Hyperfine transition dynamics of D2 line using 4-level system Hamiltonian and QuTiP

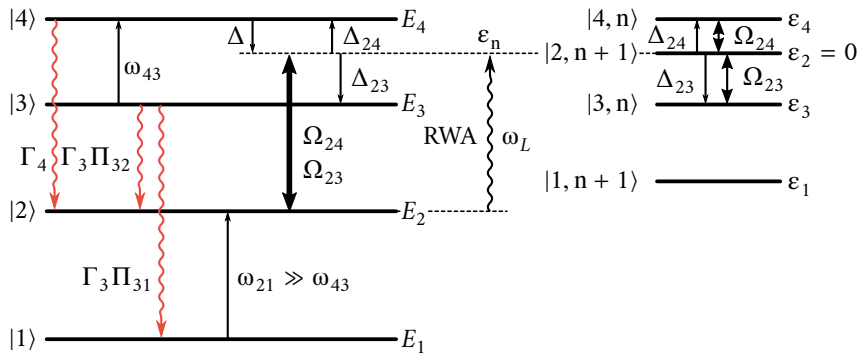
December 21, 2015

by **Martins Bruvelis**

1. Four level system

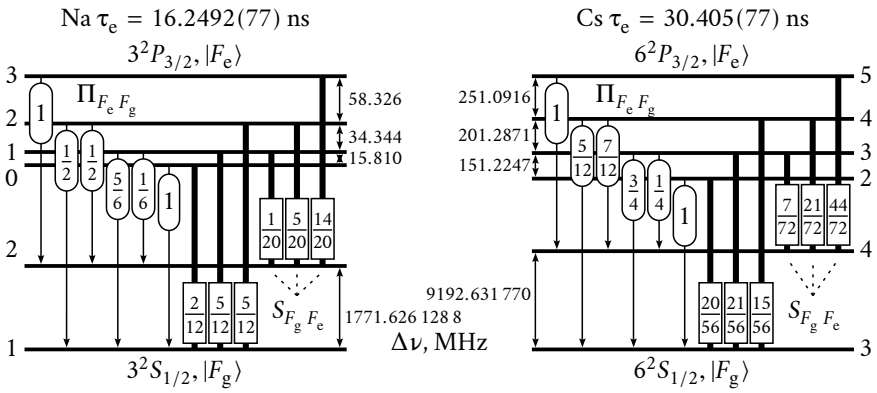
We study population dynamics and effects of optical pumping on spectral line singal for near-resonant cyclic D_2 line transitions in Cs and Na atoms with the help of simplified four-level system. A classical laser field of frequency ω_L is in resonance between highest energy ground state hyperfine component $|2\rangle$ highest energy hyperfine state component $|4\rangle$. Intensity of excitation laser field follows a Guassian pulse shape that is characterize by Rabi frequency coupling strengths $\Omega_{24}(t)$ and $\Omega_{23}(t)$ between states $|2\rangle \leftrightarrow |4\rangle$ and $|2\rangle \leftrightarrow |3\rangle$ respectively. States $|1\rangle$ is a population trapping ground state (dark state) that does not interact with the excitation laser field.

```
In [1]: %%svg
        4-level-schematics.svg
```



Corresponding D_2 line transition hyperfine level schematics in Na nd Cs atoms.

```
In [2]: %%svg
        hyperfine-level-shematics.svg
```



1.1. Atomic and interaction Hamiltonian using rotating wave approximation (RWA)

1.1.1. Lindblad master equation

The standard approach for deriving the equation of motion for a system interacting with its environment is to expand the scope of the system to include the environment. The combined quantum system is closed and its evolution is governed by the von Neumann equation

$$\begin{aligned} \dot{\rho}(t) &= -\frac{i}{\hbar}[H_{tot}, \rho_{tot}(t)] \\ H_{tot} &= H_{sys} + H_{env} + H_{int}, \end{aligned} \tag{1}$$

where the total Hamiltonian H_{tot} includes the original system Hamiltonian H_{sys}, the Hamiltonian for the environment H_{env} and the interaction Hamiltonian H_{int} between the system and its environment. To obtain the dynamics of system H_{sys}, we can perform a partial trace over the environmental degrees of freedom in von Neumann equation. The most general trace-preserving and completely positive form of this evolution is the Lindblad master equation [1], [2], [3]

$$\begin{aligned} \dot{\rho}(t) &= -\frac{i}{\hbar}[H(t), \rho(t)] + \sum_n \frac{1}{2}\left(2C_n\rho(t)C_n^\dagger - \rho(t)C_n^\dagger C_n - C_n^\dagger C_n\rho(t)\right) \\ H(t) &= H_{sys} + H_{int} \\ C_n &= \sqrt{\gamma_n}A_n, \end{aligned} \tag{2}$$

where C_n are wave function collapse operators, A_n are operators through which the environment couples to the system in H_{int} and γ_n are the corresponding decay rates.

1.1.2. Atomic Hamiltonian H_{sys} (RWA)

Let us obtain optical Bloch equation for four-level system. Using dressed state notation given in figure, system Hamiltonian can be written as

$$
\begin{aligned}
H_{sys} &= \varepsilon_1 |1, n+1\rangle \langle 1, n+1| + \varepsilon_2 |2, n+1\rangle \langle 2, n+1| + \varepsilon_3 |3, n\rangle \langle 3, n| + \varepsilon_4 |4, n\rangle \langle 4, n| \\
&= \begin{bmatrix} \varepsilon_1 & 0 & 0 & 0 \\ 0 & \varepsilon_2 & 0 & 0 \\ 0 & 0 & \varepsilon_3 & 0 \\ 0 & 0 & 0 & \varepsilon_4 \end{bmatrix}.
\end{aligned} \tag{3}
$$

And energies of the system Hamiltonain can be written as

$$
\varepsilon_1 = -\hbar\omega_{21} \quad \varepsilon_2 = 0 \quad \varepsilon_3 = \Delta_{23} = -\omega_{43} - \Delta \quad \varepsilon_4 = \Delta_{24} = -\Delta, \tag{4}
$$

where $\omega_{ab} = \Delta E_{ab}/\hbar = (E_a - E_b)/\hbar$ and energy level difference in SI unit system using frequency ν and cyclic frequency ω are written as $\Delta E = h\nu = \hbar\omega$.

Corresponding four-level system decay channels are following, A_1 from state $|4, n\rangle$ to state $|2, n+1\rangle$, A_2 from state $|3, n\rangle$ to state $|2, n+1\rangle$, A_3 from state $|3, n\rangle$ to state $|1, n+1\rangle$, such that

$$
\begin{aligned}
A_1 &= |2, n+1\rangle \langle 4, n| = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\
A_2 &= |2, n+1\rangle \langle 3, n| = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\
A_3 &= |1, n+1\rangle \langle 3, n| = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.
\end{aligned} \tag{5}
$$

Corresponding wave function collapse operators are

$$
\begin{aligned}
C_1 &= \sqrt{\gamma_1} A_1 \quad C_2 = \sqrt{\gamma_2} A_2 \quad C_3 = \sqrt{\gamma_3} A_3 \\
\gamma_1 &= \Pi_{42}\Gamma_4 \quad \gamma_2 = \Pi_{32}\Gamma_3 \quad \gamma_3 = \Pi_{31}\Gamma_3 \\
\Pi_{42} &= 1 \qquad\qquad\qquad \Pi_{31} = 1 - \Pi_{32},
\end{aligned} \tag{6}
$$

where $\Gamma_4 = \Gamma_3 = \Gamma = 1/\tau_e$ is decay rate of excited states, τ_e is natural lifetime of excited state and $\Pi_{F_e F_g}$ is branching ration for transition $|F_e\rangle \to |F_g\rangle$.

1.1.3. Laser-atom interaction Hamiltonian H_{int} using rotating wave approximation (RWA)

The interaction of bound particles with laser light most often originates with the electric-dipole interaction. Therefore, the atom-laser interaction Hamiltonian H_{int} in the dipole approximation is given by interaction energy operator, that is the projection of the electric dipole moment $\vec{d}$ onto the electric field

$$
\begin{aligned}
H_{int} &= -\vec{d} \cdot \vec{E}(t) \\
\vec{E}(t) &= \hat{\mathbf{e}} E(t) \cos(\omega_L t - \varphi) \\
E(t) &= E_0 \exp\left(-2\left(\frac{t}{\tau_{tr}}\right)^2\right)
\end{aligned} \tag{7}
$$

and where the atomic dipole operator $\vec{d}$ is given in terms of the atomic electron position $\vec{r}_e$ as

$$
\vec{d} = -e\vec{r}_e, \tag{8}
$$

where we denote the fundamental charge by e, so that the electron charge is $q = -e$. The dipole transition moment between states $|\psi_a\rangle$ and $|\psi_b\rangle$, projected onto the field unit vector $\hat{\mathbf{e}}$, is

$$d_{\psi_a,\psi_b} = \left\langle \psi_a \middle| \vec{d} \cdot \hat{\mathbf{e}} \middle| \psi_b \right\rangle. \tag{9}$$

For linear polarization we can take the unit vector $\hat{\mathbf{e}}$ to be real and write the interaction Hamiltonain between states $|\psi_a\rangle$ and $|\psi_b\rangle$ as [4]

$$H_{int,ab} = H_{int,ab} = -d_{\psi_a,\psi_b} E(t) \cos(\omega_L t - \varphi) \equiv \hbar \Omega_{ab}(t) \cos(\omega_L t + \varphi), \tag{10}$$

where we denote the Rabi frequency by $\Omega_{ab}(t)$. There are many definitions of Rabi frequency Ω in the literature. The chosen Rabi frequency $\Omega_{ab}(t)$ refers to the frequency of population oscillations of resonant two-state system, i.e., when the Rabi frequency remains constant, the populations of resonant two-state system undergo periodic Rabi oscillations at the Rabi frequency Ω. The Rabi frequency is defined by

$$\Omega_{ab}(t) \equiv -\frac{d_{\psi_a,\psi_b} E(t)}{\hbar}. \tag{11}$$

In rotating reference frame off-diagonal elements of the atom-laser interaction Hamiltonian $H_{int,ab}$ [4] are

$$H_{int,ab} = H^*_{int,ba} = -d_{\psi_a,\psi_b} E(t) \cos(\omega_L t - \varphi) e^{-i\omega_L t} = \frac{1}{2}\hbar \Omega_{ab}(t) \left(e^{-i\varphi} + e^{-2i\omega_L t + i\varphi}\right) \tag{12}$$

Rotating wave approximation (RWA) of atom-laser interaction is performed by disregarding the terms that vary as $2\omega_L t$ (counter-rotating terms) when added to constant terms

$$\cos(\omega_L t - \varphi) e^{-i\omega_L t} = \frac{1}{2}\left(e^{-i\varphi} + e^{-2i\omega_L t + i\varphi}\right) \xrightarrow{RWA} \frac{1}{2} e^{-i\varphi} \tag{13}$$

