \rightarrow |3\rangle$  do not occur.

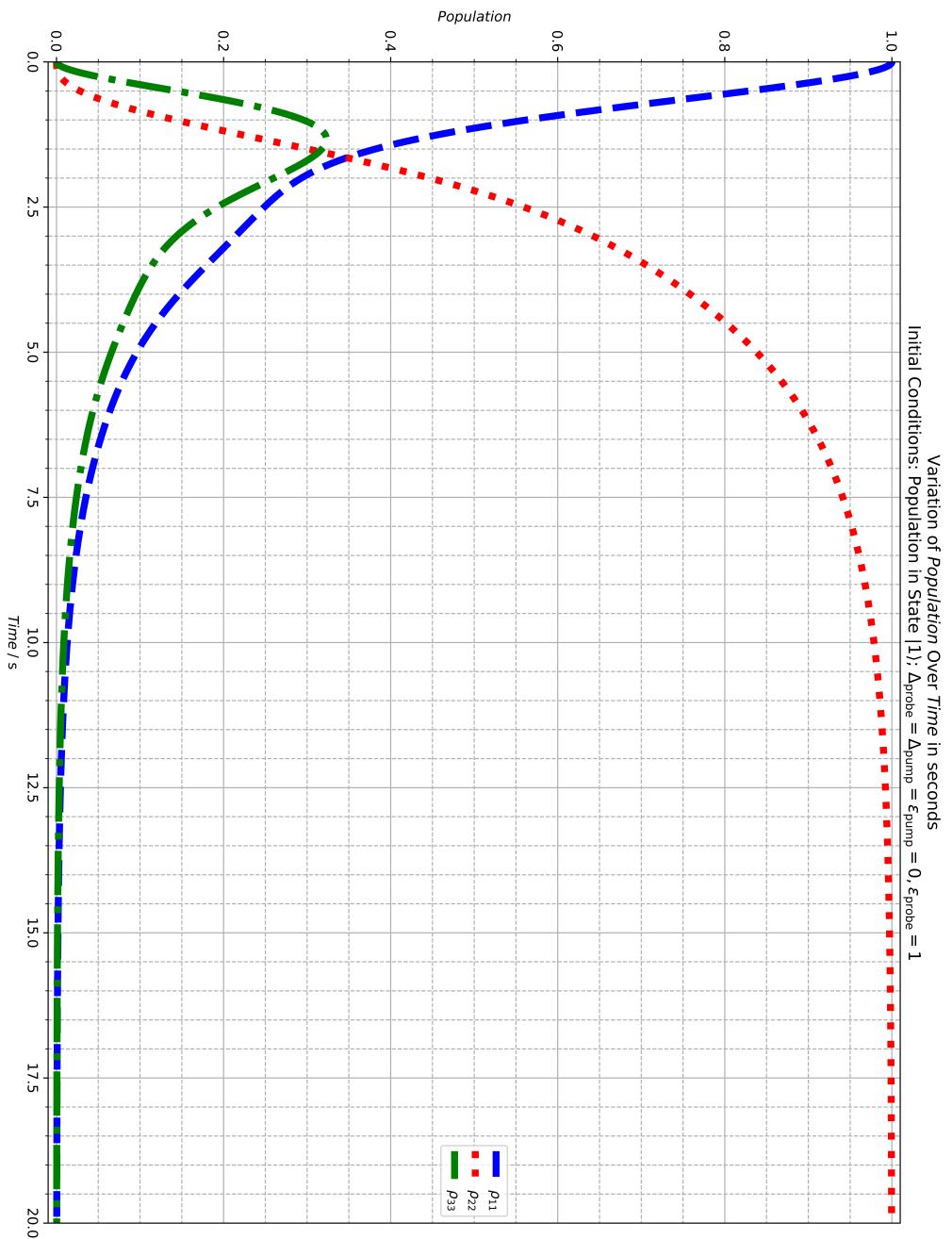
Figures 3 to 7 depict the population variation with the initial state as  $|1\rangle$  and the probe field suddenly activated at  $t = 10$ . As the subsequent graphs progress, there's an increase in the pump field. It's noticeable that with the increase in the pump field strength, the probe field becomes increasingly transparent, indicative of the occurrence of electromagnetically induced transparency (EIT). Notably, oscillations arise when the probe field is abruptly switched on.

In contrast, Figures 8 to 12 maintain the same initial conditions but introduce a slow increase in the strength of the probe field. Eventually, with the pump field surpassing the probe field significantly, EIT emerges once more. However, in this scenario, there are no oscillations; instead, a smooth transition is observed.

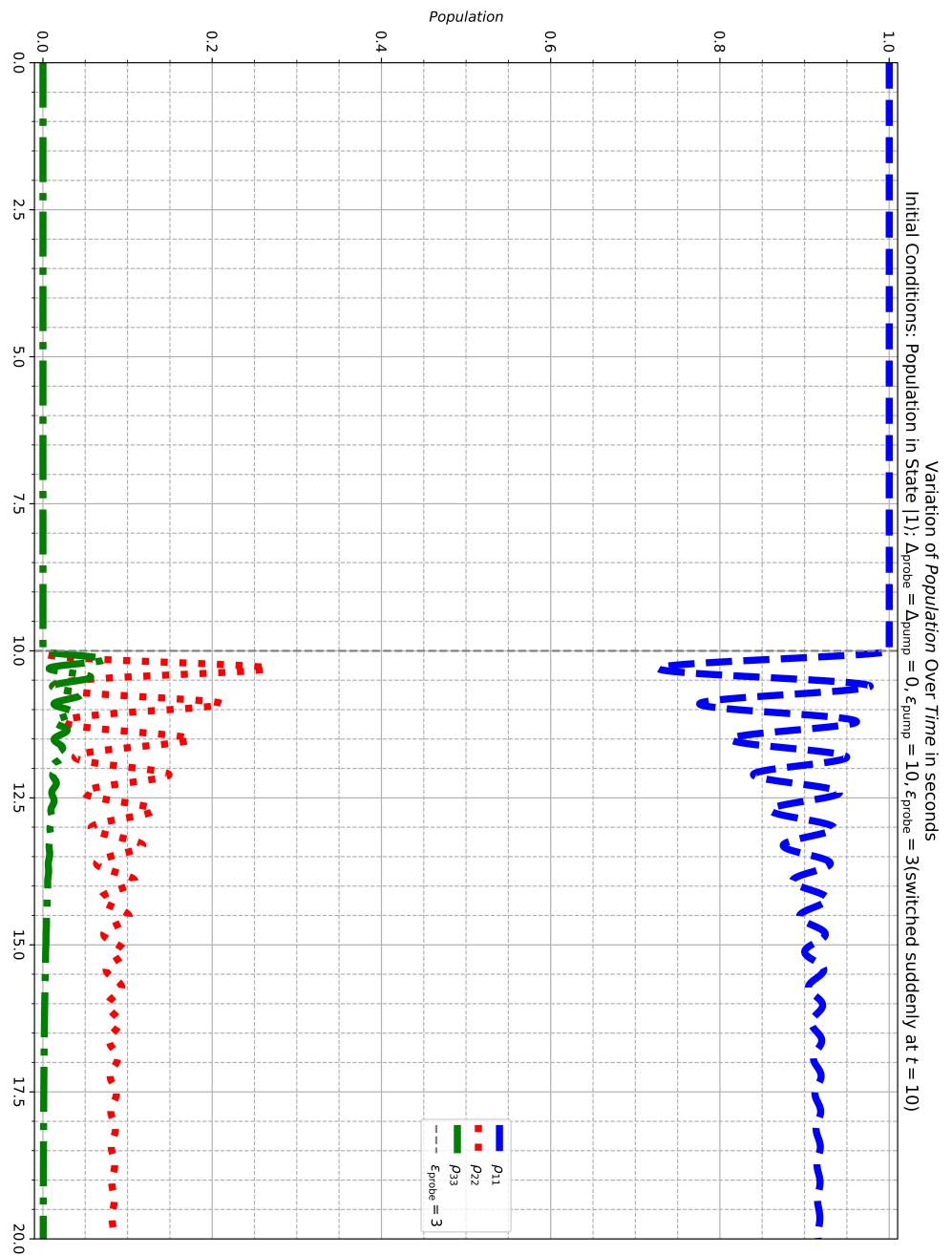
**It is important to note that throughout this results the value of the probe field strength was always much less than the value of the pump field strength. This is done in order to observe and verify the quantum phenomena EIT.**



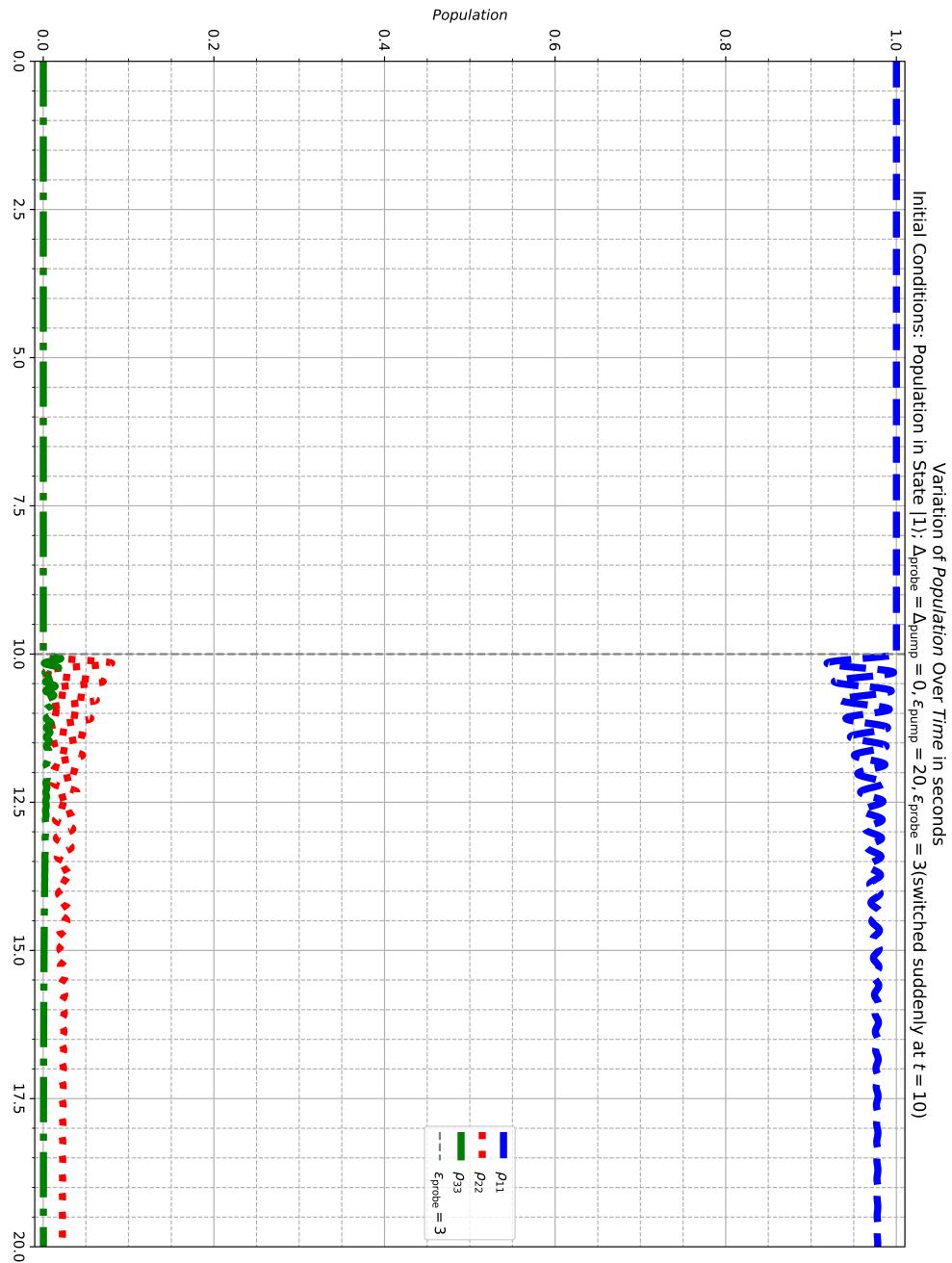
**Fig. 1.** Showing how the *Population* vary with *time* in seconds.



