Problem Set 5

1 Comparing CO and H_2

Clouds of molecular hydrogen (as H_2 is often called) are where the gas in our galaxy cools down enough to form stars. As you might imagine, if your job is to understand star formation, this is where you'd look. In contrast to the hydrogen clouds traced by the 21cm line, these are colder ($T \sim 10$ K) and more dense ($n \sim 100$ cm^{-3}).
Cooling H_2 turns out to be difficult. In this problem, we examine why.

1.1
Estimate the energies and frequencies of the first H_2 rotational and vibrational transitions. Note that, because H_2 is a quadrupole, not a dipole, $\Delta J = \pm 1$ transitions are forbidden.

1.2
Estimate Einstein A coefficients for these two transitions, using that H_2 is a symmetric molecule (e.g. quadrupole radiator).

1.3
Assuming that collisions are setting population statistics for excited/de-excited states to a temperature of 10 K and assuming these clouds are optically thin, which transition (vibrational or rotational) contributes the most to cooling this gas? To order of magnitude long would it take to cool to 5K?

1.4
Now suppose there is some CO sprinkled in with the H_2. A typical CO/H_2 ratio in the Milky Way is $6e-5$. Using the $J = 1 \rightarrow 0$ transition of CO that we examined in class and assuming collisions set the population statistics, how does the presence of CO change the cooling time you derived above?

2 Good Rovibrations

Let's see if we can get a simulation to reproduce the rovibrational spectrum of CO pictured in Figure 1. In order to get some of the answers below, it's assumed that you'll be doing some coding to work out the numerical details.

2.1
First, use the center wavenumber to estimate which vibrational transition we are talking about for CO. Assume we end in the $n = 0$ state.

2.2
Using that each rotational transition is a $\Delta J = \pm 1$ transition and using the spin degeneracy of each J state ($g = 2J + 1$), identify the state J at which $E_J \sim kT$. What does this imply T is? Don't forget that the total energy is the sum of the vibrational and rotational energies.

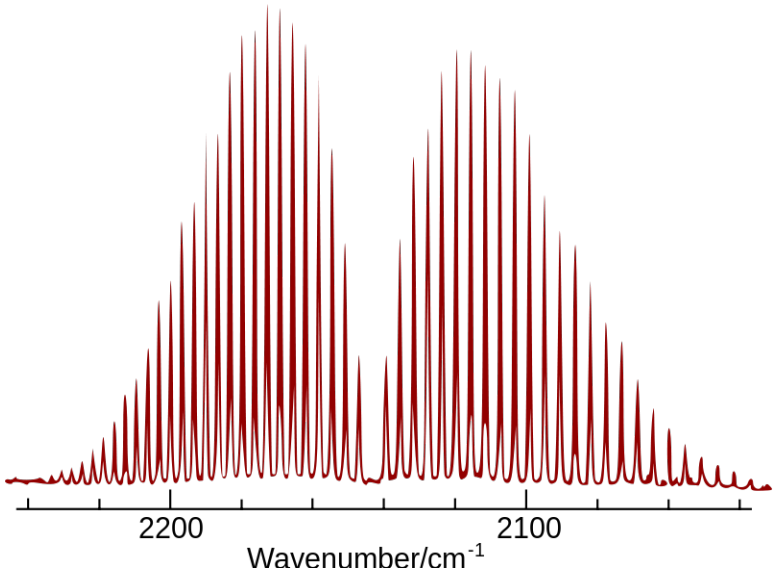

Figure 1: The vibration-rotation line spectrum of CO.

2.3

Estimate the Einstein A coefficients for this Δn transitions as a function of J. Is the variation sufficient that it needs to be included in predictions of line strength?

2.4

Now let's cook up the full simulation. For this, determine the moment of inertia I for CO as a function of n (the vibrational level) and J (the rotational level), as we worked out on our quiz. Then populate the n and J states according to Boltzmann statistics, using the T you found above. Finally, use the population of each state and (depending on your answer above) the transition strength to determine the relative intensity of each transition as a function of frequency. Plot it and compare to Figure 1.
Here's some skeleton code:

```
mu_CO # reduced mass of CO
k_CO # effective vibrational spring constant, from class, g/s^2
w0_CO # angular frequency of vibration, from mu_CO, k_CO
x_CO # atomic separation, cm

def E_vib_CO(n):
    '''Return the energy of the nth vibrational state.'''
    return E_vib

def I_CO(n, J):
    '''Return (optionally n,J-dependent) CO moment of inertia, in g*cm^2.'''
    return mom_inertia

def E_rot_CO(J):
```

```
    '''Return the energy of the Jth rotational state of CO.'''
    return E_rot

def E_rovib_CO(n, J):
    '''Return the energy of the n,Jth rovibrational state of CO.'''
    return E_rovib

def g(J):
    '''Return the degeneracy of the Jth rotational state.'''
    return degeneracy

def population(n, J, T):
    '''Fractional population in J,n rovib state state of CO.'''
    return population

def omega(J1, J0, delta_n=1):
    '''Return angular frequency of J1->J0, delta_n transition.'''
    return ang_freq

def A_CO(J1, J0, delta_n=1):
    '''Return Einstein A coefficient for J1->J0 delta_n=1 transition.'''
    return A

ns = np.array([[0],[1]]) # (2,1) array of vibrational states
Js = np.arange(20); Js.shape = (1,-1) # (1,10) array of rotational states
I_COs # (2,10) array of moments of inertia
E_COs # (2,10) array of energies
gs # (1,10) array of degeneracies
populations # (2,10) array of population fractions relative to n=0, J=0

transitions = []
for index,j in enumerate(Js.flat):
    # for each of Delta J = 1, -1
    dE # energy difference between states
    A # Einstein A of transition
    pop # population engaging in this transition
    transitions.append((w, intensity_of_line))
transitions = np.array(transitions)
nus,intensities = transitions[:,0], transitions[:,1]
```

2.5

Are there remaining discrepancies? What effects haven't you included that would improve your simulation?