The Measurement and Computational Model of Electromagnetically Induced Transparency in Potassium

Henry Pacheco Cachon*

Colby College

Waterville, Me

(Dated: 15 May 2021)

Abstract

The use of symbolic computation has given us a way to create a computational model for electromagnetically induced transparency inside of any N-level system. By using a symbolic computation approach, we created a computational model of EIT in a four level system. Through this model, we were able to reproduce some of the behaviors found within our experimental data for EIT in potassium in the $4s - 4p - 6s$ system, and as a result we were able to determine the effects of the Rabi rate on EIT. Specifically, we found that increasing the probe laser power not only increases the EIT signal amplitude, but also hides its structure. We were also able to see that there is a point in which our probe laser strength is high enough to completely merge the peaks within our EIT signal. Finally, our computational model predicts that increasing the coupling laser strength increases the amplitude of our EIT signal while also preserving its structure.

* Author's email: hpache22@colby.edu

I. INTRODUCTION

Electromagnetically induced transparency (EIT) describes a process in which the different excitation pathways of a multi-level atom impacts that atom's response to light. In general, the absorption from a ground-state to an excited state can be reduced by coupling the excited state to a third state with a second laser. EIT has many applications for example. In Quantum Computing it has been shown that EIT can be used as a mechanism for light storage [1]. EIT has a rich history of both theoretical and experimental literature exploring the main mechanism driving it, as well as its effects on atoms like rubidium and cesium. Recently, there has been interest in studying electromagnetically induced transparency within potassium vapors [2] to explore potential behaviors that may not be easily observed with Rb and Cs. A great example of this is the impact of hyperfine structures in potassium and how their small separations may impact the transparency signal that we get from EIT.

Potassium has a pretty complicated energy level structure due to the number of electrons that it has. The majority of the complexity comes from hyperfine structure, and for some energy levels in potassium, these hyperfine splittings are separated by a few to several hundreds of megahertz, and as a result have not been experimentally measured yet. For this experiment, we will be focusing on the $4s \rightarrow 4p \rightarrow 6s$ transition. We will also be working with a potassium cell that has both ^{41}K and ^{39}K isotopes, each of which have different hyperfine structure separations [3] [1] [4].

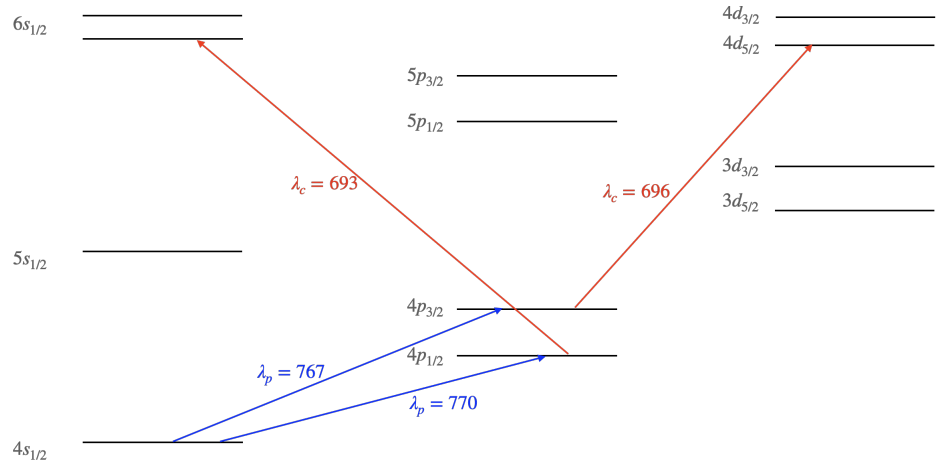


FIG. 1. The energy level structure for the $4S \rightarrow 6S$ transition and the $4S \rightarrow 4D$ transition

In this paper, we present a computational model which allows us to explore the relationship between atomic transition strength and the strength and shape of the resulting transparency signal. We will specifically move away from the usual three-level approximation commonly found in papers about Cs and Rb, and explore a 4 level system, with the possibility of adding more levels. Finally, we will present results from our experiments with a potassium cell, and compare the differences and similarities between the experimental data and our computational data.

II. EXPERIMENT

Our experiment involves a potassium cell heated at 350 K, a coupling laser set to 693 nm and a probe laser set to 770 nm. There are many parts to our experimental setup so it will help to focus on each part individually.

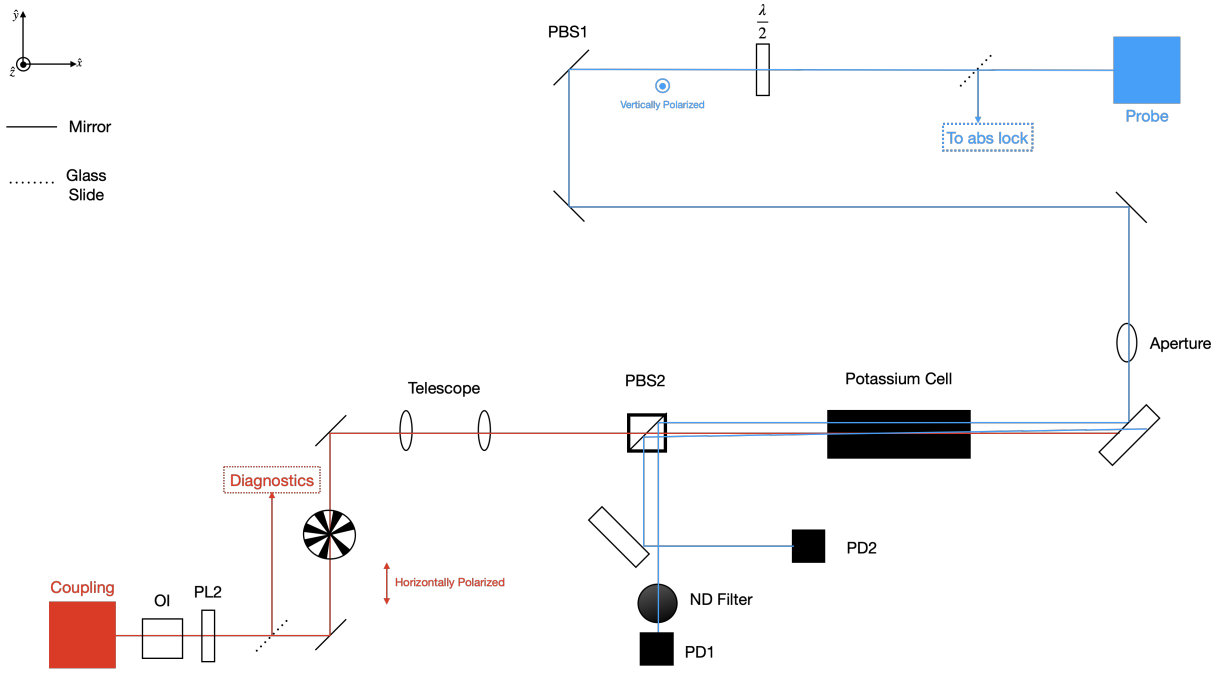


FIG. 2. Our experimental setup to measure EIT in potassium

The first part of our experimental setup is the optics that we are using for EIT. As stated earlier, we have a probe laser and the optics for both lasers are different. First, we have the probe laser in resonance with the $4s_{1/2} \rightarrow 4p_{1/2}$ transition in potassium. This requires us to create a feedback system which locks the laser at a given wavelength and a given hyperfine

transition within the $4p_{1/2}$ energy level. In our setup, we have the probe laser go through a glass slide, which will allow the laser to go through the glass and also be deflected towards our locking system. From there, the laser goes through a $\frac{\lambda}{2}$ wave-plate and polarizing beam splitter, which allows us to change the intensity of the laser while also making our laser vertically polarized. Finally, the probe laser goes through an aperture and towards a thick glass plate. Here, the glass plate makes two copies of our probe laser, one copy will be used as a reference laser, and the other will be our signal laser which is the result of the interaction with the coupling laser inside of the potassium cell.