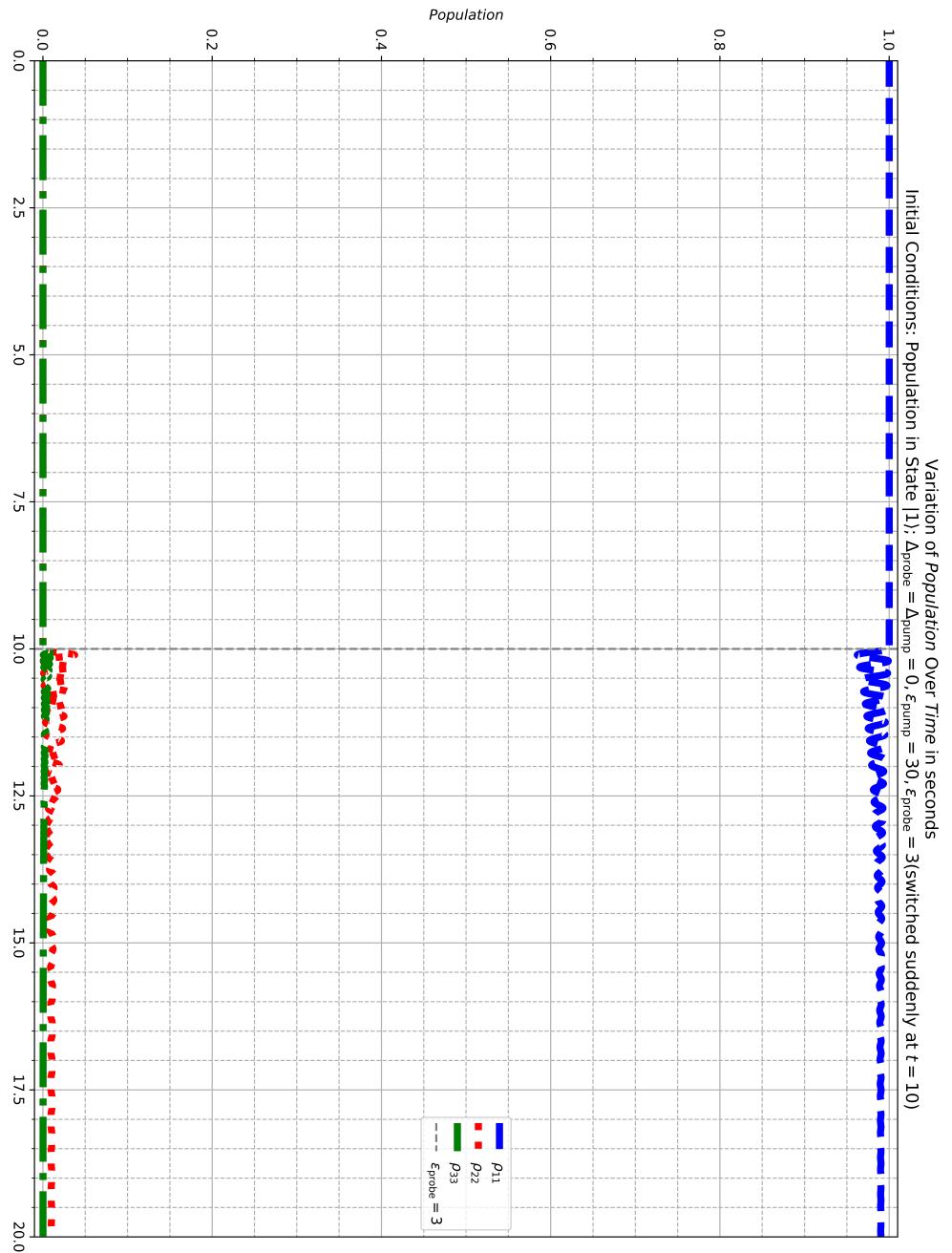
**Fig. 2.** Showing how the *Population* vary with *time* in seconds.



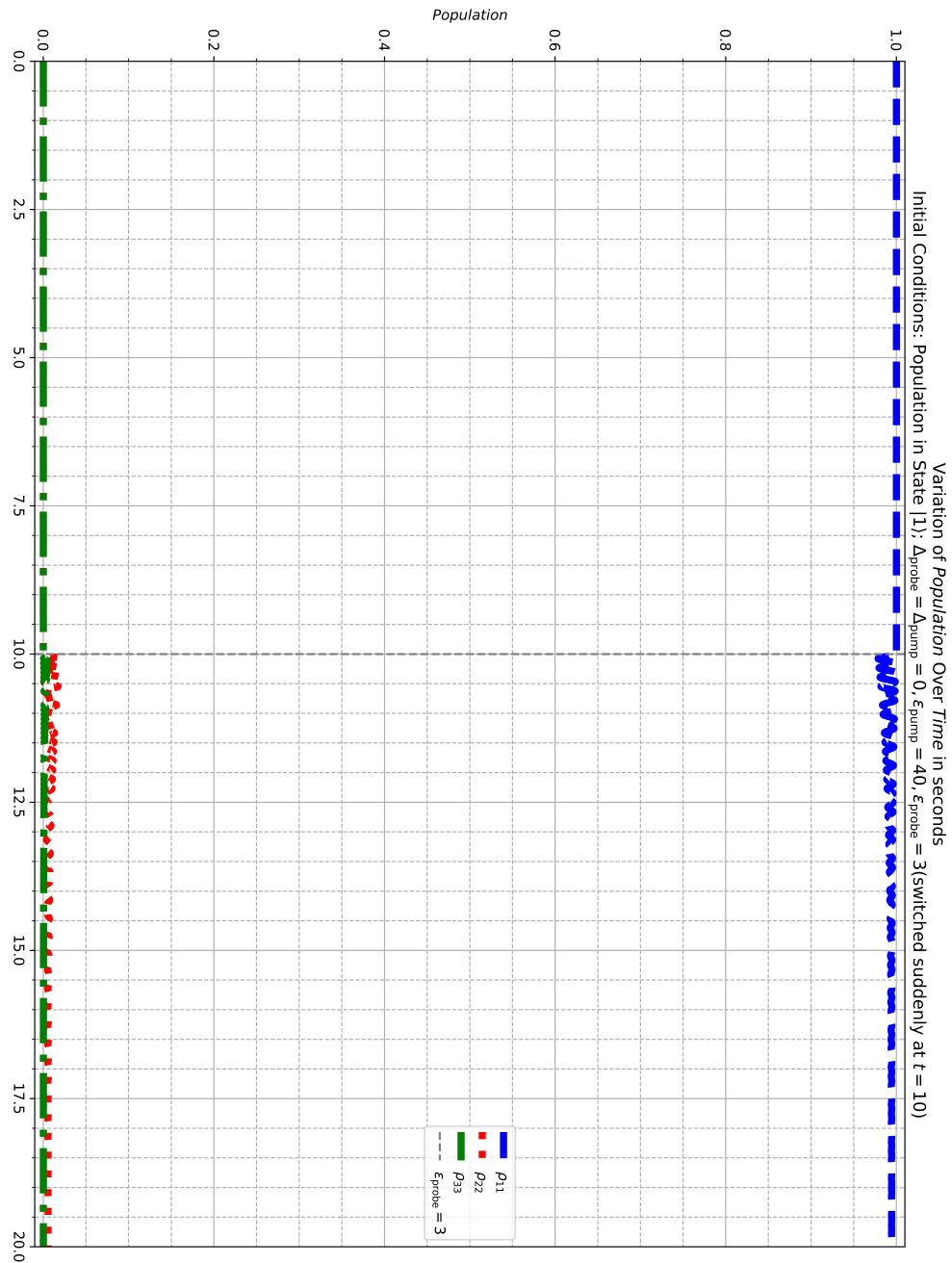
**Fig. 3.** Showing how the *Population* vary with *time* in seconds.



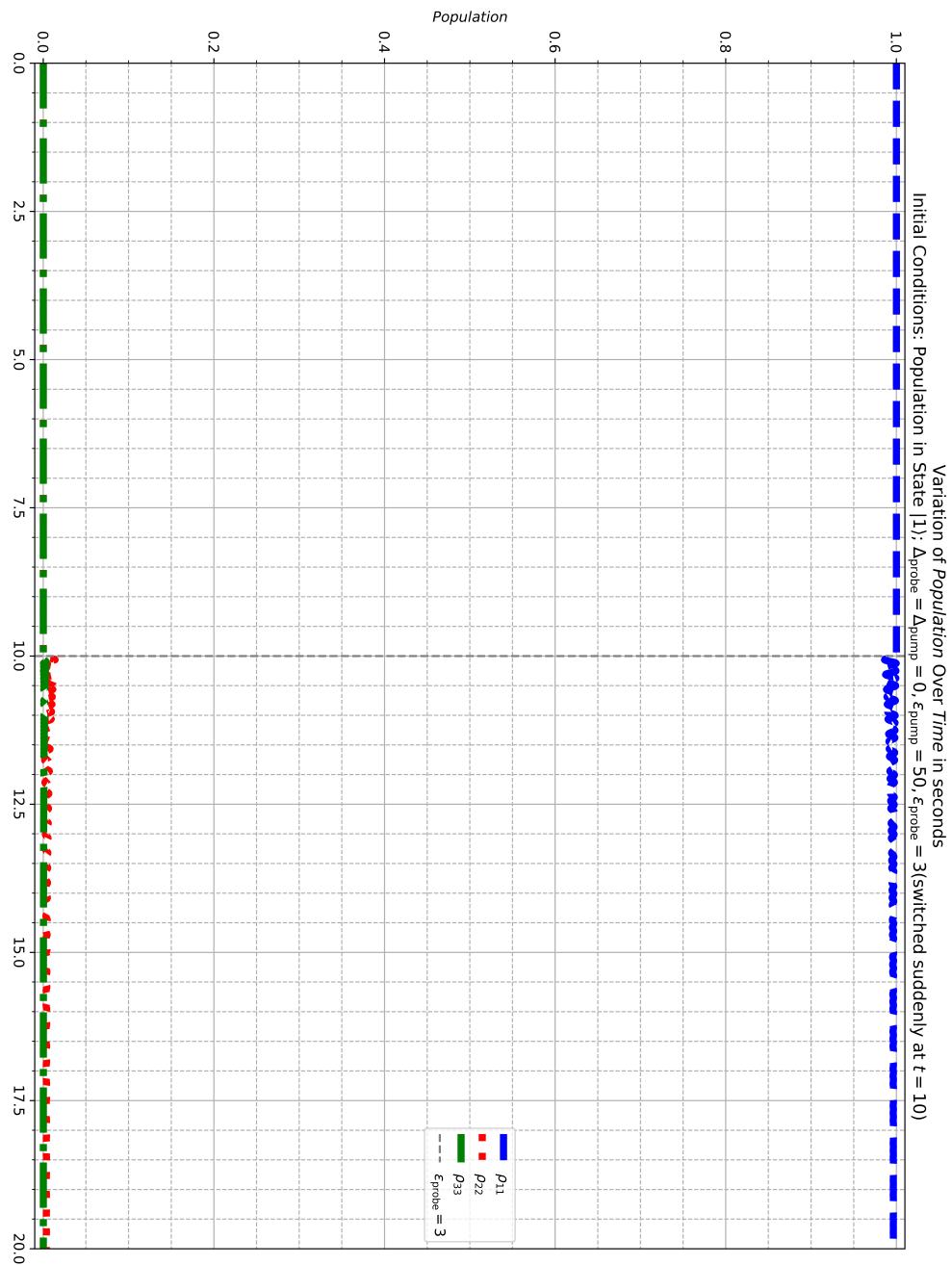
**Fig. 4.** Showing how the *Population* vary with *time* in seconds.



