Quantum Optics Final Project

Matthew Chow

1 Damped Rabi Oscillations

In this problem, we are asked to find the dynamics of a resonantly driven two level atom, including spontaneous emission from the excited state via a Quantum Monte-Carlo (QMC) method. The given master equation is

$$\frac{\partial \rho}{\partial t} = -\frac{i}{\hbar}[H_{eff}, \rho]' + \Gamma \sigma_- \rho \sigma_+ \tag{1}$$

where the effective Hamiltonian is

$$H_{eff} = -i\frac{\hbar\Gamma}{2} |e\rangle \langle e| - \frac{\hbar\Omega}{2}(\sigma_- + \sigma_+) \tag{2}$$

Here $|e\rangle$ is the excited state of the two level atom, Γ is the natural linewidth of the transition, ρ is the density matrix, $\hbar$ is reduced Plank's constant Ω is the Rabi frequency, and $\sigma_{-,+}$ are the raising and lowering operators. First, we are asked to follow exactly Molmer's notes[2] and reproduce his Fig. 3. The algorithm detailed by Molmer tells us how to find the propagation of a quantum state by calculation of many stochastic quantum trajectories. [1] To find a single trajectory, we take small steps in time and at each step, we draw a random number[2], ϵ, and compare to the probability of the making a jump during that time interval[2]:

$$\delta p = \delta t \frac{i}{\hbar} \langle \psi(t)| (H_{eff} - H_{eff}^\dagger) |\psi(t)\rangle \tag{3}$$

where $\psi(t)$ is the instantaneous wavefunction. In the case of damped Rabi oscillations, this is simply given by the population of the excited state, multiplied by the decay rate and the time interval[2].

$$\delta p = \Gamma |c_e|^2 dt \tag{4}$$

If $\epsilon > \delta p$, then we have no quantum jump (as must be true in most cases if δt is to model a differential), and the wavefunction evolves under the effective Hamiltonian. For a sufficiently short time [2],

$$|(\psi(t+\delta t)\rangle = \frac{(1 - \delta t \frac{i}{\hbar} H_{eff}) |\psi(t)\rangle}{||(1 - \delta t \frac{i}{\hbar} H_{eff}) |\psi(t)\rangle ||} = \frac{(1 - \delta t \frac{i}{\hbar} H_{eff}) |\psi(t)\rangle}{\sqrt{1 - \delta p}} \tag{5}$$

where we must re-normalize the state, given the non-Hermitian nature of H_{eff}. The effects of this non-Hermitian evolution will play a strong role in the discussion of optical pumping in Section 2.

If $\epsilon < \delta p$, then we *do* have a quantum jump. To realize the jump, we apply our Lindblad jump operator. In the case of the spontaneous emission in the two level atom, this simply results in the atom going to its ground state $|\psi(t+\delta t)\rangle = |g\rangle$. Generally, if we have many possible jumps, then we must choose which jump happens, and apply the appropriate jump operator. Again, further discussion is reserved for Section 2.

[1]Note that in this discussion, I assume an initially pure state. If the initial state is a mixed state, the result would be the convex combination of pure states, each propagated by the QMC algorithm.

[2]All random numbers discussed are sampled uniformly on [0,1)

As described in lecture, note that at all times, the state of the system in a single trajectory is a pure state. By averaging the results of many trajectories, we converge towards the solution of the master equation. This is equivalent to averaging over many measurement records, or tracing out the environment. In the case of the two level atom and the damped Rabi oscillations, we can think of each individual trajectory as a run of the experiment, where we collect the emitted photons with unit efficiency. Averaging over many trajectories is then equivalent to running the experiment many times and averaging the results. Our statistical error on the expectation value of an operator (A) after averaging m trajectories is given in Molmer [2]

$$\delta A_m = \frac{\Delta A_m}{\sqrt{m}} \tag{6}$$

where ΔA_m is the standard deviation of the sample. Note that the operator in question for this project is simply the projector onto some state that we care about.