Note that the absolute value of the phase φ is not controllable and only when one compares two pulses, both within a coherence time, or pulses affecting two locations, does one need to keep track of phases, therefore, phase φ can be taken as zero at any convenient time

$$\frac{1}{2} e^{-i\varphi} \to \frac{1}{2}. \tag{14}$$

Thus the Rabi frequency can be made real $\Omega_{ab}(t)$ such that $\Omega^*_{ab}(t) = \Omega_{ab}(t)$. Thus we obtain the atom-laser interaction Hamiltonain in the rotating wave approximation between state $|\psi_a\rangle$ and $|\psi_b\rangle$ as

$$H_{int,ab} = \frac{1}{2}\hbar \Omega_{ab}(t). \tag{15}$$

And the atom-laser interaction Hamiltonain in the rotating wave approximation is given by

$$H_{int} = \frac{\hbar\Omega_{23}(t)}{2} |2, n+1\rangle \langle 3, n| + \frac{\hbar\Omega^*_{23}(t)}{2} |3, n\rangle \langle 2, n+1| + \frac{\hbar\Omega_{24}(t)}{2} |2, n+1\rangle \langle 4, n| + \frac{\hbar\Omega^*_{24}(t)}{2} |4, n\rangle \langle 2, n+1| \tag{16}$$

When simulating the dynamics of the Hamiltonian we are modeling hyperfine transitions between states $|F_g\rangle \to |F_e\rangle$ without taking into account Zeeman sublevels, therefore, for linearly polarized excitation laser, effective reduced dipole moment $d_{eff(F_g \to F_e)}$ [5], [6] can be expressed as

$$\left|d_{eff(F_g\to F_e)}\right|^2 = \left|\left\langle F_g \middle\| \vec{d}\cdot\hat{\mathbf{e}} \middle\| F_e\right\rangle\right|^2 = S_{F_gF_e}\left|\left\langle J_g \middle\| \vec{d}\cdot\hat{\mathbf{e}} \middle\| J_e\right\rangle\right|^2$$
$$S_{F_gF_e} = (2F_e+1)(2J_g+1)\begin{Bmatrix} J_g & J_e & 1\\ F_e & F_g & 1\end{Bmatrix}^2 \tag{17}$$
$$\sum_{F_e} S_{F_gF_e} = 1,$$

where $S_{F_gF_e}$ are dimensionless relative hyperfine transition-strength factors of each of the $|F_g\rangle \to |F_e\rangle$ transitions. Numerical value of the reduced dipole matrix element $\left\langle J_g \middle\| \vec{d}\cdot\hat{\mathbf{e}} \middle\| J_e\right\rangle$ can be calculated using excited state lifetime from expression [7]

$$\frac{1}{\tau_e} = \Gamma_{J_eJ_g} = \frac{\omega_0^3}{3\pi_0\hbar c^3}\frac{2J_g+1}{2J_e+1}\left|\left\langle J_g \middle\| \vec{d} \middle\| J_e\right\rangle\right|^2. \tag{18}$$

It is convenient to rewrite Rabi frequency $\Omega_{ab}(t)$ using reduced Rabi frequency $\Omega_{red}(t)$ and relative hyperfine transition-strength factors $S_{F_gF_e}$, where reduced Raby frequency is defined by

$$\Omega_{red}(t) \equiv \frac{E(t)\left\langle J_g \middle\| \vec{d} \middle\| J_e\right\rangle}{\hbar}. \tag{19}$$

This allows us to obtain Rabi frequency between states $|\psi_a\rangle$ and $|\psi_b\rangle$ as

$$\Omega_{ab}(t) = \Omega_{red}(t)\sqrt{S_{F_aF_b}} \tag{20}$$

1.2. Remarks

1.2.1. Critical Rabi frequencies for reduced Rabi frequency Ω_{red}

Critical Rabi frequency $\Omega_{red,cr}$ is used to estimate at what reduced Rabi frequency value $\approx 63\%$ of the population will be trapped in a dark state after laser-atom interaction and effects of optical pumping become pronounced in spectral line signal.

$$\Omega_{red,cr} = \sqrt{\frac{4\tau_e\omega_{43}^2}{\tau_{tr}\Pi_{31}S_{32}\sqrt{\pi}}} \tag{21}$$

Critical Rabi frequency $\Omega_{red,cr99}$ is used to estimate at what reduced Rabi frequency value $\approx 99\%$ of the population will be trapped in a dark state after laser-atom interaction, and leading to population depletion in time period smaller than the interaction time τ_{tr}.

$$\Omega_{red,cr99} = \sqrt{-\ln\left(1-\frac{99}{100}\right)}\Omega_{red,cr} \approx 2.15\Omega_{red,cr} \tag{22}$$

1.2.2. Spectral line singal

Spectral line signal $J(\Delta, \Omega_{red})$ is given by a number of photons emitted from an atom during transit time through the excitation laser zone, therefore we can express spectral line signal as

$$J(\Delta, \Omega_{red}) = \Gamma\int_{-\infty}^{\infty}(\rho_{4,4}(t)+\rho_{3,3}(t))\,dt. \tag{23}$$

By using adiabatic aproximation and optical Bloch equations from Lindblad master equation, we can obtain approximate analitic expression of spectral line signal $J(\Delta, \Omega_{red})$ as

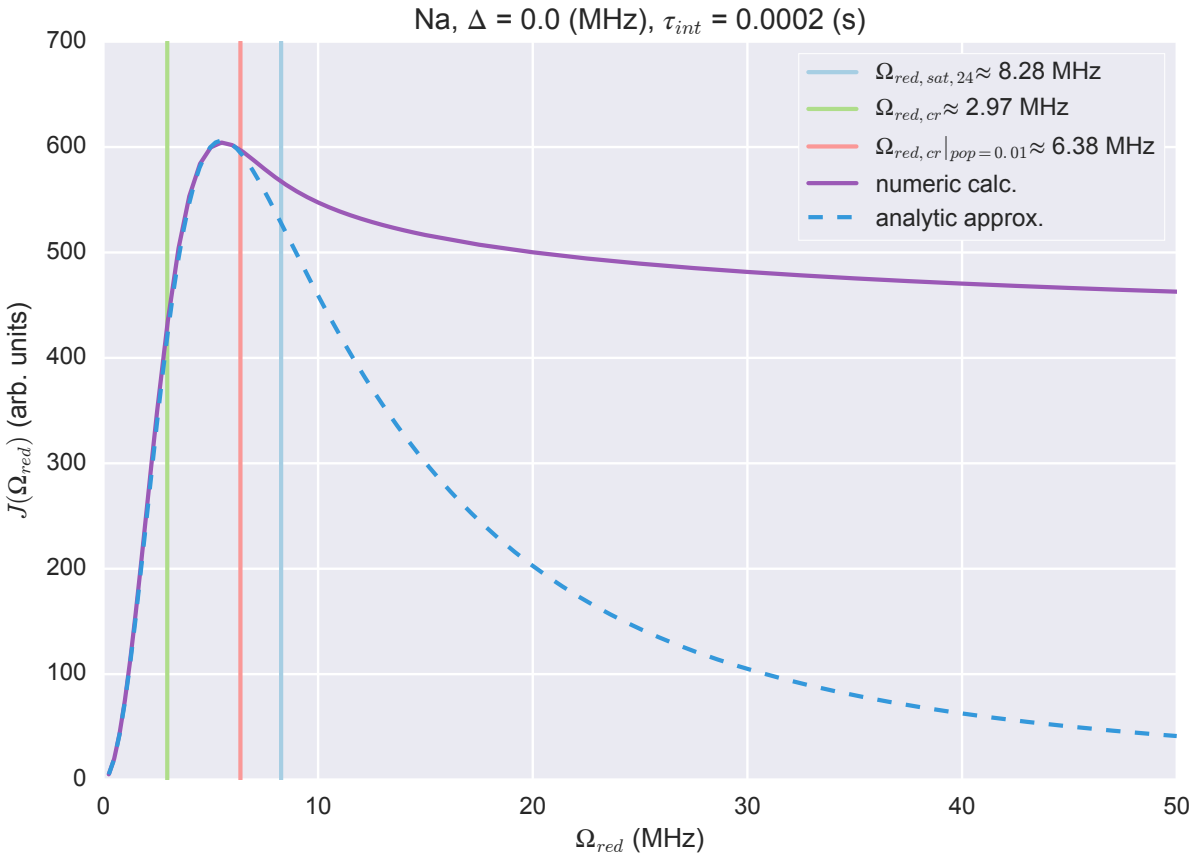
$$J(\Delta, \Omega_{red}) = \frac{S_{24} 4 \left(-\omega_{43} - \Delta\right)^2}{S_{23} \left(4 \left(-\Delta\right)^2 + S_{24} \Omega_{red}^2 + \Gamma^2\right) \Pi_{31}} \times \left(1 - \exp\left(-\tau_{tr} \frac{\sqrt{\pi} \Gamma \Pi_{31} S_{23} \Omega_{red}^2}{8 \left(-\omega_{43} - \Delta - \frac{1}{2}\sqrt{S_{24}} \Omega_{red} - \frac{1}{2}\sqrt{S_{23}} \Omega_{red}\right)^2}\right)\right). \tag{24}$$