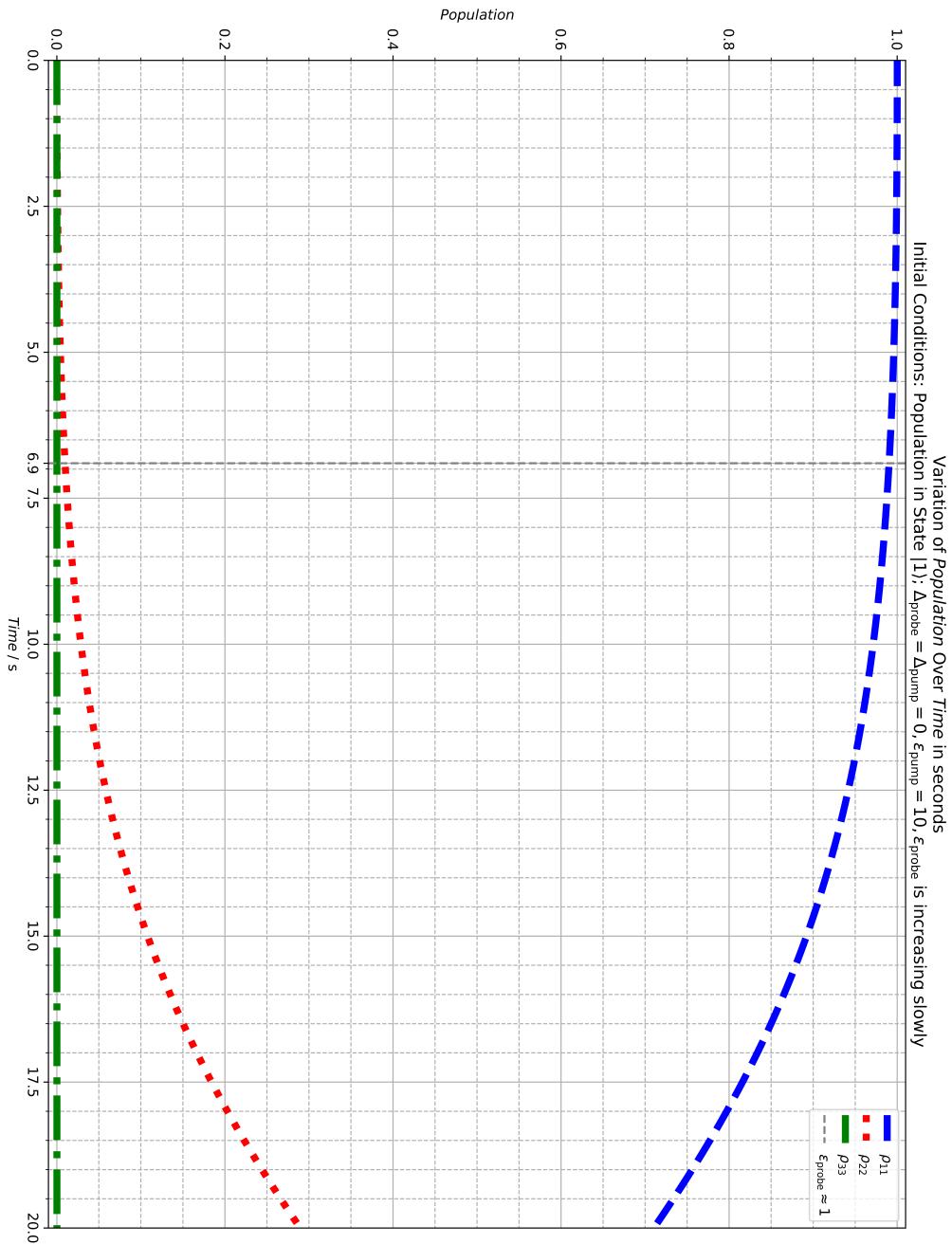
**Fig. 5.** Showing how the *Population* vary with *time* in seconds.



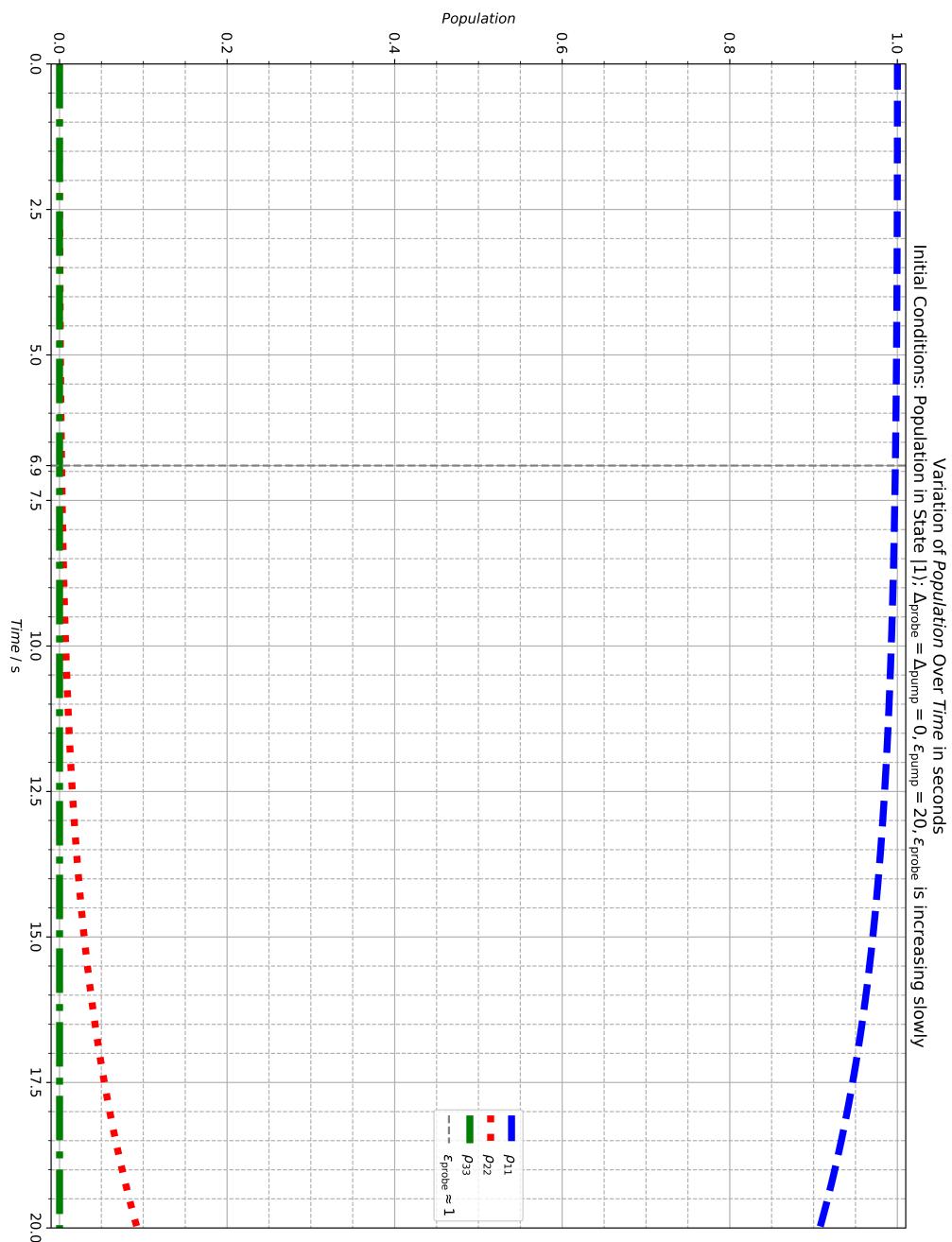
**Fig. 6.** Showing how the *Population* vary with *time* in seconds.