To recreate Molmer's Figure 3, I wrote a Monte Carlo script to calculate 100 trajectories from time 0 to $4/\Gamma$ with 1000 steps, assuming a Rabi frequency of $\Omega = 6\Gamma$. The results of this script are shown in Figure 1, and the code is included in the appendix.

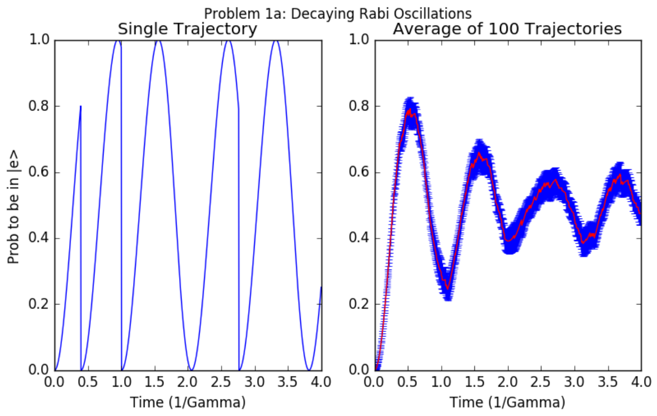

Figure 1: Solution to 1a: Recreation of Figure 3 from Molmer, using the QMC method. On the left is a single trajectory, in which we see Rabi flops, interrupted by random jumps to the ground state. On the right is an average of 100 such trajectories, which results in damped Rabi oscillations. The red line is the mean, blue error bars plotted at each point appear as a statistical confidence interval at each point. Note that as the oscillations decay, the error increases. This is because after sufficient time, the phase of the Rabi cycle is randomized due to the random timing of the jumps, leading to variation when considering an ensemble average.

Next, we are asked to modify the calculation by sampling from the wait time distribution to find *when* the next jump occurs, as opposed to selecting a random number to determine jump/no jump at each time step. To determine the time of the jump, we draw a random number and compare to the the probability that a jump has happened in the time interval t_0 to $t_0 + \tau$. The probability that no jump has occurred is given by [1]

$$\int_{t_0}^{t+\tau} || \frac{H_{eff}}{\hbar} \left|\psi(t)\right\rangle || dt \tag{7}$$

The probability that at least one jump has occurred is the compliment of the above. Thus, we draw our random number, ϵ, and when

$$1 - \int_{t_0}^{t+\tau} || \frac{H_{eff}}{\hbar} \left|\psi(t)\right\rangle || dt > \epsilon \tag{8}$$

We jump!

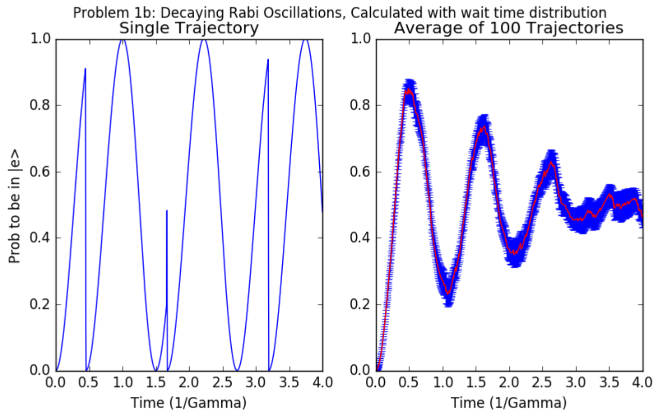

Figure 2: Solution to 1b: Second recreation of Figure 3 from Molmer, calculated using the sample from wait time distribution method. This matches well with the results from Problem 1a. See appendix for code.

I wrote a new "unravel" function to reflect this modification to the algorithm, and the results are summarized in Fig. 2. Note that I have numerically integrated when monitoring the norm, but that an analytic solution may improve computation time.