Next, we have the coupling laser. The coupling laser has more optics involved, and the purpose behind these extra optics are to improve our data collection process and the overall quality/intensity of our coupling laser. First, the coupling laser goes through an optical isolator and then to our diagnostic tools. For our diagnostics, we use Fabry-Pérot peaks and a wavemeter. These peaks don't interact with our potassium cell at all, but we do use them as a way to plot our data with respect to frequency instead of time. Next, our laser goes through a polarizer, which only makes our laser become vertically polarized. Next, our laser goes through a telescope, which is there to increase the intensity of our coupling laser and reduces its beam diameter by two times. Finally, the coupling laser goes through a PBS beam splitter and through the potassium cell which is where the coupling laser will interact with the probe laser.

The next part of our experiment involves the optics which helps us measure the probe laser transmission, along with some of the electronics that we use to make sure that we can get the best signal possible. First, when discussing the probe laser optics, we talked about how the probe laser is copied into two different beams. These two copies are reflected by the same beam splitter that our coupling laser goes through, which directs our two copies towards two photodiodes. The reference copy goes through a neutral density filter before hitting an amplified photodiode, and the signal copy reflects off a mirror and into a second photodiode. From there, the signals of both photodiodes go through an amplifier which create a difference of the two signals, amplifies the difference, and filters out some of the high-frequency noise within our signal.

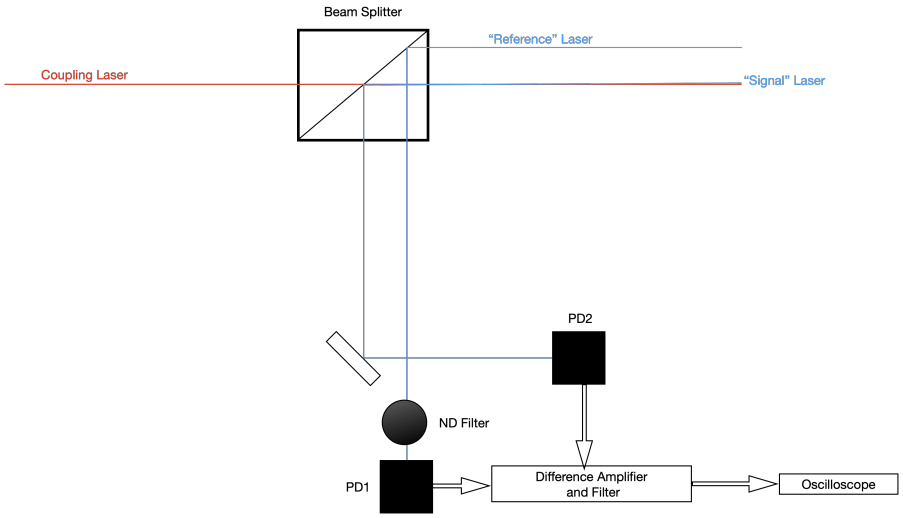
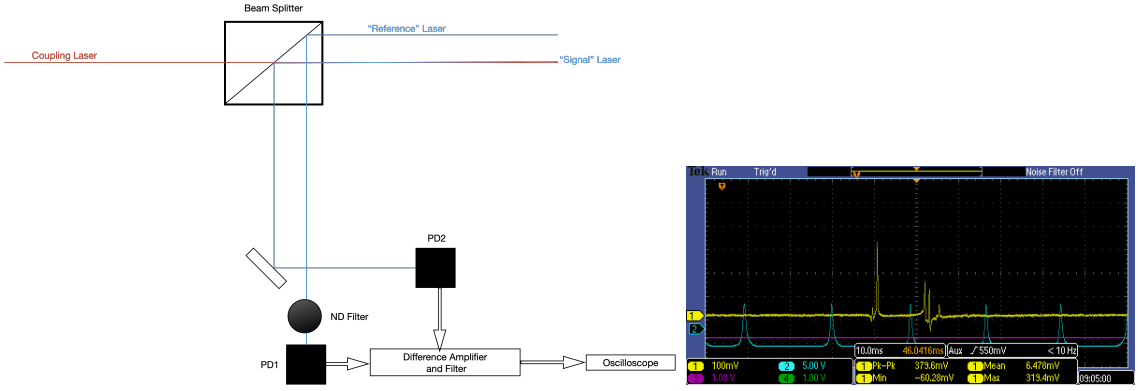


FIG. 3. Diagram of the optics involved with the measurement of our probe laser transmission.



(a) (b)

FIG. 4. (a) Diagram of the optics involved with the measurement of our probe laser transmission. (b) A sample dataset from one of our experiments, the yellow plot is our probe laser transmission signal and the blue plot shows our Fabry-Pérot peaks.

III. THEORETICAL EXPLORATION

A majority of the theoretical exploration is centered around the experiment which already has a model. We are seeking to understand how the atomic transition strength affects the transparency signal for our probe laser. In order to do this, we must come up with a theoretical model that closely describes our experimental setup, and also allow us to calculate the transmission of our probe laser.

A. The Basic Model: The Three-Level System

The basic system that we are interested in is known as a three-level ladder quantum system. If we had a particle in this system, and we defined our three states to be the three orthogonal kets - $|0\rangle$, $|1\rangle$, $|2\rangle$ - then our atom's quantum wave function can be defined as the following:

$$|\Psi(t)\rangle = C_0 e^{i\omega_0 t}|0\rangle + C_1 e^{i\omega_1 t}|1\rangle + C_2 e^{i\omega_2 t}|2\rangle \tag{1}$$

Equation 1 is the Schrodinger representation with state $|1\rangle$ defined as the state where our total energy is zero. Since we are going to be in resonance with the $4s_{1/2} \rightarrow 4p_{1/2}$ transition, we have $\omega_1 = 0$ and $\omega_0 = \omega_1 \neq 0$. These three states can be arranged in many different manners, but for the three-level ladder system, we want $|0\rangle$ to describe our ground state, $|2\rangle$ to describe our excited state, and $|1\rangle$ to describe an intermediary state between the ground state and the excited state. We also want our system to transition between these states. Finally, we want to include the spontaneous decay that happens between our three states, specifically the spontaneous decay from $|2\rangle$ to $|1\rangle$ and from $|1\rangle$ to $|0\rangle$. The transitions have to be limited though; we want to be able to transition from the ground state to the intermediate state, but not from the ground state to the excited state.

The final thing we want to add to our system is the interaction with the lasers. The basic set up for EIT is to have two electromagnetic fields (two lasers); one driving the transition from the ground state to the intermediary state (with Rabi rate Ω_p and wavelength λ_p), and the other driving the transition from the intermediary state to the excited state (with Rabi rate Ω_c and wavelength λ_c) as shown in figure 5.