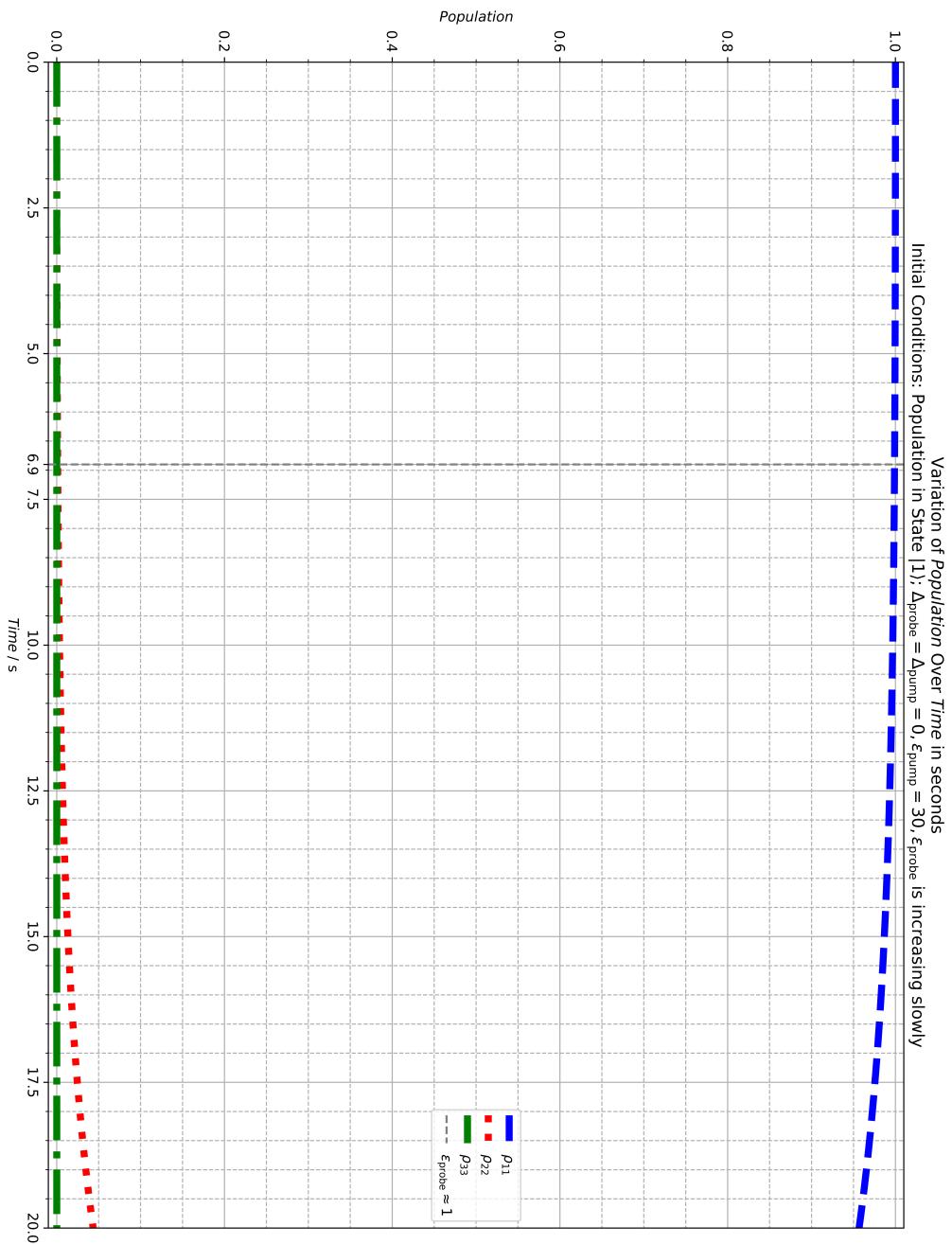
**Fig. 7.** Showing how the *Population* vary with *time* in seconds.



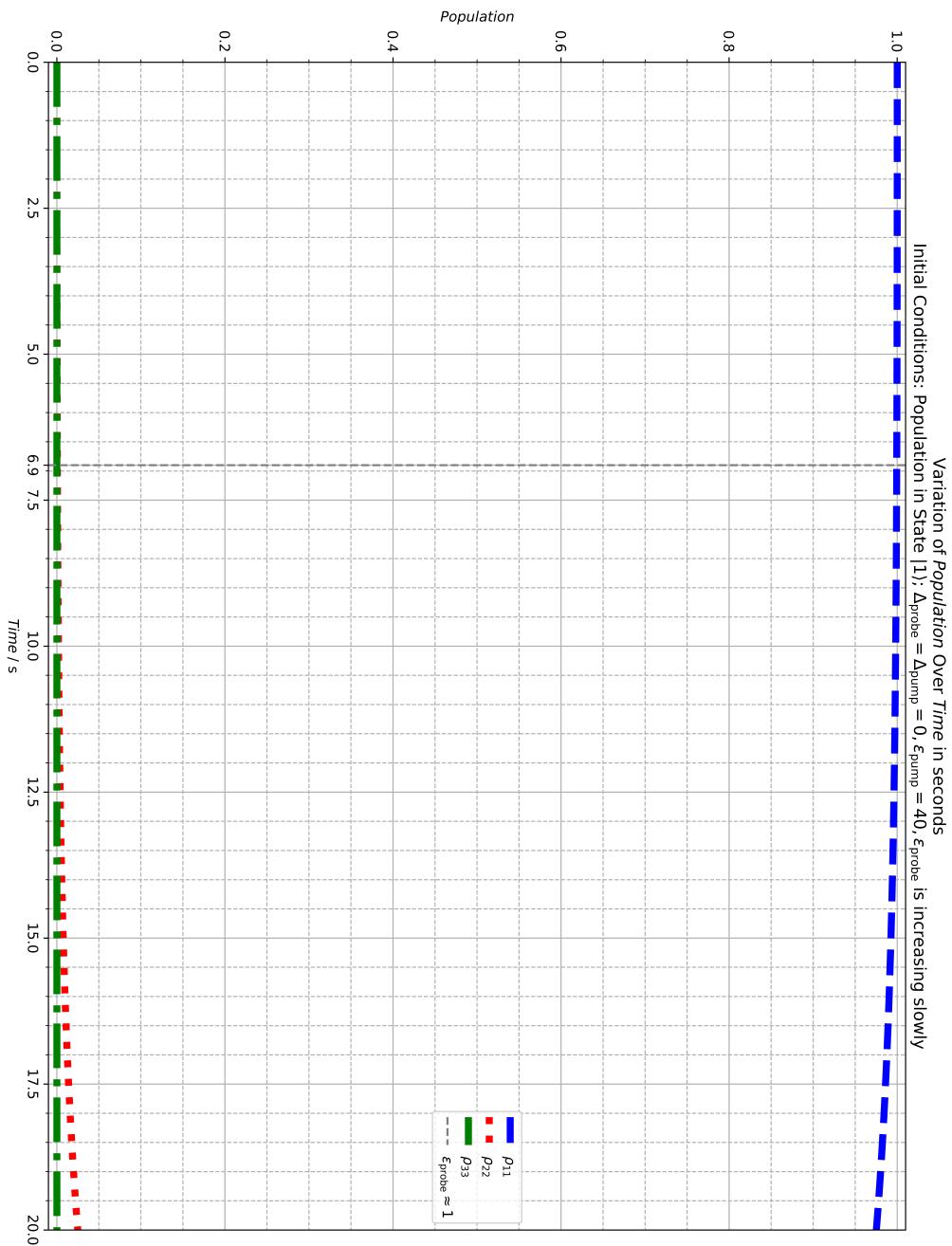
**Fig. 8.** Showing how the *Population* vary with *time* in seconds.



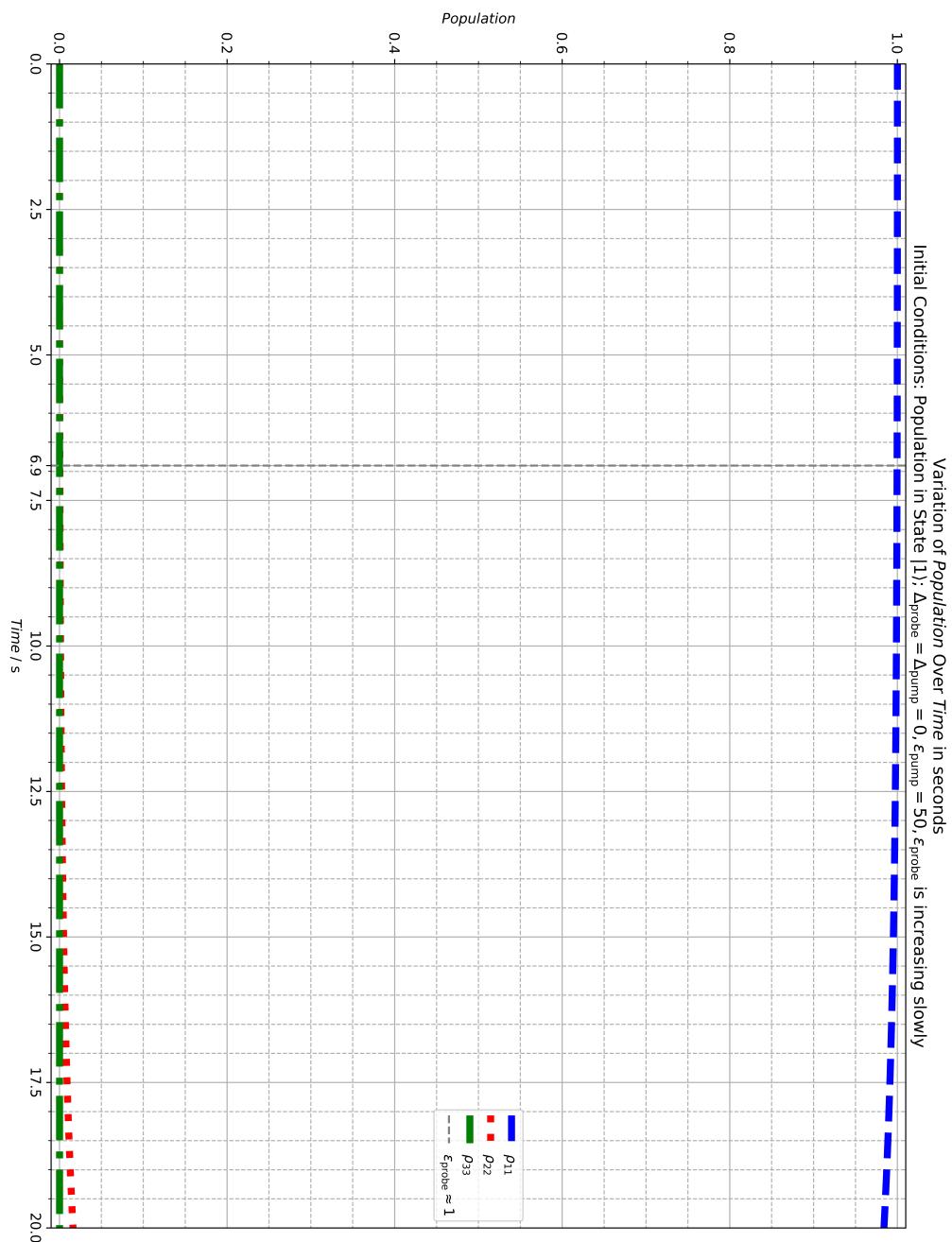
**Fig. 9.** Showing how the *Population* vary with *time* in seconds.