2 Coherent Population Trapping in Zeeman Degenerate Atom

In this problem, we consider the evolution of an atom with degenerate Zeeman levels in the ground and excited state manifolds, driven by linearly polarized light on resonance. Both the upper and lower states have total angular momentum J=1. I will label and order the states $\{|g-\rangle, |g0\rangle, |g+\rangle, |e-\rangle, |e0\rangle, |e+\rangle\}$, where g and e denote ground and excited state manifolds, and +, -, 0 denote quantum number m_J = +1, -1, and 0 respectively. The level structure is depicted in Figure 10 of Molmer:

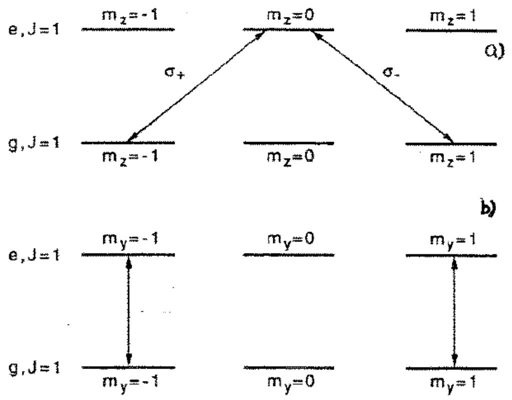

Figure 10 from Molmer[2], ripped off without permission. Level structure of our toy model atom. In (a), level diagram in the Z basis. Right and left circularly polarized light about the quantization axis drive σ_+ and σ_- transitions. This couples the $|g-\rangle_z$ and $|g+\rangle_z$ to $|e0\rangle_z$. In (b), level diagram in the Y basis. Linearly polarized light along the quantization axis drives π transitions. This couples $|g-\rangle_y$, $|g+\rangle_y$ to $|e-\rangle_y$, $|e+\rangle_y$ respectively. Note that the J=1, m=0 ground to J=1, m=0 excited transition is dipole forbidden.

Taking the light polarization to be along the $\mathbf{e}_y$ direction, the system is described by Hamiltonian

$$H_{eff} = H - i\frac{\hbar\Gamma}{2}P_e \tag{9}$$

Where

$$P_e = \sum_{m=-1,0,+1} |em_e\rangle \langle em_e| \tag{10}$$

$$H = -\frac{\hbar\Omega}{2}(D_y + D_y^\dagger) \tag{11}$$

$$D_y^\dagger = \sum_{q=-1,0,+1} D_q^\dagger \mathbf{e}_y \dot{} \mathbf{e}_q \tag{12}$$

Where $\mathbf{e}_q$ are the spherical basis vectors. The Lindblad operators are given by

$$D_q^\dagger = \sum_{m_e, m_g} \langle J_e = 1, m_e | J_g = 1, m_g; J = 1, q\rangle |m_e\rangle \langle m_g| \tag{13}$$

$$\doteq \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \langle -1|-1;q\rangle & \langle -1|0;q\rangle & 0 & 0 & 0 & 0 \\ \langle 0|-1;q\rangle & 0 & \langle 0|+1;q\rangle & 0 & 0 & 0 \\ 0 & \langle +1|0;q\rangle & \langle +1|+1;q\rangle & 0 & 0 & 0 \end{bmatrix} \tag{14}$$

Again, with ordering $\{|g-\rangle, |g0\rangle, |g+\rangle, |e-\rangle, |e0\rangle, |e+\rangle\}$. Here I have used a shorthand for the Clebsch-Gordan(CG) coefficients by listing only the m values, since all J values are 1. Notice that $D_q^\dagger$ is zero except for the lower left quadrant, in the discussion to follow, I show only the lower left quadrant of $D^\dagger$ matrices. Additionally, $|q| <= 1$ gives, $(D_q^\dagger)_{6,1} = (D_q^\dagger)_{4,3} = 0$ and our dipole forbidden transition $(\Delta J = 0, \Delta m = 0, m = 0)$ gives $(D_q^\dagger)_{5,2} = 0$. The relevant CG coefficients are listed in the appendix for the remaining elements. The lower left quadrants are shown for each value of q below

$$D_{q=-1}^\dagger \doteq \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & +1 & 0 \\ 0 & 0 & +1 \\ 0 & 0 & 0 \end{bmatrix} \tag{15}$$

$$D_{q=0}^\dagger \doteq \frac{1}{\sqrt{2}} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{16}$$

$$D_{q=+1}^\dagger \doteq \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \tag{17}$$