B. The Basic Model: The Hamiltonian

As with any quantum mechanics problem, the first thing we should do is derive the Hamiltonian for our three-level system. The first thing to note is that we are going to work from the laser's reference frame, which allows us to represent a transition frequency as a detuning. This is known as the field-interaction representation, and it actually significantly simplifies the Hamiltonian. We can split up the Hamiltonian into two parts: the first part describes our system where the laser intensities are zero, and the second part describes our

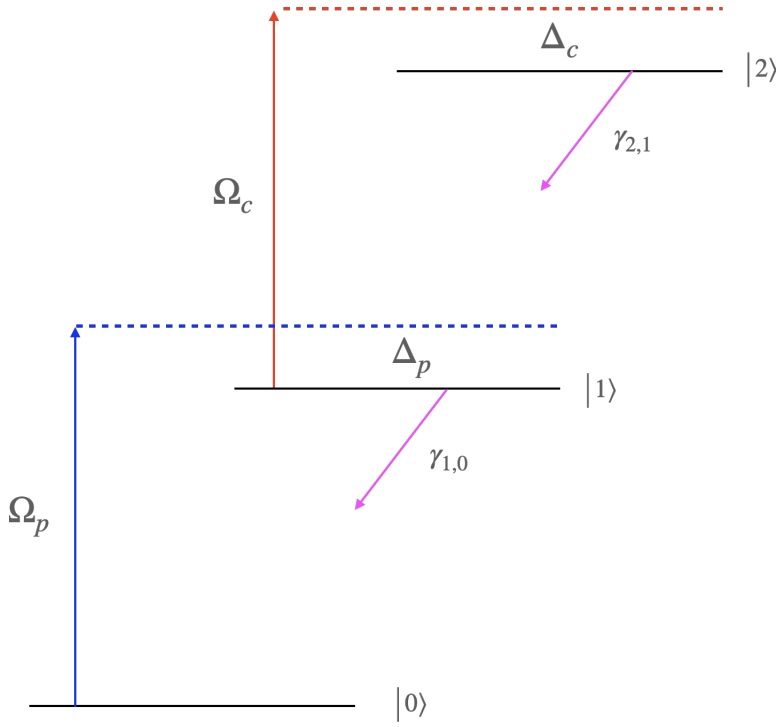


FIG. 5. three-level system interacting with two separate electromagnetic fields. Ω_p and Ω_c are the Rabi rates for the two lasers. Δ_p describes the detuning of the probe laser from the intermediate state, likewise Δ_c describes the detuning of the coupling laser from the excited state. Finally, $\gamma_{1,0}$ and $\gamma_P 2, 0$ are the decay rates.

system interacting with the two lasers.

$$H(t) = H_0 + V(t) \tag{2}$$

The first part is H_0, which is the Hamiltonian for our system if it were all by itself. Since we are dealing with three levels, H_0 will only have two non-zero elements. If we take the intermediary state to be the point where our energy is 0, we get the following Hamiltonian:

$$H_0 = \begin{bmatrix} -\delta_p & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \delta_c \end{bmatrix} \tag{3}$$

The next part is $V(t)$, which is the field-interaction part of our Hamiltonian. This part contains information for how our system interacts with the two lasers. There are many different approximations for this part, but the one that we will be using is known as the

Rotating-Wave Approximation (RWA). It can be shown that under the RWA, the field-interaction part of our Hamiltonian can be expressed as the following:

$$V(t) = \begin{bmatrix} 0 & \frac{\Omega_p}{2} & 0 \\ \frac{\Omega_p}{2} & 0 & \frac{\Omega_c}{2} \\ 0 & \frac{\Omega_c}{2} & 0 \end{bmatrix} \tag{4}$$

Notice that in Eq.4 we have Ω_p and Ω_c. These terms are known as the Rabi frequencies, and it represents atom-field interaction strength with frequency units. Experimentally, this can be changed by increasing or decreasing the power of our probe or coupling lasers. Adding both H_0 and $V(t)$ will give us the full Hamiltonian for our system.

Since we are working with a heated vapor cell, we also need to take Doppler shifts into account. This is because different atoms in our vapor cell will move at different speeds and at different directions, which will impact the coupling and probe lasers' wavelengths. Taking Doppler shifts into account is actually a pretty simple thing to do, we just need to substitute the following expressions for our probe and coupling detunings.

$$\begin{aligned} \delta_p &= \Delta_p + \vec{k_p} \cdot \vec{v} \\ \delta_c &= \Delta_c + \vec{k_c} \cdot \vec{v} \end{aligned} \tag{5}$$

Here, we $\vec{k_p}$ and $\vec{k_c}$ are the k-vectors of our probe and coupling lasers, and $\vec{v}$ is the velocity of a potassium atom. Since we are taking Dopple shifts into account, we need to integrate over the Maxwell-Boltzman Distribution of velocities in order to calculate the probe transmission; there will be a deep discussion of this process in a later section.

C. The Basic Model: Finding Steady State Solutions

Now that we have the Hamiltonian, we can focus on finding the steady state solutions for our system. Before moving on though, we need to take decay rate into account! The reason we don't take decay into account within our Hamiltonian is because the Schrodinger equation does not offer us a self-consistent way to handle decay within our system. As a result, we need to use the density matrix formalism as well as the Von-Neumann equation in order to incorporate both decay and laser excitation.

The first focus will be on the density matrix formalism. This formalism requires us to use an operator known as the density matrix operator. This operator is defined in the following manner:

$$\hat{\rho} = |i\rangle \langle j| \tag{6}$$

Here, i and j denote two states in our system. A very important thing to notice is that $\hat{p}$ is a matrix; the diagonal entries of this matrix gives us information about the population density of all our states (i.e. p_{ii} will give us the population density of the state $|i\rangle$). The off-diagonal elements gives us information about the coherences which is basically information about how our different states interact with each other. With the density matrix formalism, we now need to solve the following differential equation.

$$i\hbar\dot{\hat{\rho}} = [H, \hat{\rho}] \tag{7}$$

Equation. 7 is known as the Von-Neumann Equation, which we will be using since this is a self-consistent way of handling spontaneous decay and laser excitation. In order to incorporate decay, we will use the Lindblad operator which is defined in the following manner [5]

$$\mathcal{L}_{decay} = \sum_{i,f} \left\{ \Gamma_{i,f} \left[|f\rangle \langle i| \rho |i\rangle \langle f| - \frac{1}{2} \left(|i\rangle \langle i| \rho + \rho |i\rangle \langle i| \right) \right] \right\} \tag{8}$$

The Lindblad operator takes the allowed transitions into account in order to determine how our system will decay from one state to another. When we apply our states to the Lindblad operator, we end up getting a matrix which we can add to the Von-Neumann equation. Note $\Gamma_{i,f}$ in equation 8, this variable describes the decay rate from state $|i\rangle$ to $|f\rangle$.

$$i\hbar\dot{\hat{\rho}} = [H, \hat{\rho}] + \mathcal{L}(\hat{\rho}) \tag{9}$$

Here, we define $\mathcal{L}(\hat{\rho})$ to be a matrix with elements given by the following expression:

$$\mathcal{L}(\hat{\rho})_{jk} = \langle j| \mathcal{L}_{decay} |k\rangle \tag{10}$$

According to equation 10, the diagonal entries of $\mathcal{L}(\hat{\rho})$ describe the population transfer and the off-diagonal entries describe the decoherence. It is important to note that the

Von-Neumann equation is a differential equation which involves matrices. The differential equations that we are interested in come from the individual elements of the matrices within the Von-Neumann equation, and thus this requires us to "unpack" the Von-Neumann equation. For our three-level system, we have a system of 9 coupled differential equations. We can numerically solve these differential equations and look for the steady state solutions, but since we want to expand this to a four-level system, numerically solving the differential equations will be a computationally expensive process. We can get around this problem by turning this into a linear algebra problem. It can be shown that our system of differential equations can be transformed into the following inhomogeneous matrix equation:

$$\dot{\vec{\rho}} = \mathbf{A}\vec{\rho} + \vec{b} \tag{11}$$

Here, $\vec{\rho}$ is understood to be a vector containing all the elements of the density matrix operator.

$$\vec{\rho} = \begin{bmatrix} \rho_{00} \\ \rho_{01} \\ \rho_{02} \\ . \\ . \\ . \\ \rho_{22} \end{bmatrix} \tag{12}$$

Since we are looking for the steady state solutions, we can let $\dot{\vec{\rho}} = 0$, and we can also use the fact that all of our population densities add up to 1. Applying these conditions transforms our previous inhomogeneous matrix equation into the following homogeneous matrix equation:

$$-\vec{b} = \mathbf{A}'\vec{\rho} \tag{13}$$

Here, $\mathbf{A}'$ is an 8×8 matrix, which helps a bit with the computations. Solving this matrix equation will give us the steady state solutions, which in turn will allow us to calculate the transparency of our probe laser.