**Fig. 10.** Showing how the *Population* vary with *time* in seconds.



**Fig. 11.** Showing how the *Population* vary with *time* in seconds.



**Fig. 12.** Showing how the *Population* vary with *time* in seconds.

## 5 ANALYSIS

### 5.1 Task 1A

Setting the following substitutions for equation 1:

$$\Omega_1 = \omega_1 - \Delta_{\text{probe}}$$

$$\Omega_2 = \omega_2 - \Delta_{\text{pump}}$$

$$\Omega_3 = \omega_3.$$

$$|j_i\rangle \rightarrow e^{-i\Omega_j t}|j_i\rangle \quad (\text{for } j = 1, 2, 3),$$

$$\hat{H} \rightarrow \hat{H} - \sum_j \hbar\Omega_j |j_i\rangle\langle j_i|.$$

then we get:

$$\hat{H}_0 = \hbar \left( \omega_1 |1\rangle\langle 1| + \omega_2 |2\rangle\langle 2| + \omega_3 |3\rangle\langle 3| \right) \quad (5)$$

$$\hat{H}_{\text{pump}} = \hbar\varepsilon_{\text{pump}} \left( e^{-it(\Omega_3 - \Omega_2 + \Omega_2 - \Omega_3)} |2\rangle\langle 3| + e^{it(\Omega_3 - \Omega_2 - \Omega_3 + \Omega_2)} |3\rangle\langle 2| \right) \quad (6)$$

simplifying,

$$\hat{H}_{\text{pump}} = \hbar\varepsilon_{\text{pump}} \left( |2\rangle\langle 3| + |3\rangle\langle 2| \right) \quad (7)$$

$$\hat{H}_{\text{probe}} = \hbar\varepsilon_{\text{probe}} \left( e^{-it(\Omega_3 - \Omega_1 + \Omega_1 - \Omega_3)} |1\rangle\langle 3| + e^{it(\Omega_3 - \Omega_1 - \Omega_3 + \Omega_1)} |3\rangle\langle 1| \right) \quad (8)$$

$$\hat{H}_{\text{probe}} = \hbar\varepsilon_{\text{probe}} \left( |1\rangle\langle 3| + |3\rangle\langle 1| \right) \quad (9)$$

moreover,

$$\sum_j \hbar\Omega_j |j_i\rangle\langle j_i| = \hbar \left[ \left( \omega_1 - \Delta_{\text{probe}} \right) |1\rangle\langle 1| + \left( \omega_2 - \Delta_{\text{pump}} \right) |2\rangle\langle 2| + \omega_3 |3\rangle\langle 3| \right] \quad (10)$$

Therefore, The new Hamiltonian is given by,

$$\hat{H} - \sum_j \hbar \Omega_j |j_i\rangle\langle j_i| = \hbar \varepsilon_{\text{pump}} (|2\rangle\langle 3| + |3\rangle\langle 2|) + \hbar \varepsilon_{\text{probe}} (|1\rangle\langle 3| + |3\rangle\langle 1|) + \hbar (\Delta_{\text{probe}} |1\rangle\langle 1| + \Delta_{\text{pump}} |2\rangle\langle 2|) \quad (11)$$

## 5.2 Task 2

The resultant Hamiltonian can be easily represented in matrix form, since the outer product is a representation of the row (ket) and the column (bra) in a matrix.

$$\hat{H} = \begin{pmatrix} \hbar \Delta_{\text{probe}} & 0 & \hbar \varepsilon_{\text{probe}} \\ 0 & \hbar \Delta_{\text{pump}} & \hbar \varepsilon_{\text{pump}} \\ \hbar \varepsilon_{\text{probe}} & \hbar \varepsilon_{\text{pump}} & 0 \end{pmatrix}$$

Moreover the density matrix can be defined as,

$$\rho = \begin{pmatrix} \rho_{11} & \rho_{12} & \rho_{13} \\ \rho_{21} & \rho_{22} & \rho_{23} \\ \rho_{31} & \rho_{32} & \rho_{33} \end{pmatrix}$$

Where the diagonal elements represent the population in states 1, 2 and 3 respectively.

## 5.3 Task3

Starting from,

$$\dot{\rho} = \frac{i}{\hbar} [\rho, \hat{H}] + \mathcal{L}[\rho], \quad (12)$$

where

$$\mathcal{L}[\rho] = \gamma \sum_{j=1,2} |j\rangle\langle 3| \rho |3\rangle\langle j| - \frac{1}{2} |3\rangle\langle 3| \rho + \rho |3\rangle\langle 3| \quad (13)$$

The Hamiltonian and density matrix were defined as in Task 2. Now the Lindblad  $\mathcal{L}[\rho]$  was easily found by setting  $\gamma = 1$  and simplifying the expression in equation 13 using Mathematica,

```

1 ket1 = {{1}, {0}, {0}};
2 bra1 = {1, 0, 0};
3
4 ket2 = {{0}, {1}, {0}};
5 bra2 = {0, 1, 0};
6

```

```

7 ket3 = {{0}, {0}, {1}};
8 bra3 = {0, 0, 1};
9
10 L = Dot[ket1, Dot[bra3, Dot[p, Dot[ket3, bra1]]]] + 
11     Dot[ket2, Dot[bra3, Dot[p, Dot[ket3, bra2]]]] - 
12     Dot[ket3, Dot[bra3, p]] - Dot[p, Dot[ket3, bra3]];

```

Thus,

$$L[\rho] = \begin{pmatrix} \rho_{33} & 0 & -\rho_{13} \\ 0 & \rho_{33} & -\rho_{32} \\ -\rho_{31} & -\rho_{32} & -2\rho_{33} \end{pmatrix}$$

Then we proceeded in finding the commutator in equation 12 using Python, along with splitting the expression in real part and imaginary part and solving the differential equation. The complex function was defined as follow,

```

1
2 def combined_differential_equations(current_rho_values, t):
3     real_part = current_rho_values[:9].reshape((3, 3)).astype(float)
4     imag_part = current_rho_values[9:].reshape((3, 3)).astype(float)
5
6     rho = real_part + 1j * imag_part
7
8     L_values = L.subs({
9         rho_11: rho[0, 0], rho_12: rho[0, 1], rho_13: rho[0, 2],
10        rho_21: rho[1, 0], rho_22: rho[1, 1], rho_23: rho[1, 2],
11        rho_31: rho[2, 0], rho_32: rho[2, 1], rho_33: rho[2, 2]
12    })
13
14     dp = np.matmul(rho, H) - np.matmul(H, rho) + L_values
15     dp_real = sp.re(sp.expand(dp))
16     dp_imag = sp.im(sp.expand(dp))
17
18     dp_real_values = np.array(dp_real).astype(float).flatten()
19     dp_imag_values = np.array(dp_imag).astype(float).flatten()
20
21     return np.concatenate((dp_real_values, dp_imag_values)).tolist()

```

Furthermore, this function was solved over a period of time  $t$  and reshaped into a matrix form, from which the diagonal elements were extracted.

```

1
2 # Initial conditions
3 t = np.linspace(0, 20, 2000)

```

```

4 y0 = np.array([0, 0, 0, 0, 1, 0, 0, 0, 0] + [0, 0, 0, 0, 0, 0, 0, 0, 0])
5
6 # Solving complex differential equation
7 Y = odeint(combined_differential_equations, y0, t)
8
9 # Extracting real and imaginary parts from the results
10 real_part = Y[:, :9].reshape((len(t), 3, 3))
11 imag_part = Y[:, 9:].reshape((len(t), 3, 3))
12
13 rho_matrices = real_part + 1j * imag_part
14
15 rho_11_values_all = rho_matrices[:, 0, 0]
16 rho_22_values_all = rho_matrices[:, 1, 1]
17 rho_33_values_all = rho_matrices[:, 2, 2]

```

The graphs were then plotted, showing how the variation of the population varies from state to state,

```

1
2 # Plotting, labelling.
3 plt.figure(figsize=(15, 11))
4 plt.ylim(-0.01, 1.01)
5 plt.xlim(0, 20)
6 plt.plot(t, rho_11_values_all, 'b--', label=r'$\rho_{11}$', linewidth=5)
7 plt.plot(t, rho_22_values_all, 'r:', label=r'$\rho_{22}$', linewidth=5)
8 plt.plot(t, rho_33_values_all, 'g-.', label=r'$\rho_{33}$', linewidth=5)
9
10 plt.minorticks_on()
11 plt.grid(visible=True, which='major', linestyle='--')
12 plt.grid(visible=True, which='minor', linestyle='---')
13
14 plt.title('Variation of $Population$ Over $Time$ in seconds \n Initial Conditions:
    $Population$ in State $|2\rangle; $\Delta_{\mathrm{probe}} = \Delta_{\mathrm{pump}} = \epsilon_{\mathrm{pump}} = 0 , \epsilon_{\mathrm{probe}} = 0.5$')
15
16 plt.xlabel(r'$Time$/ s')
17 plt.ylabel(r'$Population$')
18 plt.legend()
19 plt.savefig('Graph1.png', dpi=1000)
20 plt.show()

```

This process was repeated for different values and behaviour of the pump field and probe field.

## 6 DISCUSSION

The study presents simulations conducted to examine the dynamics of a 3-level ( $\Lambda$ ) lambda system under various field conditions. The results obtained from these simulations closely align with the theoretical predictions based on quantum mechanics, specifically on the strict selection rules that govern atomic transitions. The plots derived from our tests consistently showed trends and behaviours that practically were comparable to the anticipated results. Furthermore the agreement between the simulated results and the expected theoretical predictions highlights the reliability and precision of our computational model in accurately representing the complex behaviours of the system. This not only supports the adherence to fundamental rules in quantum physics, but also strengthens the reliability of the computational approach in recreating complicated atomic transitions in the  $\Lambda$  system.

In order to enhance the study and eliminate limitations, it is suggested to broaden the investigation by integrating a more authentic model that encompasses the influences of decoherence ( $\Delta_{\text{pump}} = \Delta_{\text{probe}} \neq 0$ ) and dephasing mechanisms within the lambda system. By incorporating these elements into the simulation, the research could further explore the behaviour of the system under more realistic circumstances, providing insights into the influence of environmental factors on the phenomena of electromagnetically induced transparency (EIT) (Finkelstein et al., 2023). Gaining insight into these decoherence effects could reveal the strength of EIT against many sources of noise or disruptions, thereby facilitating practical implementations in quantum information processing or quantum computing, where preserving coherence is crucial. Expanding the simulation will not only increase the range of the study but also provide essential insights into the practicality and reliability of using EIT based devices in everyday circumstances.

An interesting phenomenon was shown in the experiment. The probe field became transparent when a strong pump field was applied. This interesting result showed an expected correlation between the probe field's behaviour and the pump field's strength, known as electromagnetically induced transparency. Essentially, when the pump field was strong enough, it acted as a strong field (control field), affecting the state 3 'level' by splitting into more bands that are now incoherent with the probe field (Finkelstein et al., 2023). Resulting in the probe field becoming transparent to the medium. Moreover, the experimental findings repeatedly showed that there are no transitions from state 1 to state 2. This lack of presence confirmed the vital concept that changes between these states were not possible because of the constraints set by the selection rules, i.e.  $l = \pm 1$  and  $m_l = 0$  or  $\pm 1$ .

The experiment on pump-probe interaction in the ( $\Lambda$ ) lambda system yielded promising results that align with theoretical predictions. Strong pump fields can induce transparency to the probe field, as suggested by electromagnetically induced transparency theory (EIT). Interestingly, the strong control field targeting the state 2 and 3 transition effectively stops light absorption from the probe field in the state 1 and 3 transition, as predicted. To determine the optimal circumstances for transparency, one can enhance the experiment by exploring variations in the intensity and duration of the pump field. By altering the initial atomic states or field frequencies, one can also investigate the dynamic changes of the system.