Writing out $D_y^\dagger$ explicitly in the Z basis[3]

$$D_y^{\dagger(z)} = \frac{-i}{\sqrt{2}}(D_{q=-1}^\dagger + D_{q=+1}^\dagger) \tag{18}$$

[3]Note that this is still just the lower left quadrant of $D^\dagger$ since other terms are 0.

$$D_y^{\dagger(z)} \doteq \frac{i}{2} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \tag{19}$$

From here we have the effective Hamiltonian in the Z basis:

$$H_{eff}^{(z)} \doteq -\frac{i\hbar}{2} \begin{bmatrix} 0 & 0 & 0 & 0 & -\frac{\Omega}{2} & 0 \\ 0 & 0 & 0 & \frac{\Omega}{2} & 0 & -\frac{\Omega}{2} \\ 0 & 0 & 0 & 0 & \frac{\Omega}{2} & 0 \\ 0 & -\frac{\Omega}{2} & 0 & \Gamma & 0 & 0 \\ \frac{\Omega}{2} & 0 & -\frac{\Omega}{2} & 0 & \Gamma & 0 \\ 0 & \frac{\Omega}{2} & 0 & 0 & 0 & \Gamma \end{bmatrix} \tag{20}$$

From this matrix, it's easy enough to find the eigenvector with eigenvalue zero. This is the dark state in the Z basis.

$$|\psi_{dark}\rangle_z \doteq \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \doteq \frac{1}{\sqrt{2}}(|g-\rangle_z + |g+\rangle_z) \tag{21}$$

Similarly, we can repeat this process in the Y basis[3]. After changing the spherical basis vectors to be oriented along $\mathbf{e}_y$, only q=0 contributes to $D_y^{\dagger(y)}$

$$D_y^{\dagger(y)} = D_{q=0}^{\dagger} \doteq \frac{1}{\sqrt{2}} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{22}$$

$$H_{eff}^{(y)} \doteq -\frac{\hbar}{2} \begin{bmatrix} 0 & 0 & 0 & -\frac{\Omega}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{\Omega}{\sqrt{2}} \\ -\frac{\Omega}{\sqrt{2}} & 0 & 0 & i\Gamma & 0 & 0 \\ 0 & 0 & 0 & 0 & i\Gamma & 0 \\ 0 & 0 & \frac{\Omega}{\sqrt{2}} & 0 & 0 & i\Gamma \end{bmatrix} \tag{23}$$

$$|\psi_{dark}\rangle_y \doteq \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \doteq |g0\rangle_y \tag{24}$$

Now we are asked to run our Monte-Carlo algorithm on this multilevel atom in both the Z and the Y basis, assuming that the state starts in $|g-\rangle_z$. We hope to replicate the known result of optical pumping to the dark state, using the QMC approach. Before showing the results of running the algorithm, I should first explain the process of choosing between multiple jumps. The algorithm is the same as that described for the Rabi oscillations, except that in the event of a jump, we choose randomly amongst the Lindblad jump operators, weighted by the probability of each one. Generally, the probability of a jumping with particular Lindblad jump operator L_k, (given that a jump happened) for wavefunction $|\psi(t)\rangle$ is given in Molmer [2]

$$Prob(k|jump) = \frac{\delta p_k}{\delta p = \sum_k \delta p_k} = \frac{\langle\psi(t)| L_k^\dagger L_k |\psi(t)\rangle}{\delta p} \tag{25}$$

Where δp is given in Eq 3. In order to select a jump with such weighting, I iterate through a list of all the jump operators, and at each step add δp_k to a sum variable ζ. At each step, I compare ζ to the random number that was drawn that caused the jump, ϵ. After adding δp_k, if $\zeta > \epsilon$, then I select the jump with operator L_k and exit the loop. This works because ϵ was sampled uniformly on $[0, 1)$, and thus also sampled uniformly on $[0, \delta p)$. The python implementation of the algorithm produced the results in Figure 3 using the initial state, $|\psi_0\rangle = |g-\rangle_z$, the Lindblad operators given by Eq. 13 and the Hamiltonian in Eq. 9. The code is included in the appendix.