D. The Basic Model: Calculating Transparency and Doppler Averaging

Now that we have our steady state solutions for a given detuning and atom velocity, we can finally calculate the transmission of our probe laser. In order to do that we take the steady state for one of our differential equations, specifically the steady state of ρ_{10} and use the following equation [6]:

$$\alpha(v) = \alpha_0 \cdot \left(\frac{N(v)}{N_0}\right) \cdot \left[\frac{\gamma_{10}}{\Omega_p} \cdot \text{Real}(i \cdot \rho_{10})\right] \tag{14}$$

Here, N_0 is the density of potassium in our vapor cell, γ_{10} is the decay rate from the intermediary state to the ground state, Ω_p is the Rabi frequency of our probe laser, and $N(v)$ is the Maxwell-Boltzmann distribution.

Since we are working with a heated vapor cell, our model requires us to do Doppler averaging in order to get the real transmission for our probe laser. In order to do that, we must average $\alpha(v)$ over the velocity distribution of the atoms in our vapor cell. In this model, we use the Maxwell-Boltzmann distribution to get our velocities, and averaging can simply be done by numerically integrating the following expression [6]:

$$\int_{-\infty}^{\infty} \alpha(v) N(v)\, dv \tag{15}$$

For our model, we will be using -10000 cm/s for our slowest velocity and 10000 cm/s for our fastest velocity. We chose these limits since they are on the tail end of the Maxwell-Boltzman Distribution of velocities, which allows us to study a wide range of atom velocities. After going through all this analysis, we can finally model EIT for our basic three-level model. We are interested in the probe laser transmission, which can be a function of either the probe laser detuning or the coupling laser detuning as seen in figure 6.

Figure 6 are the resulting plots for our three-level system and the behavior of the plot depends on whether we are plotting the absorption signal or transmission signal. For the absorption signal, we see the usual EIT behavior which is a dip in the absorption signal strength followed by a small peak, and then followed by another dip. This kind of behavior is the same for a model which doesn't take Doppler averaging into account [6]. The transmission signal plot has an interesting behavior in that there is a dip in the transmission signal before and after there is a peak, this behavior is unique to a model which takes Doppler averaging into account, and it isn't seen if you don't take Doppler averaging into account

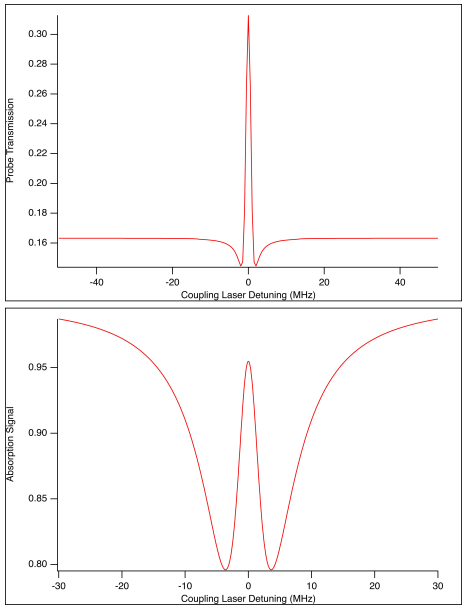


FIG. 6. Plot for the probe laser transmission as a function of coupling laser detuning (top), and plot for probe laser absorption as a function of coupling laser detuning (bottom) for a three level system. Here we let $\Omega_p = 0.5$, $\Omega_c = 3.0$, and $\Delta_p = 0.0$

[6]. These two dips seen in the transmission signal plot will become extremely important when we move into a four-level system and compare that model to our experimental results.

For our experiment, we kept the probe laser in resonance with the $4p_{1/2}$ energy level, and we varied the coupling laser detuning. It is important to note that for our four level model and our experiments, we kept the probe laser detuning at 0 MHz (resonant), and varied the coupling laser detuning.

IV. COMPUTATIONAL MODEL

Our computational model follows along closely with our theoretical exploration in the previous section. The main difference is that instead of working with a three-level system, we worked with a 4-level system. First, we will talk about our four-level model, and then talk about the implementation of our model using python.

A. Computational Model: Our Four-Level System

Our main goal is to understand how the atomic transition strengths and closely spaced excited states impact the resulting strength and shape of our EIT signal. We also want to see how the hyperfine structure of potassium can impact our EIT signal. As such, our four-level system will model the hyperfine structure within the $6s_{1/2}$ energy level.

For our model, we will have 4 different states, and we will denote the states as the orthogonal kets, $|0\rangle$, $|1\rangle$, $|2\rangle$, $|3\rangle$. Our ground state will be represented by $|0\rangle$, and our intermediary state will be represented by $|1\rangle$. Our excited state will have a hyperfine structure, which will be represented by $|2\rangle$ and $|3\rangle$. It is important to note that for our computational model, $|2\rangle$ and $|3\rangle$ will be separated by 10 MHz. Like our three-level model, we will have a probe laser driving the transition from $|0\rangle$ to $|1\rangle$, and we will have a coupling laser driving the transitions between $|1\rangle$ to our excited states. Finally, we will have decay from our excited state, $|2\rangle$ and $|3\rangle$, to our intermediary state, $|1\rangle$, and we will have decay from our intermediary state, $|1\rangle$, to our ground state, $|0\rangle$.

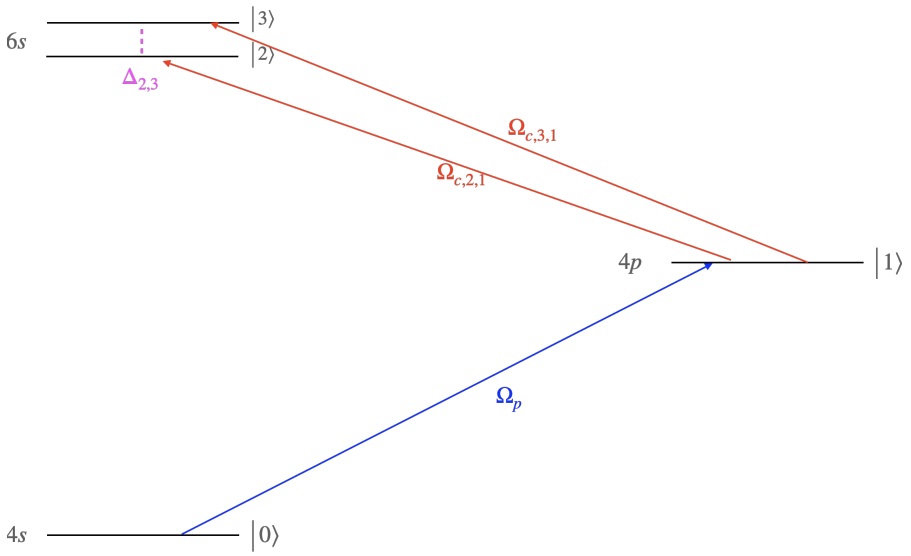


FIG. 7. Four level model of the $4s \rightarrow 4p \rightarrow 6s$ transition

Figure 7 shows the four level model that we are working with. It is important to note that our four level model takes the laser linewidths of the probe and coupling lasers into account. This is done through an additional super operator added to the Von-Neumann equation [5].

B. Computational Model: The Implementation

As seen in subsections III B - III D, the analysis required to model EIT involves a lot of math and linear algebra. It is possible to extend our analysis to a four level system and beyond, but the analysis is tedious and prone to error. A solution that I came up with was to use symbolic computation to derive the homogeneous matrix equation required to computationally model EIT (see eq. 13). My implementation is a library called nStates.py, which uses the Sympy library as its foundation [7].