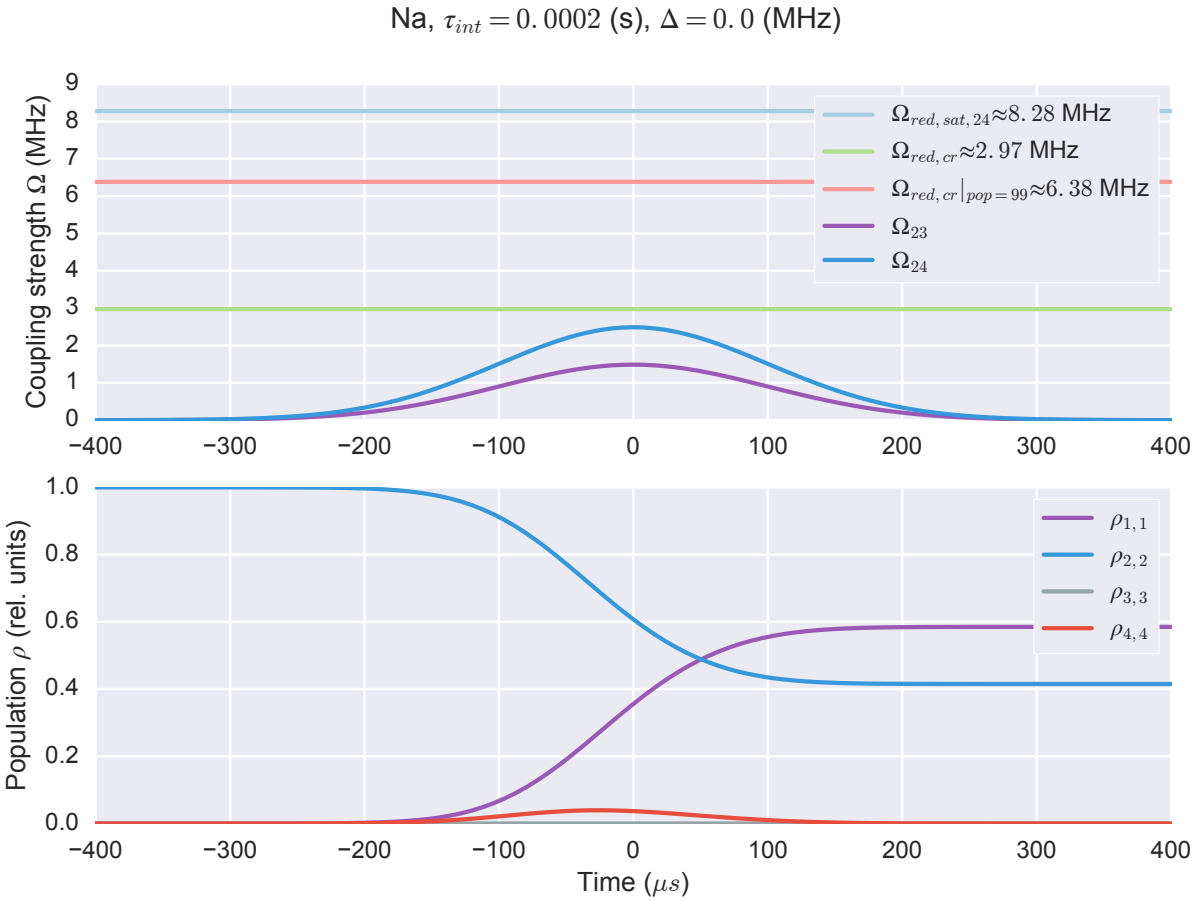
2. Resulsts

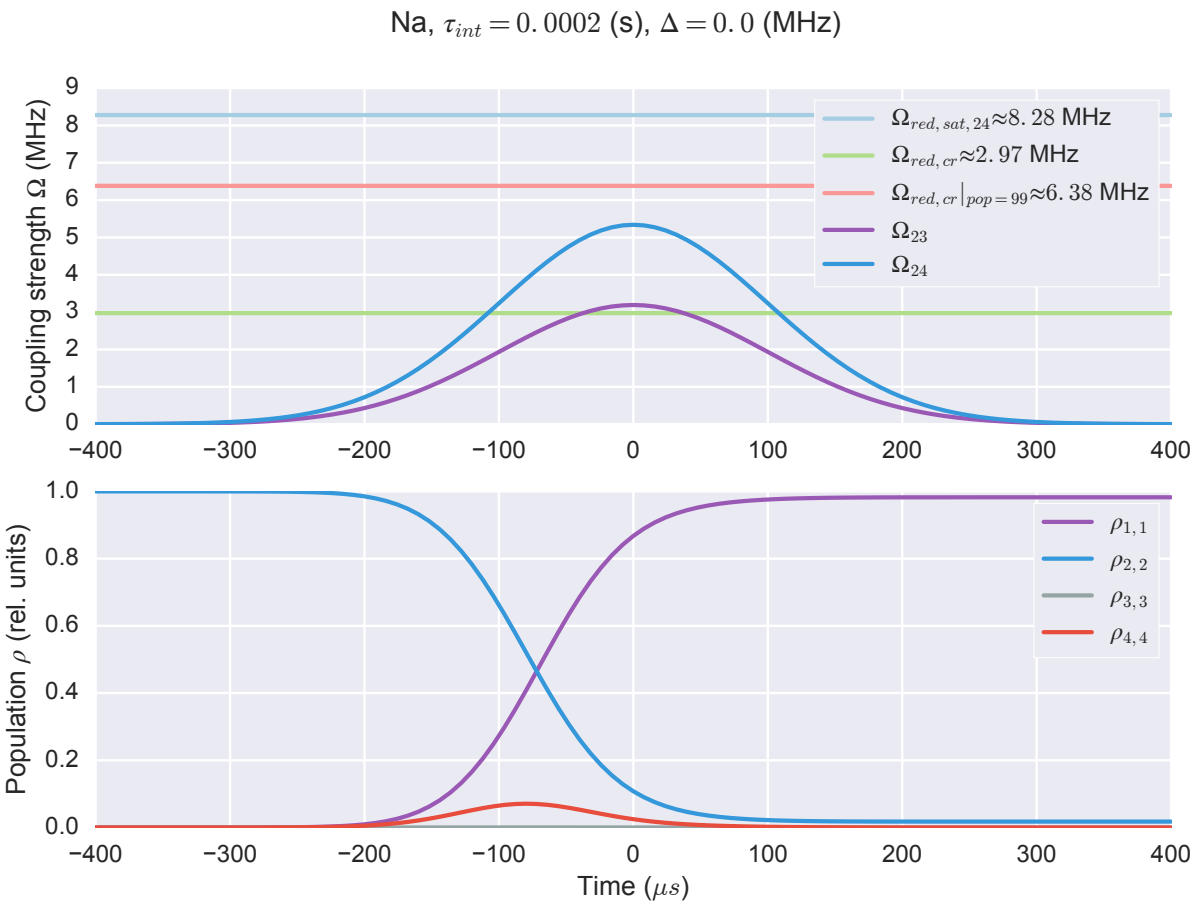
2.1. Compare and review $J(\Delta, \Omega_{red})$ evaluated numerically from density matrix with approximate analytic expression.

2.1.1. Compare spectral singnal $J(\Omega_{red})$ and review corresponding population dinamics in case of Na atoms

```
In [7]: las_plot_jored(na_jored_expcase_list_1)
        plot4lme(expcase_list[0], expcase_list[0].result)
        plot4lme(expcase_list[1], expcase_list[1].result)
```

