ASTP 720 - Homework 3 - Multiwavelength Galactic Structure Part II and ODE Solvers

Due September 17th, 2020

For the first part of this homework, you will be continuing on with some matrix stuff with some applications. For the second part, you will be building some ODE solvers, which you will begin applying next time to some equations of stellar structure and mass-radius relations.

[1.] As promised, unit testing time! Please write a unit test class (e.g., in a file called `test_matrix.py` if your `Matrix` class in in a file called `matrix.py`) for your Matrix class. I *highly* recommend you use Python's built-in `unittest` framework, e.g., `https://docs.python.org/3/library/unittest.html`, though if you want to use just your own `assert` commands you may do so. Do not overdo it with these, simply compare for each of the above items that you get a sensible result, i.e., you only need to do one test per item above, so don't make 50 tests of whether you have or have not transposed a matrix or not. The goal is to just learn about the practicalities of unit testing and make it easier for you to debug your `Matrix` class. For this part of the assignment, you may absolutely use `numpy` or `scipy`'s built-in functionality for the purpose of checking, though if you want to put in 3×3 matrices since you can do the checks by hand, that is of course fine as well.

The Content of Galaxies

In class we discussed very briefly the equilibrium of different states of a gas like Hydrogen in the interstellar medium and the concept of *detailed balance*. Let's look at this in more detail.

Let's again define the *Einstein coefficients* and go into them in a little more detail. For an upper state u and a lower state l, they are:

- A_{ul} is the transition probability per unit time for *spontaneous emission*.
- $B_{lu}\bar{J}$ is the transition probability per unit time for *absorption* of radiation, with $\bar{J}$ being the mean intensity of the radiation field (related to the energy density).
- $B_{ul}\bar{J}$ is the transition probability per unit time for *stimulated emission*, where an electron will jump to a lower energy configuration in the presence of other electromagnetic radiation.

Again, we will assume here that collisional excitation and de-excitation, often given with C_{lu} and C_{ul}, respectively (but not often called Einstein Coefficients), are negligible given the low densities for the purposes of this exercise.

And, again for posterity, a three-level system in equilibrium will have equations that look like:

$$\begin{aligned} n_1(B_{12}+B_{13})\bar{J} &= n_2(A_{21}+B_{21}\bar{J})+n_3(A_{31}+B_{31}\bar{J}) \\ n_2(B_{21}\bar{J}+A_{21}+B_{23}\bar{J}) &= n_1B_{12}\bar{J}+n_3(B_{32}\bar{J}+A_{32}) \\ n_3(B_{31}\bar{J}+A_{31}+B_{32}\bar{J}+A_{32}) &= n_1B_{13}\bar{J}+n_2B_{23}\bar{J} \end{aligned}$$

And also again, since the third equation is related to the other two, so in order to solve for the three densities, we need one more constraining equation. That one can just be given by the total nuber density: $N = n_1 + n_2 + n_3$. But if we deal with line ratios, then we don't actually care, which is often the case. So then you only have to worry about what n_2/n_1 is and n_3/n_1 is, for example (but often in observable spectral line units).

The Einstein coefficients are all related to each other. If you have a two-level system in thermodynamic equilibrium, you can work out the *Einstein relations* which *must* hold true for all temperatures and thus are *atomic properties* and not dependent on the environment. These relations are:

$$g_l B_{lu} = g_u B_{ul} \tag{1}$$

$$A_{ul} = \frac{2h\nu^3}{c^2} B_{ul} \tag{2}$$

where h is Planck's constant, c is the speed of light, ν is the frequency of the emission line, and g_n are statistical weights for each level (the degeneracy for each). One can work out that for Hydrogen in state $n = 1, 2, \cdots$, $g_n = 2n^2$ (Sorry about another "n").

What we see above is that once we know one of the Einstein coefficients, we know them all, which is convenient. That doesn't mean we know all of the *rates* though because those are dependent on $\bar{J}$. Technically, this is given by

$$\bar{J} \equiv \int_0^\infty J_\nu \phi(\nu)\, d\nu, \tag{3}$$

where J_ν is the spherically averaged mean intensity at a frequency ν (often written this way and not as $J(\nu)$ by convention) and $\phi(\nu)$ is the normalized line profile function that is the sum of many components, including a natural (Lorentzian) broadening, Doppler broadening, collisional broadening, etc. We can make an approximation that it is nearly a delta function in shape, centered at some frequency ν_0, and so we have that $\bar{J} \approx J_{\nu_0}$. In thermodynamic equilibrium though, $J_\nu = B_\nu$, the Planck blackbody function (sorry about another "B"), or

$$B_\nu(T) = \frac{2h\nu^3}{c^2} \frac{1}{e^{h\nu/kT} - 1}, \tag{4}$$

where k is Boltzmann's constant. Therefore,

$$\bar{J} \approx \frac{2h\nu_0^3}{c^2} \frac{1}{e^{h\nu_0/kT} - 1}. \tag{5}$$

For Hydrogen, the frequencies of the transitions between different states are easy to calculate: $E = -13.6 \text{ eV}/n^2$, again for $n = 1, 2, \cdots$. The frequency of emission or absorption is just going to be the difference between the energy levels, so between two states i and j, the frequency of the transition is given by $\Delta E_{ij} = h\nu_{ij}$, where $\nu_{ij} = \nu_0$ above.