## REFERENCES

- Finkelstein, R Bali, S Firstenberg, O & Novikova, I (2023) A practical guide to electromagnetically induced transparency in atomic vapor. *New Journal of Physics*, 25(3) 035001. <https://doi.org/10.1088/1367-2630/acbc40>
- Gallagher, R & Ingram, P (2015) *Complete chemistry for cambridge igcse®*. Oxford University Press-Children.
- Mann, A (2022) *What is quantum mechanics?* <https://www.livescience.com/33816-quantum-mechanics-explanation.html> Accessed on December 13, 2023.
- Morin, D (2008) Introduction to quantum mechanics. *Ch, 10*, 1–20.
- Romforn, L (2023) *What is a quantum mechanical model?* <https://academichelp.net/stem/physics/quantum-mechanical-model.html> Accessed on December 12, 2023.
- Sakurai, J J & Commins, E D (1995) Modern quantum mechanics, revised edition.
- Silberberg, M S Amateis, P Venkateswaran, R & Chen, L (2006) *Chemistry: The molecular nature of matter and change* (Vol. 4) McGraw-Hill New York.
- Svelto, O Hanna, D C et al. (2010) *Principles of lasers* (Vol. 1) Springer.

# APPENDIX

## 1 CODE LISTINGS

### 1.1 Python Code

```

1 import sympy as sp
2 import numpy as np
3 from scipy.integrate import odeint
4 import matplotlib.pyplot as plt
5 from scipy.linalg import eigh
6
7 # Define symbolic variables for Delta_probe, Delta_pump, epsilon_probe, epsilon_pump, rho_ij
8 Delta_probe, Delta_pump, epsilon_probe, epsilon_pump = sp.symbols('Delta_probe Delta_pump
9     epsilon_probe epsilon_pump')
10 rho_11, rho_12, rho_13, rho_21, rho_22, rho_23, rho_31, rho_32, rho_33 = sp.symbols('rho_11
11     rho_12 rho_13 rho_21 rho_22 rho_23 rho_31 rho_32 rho_33')
12
13 # Defining imaginary number
14 j = sp.I
15
16 # Define specific values
17 values = {
18     Delta_probe: 0,
19     Delta_pump: 0,
20     epsilon_probe: 0.5,
21     epsilon_pump: 0
22 }
23
24 # Defining the Hamiltonian, Density matrix and lindblad matrix.
25 H = sp.Matrix([[Delta_probe * j, 0, epsilon_probe * j],
26                 [0, Delta_pump * j, epsilon_pump * j],
27                 [epsilon_probe * j, epsilon_pump * j, 0]]))
28
29 rho = sp.Matrix([[rho_11, rho_12, rho_13],
30                  [rho_21, rho_22, rho_23],
31                  [rho_31, rho_32, rho_33]])
32
33 L = sp.Matrix([[rho_33, 0, -1 * rho_13],
34                 [0, rho_33, -1 * rho_23],
35                 [-1 * rho_31, -1 * rho_32, -2 * rho_33]])
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
748
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
778
779
779
780
781
782
783
784
785
786
787
788
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
817
818
819
819
820
821
822
823
824
825
826
827
828
828
829
829
830
831
832
833
834
835
836
837
838
838
839
839
840
841
842
843
844
845
846
847
847
848
849
849
850
851
852
853
854
855
856
857
858
858
859
859
860
861
862
863
864
865
866
867
867
868
868
869
869
870
871
872
873
874
875
875
876
876
877
877
878
878
879
879
880
881
882
883
884
885
885
886
886
887
887
888
888
889
889
890
891
892
893
894
894
895
895
896
896
897
897
898
898
899
899
900
901
902
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
```

```

35 H = H.subs(values)
36 L = L.subs(values)
37 rho = rho.subs(values)
38
39 # Defining the complex differential equation
40 def combined_differential_equations(current_rho_values, t):
41     real_part = current_rho_values[:9].reshape((3, 3)).astype(float)
42     imag_part = current_rho_values[9:].reshape((3, 3)).astype(float)
43
44     rho = real_part + 1j * imag_part
45
46     L_values = L.subs({
47         rho_11: rho[0, 0], rho_12: rho[0, 1], rho_13: rho[0, 2],
48         rho_21: rho[1, 0], rho_22: rho[1, 1], rho_23: rho[1, 2],
49         rho_31: rho[2, 0], rho_32: rho[2, 1], rho_33: rho[2, 2]
50     })
51
52     dp = np.matmul(rho, H) - np.matmul(H, rho) + L_values
53     dp_real = sp.re(sp.expand(dp))
54     dp_imag = sp.im(sp.expand(dp))
55
56     dp_real_values = np.array(dp_real).astype(float).flatten()
57     dp_imag_values = np.array(dp_imag).astype(float).flatten()
58
59     return np.concatenate((dp_real_values, dp_imag_values)).tolist()
60
61
62 # Initial conditions
63 t = np.linspace(0, 20, 2000)
64 y0 = np.array([0, 0, 0, 0, 1, 0, 0, 0, 0] + [0, 0, 0, 0, 0, 0, 0, 0, 0])
65
66 # Solving complex differential equation
67 Y = odeint(combined_differential_equations, y0, t)
68
69 # Extracting real and imaginary parts from the results
70 real_part = Y[:, :9].reshape((len(t), 3, 3))
71 imag_part = Y[:, 9:].reshape((len(t), 3, 3))
72
73 rho_matrices = real_part + 1j * imag_part
74
75 rho_11_values_all = rho_matrices[:, 0, 0]

```

```
76 rho_22_values_all = rho_matrices[:, 1, 1]
77 rho_33_values_all = rho_matrices[:, 2, 2]
78
79
80 # Plotting, labelling.
81 plt.figure(figsize=(15, 11))
82 plt.ylim(-0.01, 1.01)
83 plt.xlim(0,20)
84 plt.plot(t, rho_11_values_all, 'b--', label=r'$\rho_{11}$', linewidth=5)
85 plt.plot(t, rho_22_values_all, 'r:', label=r'$\rho_{22}$', linewidth=5)
86 plt.plot(t, rho_33_values_all, 'g-.', label=r'$\rho_{33}$', linewidth=5)
87
88 plt.minorticks_on()
89 plt.grid(visible=True, which='major', linestyle='-')
90 plt.grid(visible=True, which='minor', linestyle='--')
91
92 plt.title('Variation of $Population$ Over $Time$ in seconds \n Initial Conditions:
93     $Population$ in State $|2\rangle; $\Delta_{\mathrm{probe}} = \Delta_{\mathrm{pump}} = \
94     \epsilon_{\mathrm{probe}} = 0 , \epsilon_{\mathrm{pump}} = 0.5$')
95 plt.xlabel(r'$Time$/ s ')
96 plt.ylabel(r'$Population$')
97 plt.legend()
98 plt.savefig('Graph1.png', dpi=1000)
99 plt.show()
100
101
102
103 values = {
104     Delta_probe: 0,
105     Delta_pump: 0,
106     epsilon_probe: 1,
107     epsilon_pump: 0
108 }
109
110 H = sp.Matrix([[Delta_probe * j, 0, epsilon_probe * j],
111                 [0, Delta_pump * j, epsilon_pump * j],
112                 [epsilon_probe * j, epsilon_pump * j, 0]])
113
114 rho = sp.Matrix([[rho_11, rho_12, rho_13],
```

```

115         [rho_21, rho_22, rho_23],
116         [rho_31, rho_32, rho_33]])
117
118 L = sp.Matrix([[rho_33, 0, -1 * rho_13],
119                 [0, rho_33, -1 * rho_23],
120                 [-1 * rho_31, -1 * rho_32, -2 * rho_33]])
121
122 H = H.subs(values)
123 L = L.subs(values)
124 rho = rho.subs(values)
125
126 def combined_differential_equations(current_rho_values, t):
127     real_part = current_rho_values[:9].reshape((3, 3)).astype(float)
128     imag_part = current_rho_values[9:].reshape((3, 3)).astype(float)
129
130     rho = real_part + 1j * imag_part
131
132     L_values = L.subs({
133         rho_11: rho[0, 0], rho_12: rho[0, 1], rho_13: rho[0, 2],
134         rho_21: rho[1, 0], rho_22: rho[1, 1], rho_23: rho[1, 2],
135         rho_31: rho[2, 0], rho_32: rho[2, 1], rho_33: rho[2, 2]
136     })
137
138 dp = np.matmul(rho, H) - np.matmul(H, rho) + L_values
139 dp_real = sp.re(sp.expand(dp))
140 dp_imag = sp.im(sp.expand(dp))
141
142 dp_real_values = np.array(dp_real).astype(float).flatten()
143 dp_imag_values = np.array(dp_imag).astype(float).flatten()
144
145 return np.concatenate((dp_real_values, dp_imag_values)).tolist()
146
147
148 # Initial conditions
149 t = np.linspace(0, 20, 2000)
150 y0 = np.array([1, 0, 0, 0, 0, 0, 0, 0, 0] + [0, 0, 0, 0, 0, 0, 0, 0, 0])
151 Y = odeint(combined_differential_equations, y0, t)
152
153
154 real_part = Y[:, :9].reshape((len(t), 3, 3))
155 imag_part = Y[:, 9:].reshape((len(t), 3, 3))

```

```
156  
157 rho_matrices = real_part + 1j * imag_part  
158  
159 rho_11_values_all = rho_matrices[:, 0, 0]  
160 rho_22_values_all = rho_matrices[:, 1, 1]  
161 rho_33_values_all = rho_matrices[:, 2, 2]  
162  
163  
164 # Plotting  
165 plt.figure(figsize=(15, 11))  
166 plt.ylim(-0.01, 1.01)  
167 plt.xlim(0, 20)  
168 plt.plot(t, rho_11_values_all, 'b--', label=r'$\rho_{11}$', linewidth=5)  
169 plt.plot(t, rho_22_values_all, 'r:', label=r'$\rho_{22}$', linewidth=5)  
170 plt.plot(t, rho_33_values_all, 'g-.', label=r'$\rho_{33}$', linewidth=5)  
171  
172 plt.minorticks_on()  
173 plt.grid(visible=True, which='major', linestyle='--')  
174 plt.grid(visible=True, which='minor', linestyle='---')  
175  
176 plt.title('Variation of $Population$ Over $Time$ in seconds \nInitial Conditions: Population  
in State $|1\rangle; $\Delta_{\mathrm{probe}} = \Delta_{\mathrm{pump}} = \epsilon_{\mathrm{probe}} = 0$, $\epsilon_{\mathrm{pump}} = 1$')  
177 plt.xlabel(r'$Time$/ s')  
178 plt.ylabel(r'$Population$')  
179 plt.legend()  
180 plt.savefig('Graph2.png', dpi=1000)  
181 plt.show()  
182  
183  
184  
185 # Question 4i)  
186  
187  
188  
189 H = sp.Matrix([[Delta_probe * j, 0, epsilon_probe * j],  
190 [0, Delta_pump * j, epsilon_pump * j],  
191 [epsilon_probe * j, epsilon_pump * j, 0]])  
192  
193 rho = sp.Matrix([[rho_11, rho_12, rho_13],  
194 [rho_21, rho_22, rho_23],
```

```

195             [rho_31, rho_32, rho_33]])
196
197 L = sp.Matrix([[rho_33, 0, -1 * rho_13],
198                 [0, rho_33, -1 * rho_23],
199                 [-1 * rho_31, -1 * rho_32, -2 * rho_33]])
200
201
202 L = L.subs(values)
203 rho = rho.subs(values)
204
205
206
207
208 def combined_differential_equations(current_rho_values, t):
209     real_part = current_rho_values[:9].reshape((3, 3)).astype(float)
210     imag_part = current_rho_values[9:].reshape((3, 3)).astype(float)
211
212     rho = real_part + 1j * imag_part
213
214     L_values = L.subs({
215         rho_11: rho[0, 0], rho_12: rho[0, 1], rho_13: rho[0, 2],
216         rho_21: rho[1, 0], rho_22: rho[1, 1], rho_23: rho[1, 2],
217         rho_31: rho[2, 0], rho_32: rho[2, 1], rho_33: rho[2, 2]
218     })
219
220     if t > 10:
221
222         epsilon_probe_values = 3
223     else:
224         epsilon_probe_values = 0
225
226     H_values = H.subs({
227         Delta_probe: 0,
228         Delta_pump: 0,
229         epsilon_probe: epsilon_probe_values,
230         epsilon_pump: 10
231     })
232
233 dp = np.matmul(rho, H_values) - np.matmul(H_values, rho) + L_values
234 dp_real = sp.re(sp.expand(dp))
235 dp_imag = sp.im(sp.expand(dp))