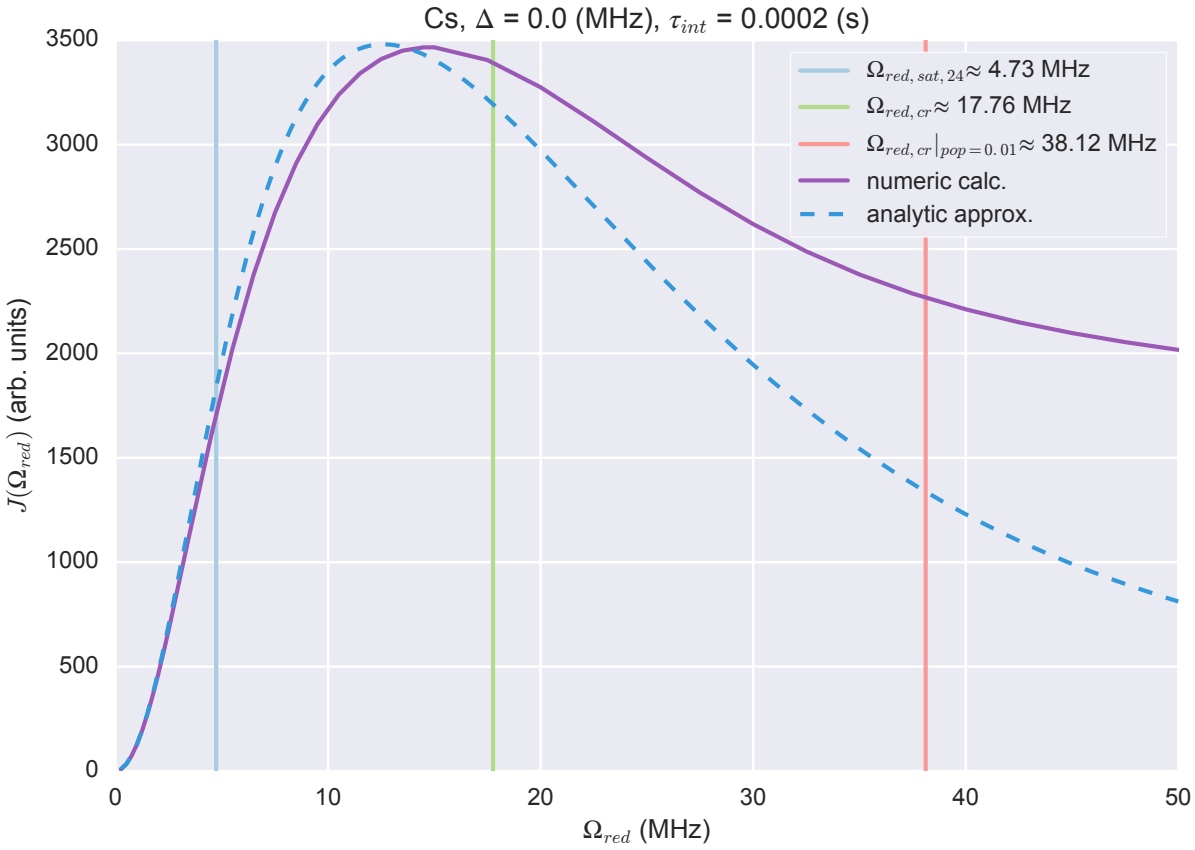


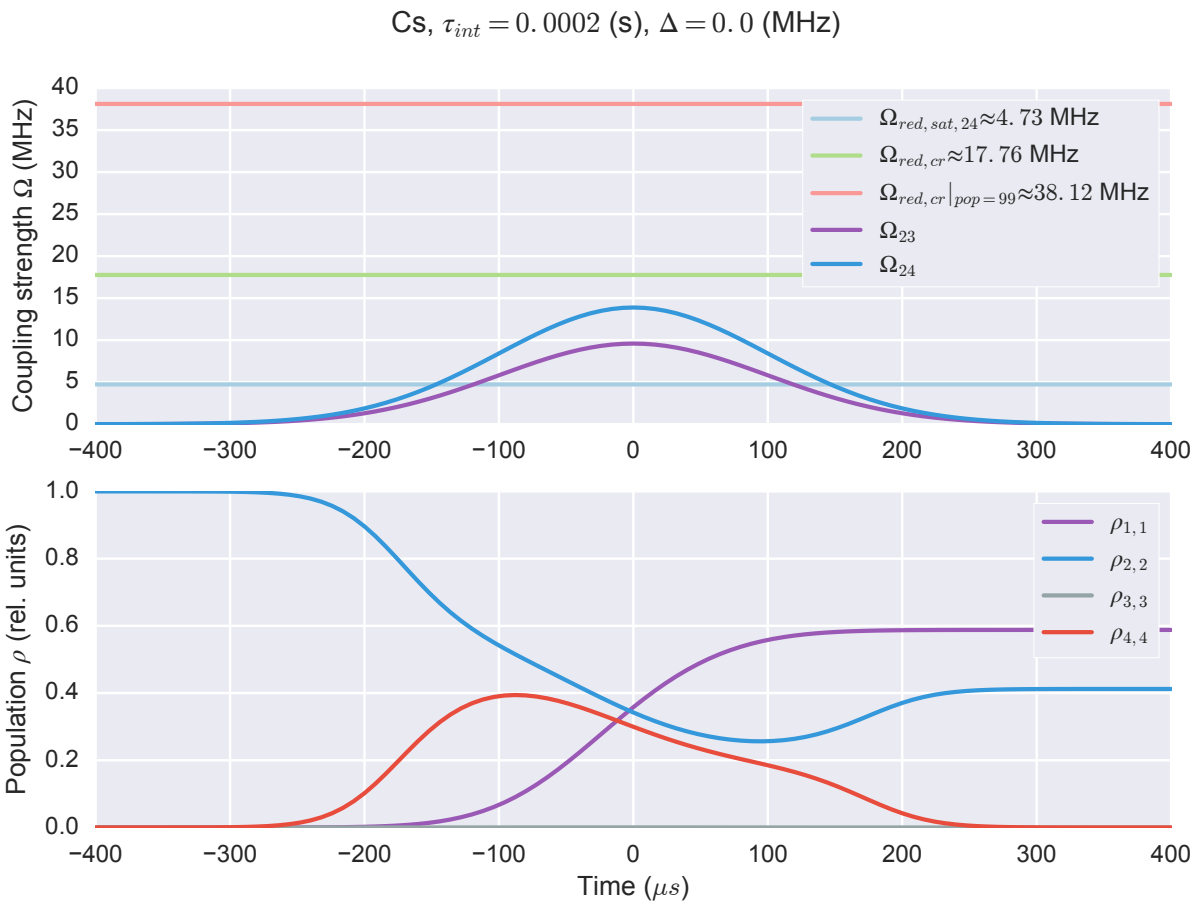

Na, $\tau_{int} = 0.0002$ (s), $\Delta = 0.0$ (MHz)
Coupling strength Ω (MHz)
$\Omega_{red,\,sat,\,24} \approx 8.28$ MHz
$\Omega_{red,\,cr} \approx 2.97$ MHz
$\Omega_{red,\,cr}|_{pop=99} \approx 6.38$ MHz
Ω_{23}
Ω_{24}
Population ρ (rel. units)
$\rho_{1,1}$
$\rho_{2,2}$
$\rho_{3,3}$
$\rho_{4,4}$
Time (μs)

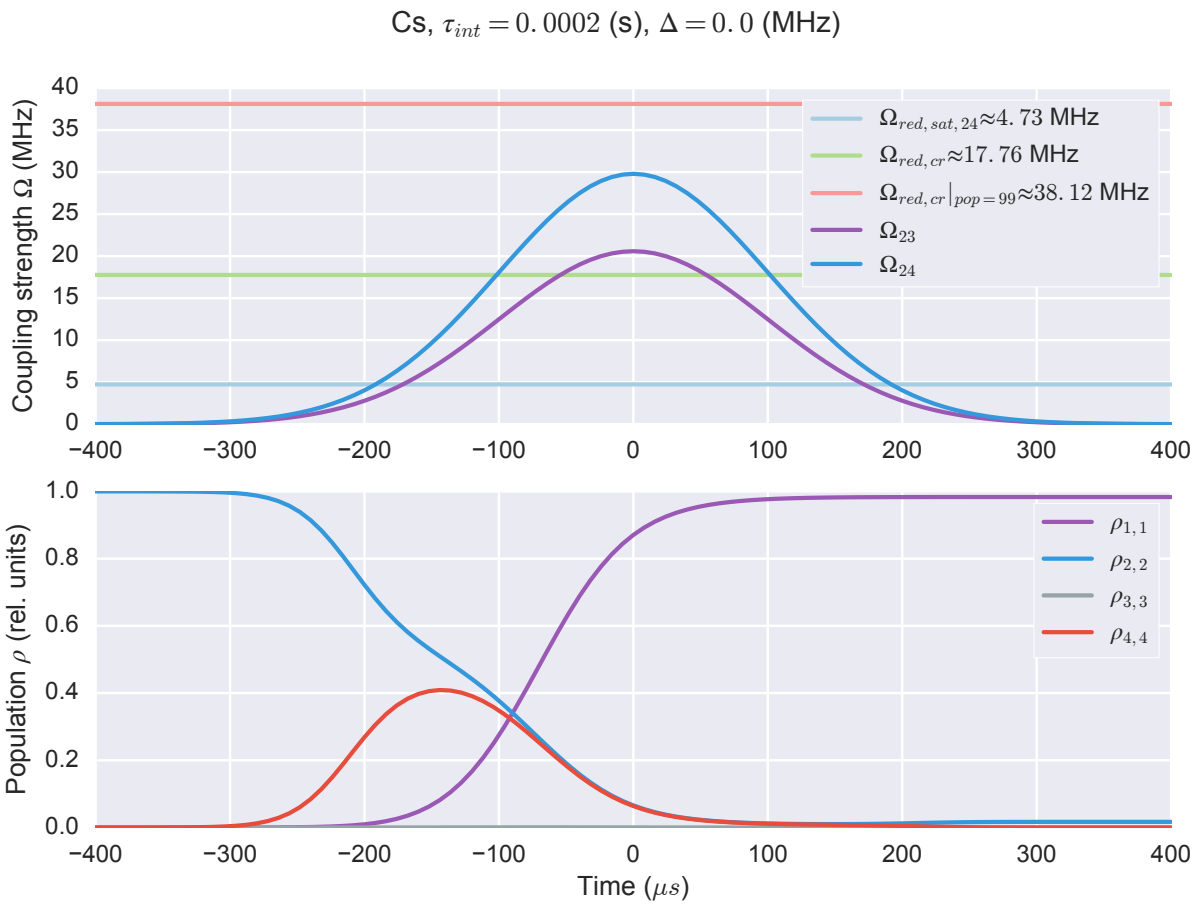


2.1.2. Compare spectral singnal $J(\Omega_{red})$ and review corresponding population dinamics in case of Cs atoms

```
In [8]: las_plot_jored(cs_jored_expcase_list_1)
        plot4lme(expcase_list[2], expcase_list[2].result)
        plot4lme(expcase_list[3], expcase_list[3].result)
```

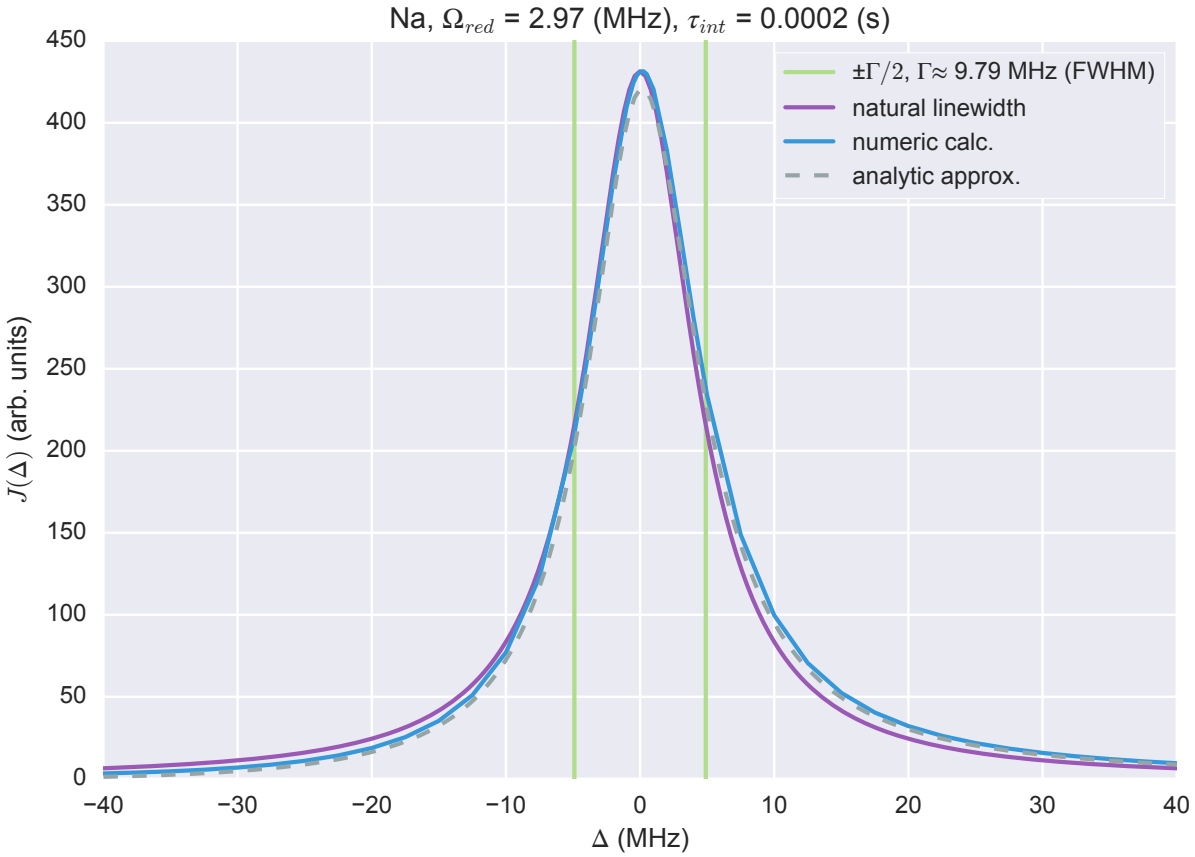

Cs, Δ = 0.0 (MHz), τ_{int} = 0.0002 (s)
$\Omega_{red,\,sat,\,24}\approx$ 4.73 MHz
$\Omega_{red,\,cr}\approx$ 17.76 MHz
$\Omega_{red,\,cr}|_{pop=0.01}\approx$ 38.12 MHz
numeric calc.
analytic approx.
$J(\Omega_{red})$ (arb. units)
Ω_{red} (MHz)
0
500
1000
1500
2000
2500
3000
3500
0
10
20
30
40
50