Below is a table of Einstein coefficients A_{ul} from `http://www.ioffe.ru/astro/QC/CMBR/sp_tr.html` (Kholupenko, Ivanchik, and Varshalovich 2005), sanity checked against the more specific CHIANTI database (Dere et al. 1997, Landi et al. 2013; has more specific individual transitions), and sanity checked against some other values found elsewhere. The values are given in units of s^{-1}.

l \ u	2	3	4	5	6	7	8	9
1	4.70×10^8	5.57×10^7	1.28×10^7	4.12×10^6	1.64×10^6	7.57×10^5	3.87×10^5	2.14×10^5
2	$--$	4.41×10^7	8.42×10^6	2.53×10^6	9.73×10^5	4.39×10^5	2.21×10^5	1.22×10^5
3	$--$	$--$	8.98×10^6	2.20×10^6	7.78×10^5	3.36×10^5	1.65×10^5	8.90×10^4
4	$--$	$--$	$--$	2.70×10^6	7.71×10^5	3.04×10^5	1.42×10^5	7.46×10^4
5	$--$	$--$	$--$	$--$	1.02×10^6	3.25×10^5	1.39×10^5	6.90×10^4
6	$--$	$--$	$--$	$--$	$--$	4.56×10^5	1.56×10^5	7.06×10^4
7	$--$	$--$	$--$	$--$	$--$	$--$	2.27×10^5	8.23×10^4
8	$--$	$--$	$--$	$--$	$--$	$--$	$--$	1.23×10^5

These are also provided to you in a more convenient file to read from called `A_coefficients.dat`. You can choose to read from this file and parse it yourself, or I have provided the convenience function inside `coefficients_reader.py` which will return a Python dictionary of the values. See the documentation at the top of the code for more information.

[2.] For a total number density of $N = 1$ cm^{-3} for convenience, calculate the different number densities as a function of temperature T. Make a plot showing the different number densities vary as a function of T. If you're feeling adventures, try to think about the physicality of this plot and it's limitations (not required).

[3.] Write an ODE package that includes Euler's method, Heun's method, and the explicit RK4 method. You can set this up in a number of different ways but to simplify things, it should take an initial guess and a list of times to solve for. Note that this is *not* the same as the step size. Feel free to assume that the function requires user input such as the step size (or also feel free to set it to be an optional argument, e.g., make each step size 0.1 or 0.01 or 0.001 of the interval), number of Picard iterations, etc.

Your ODE solvers should each be able to handle up to two variables simultaneously, so that you can solve for example the force equations we discussed in class, or the case of a damped pendulum that you will be testing against below. There are different ways you can approach this. You could write two different sets of functions, those where you consider a first-order ODE, and ones where you consider a second-order ODE broken into two. Or, you could write these such that they look at the inputs to the function and determine what to do. Or, for example, in the spirit of `scipy`'s `odeint()` solver, you could write

```
def gravity(variables, t, M):
    x, v = variables
    dvariables_dt = [v, -G*M/x**2]
    return dvariables_dt
```

since in the case of Newtonian gravity, $x' = v$ and $v' = a = -GM/x^2$. In this case, you are supplying a vector of variables/equations (see this notation in the very beginning of our discussion on ODEs), where you are saying that the first indices deal with the x variable and its derivative, and the second indices deal with the x' variable and its derivative. See the documentation: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.odeint.html` and the examples for how they structure their functions. This has the nice capability that you can arbitrarily add in more variables. But again, you are not required to generalize beyond having two variables.

[4.] Test your three functions against the damped pendulum example in `scipy`'s `odeint()` documentation. Of course, your functions do not have to receive input in the same way that `odeint()` does. Make some plots to demonstrate that things are working as they should.

[5.] Test your three methods against the following stiff ODE:

$$\frac{dy}{dt} = -\lambda(y - \cos(t)), \qquad y(0) = 0 \tag{6}$$

where λ is a constant. As λ increases, the equation typically becomes stiffer, so pick maybe $\lambda \geq 10$. So that you can check your results, the solution to this equation is:

$$y(t) = -\frac{\lambda^2}{1+\lambda^2}e^{-\lambda t} + \frac{\lambda}{1+\lambda^2}\sin(t) + \frac{\lambda^2}{1+\lambda^2}\cos(t). \tag{7}$$

Provide some comments about the stability of each.