```

```

236
237     dp_real_values = np.array(dp_real).astype(float).flatten()
238     dp_imag_values = np.array(dp_imag).astype(float).flatten()
239
240     return np.concatenate((dp_real_values, dp_imag_values)).tolist()
241
242
243 # Initial conditions
244 t = np.linspace(0, 20, 2000)
245 y0 = np.array([1, 0, 0, 0, 0, 0, 0, 0, 0] + [0, 0, 0, 0, 0, 0, 0, 0, 0])
246
247 Y = odeint(combined_differential_equations, y0, t)
248
249
250 real_part = Y[:, :9].reshape((len(t), 3, 3))
251 imag_part = Y[:, 9:].reshape((len(t), 3, 3))
252
253 rho_matrices = real_part + 1j * imag_part
254
255 rho_11_values_all = rho_matrices[:, 0, 0]
256 rho_22_values_all = rho_matrices[:, 1, 1]
257 rho_33_values_all = rho_matrices[:, 2, 2]
258
259 # Plotting
260 plt.figure(figsize=(15, 11))
261 plt.ylim(-0.01, 1.01)
262 plt.xlim(0, 20)
263 plt.plot(t, rho_11_values_all, 'b--', label=r'$\rho_{11}$', linewidth=5)
264 plt.plot(t, rho_22_values_all, 'r:', label=r'$\rho_{22}$', linewidth=5)
265 plt.plot(t, rho_33_values_all, 'g-.', label=r'$\rho_{33}$', linewidth=5)
266
267 plt.minorticks_on()
268 plt.grid(visible=True, which='major', linestyle='--')
269 plt.grid(visible=True, which='minor', linestyle='---')
270
271 plt.title('Variation of $Population$ Over $Time$ in seconds\nInitial Conditions: Population
    in State $|1\rangle; \Delta_{\mathrm{probe}} = \Delta_{\mathrm{pump}} = 0, \epsilon_{\mathrm{probe}} = 10, \epsilon_{\mathrm{pump}} = 3$ (switched suddenly at $t=10$)')
272 plt.xlabel(r'$Time$/ s')
273 plt.ylabel(r'$Population$')
274 plt.axvline(x=10, color='grey', linestyle='--', label='$\epsilon_{\mathrm{probe}} = 3$')

```

```
275 plt.legend()
276 plt.savefig('Graph3.png', dpi=1000)
277 plt.show()
278
279
280 # Question 4ii)
281
282
283 H = sp.Matrix([[Delta_probe * j, 0, epsilon_probe * j],
284                 [0, Delta_pump * j, epsilon_pump * j],
285                 [epsilon_probe * j, epsilon_pump * j, 0]]])
286
287 rho = sp.Matrix([[rho_11, rho_12, rho_13],
288                   [rho_21, rho_22, rho_23],
289                   [rho_31, rho_32, rho_33]])
290
291 L = sp.Matrix([[rho_33, 0, -1 * rho_13],
292                 [0, rho_33, -1 * rho_23],
293                 [-1 * rho_31, -1 * rho_32, -2 * rho_33]])
294
295 L = L.subs(values)
296 rho = rho.subs(values)
297
298 def expo(t, k):
299     return np.exp(k*t)-1
300
301
302 def combined_differential_equations(current_rho_values, t):
303     real_part = current_rho_values[:9].reshape((3, 3)).astype(float)
304     imag_part = current_rho_values[9:].reshape((3, 3)).astype(float)
305
306     rho = real_part + 1j * imag_part
307
308     L_values = L.subs({
309         rho_11: rho[0, 0], rho_12: rho[0, 1], rho_13: rho[0, 2],
310         rho_21: rho[1, 0], rho_22: rho[1, 1], rho_23: rho[1, 2],
311         rho_31: rho[2, 0], rho_32: rho[2, 1], rho_33: rho[2, 2]
312     })
313
314
315     epsilon_probe_values = expo(t, k=0.1)
```

```

316
317     H_values = H.subs({
318         Delta_probe: 0,
319         Delta_pump: 0,
320         epsilon_probe: epsilon_probe_values,
321         epsilon_pump: 10
322     })
323
324     dp = np.matmul(rho, H_values) - np.matmul(H_values, rho) + L_values
325     dp_real = sp.re(sp.expand(dp))
326     dp_imag = sp.im(sp.expand(dp))
327
328     dp_real_values = np.array(dp_real).astype(float).flatten()
329     dp_imag_values = np.array(dp_imag).astype(float).flatten()
330
331     return np.concatenate((dp_real_values, dp_imag_values)).tolist()
332
333
334 # Initial conditions
335 t = np.linspace(0, 20, 2000)
336 y0 = np.array([1, 0, 0, 0, 0, 0, 0, 0, 0] + [0, 0, 0, 0, 0, 0, 0, 0, 0])
337
338 Y = odeint(combined_differential_equations, y0, t)
339
340 real_part = Y[:, :9].reshape((len(t), 3, 3))
341 imag_part = Y[:, 9: ].reshape((len(t), 3, 3))
342
343 rho_matrices = real_part + 1j * imag_part
344
345 rho_11_values_all = rho_matrices[:, 0, 0]
346 rho_22_values_all = rho_matrices[:, 1, 1]
347 rho_33_values_all = rho_matrices[:, 2, 2]
348
349 # Plotting
350 plt.figure(figsize=(15, 11))
351 plt.ylim(-0.01, 1.01)
352 plt.xlim(0, 20)
353 plt.plot(t, rho_11_values_all, 'b--', label=r'$\rho_{11}$', linewidth=5)
354 plt.plot(t, rho_22_values_all, 'r:', label=r'$\rho_{22}$', linewidth=5)
355 plt.plot(t, rho_33_values_all, 'g-.', label=r'$\rho_{33}$', linewidth=5)
356

```

```
357 plt.minorticks_on()
358 plt.grid(visible=True, which='major', linestyle='--')
359 plt.grid(visible=True, which='minor', linestyle='--')
360
361 plt.title('Variation of $Population$ Over $Time$ in seconds \nInitial Conditions: Population
            in State $|1\rangle; \Delta_{\mathrm{probe}} = \Delta_{\mathrm{pump}} = 0 , \
            \epsilon_{\mathrm{probe}} = 10 $, $\epsilon_{\mathrm{pump}}$ (is increasing slowly),')
362 plt.xlabel(r'$Time$; $ s $')
363 plt.ylabel(r'$Population$')
364
365 plt.axvline(x=6.9, color='grey', linestyle='--', label=r'$\epsilon_{\mathrm{probe}} \approx 1$')
366 plt.legend()
367 plt.xticks(ticks=[*plt.xticks()[0], 6.9])
368 plt.savefig('Graph4.png', dpi=1000)
369 plt.show()
370
371
372 # Creating Gifs for question 4i and 4ii
373
374 import sympy as sp
375 import numpy as np
376 from scipy.integrate import odeint
377 import matplotlib.pyplot as plt
378 from scipy.linalg import eigh
379 import imageio
380
381 # Define symbolic variables for Delta_probe, Delta_pump, epsilon_probe, epsilon_pump, rho_ij
382 Delta_probe, Delta_pump, epsilon_probe, epsilon_pump = sp.symbols('Delta_probe Delta_pump
            epsilon_probe epsilon_pump')
383 rho_11, rho_12, rho_13, rho_21, rho_22, rho_23, rho_31, rho_32, rho_33 = sp.symbols('rho_11
            rho_12 rho_13 rho_21 rho_22 rho_23 rho_31 rho_32 rho_33')
384
385 # Defining imaginary number
386 j = sp.I
387
388 # Define specific values
389 values = {
390     Delta_probe: 0,
391     Delta_pump: 0,
392 }
```

```

393
394 H = sp.Matrix([[Delta_probe * j, 0, epsilon_probe * j],
395                 [0, Delta_pump * j, epsilon_pump * j],
396                 [epsilon_probe * j, epsilon_pump * j, 0]]])
397
398 rho = sp.Matrix([[rho_11, rho_12, rho_13],
399                   [rho_21, rho_22, rho_23],
400                   [rho_31, rho_32, rho_33]])
401
402 L = sp.Matrix([[rho_33, 0, -1 * rho_13],
403                 [0, rho_33, -1 * rho_23],
404                 [-1 * rho_31, -1 * rho_32, -2 * rho_33]])
405
406
407 L = L.subs(values)
408 rho = rho.subs(values)
409
410
411 # Generating plots for various epsilon_pump values
412 epsilon_pump_values = [10, 20, 30, 40, 50] # Replace with desired epsilon_pump values
413 images = []
414
415 for epsilon_pump_value in epsilon_pump_values:
416     def combined_differential_equations(current_rho_values, t):
417         real_part = current_rho_values[:9].reshape((3, 3)).astype(float)
418         imag_part = current_rho_values[9:].reshape((3, 3)).astype(float)
419
420         rho = real_part + 1j * imag_part
421
422         L_values = L.subs({
423             rho_11: rho[0, 0], rho_12: rho[0, 1], rho_13: rho[0, 2],
424             rho_21: rho[1, 0], rho_22: rho[1, 1], rho_23: rho[1, 2],
425             rho_31: rho[2, 0], rho_32: rho[2, 1], rho_33: rho[2, 2]
426         })
427
428         if t > 10:
429
430             epsilon_probe_values = 3
431         else:
432             epsilon_probe_values = 0
433