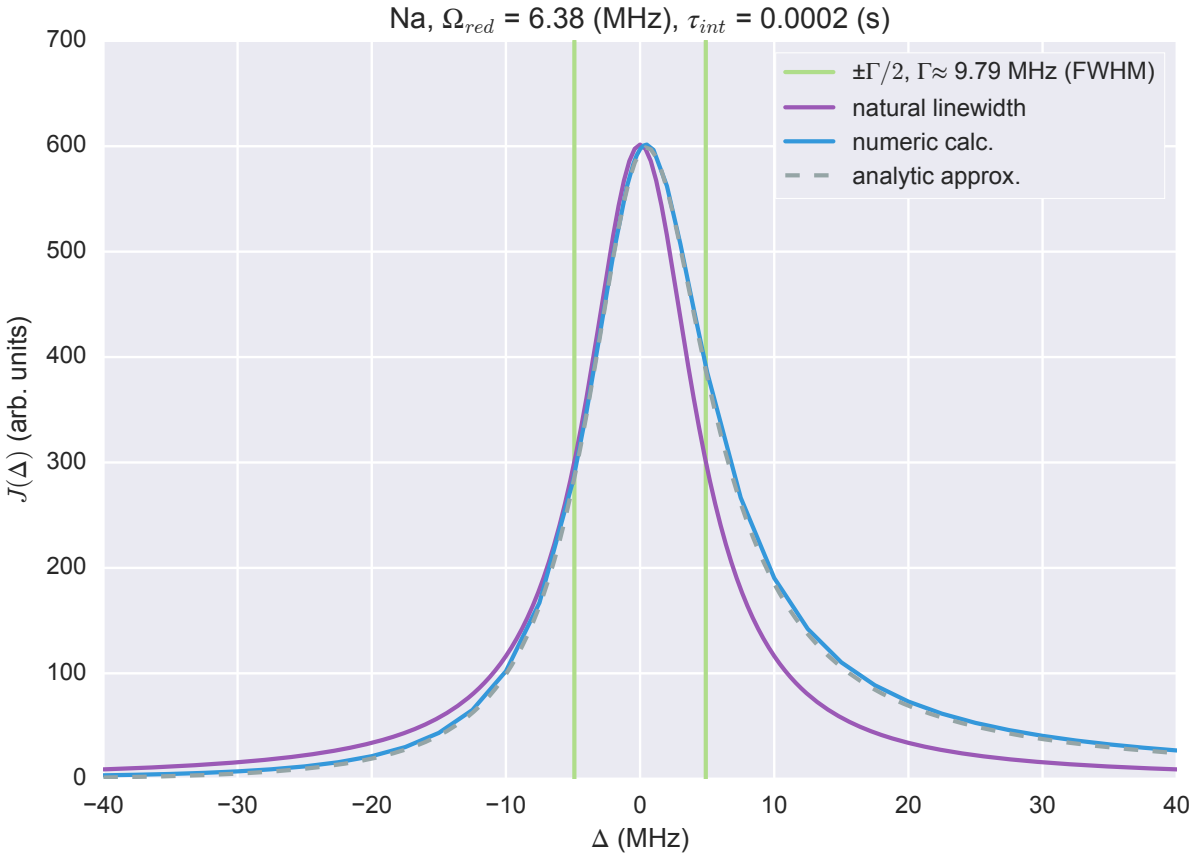

Cs, $\tau_{int} = 0.0002$ (s), $\Delta = 0.0$ (MHz)
Coupling strength Ω (MHz)
$\Omega_{red,sat,24} \approx 4.73$ MHz
$\Omega_{red,cr} \approx 17.76$ MHz
$\Omega_{red,cr}|_{pop=99} \approx 38.12$ MHz
Ω_{23}
Ω_{24}
Population ρ (rel. units)
$\rho_{1,1}$
$\rho_{2,2}$
$\rho_{3,3}$
$\rho_{4,4}$
Time (μs)

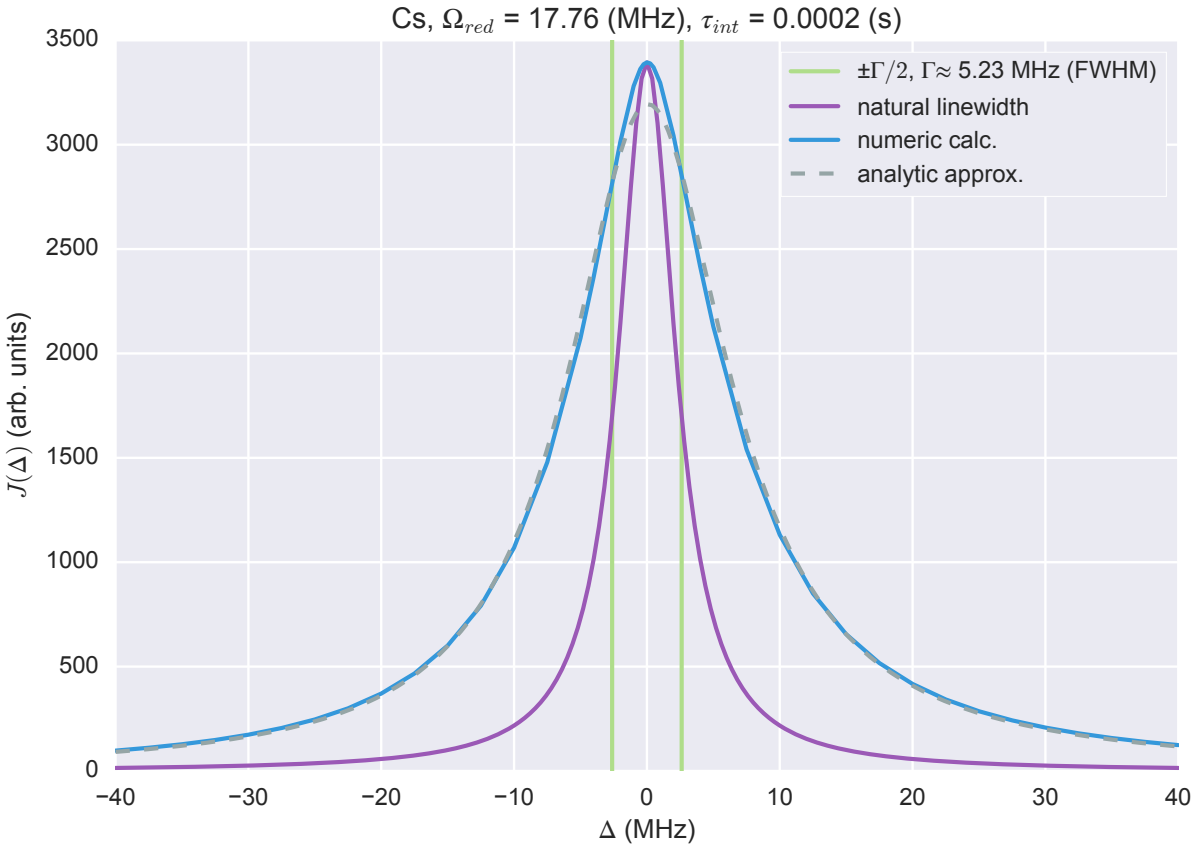


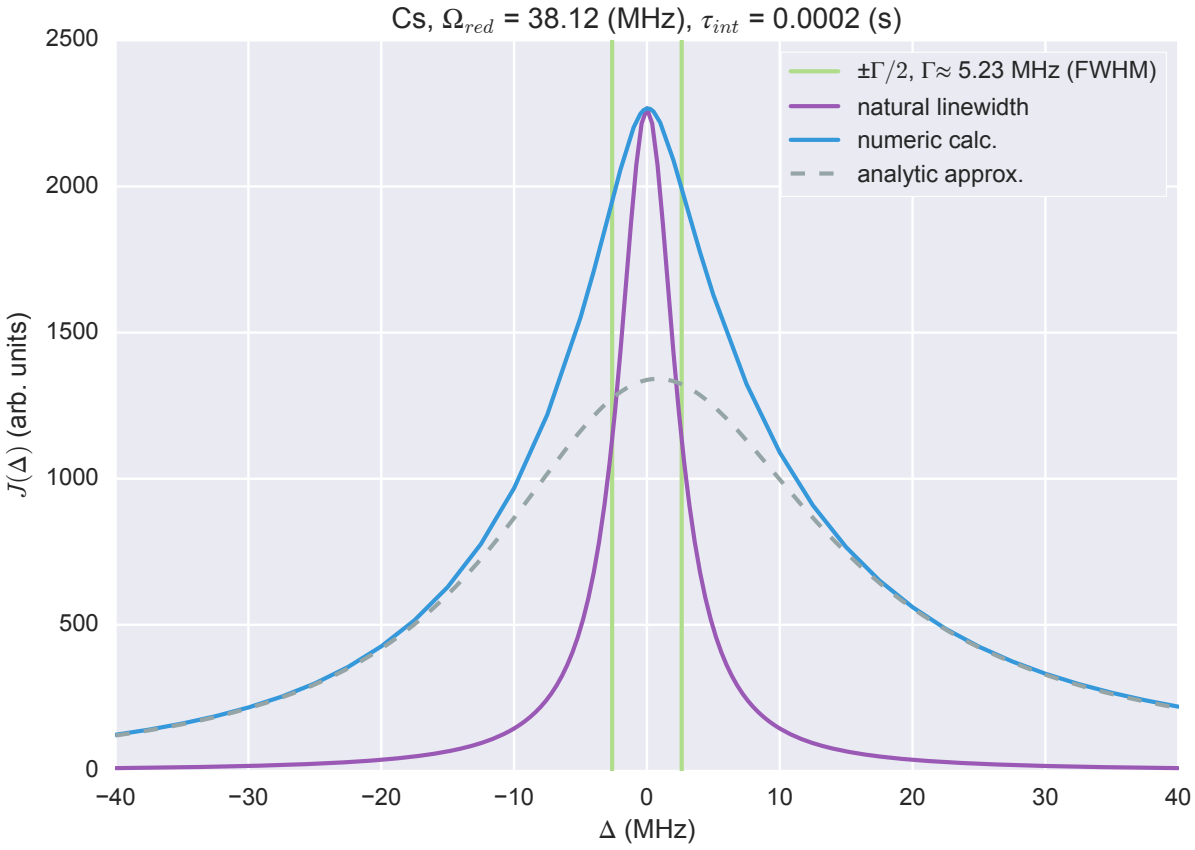
2.1.3. Compare spectral singnal $J(\Delta)$ of Na and Cs atoms

```
In [9]: las_plot_jdelta(na_jdelta_expcase_list_1)
        las_plot_jdelta(na_jdelta_expcase_list_2)
        las_plot_jdelta(cs_jdelta_expcase_list_1)
        las_plot_jdelta(cs_jdelta_expcase_list_2)
```


Na, Ω_{red} = 2.97 (MHz), τ_{int} = 0.0002 (s)
$\pm\Gamma/2$, $\Gamma\approx$ 9.79 MHz (FWHM)
natural linewidth
numeric calc.
analytic approx.
$J(\Delta)$ (arb. units)
Δ (MHz)
450
400
350
300
250
200
150
100
50
0
−40
−30
−20
−10
0
10
20
30
40


Na, Ω_{red} = 6.38 (MHz), τ_{int} = 0.0002 (s)
$\pm\Gamma/2$, $\Gamma\approx$ 9.79 MHz (FWHM)
natural linewidth
numeric calc.
analytic approx.
$J(\Delta)$ (arb. units)
Δ (MHz)
700
600
500
400
300
200
100
0
−40
−30
−20
−10
0
10
20
30
40


Cs, Ω_{red} = 17.76 (MHz), τ_{int} = 0.0002 (s)
$\pm\Gamma/2$, $\Gamma\approx$ 5.23 MHz (FWHM)
natural linewidth
numeric calc.
analytic approx.
$J(\Delta)$ (arb. units)
Δ (MHz)
3500
3000
2500
2000
1500
1000
500
0
−40
−30
−20
−10
0
10
20
30
40



3. Simulating the dynamics of 4-level system with QuTiP

Please note that when simulating the dynamics of the total Hamiltonian:

1. We set the reduced Plank constant value from QuTiP backend, i.e., it is set to

$$\hbar = 1 \tag{25}$$

2. For laser-atom interaction Hamiltonian we are using a precomputed product of all of the coefficients before time-depenent exponent function

$$hc f_{ab} = \frac{\hbar \Omega_{red} \sqrt{S_{F_a F_b}}}{2} \tag{26}$$

3.1. Simulating the dynamics for following experimental conditions:

1. Compute population dinamics for $\Omega_{red} = \Omega_{red,cr}$ and $\Omega_{red} = \Omega_{red,cr99}$.
2. Compute $J(\Omega_{red})$ for Ω_{red} from 0.25 MHz to 50 MHz.
3. Compute $J(\Delta)$ for Δ from -40 MHz to 40 MHz.