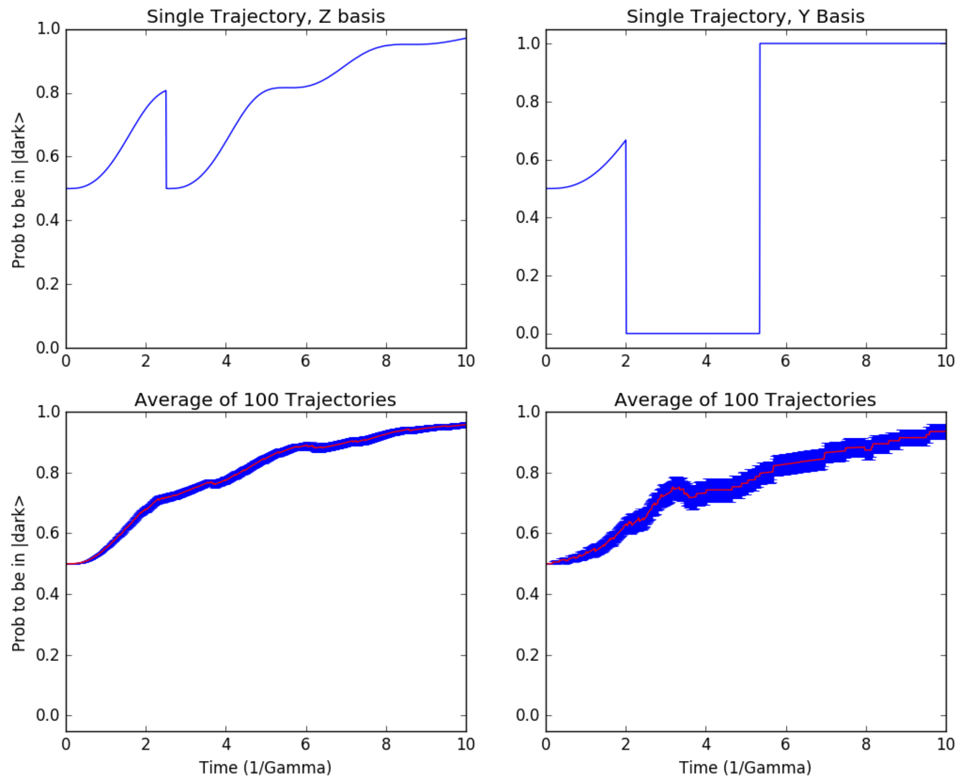

Figure 3: Reproduction of Figure 11 from Molmer[2]. Evolution towards the dark state, calculated with the QMC approach. The initial state is $|g-\rangle_z$, the Rabi rate is 3Γ. Notice that in the Z basis (left), the no-jump evolution drives the state towards the dark state, whereas in the Y basis(right), the dominant driver to the dark state is the sigma jump. Notice too that the error bars in the Z basis are smaller, given the "smoother" evolution.

Upon inspection of Figure 3, it is apparent that the measurement basis we choose greatly impacts our understanding of the quantum state.[4] In the Y basis, we see sudden jumps to the dark state (upon measurement of a sigma photon), whereas in the Z basis, we see that jumps take us to 50% probability of being in the dark state. In fact, when looking at the Z basis, it is the *no jump* evolution that drives the state towards the dark state. If we are measuring in the Z basis, any photon we detect projects the state onto an equal superposition of bright and dark states. For each time step that we don't see a photon, when we update the state, we think we are a little more likely to be in the dark state. As Ivan mentioned in lecture, this is like updating our prior with "quantum Bayes rule." In either basis, in the limit of infinite time, it must be possible to evolve to the dark state without emitting a photon, since we start in a 50-50 superposition of bright and dark. This is achieved by evolution under H_{eff} during no jump times.