The general function of nStates.py is to first initialize all the matrices that we need for the Von-Neumann equation, create a super matrix from the Von-Neumann equation, extract the system of differential equations from the super matrix, apply any numerical parameters, and finally create the homogeneous matrix equation required to find the steady state solutions.

I designed nStates.py in a way in which we can numerically solve the homogeneous matrix equation and do Doppler averaging, but the time complexity for this process is roughly $O(n^2)$. Given that Python is a pretty slow programming language, the run time for our computational models can go from minutes to hours. A workaround for this problem was to have nStates symbolically derive the $\mathbf{A}'$ matrix of Eq. 13, and then write that matrix as a C++ method. This allowed us to create a C++ class responsible for the computational modeling, which reduced our run time to seconds. More information about nStates.py and the C++ class can be found in the appendix.

V. RESULTS

A. Experimental Results

For this experiment, we only varied the probe laser strength.

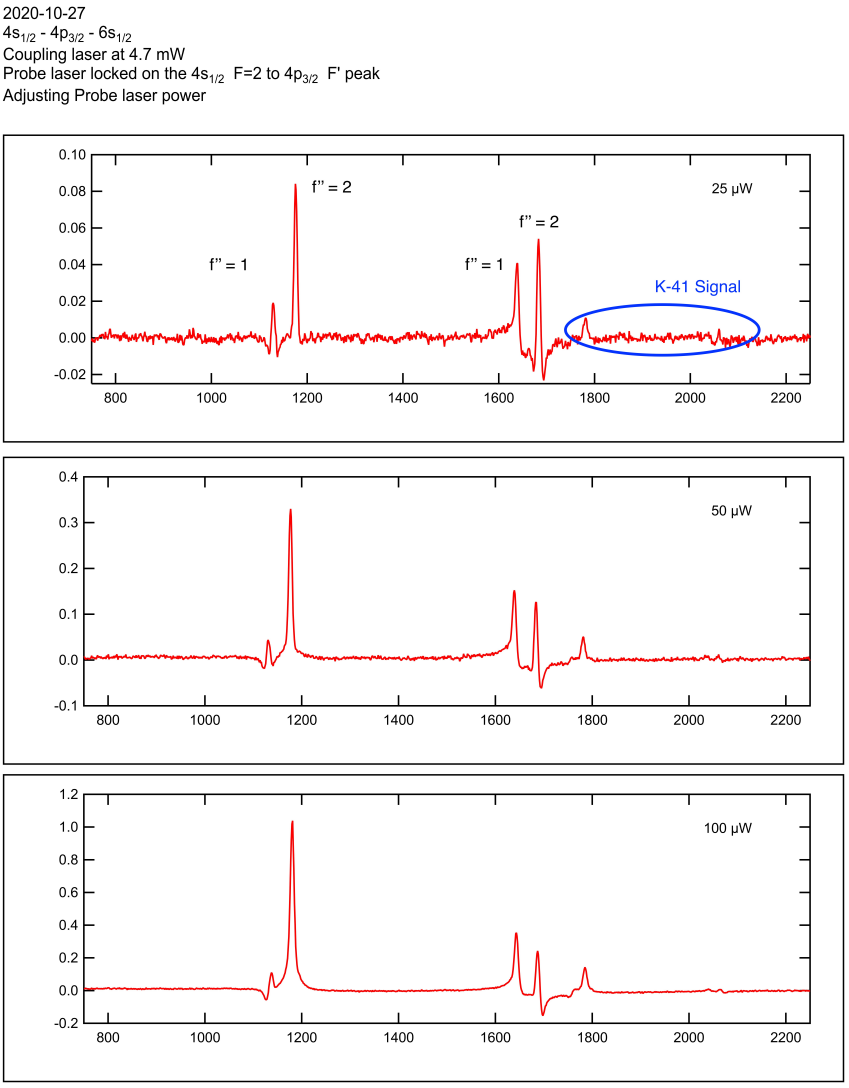


FIG. 8. Experimental data showing probe laser transmission as a function of coupling laser detuning. The different plots show the effects of increasing the probe laser power.

As stated earlier, our potassium vapor cell contains ^{39}K and ^{41}K isotopes, and we can see their respective peaks in figure 8. Since ^{39}K is more abundant than ^{41}K, the peaks for this isotope will be large, and in figure 8, they are the 4 leftmost peaks. ^{41}K is not as abundant as ^{39}K, in fact the potassium cell we are working with contains about 7% ^{41}K, and as a result,

the peaks for this isotope will be very small. In figure 8, the peaks for the ^{41}K isotope are the smallest rightmost peaks. In our dataset, we can see the peaks for the $f'' = 1$ and $f'' = 2$ hyperfine structure of the $4s_{1/2} \rightarrow 4p_{3/2}$ transition, since the $4s_{1/2}$ energy level has an $F = 1$ and $F = 2$ hyperfine structure, we see two peaks; of which the $F = 2$ is the stronger peak.

The first thing to take note of is the structure of our data set. Throughout all the plots, we see an asymmetry with the peak structure of the $F = 2$ transition; we first see a dip to the left of the peaks, but we don't see a dip to the right of the peaks. Another interesting thing we see is that as we increase the power of the probe laser, some of the structure of our peaks begins disappearing, this is mostly seen in between our peaks for both the $F = 2$ and $F = 1$ transitions.

B. Computational Results

For our computational model, we looked at varying probe laser strengths and coupling laser strengths. The main focus for our model was on the $F = 2$ transition that we saw in figure 8 which was the leftmost pair of peaks.