4. Compare $J(\Delta, \Omega_{red})$ evaluated numerically from density matrix with approximate analytic expression.

3.2. Setup imports

```
In [3]: %pylab inline
        %config InlineBackend.figure_formats = {'svg',}
        import seaborn as sns
        flatui = ["#9b59b6", "#3498db", "#95a5a6", "#e74c3c", "#34495e", "#2ecc71"]
        # rc('axes', prop_cycle=cycler('color', flatui))
        # sns.palplot(sns.color_palette(flatui))
        sns.set(palette=flatui)
        rc('svg', fonttype='none')
        from qutip import *
        from fractions import Fraction
        from IPython.display import Markdown
```

```
Populating the interactive namespace from numpy and matplotlib
```

3.3. Define functions for computing 4-level RWA Hamiltonian

```
In [4]: def solve4lme(args):
            '''
            Solve RWA Hamiltonian for 4-level system.
            RWA Hamiltonian states |1,n+1⟩, |2,n+1⟩, |3,n⟩, |4,n⟩.
            RWA Hamiltonian energies ε1=-w21, ε2=0, ε3=-w43-Δ, ε4=-Δ.
            A classical laser field is in resonance between states |2⟩↔|4⟩.
            Excitation schematics (bare atomic states picture)
                #           |4⟩
                #       -----------E4
                #         ↓Δ
                #       -----------             |3⟩
                #       / ↓Γ*П42          ---------------E3
                #  Ω24 ↕  ↓              / ↓П32*Γ      ↓(1-П32)*Γ
                #     /   ↓          Ω23↕  ↓           ↓
                #    ------------------------E2       ↓
                #             |2⟩               -----------E1
                #                                    |1⟩

            Usage::

                >>> args = {'delta': 0,  # detuning Δ from excited state |4⟩ in s^-1
                        'gamma': 1,      # excited state decay rate Γ
                        'tint': 5920,    # laser-atom interaction time, np.float32
                        'w21': 1756,     # hyperfile splitting 1↠2
                        'w43': 48,       # hyperfile splitting 3↠4
```

```
            'hcf23': 3,      # interaction coeff 2↔3, np.float32
            'hcf24': 5,      # interaction coeff 2↔4, np.float32
            'cbr1': 1,       # branching ratio П42 for |2⟩→|4⟩
            'cbr2': 7/12,    # branching ratio П32 for |3⟩→|2⟩
            'cbr3': 5/12,    # branching ratio П31=(1-П32) for |3⟩→|1⟩
            'nsteps': 50000} # Max. number of internal ode steps/call.
    >>> results = solve4lme(args)

:param args: A dictionary of the form ``(parameter, value)``.
:return: QuTiP Result object with mesolve data.
:rtype: A :class:`qutip.solver.Result`
'''

# Define atomic states |1>, |2>, |3>, |4>
st1, st2, st3, st4 = map(lambda st: basis(4, st), range(4))

# Operators for the diagonal elements of atomic Hamiltonian
# outer product of |1><1|, |2><2|, |3><3|, |4><4|
sig11, sig22, sig33, sig44 = map(ket2dm, [st1, st2, st3, st4])

# Operators for the off-diagonal elements of Hamiltonian
# Interaction Hamiltonian |2>↔|3> and |2>↔|4>
# Decay modes (collapse operators) for |4>→|2>, |3>→|2>, |3>→|1>
sig24 = st2 * st4.dag()  # |2><4|
sig23 = st2 * st3.dag()  # |2><3|
sig13 = st1 * st3.dag()  # |1><3|

# Define time independent collapse operators
# collapse operator for 4->2, sqrt(П42*Г) * |2><4|
C1 = np.sqrt(args['cbr1'] * args['gamma']) * sig24
# collapse operator for 3->2, sqrt(П32*Г) * |2><3|
C2 = np.sqrt(args['cbr2'] * args['gamma']) * sig23
# collapse operator for 3->1, sqrt(П31*Г) * |1><3|
C3 = np.sqrt(args['cbr3'] * args['gamma']) * sig13

# Define list of collapse operators
c_op_list = [C1, C2, C3]

# Define time vector
t = linspace(-args['tint']*2, args['tint']*2, 101)

# Set up the time independent system Hamiltonians
# ε1=-w21, ε2 = 0, ε3=-w43-Δ, ε4=-Δ.
# state1 energy level position ε1=-w21
HS1 = -args['w21'] * sig11
# state3 energy level position ε3=-w43-Δ
```

```
    HS3 = (-args['w43'] - args['delta']) * sig33
    # state4 energy level position ε4=-Δ
    HS4 = -args['delta'] * sig44

    # Set up operators for the time varying Hamiltonians
    # for laser-atom interaction Ω23(t) and Ω24(t)
    HI1 = sig23.dag() + sig23
    HI2 = sig24.dag() + sig24

    # Set up the time varying RWA Hamiltonian with time dependant
    # coefficients based on QuTip Cython string functions format
    HRWA = [HS1, HS3, HS4,
            [HI1, 'hcf23 * exp(-2*(t / tint) ** 2)'],
            [HI2, 'hcf24 * exp(-2*(t / tint) ** 2)']]

    # Define initial state as state |2>
    psi0 = st2

    # Define ODE solver options
    opts=Odeoptions()
    opts.nsteps = args['nsteps']

    # Workaround for QuTip version 3.1.0 error with Cython string functions
    # Convert Cython string function variable values to numpy.float32
    # # error: two or more data types in declaration specifiers
    # # typedef npy_double _Complex __pyx_t_npy_double_complex;
    args['hcf23'] = np.float32(args['hcf23'])
    args['hcf24'] = np.float32(args['hcf24'])

    # Solve RWA Hamiltonian
    output = mesolve(HRWA, psi0, t, c_op_list, [sig11, sig22, sig33, sig44],
                     args=args, options=opts, progress_bar=True)
    # return mesolve
    return output

def plot4lme(atomdata, results=None, saveplot=None):
    '''Plot QuTiP Result object.'''

    if results != None:
        rs = results
    else:
        rs = qload(atomdata.filename)

    rho11, rho22, rho33, rho44 = rs.expect
    timescale = 1e6
    t = rs.times * timescale
```

```
# Define pump strength as a function of time for plotting
wp = lambda t, tw, A: A * np.exp(-2*(t / tw) ** 2)

# Plot the results
fig = figure()
subplot(211)
if atomdata != None:
    # colors #a6cee3 #2078b4 #afdd8a #35a12e #fa9897 #e31a1c
    detuning = atomdata.delta
    detuning_MHz = detuning / (timescale * 2*pi)
    detuning_MHz_str = str(round(detuning_MHz,2))
    fig.suptitle(atomdata.symbol + ', $τ_{int} = %s$ (s), ' % str(atomdata.tint) \
                 + '$Δ = %s$ (MHz)' % detuning_MHz_str)

    ored_sat_24_MHz = atomdata.osat / (timescale * 2 * pi * np.sqrt(atomdata.st24))
    ored_sat_24_MHz_str = str(round(ored_sat_24_MHz,2))
    axhline(y=ored_sat_24_MHz, color='#a6cee3',
            label='$Ω_{red,sat,24} ≈ %s$ MHz' % ored_sat_24_MHz_str)

    ored_cr_MHz = atomdata.__class__(atomdata.tint).ocr \
                / (timescale * 2 * pi)
    ored_cr_MHz_str = str(round(ored_cr_MHz,2))
    axhline(y=ored_cr_MHz, color='#afdd8a',
            label='$Ω_{red,cr} ≈ %s$ MHz' % ored_cr_MHz_str)

    ored_cr_pop_99_MHz = atomdata.__class__(atomdata.tint, pop=99).ocr \
                       / (timescale * 2 * pi)
    ored_cr_pop_99_MHz_str = str(round(ored_cr_pop_99_MHz,2))
    axhline(y=ored_cr_pop_99_MHz, color='#fa9897',
            label='$Ω_{red,cr}|_{pop=99} ≈ %s$ MHz' % ored_cr_pop_99_MHz_str)
plot(t, wp(t, atomdata.tint * timescale,
           atomdata.hcf23 * 2 / (timescale * 2*pi)), '-', label='$Ω_{23}$')
plot(t, wp(t, atomdata.tint * timescale,
           atomdata.hcf24 * 2 / (timescale * 2*pi)), '-', label='$Ω_{24}$')
ylabel('Coupling strength $Ω$ (MHz)')
lg_1 = legend()
lg_1.draw_frame(True)

subplot(212)
plot(t, rho11, '-', label='$ρ_{1,1}$')
plot(t, rho22, '-', label='$ρ_{2,2}$')
plot(t, rho33, '-', label='$ρ_{3,3}$')
plot(t, rho44, '-', label='$ρ_{4,4}$')
ylabel('Population $ρ$ (rel. units)')
xlabel('Time ($\mu s$)')
```

```
    lg_2 = legend()
    lg_2.draw_frame(True)
    # check if filename for exporting figure is provided
    if saveplot != None:
        savefig(saveplot)
    show()

class AtomData4Levels:
    '''Setup parameters for 4-level system RWA Hamiltonian

    Usage::
        >>> class Atom4levelD2line(AtomData4Levels):
                gamma = 1     # excited state decay rate Γ in s^-1
                taue = 1      # exctied state lifetime in s
                w21 = 1756    # hyperfile splitting 1⇤⇥2 in s^-1
                w43 = 48      # hyperfile splitting 3⇤⇥4 in s^-1
                cbr1 = 1      # branching ratio П42 for |2⟩→|4⟩
                cbr2 = 1/2    # branching ratio П32 for |3⟩→|2⟩
                cbr3 = 1/2    # branching ratio П31=(1-П32) for |3⟩→|1⟩
                st23 = 1      # rel. HF trans. strength factor
                st24 = 1      # rel. HF trans. strength factor
        >>> tint = 5920       # laser-atom interaction time in s
        >>> csd2 = Atom4levelD2line(tint, delta)

    :param tint : laser-atom interaction time in s
    :param delta: detuning Δ from excited state |4⟩ in s^-1
    :param gamma: excited state decay rate Γ in s^-1
    :param taue : exctied state lifetime in s
    :param w21  : hyperfile splitting 1⇤⇥2 in s^-1
    :param w43  : hyperfile splitting 3⇤⇥4 in s^-1
    :param cbr1 : branching ratio П42 for |2⟩→|4⟩
    :param cbr2 : branching ratio П32 for |3⟩→|2⟩
    :param cbr3 : branching ratio П31=(1-П32) for |3⟩→|1⟩
    :param st23 : HF transition strength factors divided by
    :param st24 : square of reduced dipole matrix element
    :return: Parameters for 4-level system RWA Hamiltonian.
    :rtype: A :class:`AtomData4Levels`
    '''

    listargs = ('delta', 'tint', 'gamma', 'taue', 'w21', 'w43', 'hcf23', 'hcf24',
                'cbr1', 'cbr2', 'cbr3', 'nsteps')

    def __init__(self, tint, delta=0, ored=None, pop=None, nsteps=50000):
        self.tint = tint
        self.delta = delta
        self.ored = ored
```

```
        self.pop = pop
        self.nsteps = nsteps
        self.update()

    def update(self):
        '''Update parameters'''
        self.osat = self.omega_saturation(self.gamma)
        if self.pop == None:
            self.pop = (1-np.exp(-1))*100
            self.pop_coef = 1
        else:
            self.pop_coef = - np.log(1-self.pop/100)
        self.ocr = self.omega_critical(self.tint, self.taue, self.w43-self.delta,
                                       self.st23, self.cbr3, self.pop_coef)
        if self.ored == None:
            self.ored = self.ocr
        self.hcf23 = self.hcf(self.ored, self.st23)
        self.hcf24 = self.hcf(self.ored, self.st24)
        self.argsaslist = (self.delta, self.tint, self.gamma, self.taue, self.w21,