```

```

434     H_values = H.subs({
435         Delta_probe: 0,
436         Delta_pump: 0,
437         epsilon_probe: epsilon_probe_values,
438         epsilon_pump: epsilon_pump_values
439     })
440
441     dp = np.matmul(rho, H_values) - np.matmul(H_values, rho) + L_values
442     dp_real = sp.re(sp.expand(dp))
443     dp_imag = sp.im(sp.expand(dp))
444
445     # Evaluate symbolic expressions with numerical values
446     dp_real_values = np.array(dp_real.subs({Delta_probe: 0, Delta_pump: 0, epsilon_probe
447 : 3, epsilon_pump: epsilon_pump_value})).astype(float).flatten()
448     dp_imag_values = np.array(dp_imag.subs({Delta_probe: 0, Delta_pump: 0, epsilon_probe
449 : 3, epsilon_pump: epsilon_pump_value})).astype(float).flatten()
450
451
452     # Initial conditions
453     t = np.linspace(0, 20, 2000)
454     y0 = np.array([1, 0, 0, 0, 0, 0, 0, 0, 0] + [0, 0, 0, 0, 0, 0, 0, 0, 0])
455
456     Y = odeint(combined_differential_equations, y0, t)
457
458
459     real_part = Y[:, :9].reshape((len(t), 3, 3))
460     imag_part = Y[:, 9: ].reshape((len(t), 3, 3))
461
462     rho_matrices = real_part + 1j * imag_part
463
464     rho_11_values_all = rho_matrices[:, 0, 0]
465     rho_22_values_all = rho_matrices[:, 1, 1]
466     rho_33_values_all = rho_matrices[:, 2, 2]
467
468     # Plotting
469     plt.figure(figsize=(15, 11))
470     plt.ylim(-0.01, 1.01)
471     plt.xlim(0, 20)
472     plt.plot(t, rho_11_values_all, 'b--', label=r'$\rho_{11}$', linewidth=5)

```

```

473 plt.plot(t, rho_22_values_all, 'r:', label=r'$\rho_{22}$', linewidth=5)
474 plt.plot(t, rho_33_values_all, 'g-.', label=r'$\rho_{33}$', linewidth=5)
475
476 plt.minorticks_on()
477 plt.grid(visible=True, which='major', linestyle='--')
478 plt.grid(visible=True, which='minor', linestyle='---')
479
480 plt.title(f'Variation of $Population$ Over $Time$ in seconds\nInitial Conditions:
481 Population in State $|1\rangle$; $\Delta_{\{\mathrm{probe}\}} = \Delta_{\{\mathrm{pump}\}} = 0$, $\epsilon_{\{\mathrm{pump}\}} = \epsilon_{\mathrm{pump\_value}}$, $\epsilon_{\{\mathrm{probe}\}} = 3$ (switched suddenly at $t=10$)')
482
483 plt.ylabel(r'$Population$')
484 plt.axvline(x=10, color='grey', linestyle='--', label='$\epsilon_{\{\mathrm{probe}\}} = 3$')
485 plt.legend()
486 plt.savefig(f'Graph 3_epsilon_pump_{epsilon_pump_value}.png', dpi = 1000)
487 plt.close()
488 images.append(imageio.imread(f'Graph 3_epsilon_pump_{epsilon_pump_value}.png'))
489
490 # Saving the images as a GIF
491 imageio.mimsave('graph_3_epsilon_pump_variation.gif', images, duration=3)
492
493 # Question 4ii GIF
494
495 # Define symbolic variables for Delta_probe, Delta_pump, epsilon_probe, epsilon_pump, rho_ij
496 Delta_probe, Delta_pump, epsilon_probe, epsilon_pump = sp.symbols('Delta_probe Delta_pump
497 epsilon_probe epsilon_pump')
498 rho_11, rho_12, rho_13, rho_21, rho_22, rho_23, rho_31, rho_32, rho_33 = sp.symbols('rho_11
499 rho_12 rho_13 rho_21 rho_22 rho_23 rho_31 rho_32 rho_33')
500
501 # Defining imaginary number
502 j = sp.I
503
504 # Define specific values
505 values = {
506     Delta_probe: 0,
507     Delta_pump: 0,
508 }
509
510 H = sp.Matrix([[Delta_probe * j, 0, epsilon_probe * j],
511                 [0, Delta_pump * j, epsilon_pump * j],
512                 [epsilon_probe * j, epsilon_pump * j, 0]])

```

```

509
510 rho = sp.Matrix([[rho_11, rho_12, rho_13],
511                   [rho_21, rho_22, rho_23],
512                   [rho_31, rho_32, rho_33]]))
513
514 L = sp.Matrix([[rho_33, 0, -1 * rho_13],
515                  [0, rho_33, -1 * rho_23],
516                  [-1 * rho_31, -1 * rho_32, -2 * rho_33]]))
517
518
519 L = L.subs(values)
520 rho = rho.subs(values)
521
522
523 # Generating plots for various pump field values
524 epsilon_pump_values = [10, 20, 30, 40, 50] # Replace with desired epsilon_pump values
525 images = []
526
527 def expo(t, k):
528     return np.exp(k*t)-1
529
530 for epsilon_pump_value in epsilon_pump_values:
531     def combined_differential_equations(current_rho_values, t):
532         real_part = current_rho_values[:9].reshape((3, 3)).astype(float)
533         imag_part = current_rho_values[9:].reshape((3, 3)).astype(float)
534
535         rho = real_part + 1j * imag_part
536
537         L_values = L.subs({
538             rho_11: rho[0, 0], rho_12: rho[0, 1], rho_13: rho[0, 2],
539             rho_21: rho[1, 0], rho_22: rho[1, 1], rho_23: rho[1, 2],
540             rho_31: rho[2, 0], rho_32: rho[2, 1], rho_33: rho[2, 2]
541         })
542
543         epsilon_probe_values = expo(t, k=0.1)
544
545         H_values = H.subs({
546             Delta_probe: 0,
547             Delta_pump: 0,
548             epsilon_probe: epsilon_probe_values,
549             epsilon_pump: epsilon_pump_values

```

```

550     } )

551

552     dp = np.matmul(rho, H_values) - np.matmul(H_values, rho) + L_values
553     dp_real = sp.re(sp.expand(dp))
554     dp_imag = sp.im(sp.expand(dp))

555

556     # Evaluate symbolic expressions with numerical values
557     dp_real_values = np.array(dp_real.subs({Delta_probe: 0, Delta_pump: 0, epsilon_probe
558 : 3, epsilon_pump: epsilon_pump_value})).astype(float).flatten()
559     dp_imag_values = np.array(dp_imag.subs({Delta_probe: 0, Delta_pump: 0, epsilon_probe
560 : 3, epsilon_pump: epsilon_pump_value})).astype(float).flatten()

561

562     # Initial conditions
563     t = np.linspace(0, 20, 2000)
564     y0 = np.array([1, 0, 0, 0, 0, 0, 0, 0, 0] + [0, 0, 0, 0, 0, 0, 0, 0, 0])
565

566     Y = odeint(combined_differential_equations, y0, t)

567     real_part = Y[:, :9].reshape((len(t), 3, 3))
568     imag_part = Y[:, 9: ].reshape((len(t), 3, 3))

569     rho_matrices = real_part + 1j * imag_part

570

571     rho_11_values_all = rho_matrices[:, 0, 0]
572     rho_22_values_all = rho_matrices[:, 1, 1]
573     rho_33_values_all = rho_matrices[:, 2, 2]

574

575     # Plotting
576     plt.figure(figsize=(15, 11))
577     plt.ylim(-0.01, 1.01)
578     plt.xlim(0, 20)
579     plt.plot(t, rho_11_values_all, 'b--', label=r'$\rho_{11}$', linewidth=5)
580     plt.plot(t, rho_22_values_all, 'r:', label=r'$\rho_{22}$', linewidth=5)
581     plt.plot(t, rho_33_values_all, 'g-.', label=r'$\rho_{33}$', linewidth=5)
582

583     plt.minorticks_on()
584     plt.grid(visible=True, which='major', linestyle='--')
585     plt.grid(visible=True, which='minor', linestyle='--')

```

```

589 plt.title(f'Variation of $Population$ Over $Time$ in seconds\nInitial Conditions:
590   Population in State $|1\rangle; \Delta_{\{\mathrm{probe}\}} = \Delta_{\{\mathrm{pump}\}} = 0 , \epsilon_{\{\mathrm{pump}\}} = \epsilon_{\mathrm{pump\_value}}, \epsilon_{\{\mathrm{probe}\}}$ is increasing slowly')
591
592 plt.xlabel(r'$Time$/ s ')
593 plt.ylabel(r'$Population$')
594
595 plt.axvline(x=6.9, color='grey', linestyle='--', label=r'$\epsilon_{\{\mathrm{probe}\}} \approx 1$')
596 plt.legend()
597 plt.xticks(ticks=[*plt.xticks()[0], 6.9])
598 plt.savefig(f'Graph 4_epsilon_pump_{epsilon_pump_value}.png', dpi = 1000)
599 plt.close()
600 images.append(imageio.imread(f'Graph 4_epsilon_pump_{epsilon_pump_value}.png'))
601
602 #Saving the images as a GIF
603 imageio.mimsave('graph4_epsilon_pump_variation.gif', images, duration=3)

```

**Listing 1.** Python code used for Task 1, 2, 3.

## 1.2 Mathematica computational code

```

1 ket1 = {{1}, {0}, {0}};
2 bra1 = {{1, 0, 0}};
3
4 ket2 = {{0}, {1}, {0}};
5 bra2 = {{0, 1, 0}};
6
7 ket3 = {{0}, {0}, {1}};
8 bra3 = {{0, 0, 1}};
9
10 (*Compute Hamiltonian, H, by first principles. Dot[] is used for \
11 general matrix multiplication*)
12 H = h*(Delta_pump* Dot[ket1, bra1] + Delta_pump *Dot[ket2, bra2] +
13     epsilon_pump *(Dot[ket2, bra3] + Dot[ket3, bra2]) +
14     epsilon_probe* (Dot[ket1, bra3] + Dot[ket3, bra1]));
15 (*Define a general density matrix*)
16 p = {{rho11, rho12, rho13}, {rho21, rho22, rho23}, {rho31, rho32,
17     rho33}};
18

```

```

19 (*To simplify the process, first calculate the commutator and L[p]*)  

20 Commutator = Dot[p, H] - Dot[H, p];  

21 L = Dot[ket1, Dot[bra3, Dot[p, Dot[ket3, bra1]]]] +  

22 Dot[ket2, Dot[bra3, Dot[p, Dot[ket3, bra2]]]] -  

23 Dot[ket3, Dot[bra3, p]] - Dot[p, Dot[ket3, bra3]];  

24  

25 (*rho-dot is the derivative of the density matrix, the imaginary \  

26 number,i, can be entered using (ESC), then (i), and then (ESC) again*)  

27  

28 RhoDot = Simplify[((i/h)*Commutator) + L];  

29 MatrixForm[RhoDot]

```

**Listing 2.** Mathematica code to recheck the Master equation calculations

## 2 APPENDIX B

Throughout listing 1, we imported the packages that were used in this experiment. The Hamiltonian, the density matrix, and the Lindblad operators were defined and used to solve the master differential equation. Before solving, however, the real and imaginary parts were separated and solved separately but simultaneously. After solving, the diagonal elements of the density matrix depicted how the population was changing throughout states 1, 2, and 3. The population was plotted against the evolution of time. The same code was repeated a couple of times with different pump and probe field strengths. Then the process was repeating over increasing values of the pump field creating "snapshots".

Additionally listing 2 shows the calculation performed on the Mathematica field to calculate the Hamiltonian, and the Master equation. It was used as a second reference, to be sure on the calculations.