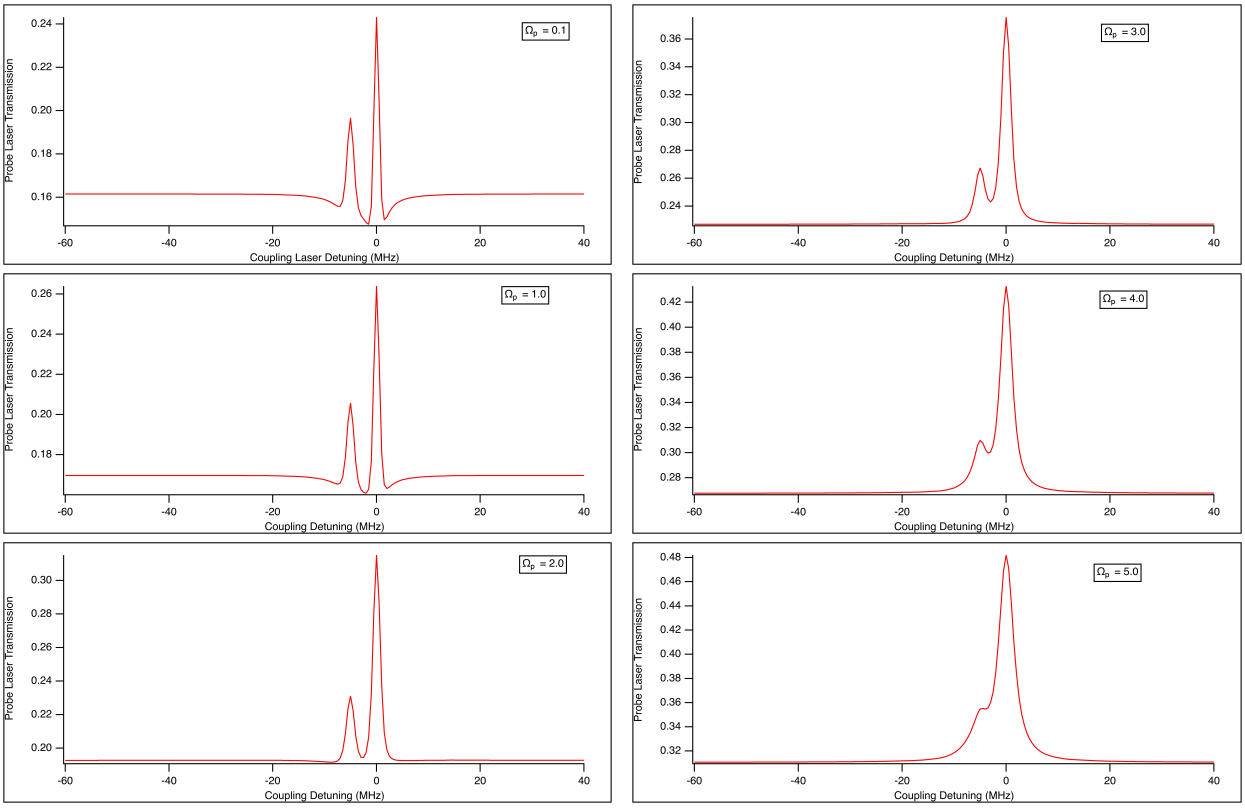


FIG. 9. Computational model of a 4 level system with varying probe laser strength. The plots show probe laser transmission as a function of coupling laser detuning

For figure 9, we let $\Omega_{c,2,1} = \Omega_{c,3,1} = 2.0$ and $\Delta_{2,3} = 10.0$, we also increased the probe laser intensity. The first thing to note is that we still have an asymmetry, but it is not like the asymmetry in our experimental dataset; there are dips on both sides of the peaks, but the left dip is slightly smaller than the right dip. As we increase the power of our probe laser, these dips begin to disappear and the asymmetry is no longer there. Furthermore, the structure between our peaks actually begins to disappear as we increased the power of the probe laser, which is something that we observed in our experimental dataset. An interesting thing to note is that as we increase the probe laser power, our two peaks seem to merge together and become one large peak.

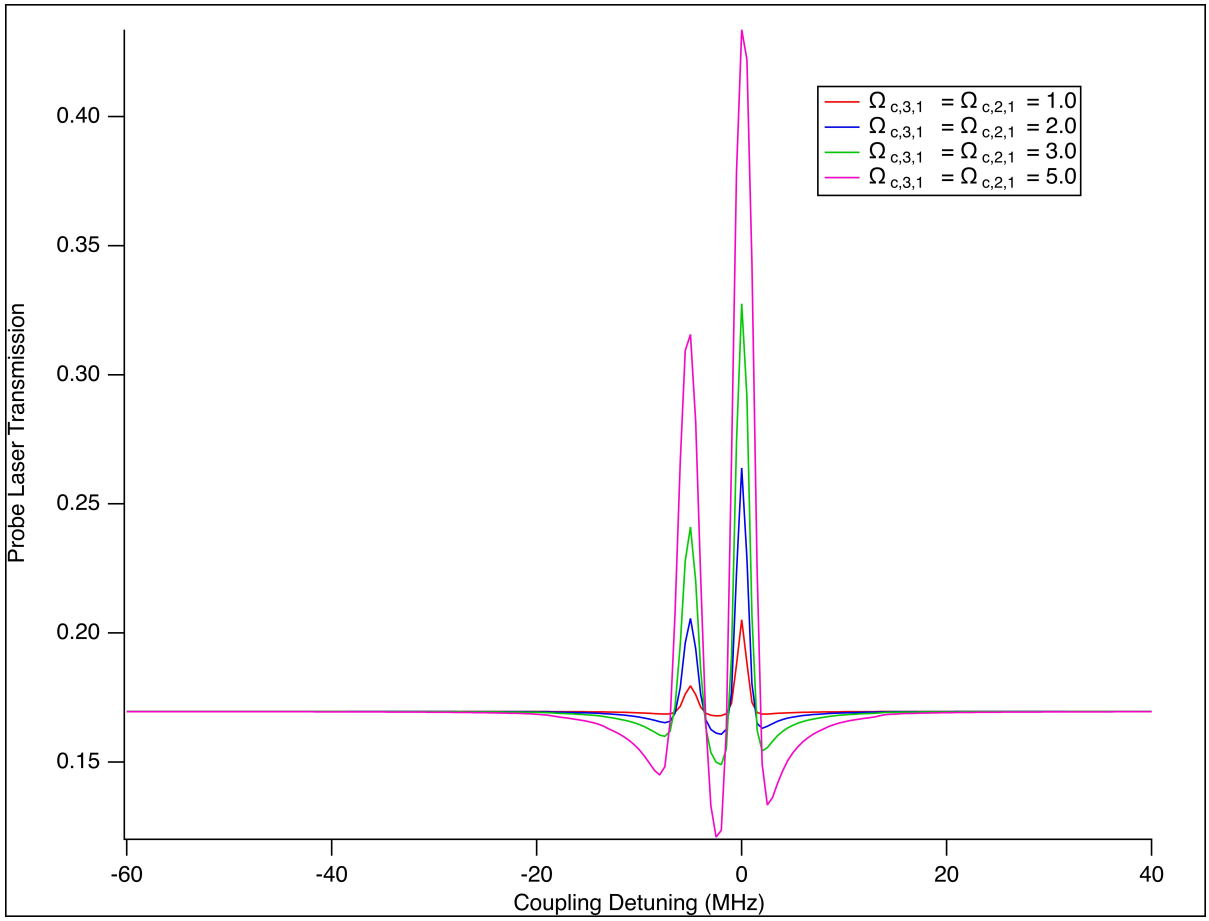


FIG. 10. Computational model of a 4 level system with varying coupling laser strength. The plot shows probe laser transmission as a function of coupling laser detuning

For figure 10, we let $\Omega_p = 0.5$ and $\Delta_{2,3} = 10.0$. Unlike the behavior for varying probe laser strength, varying coupling laser strength only really changed the intensity of the EIT signal. Although the peaks look different in figure 10, this is purely due to the scale of the graph and not any physics involving the coupling laser and its strength.

VI. DISCUSSION

From the experimental results in figure 8, we see that increasing the probe laser strength also increases the amplitude for our EIT signal. Another interesting behavior is that our peaks widen, and some of the structure in between the peaks are lost. We also see some asymmetric behavior within our data set: the left and right behavior of the peaks do not match. The left side of the peak does not experience the signature EIT dip, but the right side of the peak does.

When we run the computational model and vary the probe laser strength, we see a lot of the same behavior that we observed in the experiment. Increasing the probe laser strength in the model also increases the amplitude of our EIT signal, and we also see the widening of our peaks. There is some asymmetry within our model, but this is seen through the strength of the dips before and after the peaks. Unlike the experiment, our model predicts that the signature EIT dips are still present, but the dip is strong on the right side of the peaks and weaker on the left side of the peaks. Finally, we also observe that the structure between our peaks disappears as we increase the probe laser strength. An interesting thing to note is that if increase our probe laser strength enough, the overall structure of our EIT signal disappears, and we end up with a single peak instead of the two peaks.

When we vary the coupling laser strength within our computational model, the structure remains identical. We do see that increasing the coupling laser strength also increases the amplitude of our EIT signal, but apart from that, the structure doesn't change that much.