                           self.w43, self.hcf23, self.hcf24, self.cbr1, self.cbr2,
                           self.cbr3, self.nsteps)
        self.args = self.gendict(self.listargs, self.argsaslist)
        self.filename = self.args_to_filename(self.listargs, **self.args)

    def args_to_filename(self, listargs, **kwargs):
        '''Return filename from list of args'''
        return ''.join(list(map(lambda i: i + ':' + str(kwargs[i]), listargs)))

    def omega_saturation(self, gamma):
        '''Return aturation Rabi frequency value for reduced Rabi frequency'''
        return gamma / np.sqrt(2)

    def omega_critical(self, ti, te, w43, s23, p3, pop_coef):
        '''Return critical Rabi frequency value for reduced Rabi frequency'''
        return np.sqrt(4 * pop_coef * te * (w43 ** 2) / (ti * p3 * s23))

    def hcf(self, ored, strength):
        '''Return interaction Hamiltonian coeff'''
        return ored * np.sqrt(strength) / 2

    def gendict(self, listargs, args):
        '''Return Cesium D2 Line Data as dictionary'''
        return dict(zip(listargs, args))

class CsD2data:
    '''Store cesium D2 line data.
```

```
    Data obtained from Daniel A. Steck, Cesium D Line Data (2010).
    Available online at http://steck.us/alkalidata
    '''
    symbol = 'Cs'
    nspin = Fraction(7,2) # nuclear spin
    jg = Fraction(1,2)    # ground state J
    je = Fraction(3,2)    # excited state J
    # relative hyperfine transition strength factors [D. Steck, Alkali D Line Data]
    # S_{F_{g} F_{e}}
    # F_{g} = 3
    s32, s33, s34 = Fraction(20,56), Fraction(21,56), Fraction(15,56)
    # F_{g} = 4
    s43, s44, s45 = Fraction(7,72), Fraction(21,72), Fraction(44,72)
    skeys = ((3,2), (3,3), (3,4), (4,3), (4,4), (4,5))
    svals = (s32, s33, s34, s43, s44, s45)
    sge = dict(zip(skeys, svals))
    gamma = 32.889e+6             # decay rate Γ 32.889(84)e+6 in s^-1
    taue = 30.405e-9              # lifetime 30.405(77) in s
    wg43 = 2*pi * 9192.631770e+6  # F_{g} hfs 3↤↦4 9192.631770e+6 in s^-1
    we54 = 2*pi * 251.0916e+6     # F_{e} hfs 4↤↦5 251.0916(20)e+6 in s^-1
    we43 = 2*pi * 201.2871e+6     # F_{e} hfs 3↤↦4 201.2871(11)e+6 in s^-1
    we32 = 2*pi * 151.2247e+6     # F_{e} hfs 2↤↦3 151.2247(16)e+6 in s^-1
    cbr54 = 1                     # branching ratio Π54 for |F_{5}⟩→|F_{4}⟩
    cbr43 = Fraction(5,12)        # branching ratio Π43 for |F_{4}⟩→|F_{3}⟩
    cbr44 = Fraction(7,12)        # branching ratio Π44 for |F_{4}⟩→|F_{4}⟩
    cbr33 = Fraction(3,4)         # branching ratio Π33 for |F_{3}⟩→|F_{3}⟩
    cbr34 = Fraction(1,4)         # branching ratio Π34 for |F_{3}⟩→|F_{4}⟩
    cbr23 = 1                     # branching ratio Π23 for |F_{2}⟩→|F_{3}⟩

class Cs4L(AtomData4Levels,CsD2data):
    '''Store cesium D2 line data parameters for 4-level system RWA Hamiltonian'''
    w21 = np.float32(CsD2data.wg43)
    w43 = np.float32(CsD2data.we54)
    cbr1 = np.float32(CsD2data.cbr54)
    cbr2 = np.float32(CsD2data.cbr44)
    cbr3 = np.float32(CsD2data.cbr43)
    st24 = np.float32(CsD2data.sge[4,5])
    st23 = np.float32(CsD2data.sge[4,4])

class NaD2data:
    '''Store sodium D2 line data
    Data obtained from Daniel A. Steck, Sodium D Line Data (2010).
    Available online at http://steck.us/alkalidata
    '''
    symbol = 'Na'
    nspin = Fraction(3,2) # nuclear spin
```

```
    jg = Fraction(1,2)    # ground state J
    je = Fraction(3,2)    # excited state J
    # relative hyperfine transition strength factors [D. Steck, Alkali D Line Data]
    # S_{F_{g} F_{e}}
    # F_{g} = 1
    s10 = Fraction(2,12)
    s11 = Fraction(5,12)
    s12 = Fraction(5,12)
    # F_{g} = 2
    s21 = Fraction(1,20)
    s22 = Fraction(5,20)
    s23 = Fraction(14,20)
    skeys = ((1,0), (1,1), (1,2), (2,1), (2,2), (2,3))
    svals = (s10, s11, s12, s21, s22, s23)
    sge = dict(zip(skeys, svals))
    gamma = 61.542e+6             # decay rate Γ 61.542(29)e+6 in s^-1
    taue = 16.2492e-9             # lifetime 16.2492(77) in s
    wg21 = 2*pi * 1771.6261288e+6 # F_{g} hfs 1↤↦2 1771.6261288(10)e+6 in s^-1
    we32 = 2*pi * 58.326e+6       # F_{e} hfs 2↤↦3 58.326(43)e+6 in s^-1
    we21 = 2*pi * 34.344e+6       # F_{e} hfs 1↤↦2 34.344(49)e+6 in s^-1
    we10 = 2*pi * 15.810e+6       # F_{e} hfs 0↤↦1 15.810(80)e+6 in s^-1
    cbr32 = 1                     # branching ratio Π32 for |F_{3}⟩→|F_{2}⟩
    cbr21 = Fraction(1,2)         # branching ratio Π21 for |F_{2}⟩→|F_{1}⟩
    cbr22 = Fraction(1,2)         # branching ratio Π22 for |F_{2}⟩→|F_{2}⟩
    cbr11 = Fraction(5,6)         # branching ratio Π11 for |F_{1}⟩→|F_{1}⟩
    cbr12 = Fraction(1,6)         # branching ratio Π12 for |F_{1}⟩→|F_{2}⟩
    cbr01 = 1                     # branching ratio Π01 for |F_{0}⟩→|F_{1}⟩

class Na4L(AtomData4Levels,NaD2data):
    '''Store sodium D2 line data parameters for 4-level system RWA Hamiltonian'''
    w21 = np.float32(NaD2data.wg21);
    w43 = np.float32(NaD2data.we32)
    cbr1 = np.float32(NaD2data.cbr32)
    cbr2 = np.float32(NaD2data.cbr22)
    cbr3 = np.float32(NaD2data.cbr21)
    st24 = np.float32(NaD2data.sge[2,3])
    st23 = np.float32(NaD2data.sge[2,2])

def lorentz_norm(delta, deltapeak, gamma):
    '''Return Lorentzian profile of natural linewidth norlalized to 1 at peak.'''
    g, w, w0 = gamma, delta, deltapeak
    return g ** 2 / (4 * ((w - w0)**2 + (g ** 2)/4))

def las_analytic(delta, omegared, atom):
    '''Return laser absorption signal J using approximate analytic expression.'''
    delta23 = - atom.w43 - delta
```

```
    delta24 = - delta
    delta23eff = delta23 - omegared * (np.sqrt(atom.st23) + np.sqrt(atom.st24)) / 2
    delta24effsq = (delta24 ** 2) + (atom.st24 * omegared ** 2)/4
    tau_pump = (atom.taue * 8 * (delta23eff ** 2)) \
             / (np.sqrt(pi) * atom.cbr3 * atom.st23 * omegared ** 2)
    J = (atom.st24 * 4 * (delta23 ** 2) * (1 - np.exp(-atom.tint/tau_pump))) \
      / (atom.st23 * (4 * (delta24effsq) + (atom.gamma) ** 2) * atom.cbr3)
    return J

def las_numeric_single(results, gamma):
    '''Return laser absorption signal J by integrating populations
       rho33, rho44 from QuTiP result object using Trapezoidal rule
       and multiplying it by decay rate Γ.'''
    rho11, rho22, rho33, rho44 = results.expect
    times = results.times
    return (numpy.trapz(rho33, times) + numpy.trapz(rho44, times)) * gamma

def solve4lme_list(atomlist, precomputed=True):
    '''Compute and add RWA Hamiltonian QuTiP result object
       to AtomData4Levels object in atomlist.'''
    for atom in atomlist:
        if precomputed:
            atom.result = qload(atom.filename)
        else:
            atom.result = solve4lme(atom.args)
            qsave(atom.result, atom.filename)
        atom.jnumval = las_numeric_single(atom.result, atom.gamma)

def las_plot_jored(jored_expcase_list):
    '''Plot J(Ω_{red}) for fixed Δ'''
    # colors #a6cee3 #2078b4 #afdd8a #35a12e #fa9897 #e31a1c
    atom = jored_expcase_list[0]

    delta = atom.delta
    delta_MHz = delta * AFtoMHz
    delta_MHz_str = str(round(delta_MHz,2))

    jored_list = np.array(list(map(lambda atom: atom.ored, jored_expcase_list)))
    jored_list_vals = np.array(list(map(lambda atom: atom.jnumval, jored_expcase_list)))

    jored_num_list = np.linspace(jored_list[0], jored_list[-1], 201)

    ored_sat_24 = atom.osat / np.sqrt(atom.st24)
    ored_sat_24_MHz = ored_sat_24 * AFtoMHz
    ored_sat_24_MHz_str = str(round(ored_sat_24_MHz,2))
```

```
    orec_cr = atom.__class__(atom.tint).ocr
    ored_cr_MHz = orec_cr * AFtoMHz
    ored_cr_MHz_str = str(round(float(ored_cr_MHz),2))

    orec_cr_pop_99 = atom.__class__(atom.tint,pop=99).ocr
    ored_cr_pop_99_MHz = orec_cr_pop_99 * AFtoMHz
    ored_cr_pop_99_MHz_str = str(round(float(ored_cr_pop_99_MHz),2))

    figure()
    axvline(x=ored_sat_24_MHz, color='#a6cee3',
            label='$Ω_{red,sat,24} ≈$ %s MHz' % ored_sat_24_MHz_str)
    axvline(x=ored_cr_MHz, color='#afdd8a',
            label='$Ω_{red,cr} ≈$ %s MHz' % ored_cr_MHz_str)
    axvline(x=ored_cr_pop_99_MHz, color='#fa9897',
            label='$Ω_{red,cr}|_{pop=0.01} ≈$ %s MHz' % ored_cr_pop_99_MHz_str)

    plot(jored_list * AFtoMHz,jored_list_vals, '-', label='numeric calc.')

    plot(jored_num_list * AFtoMHz, las_analytic(delta, jored_num_list, atom),
         '--', label='analytic approx.')
    ylabel('$J(Ω_{red})$ (arb. units)')
    xlabel('$Ω_{red}$ (MHz)')
    title('%s, $Δ$ = %s (MHz), $τ_{int}$ = %s (s)' \
          % (atom.symbol, delta_MHz_str, str(atom.tint)))
    lg = legend()
    lg.draw_frame(True)
    show()

def las_plot_jdelta(jdelta_expcase_list):
    '''Plot J(Δ) for fixed Ω_{red}'''
    # colors #a6cee3 #2078b4 #afdd8a #35a12e #fa9897 #e31a1c
    atom = jdelta_expcase_list[0]

    ored = atom.ored
    ored_MHz = ored * AFtoMHz
    ored_MHz_str = str(round(ored_MHz,2))

    gamma = atom.gamma
    gamma_MHz = gamma * AFtoMHz
    gamma_MHz_str = str(round(gamma_MHz,2))

    jdelta_list = np.array(list(map(lambda atom: atom.delta, jdelta_expcase_list)))
    jdelta_list_vals = np.array(list(map(lambda atom: atom.jnumval, jdelta_expcase_list)))

    jdelta_num_list = np.linspace(jdelta_list[0], jdelta_list[-1], 201)
```

```
    figure()
    axvline(x=-gamma_MHz/2, color='#afdd8a',
            label='±$Γ/2$, $Γ≈$ %s MHz (FWHM)' % gamma_MHz_str)
    axvline(x=gamma_MHz/2, color='#afdd8a')

    plot(jdelta_num_list * AFtoMHz,
         lorentz_norm(jdelta_num_list, 0, gamma) * jdelta_list_vals.max(),
         '-', label='natural linewidth')
    plot(jdelta_list * AFtoMHz, jdelta_list_vals, '-', label='numeric calc.')

    plot(jdelta_num_list * AFtoMHz, las_analytic(jdelta_num_list, ored, atom),
         '--', label='analytic approx.')
    ylabel('$J(Δ)$ (arb. units)')
    xlabel('$Δ$ (MHz)')
    title('%s, $Ω_{red}$ = %s (MHz), $τ_{int}$ = %s (s)' \
          % (atom.symbol, ored_MHz_str, str(atom.tint)))
    lg = legend()
    lg.draw_frame(True)
    show()

# converting from/to frequency in MHz to/from angular frequency s^{-1}
MHztoAF = 2 * pi * 1e+6
AFtoMHz = 1/(MHztoAF)
```