Additionally, the information gained about the system is different in the two bases. For example, if we wanted to prepare the atom in the dark state while monitoring the fluorescence, we might want to choose the Y basis, given that any time we see a sigma photon, we immediately know we are in the dark state. If we perform this experiment without looking at the measurements, it's like averaging over many runs, and we get the same result in either basis. To model what is happening in this case, we may want to perform the QMC algorithm. If we perform the calculation in the Z basis, our error bars are much smaller (since individual trajectories are less erratic) for the same number of trajectories averaged, and thus it may be less computationally expensive to choose the Z basis for such a calculation.

[4]Again assume unit detection efficiency for this discussion.

References

[1] R. Dum, A. S. Parkins, P. Zoller, and C. W. Gardiner. Monte carlo simulation of master equations in quantum optics for vacuum, thermal, and squeezed reservoirs. *Phys. Rev. A*, 46:4382–4396, Oct 1992.

[2] Klaus Molmer. Density Matrices and the Quantum Monte-Carlo Method in Quantum Optics, February 1994.

3 Appendix: Python Code

This appendix includes only the important snippets of the python script. Full code may be found at: *https://github.com/matthewnchow/Monte-Carlo-Wavefunction-Example*. This code requires numpy and matplotlib.pyplot packages.

3.1 Code for 1a

```
# Problem 1a
# Model of decaying Rabi oscillations for a 2 lvl system.
# Working in the {|g> , |e>} basis

def run_1a():
    t_initial = 0     # time variable, in units of 1/Gamma
    t_final = 4      # Molmer's plots go out to 4*(1/Gamma)

    Omega = 6       # Rabi frequency
    t_steps = 1000  # Number of time steps per unravelling
    ts = np.linspace(t_initial, t_final, t_steps)   # Array of times
    psi_0 = np.array([1, 0])    # Atom starts in the ground state
    psi = 1j*np.zeros((t_steps, 2)) # Wavefunction at each time step
    psi[0] = psi_0
    sigma_x = np.array([[0, 1], [1, 0]]) # = Sigma+ + sigma-
    proj_e = np.array([[0, 0], [0, 1]])
    h_eff = -0.5j * proj_e - 1 * Omega*0.5*sigma_x

    def unravel_rabi(h_e=h_eff, steps=t_steps, c_0=psi_0):
        # Function for calculating a single unravelling
        dt = float(t_final - t_initial) / steps  # time step
        c = 1j * np.zeros((steps, 2))  # Coefficients of wavefunction
        c[0] = c_0

        # Take a bunch of time steps in loop below to propagate wvfn
        for i in range(1, steps):
            # Prob = (Gamma=1) * |ce|^2 * dt
            prob_jump = np.abs(c[i-1, 1])**2 * dt
            jump = np.random.rand() < prob_jump
            if jump:
                # Would have to determine which jump if multiple
                # Here, all atom goes to ground if there is a jump
                c[i] = np.array([1, 0])
            else:
                # Evolve according to H_effective and re-normalize
                c[i] = c[i-1] - 1j * dt * np.matmul(h_e, c[i - 1])
                c[i] = c[i]/(np.dot(c[i], np.conjugate(c[i])))
        return c

    psi = unravel_rabi()
    p_g = np.abs(psi[:, 0])**2      # Population in the ground state
    p_e = np.abs(psi[:, 1])**2      # Population in the excited state

    m_trials = 100                 # Molmer averages 100 wvfns
    p_es = np.zeros((m_trials, len(p_e)))
    p_es[0] = p_e

    for m in range(1, m_trials):
        psi = unravel_rabi()
        p_es[m] = np.abs(psi[:, 1])**2

    # Calculate uncertainty according to Molmer
    uncertainty = np.std(p_es, 0)/np.sqrt(m_trials)
```