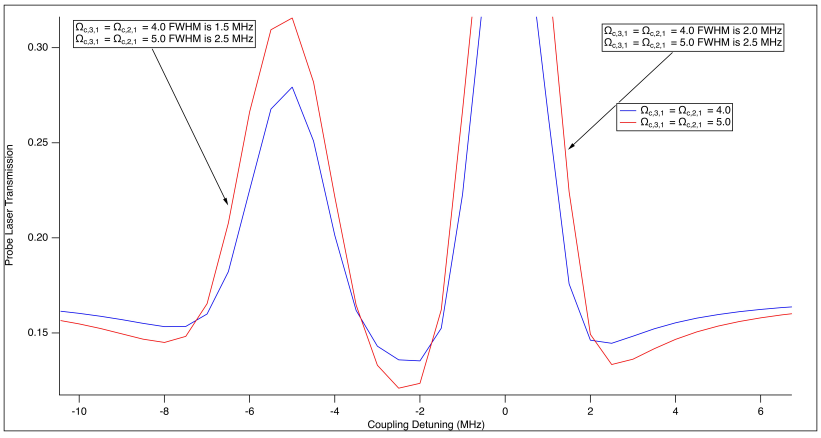


FIG. 11. Plot looking closer at the peaks for varying coupling laser strength, here we see the differences in the full width at half max of both our peaks.

As we can see in figure 11, the FWHM for each peak vary from 0.5 MHz - 1.0 MHz, which are pretty small compared to the hyperfine separation of our excited state. The discrepancy that we see in figure 10 can be due to the scaling of our plot and not due to quantum physics.

Overall, our computational model gives us a lot of information about how the atomic transition strength impacts our EIT signal. First, an increasing probe laser strength can increase the amplitude of our EIT signal, but we do risk losing the structure of the peaks within our signal. This means that if we are not careful with the probe laser strength, we can completely erase some of the peaks from hyperfine structures. If we wanted to measure the separation of these hyperfine structures, increasing the probe laser strength will do more harm than good.

There does seem to be a way to increase the amplitude of our EIT signal without losing the important structures between our peaks. This can be done through increasing the coupling laser strength which doesn't appear to significantly impact the width of our peaks.

VII. CONCLUSION

In conclusion, our computational model was able to reproduce some of the behaviors that we observed within our experimental data. Our computational model showed that increasing the probe laser strength not only increases the EIT signal amplitude, but also gets rid of some of the structures between our peaks. Our computational model also predicts that there is a point in which the probe laser strength causes our peaks to completely merge. Finally, we saw that increasing the coupling laser strength increases the amplitude of our EIT signal while also preserving the structure. If we wanted to measure some of the hyperfine structures within potassium, increasing the coupling laser strength along with having a great data gathering process will significantly aid us with our measurements.

Our computational model is limited though since we are only looking at a 4-level system. First, our computational model does not reproduce the same kind of asymmetry seen in figure 8, nor does it reproduce the same structure that we see in between the peaks. The other thing that our model does not reproduce is the structure of the ^{41}K peak; this is due to the fact that our model only looks at the ^{39}K isotope, it is unclear if the presence of ^{41}K impacts the EIT signal in any significant way, and exploring that question is a potential next step for our computational model. A very clear and important next step for our

computational model is to expand it to a higher level system so that we can further study the effects of hyperfine structures on the overall EIT signal.

Appendix A: nStates

nStates.py is a python library that I created to derive the homogeneous matrix equation necessary for the modeling of EIT in any system. The main design of nStates.py follows object oriented programming, and as such I will refer to functions and variables as methods and objects respectively.

The main goal for nStates.py was to create an accessible python library which automates the analysis seen in section III. I designed nStates.py to be modular so that a user can call on its methods and see the analysis process through code. There is also the ability for the user to see the mathematical derivation, although some matrices might be too big to fit on the command line. There are a ton of methods within nStates.py, so I will only focus on the vital methods.

The first thing I want to focus on is the initialization process. nStates.py is built using a class, and as such we must initialize nStates.py when we call it. The inputs that the library takes are the number of states, the Hamiltonian, and the parameters we wish to give a fixed numerical value. During the initialization process, nStates.py assigns the given inputs to different fields and also initializes an array containing the elements of the density matrix operator. A special thing about the array is that we can create it in a way that can help organize our matrices and the vectors that we get from solving the homogeneous matrix equation. nStates.py creates the array by putting the diagonal elements of the density matrix operator first, and then putting the off-diagonal elements last.

```
1    def __init__(self, numberStates, hamiltonian,staticDictionary):
2
3        self.N = numberStates
4        self.H = Matrix(hamiltonian)
5        self.static = staticDictionary
6        self.rhos = [Symbol("rho_{}{}".format(i,j)) for i in range(self.N)
    for j in range(self.N) if i == j]  + [Symbol("rho_{}{}".format(i,j))
  for i in range(self.N) for j in range(self.N) if i != j]
```

After the initialization process, the user has a choice to either initialize the density matrix operator, or the Lindblad matrix ($\mathcal{L}(\hat{\rho})$). The order in which the user initializes the matrices

doesn't matter, but these two matrices must be initialized before moving on. The method responsible for initializing the density matrix operator is pretty simple, it takes the density matrix operator elements, and it then builds a 2 dimensional numpy array.

```
def densityMatrix(self):

        self.rho = np.array([[Symbol('rho_{}{}'.format(i,j)) for j in
   range(self.N)] for i in range(self.N)])
```

The method responsible for building the $\mathcal{L}(\hat{\rho})$ matrix is pretty involved. Recall, that the Lindblad operator is defined as

$$\mathcal{L}_{decay} = \sum_{i,f} \left\{ \Gamma_{i,f} \left[|f\rangle \langle i| \rho |i\rangle \langle f| - \frac{1}{2} \left(|i\rangle \langle i| \rho + \rho |i\rangle \langle i| \right) \right] \right\} \tag{A1}$$

and that elements of $\mathcal{L}(\hat{\rho})$ are given by the following expression:

$$\mathcal{L}(\hat{\rho})_{jk} = \langle j| \mathcal{L}_{decay} |k\rangle \tag{A2}$$

In order to make nStates.py readable, I had to make an auxiliary method which handles the computation of equation A2. This is where the superOperatorElements method comes into play. This method takes in the allowed transitions as an input, along with two integers representing the bra and ket within equation A2. The basic algorithm for this method can be broken down in the following way:

1. Create a placeholder array to hold in the computational results of the Lindblad operator for individual transitions.

2. For each transition do the following:

   (a) Get the initial and final state within the transition.

   (b) Make the initial and final states orthogonal bras and kets.

   (c) Make a orthogonal bra for the left integer and an orthogonal ket for the right integer.

   (d) Compute the Lindblad operator for the current initial and final state, apply the left and right bra-ket, and append the result to the placeholder array.

3. Sum all the elements within the placeholder array, and return the result.

In Python, I implemented the code in the following way:

```
def superOperatorElements(self, allowedTransitions, left, right):

        # Initializing placeholder list. Using a list for the sum()
    function
        # Which just adds all the elements within a list!
        self.lindbladMaster = []

        # Initializing input variables
        self.allowedTransitions = allowedTransitions
        self.left = left
        self.right = right

        # This section goes through a possible transition, computes the
    lindblad equation
        # and it then appends that result to the lindbladMaster list.
        for transitions in self.allowedTransitions:
            self.i = transitions[0]
            self.f = transitions[1]

            # Initialize Symbols
            self.gamma = Symbol('Gamma_{}{}'.format(self.i,self.f))
            self.rho_ii = Symbol('rho_{}{}'.format(self.i,self.i))
            self.rho_iright = Symbol('rho_{}{}'.format(self.i,self.right))
            self.rho_lefti = Symbol('rho_{}{}'.format(self.left,self.i))

            # Initializing Bras and Kets for the Lindblad Equation
            self.iKet = OrthogonalKet(self.i)
            self.iBra = OrthogonalBra(self.i)
```

```
27
28              self.fKet = OrthogonalKet(self.f)
29              self.fBra = OrthogonalBra(self.f)
30
31              self.Right = OrthogonalKet(self.right)
32              self.Left = OrthogonalBra(self.left)
33
34              # Lindblad equation, look in notes to see what this is!
35              self.L = self.gamma * ( (self.Left*self.fKet).doit() * self.
    rho_ii * (self.fBra*self.Right).doit() \
36                      - 1/2*( (self.Left*self.iKet).doit() * self.
    rho_iright + self.rho_lefti * (self.iBra*self.Right).doit() ) )
37              self.lindbladMaster.append(self.L)
38
39          self.total = sum(self.lindbladMaster)
40
41          return(self.total)
```