3.4. Laser-atom interaction time 0.0002 s

```
In [5]: %%capture

expcase_list = Na4L(0.0002), Na4L(0.0002, pop=99), Cs4L(0.0002), Cs4L(0.0002, pop=99)

# set precomputed to False to avoid loading precomputed and saved qutip results
solve4lme_list(expcase_list, precomputed=True)
```

```
In [6]: %%capture

# Calculate numerical laser absorption signal J(Ω_{red})|Δ=0
# laser-atom interaction time 0.0002 s,
# Ω_{red} from 0.5 MHz to 50 MHz with vairable step size
na_jored_list = np.linspace(0.25,2.25, 9)
na_jored_list = np.concatenate((na_jored_list, np.linspace(2.5,14.5, 25)), axis=0)
na_jored_list = np.concatenate((na_jored_list, np.linspace(15,50, 15)), axis=0)

na_jored_MHz_list_1 = na_jored_list

na_jored_expcase_list_1 = list(map(lambda omegaMHz:
                                   Na4L(0.0002, ored = omegaMHz * MHztoAF),
```

```
                                    na_jored_MHz_list_1))

# set precomputed to False to avoid loading precomputed and saved qutip results
solve4lme_list(na_jored_expcase_list_1, precomputed=True)

# Calculate numerical laser absorption signal J(Ω_{red})|Δ=0
# laser-atom interaction time 0.0002 s,
# Ω_{red} from 0.5 MHz to 50 MHz with vairable step size
cs_jored_list = np.linspace(0.25,2.25, 9)
cs_jored_list = np.concatenate((cs_jored_list, np.linspace(2.5,14.5, 13)), axis=0)
cs_jored_list = np.concatenate((cs_jored_list, np.linspace(15,50, 15)), axis=0)

cs_jored_MHz_list_1 = cs_jored_list

cs_jored_expcase_list_1 = list(map(lambda omegaMHz:
                                    Cs4L(0.0002, ored = omegaMHz * MHztoAF),
                                    cs_jored_MHz_list_1))

# set precomputed to False to avoid loading precomputed and saved qutip results
solve4lme_list(cs_jored_expcase_list_1, precomputed=True)

# Calculate numerical laser absorption signal J(Δ)|Ω_{red}=Ω_{red,cr}|Δ=0
# laser-atom interaction time 0.0002 s,
# Δ from -40 MHz to 40 MHz with vairable step size
na_jdelta_list = np.linspace(-40,-5, 15)
na_jdelta_list = np.concatenate((na_jdelta_list, np.linspace(-4,-1, 4)), axis=0)
na_jdelta_list = np.concatenate((na_jdelta_list, np.linspace(-0.5,0.5, 5)), axis=0)
na_jdelta_list = np.concatenate((na_jdelta_list, np.linspace(1,4, 4)), axis=0)
na_jdelta_list = np.concatenate((na_jdelta_list, np.linspace(5,40, 15)), axis=0)

na_jdelta_MHz_list_1 = na_jdelta_list

na_jdelta_expcase_list_1 = list(map(lambda deltaMHz:
                                    Na4L(0.0002, delta = deltaMHz * MHztoAF,
                                         ored = Na4L(0.0002).ocr),
                                    na_jdelta_MHz_list_1))

# set precomputed to False to avoid loading precomputed and saved qutip results
solve4lme_list(na_jdelta_expcase_list_1, precomputed=True)

# Calculate numerical laser absorption signal J(Δ)|Ω_{red}=Ω_{red,cr}|Δ=0
# laser-atom interaction time 0.0002 s,
# Δ from -40 MHz to 40 MHz with vairable step size
na_jdelta_list_2 = np.linspace(-40,-5, 15)
na_jdelta_list_2 = np.concatenate((na_jdelta_list_2, np.linspace(-4,-1, 4)), axis=0)
na_jdelta_list_2 = np.concatenate((na_jdelta_list_2, np.linspace(-0.5,0.5, 5)), axis=0)
```

```
na_jdelta_list_2 = np.concatenate((na_jdelta_list_2, np.linspace(1,4, 4)), axis=0)
na_jdelta_list_2 = np.concatenate((na_jdelta_list_2, np.linspace(5,40, 15)), axis=0)

na_jdelta_MHz_list_2 = na_jdelta_list_2

na_jdelta_expcase_list_2 = list(map(lambda deltaMHz:
                                    Na4L(0.0002, delta = deltaMHz * MHztoAF,
                                         ored = Na4L(0.0002, pop=99).ocr),
                                    na_jdelta_MHz_list_2))

# set precomputed to False to avoid loading precomputed and saved qutip results
solve4lme_list(na_jdelta_expcase_list_2, precomputed=True)

# Calculate numerical laser absorption signal J(Δ)|Ω_{red}=Ω_{red,cr}|Δ=0
# laser-atom interaction time 0.0002 s,
# Δ from -40 MHz to 40 MHz with vairable step size
cs_jdelta_list = np.linspace(-40,-5, 15)
cs_jdelta_list = np.concatenate((cs_jdelta_list, np.linspace(-4,-1, 4)), axis=0)
cs_jdelta_list = np.concatenate((cs_jdelta_list, np.linspace(-0.5,0.5, 5)), axis=0)
cs_jdelta_list = np.concatenate((cs_jdelta_list, np.linspace(1,4, 4)), axis=0)
cs_jdelta_list = np.concatenate((cs_jdelta_list, np.linspace(5,40, 15)), axis=0)

cs_jdelta_MHz_list_1 = cs_jdelta_list

cs_jdelta_expcase_list_1 = list(map(lambda deltaMHz:
                                    Cs4L(0.0002, delta = deltaMHz * MHztoAF,
                                         ored = Cs4L(0.0002).ocr,
                                         nsteps=100000),
                                    cs_jdelta_MHz_list_1))

# set precomputed to False to avoid loading precomputed and saved qutip results
solve4lme_list(cs_jdelta_expcase_list_1, precomputed=True)

# Calculate numerical laser absorption signal J(Δ)|Ω_{red}=Ω_{red,cr}|Δ=0
# laser-atom interaction time 0.0002 s,
# Δ from -40 MHz to 40 MHz with vairable step size
cs_jdelta_list_2 = np.linspace(-40,-5, 15)
cs_jdelta_list_2 = np.concatenate((cs_jdelta_list_2, np.linspace(-4,-1, 4)), axis=0)
cs_jdelta_list_2 = np.concatenate((cs_jdelta_list_2, np.linspace(-0.5,0.5, 5)), axis=0)
cs_jdelta_list_2 = np.concatenate((cs_jdelta_list_2, np.linspace(1,4, 4)), axis=0)
cs_jdelta_list_2 = np.concatenate((cs_jdelta_list_2, np.linspace(5,40, 15)), axis=0)

cs_jdelta_MHz_list_2 = na_jdelta_list_2

cs_jdelta_expcase_list_2 = list(map(lambda deltaMHz:
                                    Cs4L(0.0002, delta = deltaMHz * MHztoAF,
```

```
                                        ored = Cs4L(0.0002, pop=99).ocr,
                                        nsteps=100000),
                                    cs_jdelta_MHz_list_2))

# set precomputed to False to avoid loading precomputed and saved qutip results
solve4lme_list(cs_jdelta_expcase_list_2, precomputed=True)
```

References

[1] G. Lindblad, Commun. Math. Phys. **48**, 119 (1976)

[2] C. Gardiner and P. Zoller, *Quantum Noise*, Springer Series in Synergetics (Springer-Verlag Berlin Heidelberg, 2004)

[3] D. Walls and G. J. Milburn, *Quantum Optics* (Springer Science + Business Media, 2008)

[4] B. W. Shore, *Manipulating Quantum Structures Using Laser Pulses* (Cambridge University Press, 2011)

[5] D. A. Steck, *Cesium D Line Data* (http://steck.us/alkalidata, 2010)

[6] D. A. Steck, *Sodium D Line Data* (http://steck.us/alkalidata, 2010)

[7] R. Loudon, *The Quantum Theory of Light Third Edition* (Oxford University Press, 2000)