3.2 Code for 1b

Everything in 1b is the same, except that we now sample from the wait time distribution to find out *when* the next jump happens, instead of asking jump no jump at each time step. Therefore only the adjusted "unravel" function is included.

```
def unravel_rabi_wait(h_e, c_0, steps=t_steps, t_i=t_initial, t_f=t_final):
    # Function for calculating a single unravelling
    dt = float(t_f - t_i) / steps  # time step
    c = 1j * np.zeros((steps, 2))  # Coefficients of wavefunction
    c[0] = c_0
    eta = np.random.rand()
    c_t = c_0*(-1j)**2

    # Take a bunch of time steps in loop below to propagate wvfn
    for i in range(1, steps):
        # calculate the norm of the wvfn and compare to random number
        norm2 = np.abs(np.dot(c_t, np.conjugate(c_t)))
        jump = (norm2 <= eta)
        if jump:
            # All population moves to ground state
            c[i] = np.array([1, 0])
            # Also pick a new random number
            eta = np.random.rand()
            # And reset numerical integration state
            c_t = c[i]
        else:
            # Evolve according to H_effective and re-normalize
            c[i] = c[i-1] - 1j * dt * np.matmul(h_e, c[i - 1])
            c[i] = c[i]/(np.dot(c[i], np.conjugate(c[i])))
            # Performing numerical integration of differential propagator
            #   to find when jump happens
            c_t -= 1j * dt * np.matmul(h_e, c_t)
    return c
```

3.3 Code for 2

Here the general "unravel" function is included, which works for arbitrary H_{eff} and jump operators. Other than that, the code is the same as 1a, but with different inputs (Hamiltonian, jump operators and initial state). We also project onto a different state (the dark state).

```
def unravel(h=h_eff, steps=t_steps, c_0=psi_0, ls=d_q):
    # Function for calculating a single unravelling
    dt = float(t_final - t_initial) / steps  # time step
    c = 1j * np.zeros((steps, len(psi_0)))  # Wavefunction coefficients
    c[0] = c_0
    h_jump = h - np.transpose(np.conjugate(h))

    # Take a bunch of time steps in loop below to propagate wvfn
    for i in range(1, steps):
        dp = dt * 1j*np.dot(np.conjugate(c[i-1]), np.matmul(h_jump, c[i-1]))
        epsilon = np.random.rand()
        jump = epsilon < dp
        if jump:
            cumulative_dpm = 0
            # Calculate each jump probability
            # If epsilon < Sum of dpm up to this point, select jump
            for L in ls:
                c[i] = np.matmul(L, c[i-1])  # Temporary, used to calc dpm
                dpm = dt*np.abs(np.dot(c[i], np.conjugate(c[i])))
                cumulative_dpm += dpm
                if epsilon < cumulative_dpm:
                    c[i] /= np.sqrt(dpm/dt)
                    break
        else:
        # Evolve according to H_effective and re-normalize
            c[i] = c[i-1] - 1j * dt * np.matmul(h, c[i - 1])
            c[i] = c[i]/np.abs((np.dot(c[i], np.conjugate(c[i]))))
    return c
```

3.4 Relevant Clebsch-Gordan Coefficients

These were copied from the Wikipedia page:

`https://en.wikipedia.org/wiki/Table_of_Clebsch%E2%80%93Gordan_coefficients`

using the J1=1, J2=1 tables. Note the table below only lists mg + q = me, since that must be true by angular momentum conservation.

m_e	m_g	q	CG
0	1	-1	$\frac{1}{\sqrt{2}}$
0	0	0	0
0	-1	1	$-\frac{1}{\sqrt{2}}$
+/-1	+/-1	0	+/-$\frac{1}{\sqrt{2}}$
+/-1	0	+/-1	-/+$\frac{1}{\sqrt{2}}$