The superOperatorElements method returns a sympy object which we can use to make the $\mathcal{L}(\hat{\rho})$ matrix. The lindbladMatrix method is the method responsible for initializing the $\mathcal{L}(\hat{\rho})$ matrix, and it does this by calling on the superOperatorElements method. The lindbladMatrix method inputs a boolean which determines if the matrix will also take finite laser linewidths into account. This input is really only useful for four-level systems, since I have not yet created the algorithm to automate the creation of the finite laser linewidth operator. The way that the lindbladMatrix operator works is that it uses python list comprehension to create a 2 dimensional array using the results of the superOperatorElements method.

```
1 def lindbladMatrix(self, allowedTransitions,laser = False):
2
3           if laser == False:
4               self.transitions = allowedTransitions
5               self.matrix = [[self.superOperatorElements(self.transitions,i,
```

```
    j) for j in range(self.N)] for i in range(self.N)]
6         elif laser == True:
7             self.transitions = allowedTransitions
8             self.matrix = [[self.superOperatorElements(self.transitions,i,
    j) for j in range(self.N)] for i in range(self.N)]
9             self.laser = [
10                [0,Symbol('gamma_p')*Symbol('rho_01'),-(Symbol('gamma_p')+
    Symbol('gamma_c'))*Symbol('rho_02'),-(Symbol('gamma_p')+Symbol('gamma_c
    '))*Symbol('rho_03')],
11                [-Symbol('gamma_p')*Symbol('rho_10'),0,-Symbol('gamma_c')*
    Symbol('rho_12'),-Symbol('gamma_c')*Symbol('rho_13')],
12                [-(Symbol('gamma_p')+Symbol('gamma_c'))*Symbol('rho_20'),-
    Symbol('gamma_c')*Symbol('rho_21'),0,0],
13                [-(Symbol('gamma_p')+Symbol('gamma_c'))*Symbol('rho_30'),-
    Symbol('gamma_c')*Symbol('rho_31'),0,0]
14            ]
15            self.matrix = Matrix(self.matrix) + Matrix(self.laser)
```

Once the user has initialized the density matrix and the $\mathcal{L}(\hat{\rho})$ matrix, the user can call on the superMatrix method. This method uses sympy to compute the Von-Neumann equation of the following form:

$$\dot{\hat{\rho}} = i \cdot [\hat{\rho}, H] + \mathcal{L}(\hat{\rho}) \tag{A3}$$

The superMatrix method uses equation A3 to create a new matrix, which is stored as a global field. This global field can be used by other methods, but most importantly, it is held in the memory until the program is shut down. From here, the user can call on the makeEquations method, which goes through each element of the super matrix and appends them to a list. After that, any variable within the equations is replaced with a numerical value that the user initialized at the very beginning, and finally we have a list of equations from which we can create a matrix.

```
1 # This function makes our list of equations by only using the static
```

```
    parameters.
    def makeEquations(self):

        # Extracting elements from super matrix and putting
        # them into a list
        self.eqns = [self.A[i,j] for i in range(self.N) for j in range(
    self.N) if i == j] + \
                    [self.A[i,j] for i in range(self.N) for j in range(
    self.N) if i != j]

        # Substituting parameters that are fixed
        for key in self.static.keys():
            self.eqns = [i.subs(key,self.static[key]) for i in self.eqns ]
```

Finally, the user can call on the changeParam method, which will make a lambda function from which we can run our computational models. This method inputs any parameter that we wish to change independently, these parameters will end up being the input of our resulting lambda function. As an aside, a lambda function in Python is basically the equivalent of a method that can only do one thing. These kinds of functions are really useful for math operations and single-step algorithms, and Sympy uses lambda functions to convert a symbolic equation into a method that Python can run with numerical values. The way the changeParam method works is that it first goes through each equation, and simplifies each equation symbolically. Then, it uses the linear_eq_to_matrix method from the sympy library, the array containing the elements of the density matrix operator, and the list of equations to create a matrix from the list of equations. After that, we remove the first column and the top row, which will give us the final matrix. Finally, we convert the matrix into a lambda function, which we can use to run our numerical models from. A thing to note is that the linear_eq_to_matrix method returns a matrix and an extra vector. Due to how we initialize our super matrix, the extra vector is actually $\vec{b}$ from section III C. Due to this, I have the method save $\vec{b}$ as another global field which I can use to create a solve method.

```
# Function changes the non static parameters
```

```
    # The inputs are a dictionary and an optional key.
    def changeParam(self,variables):

        self.change = variables

        # Just in case there were some things that were not evaluated.
        # This really only happens if you use the unevaluatedExpr()
    wrapper

        # Makes our matrix and vector
        self.final , self.vector = linear_eq_to_matrix(self.eqns,self.rhos
    )

        self.final1 = self.remove(self.final,0,0)
        #self.final1 = SparseMatrix(self.final1)
        self.vector = np.array(self.vector).astype(np.complex128)
        self.vector = np.delete(self.vector,0,axis=0)

        self.f = lambdify(self.change,self.final1,'numpy')

        return(self.f)
```

The solve method inputs a 2 dimensional array and uses the linalg.solve method from numpy [8]. This method basically uses $\vec{b}$ and the input 2 dimensional array to numerically solve the homogeneous matrix equation from section III C.

```
# This function solves our final matrix problem!
    def solve(self,matrix):
        self.inputMatrix = matrix
        # Finally it solves our modified matrix and vector
        self.solutions = np.linalg.solve(self.inputMatrix, self.vector)

```

```
7         return(self.solutions)
```

In summary, nStates.py works by initializing all the matrices needed to create the homogeneous matrix equation for EIT. After that, we can give nStates.py the variables we wish to change independently in order to create a lambda function which uses those variables as an input. This allows us to model any variation of EIT, from EIT without Doppler averaging, to EIT with finite laser linewidths and Doppler averaging.

The source code for nStates.py is available on github (`https://github.com/hpache/nStates`)!

[1] C. Liu, Z. Dutton, C. H. Behroozi, and L. V. Hau, Observation of coherent optical information storage in an atomic medium using halted light pulses, Nature **409**, 490 (2001).

[2] A. Sargsyan, P. A. Petrov, T. A. Vartanyan, and D. Sarkisyan, Electromagnetically induced transparency in potassium vapors: Features and restrictions, Optics and Spectroscopy **120**, 339 (2016).

[3] D. McKay, Potassium 5p line data, *http* : $//fermionlattice.wdfiles.com/local --files/papers/5P_structure$ (2009).

[4] S. Hörbäck, A. M. Pendrill, L. Pendrill, and M. Pettersson, Experimental and theoretical investigation of the isotope shift of the 4d level in atomic potassium, Zeitschrift für Physik A Atoms and Nuclei **318**, 285 (1984).

[5] J. B. Naber, A. Tauschinsky, H. B. van Linden van den Heuvell, and R. J. C. Spreeuw, Electromagnetically induced transparency with Rydberg atoms across the Breit-Rabi regime, SciPost Phys. **2**, 015 (2017).

[6] F. A. Tauschinsky, *Rydberg atoms on a chip and in a cell*, Phd thesis, Van der Waals-Zeeman Institute (2013).

[7] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, v. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, and A. Scopatz, Sympy: symbolic computing in python, PeerJ Computer Science **3**, e103 (2017).

[8] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del R'ıo, M. Wiebe, P. Peterson, P. G'erard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, Array programming with NumPy, Nature **585**, 357 (2020).