# AY640 Computational Experiment 3: Synchrotron, Compton, and Bremsstrahlung

Discussion: Monday April 4, 2022

In this experiment, you will use SiRTIPy to see the effects of more radiative processes. You will also see how you can automatically fit an observed spectrum to determine the physical properties of the object.

1. Download `experiment3-a.py` (for Part A), `experiment3-bc.py` (for Parts B and C), and `orion.dat` (for Parts B and C) from the Blackboard class website. Read through the Python code so you understand what each line does.

2. Part A: Synchrotron Emission and Absorption

   In this section, you will calculate the radiation that leaves a medium with a power-law distribution of electron energies within a magnetic field, which produces and absorbs synchrotron radiation. We will use a medium that has the following properties:

   Electron energy distribution:

$$N(\gamma) = \begin{cases} C\gamma^{-p} & 2 \leq \gamma \leq 10^4 \\ 0 & \text{otherwise} \end{cases}$$

   with $p = 2$, and a total free electron number density $n_e = 2 \times 10^{-3}$ cm$^{-3}$, in a magnetic field of strength $B = 0.1$ G.

   (a) Begin by running the code as-is. It will calculate the synchrotron *emission only* through $10^{19}$ cm of this medium. Look at the plots. Do they look as you would expect? What is the slope of the power law? (find 2 points on one of the curves and calculate it by hand) Do you get the expected answer?

   (b) Now add in the synchrotron self-absorption by looking for places marked **FIXME** in Part A of the code. You will need to:

   - Write the `alpha_synchrotron` function, using `j_synchrotron` as a guide.
   - Add `alpha_synchrotron` to the medium.
   - Change the title and filename of the plot.

   Run the code again. How are the spectra different? What is the slope at low frequencies? Is it what you expect? Why do the different curves lie overtop of each other at low frequencies?

   (c) Change the value of $p$, perhaps to 3 or 1. How does it change the solution? Try playing with at least one other physical parameter and see how it affects the solution. Why do the different parameters have the effects they do?

3. Part B: Bremsstrahlung

   Now we will implement functions to calculate the bremsstrahlung (free-free) emission and absorption from a thermal medium. You will compare this to the data from the Orion nebula that you analyzed in Homework 3 from Terzian & Parrish (1970), which I have digitized and placed into the file `orion.dat`. In this part, you will use the following physical parameters:

$$\langle Z^2 n_i \rangle = 2n_e$$

$$T = 1.5 \times 10^4 \text{ K}$$

$$n_e = 2 \times 10^3 \text{ cm}^{-3}$$

   Read through Part C of the `experiment3-cd.py` code. Note especially how to easily read in (and access) simple text files with columns of numbers.

   Look at the **FIXME**s to do the following:

(a) Calculate the emission per unit volume in `j_bremsstrahlung` using RL equation 5.14b. You can assume that all Gaunt factors are 1, and that all properties of the ions have been folded into the parameter `Z2x`, i.e. you can assume

$$Z^2 n_e n_i = \texttt{Z2x}\, n_e^2.$$

(b) Calculate the absorption coefficient in `alpha_bremsstrahlung` using RL equation 5.18b. You can make the same assumptions as above.

Run the code. What does the spectrum look like? What is the slope at low frequencies (calculate it from the plot) — is it what you would expect? What is the slope at the higher frequencies plotted — is it what you would expect? Why are there two different frequency regimes what do they represent? Compare the radiative transfer model to the observations of the Orion nebula. Is it a good fit?

4. Part C: Model Fitting (Note, code is labeled part D)

   There are probably a set of physical parameters that match the Orion nebula spectrum better than Part C — let's get the computer to calculate those for us!

   The python module `scipy.optimize` is full of routines for calculating best fits like this. A very useful general-purpose function is `scipy.optimize.curve_fit`, which is what we will use. It takes as input:

   - Observational $x$ and $y$ values.
   - A function that calculates the model predictions for $y$ at each given $x$.
   - Guesses for the best-fit values of the parameters of the model (in this case, temperature $T$ and electron number density $n_e$).

   Our function will do a radiative transfer calculation to make the prediction.

   Uncomment Part D of the code and read it thoroughly, especially the definition of `bremsstrahlung_spectrum`, and the call to `curve_fit`. Run the code and look at the output. How does it compare to Part C? How well does it fit the data? How do the values compare to what you estimated in Homework 3 Question 3?

   You might notice that the physical parameters aren't actually that similar to what you found when you did it by hand in the homework. This is because the code is doing a least-squares fit to the intensity *values*. But that is quite different from what you do when you look at a logarithmic plot like this for example, `curve_fit` thinks that a miss of $30 \times 10^{-26}$ W m$^{-2}$ Hz$^{-1}$ is equally bad at every point in the spectrum. Look at the spectrum do *you* think that it is just as big a miss at $10^2$ Hz as at $10^4$ Hz? Why or why not? Which part of the spectrum is the code trying hardest to fit?

   In order to give equal weight to both the high and low parts of the spectrum, we can fit the *logarithm* of the intensity values, instead of the intensity values themselves. To do that, look at the **FIXME**s inside `bremsstrahlung_spectrum` and around the `curve_fit` call, and switch to the versions of the code that use log (base 10) of the intensity. Then re-run the code and compare the best-fit values to both the previous ones, and to what you found in Homework 3. Which part of the spectrum is the code trying hardest to fit now?

   Finally, look at the `initial_parms` line. This gives initial guess for the parameters. Sometimes, fits like these are very sensitive to the initial guess, especially when thare are many parameters. Try playing with these — does it always converge on basically the same solution, or can you make it come to a different solution? Try completely crazy numbers (for example, the code does not intrinsically know that negative densities and temperatures are unphysical! However, it may complain if it needs to take the logarithm of a negative number).

   Note: Doing best-fit values to physical parameters for real astronomical objects, like in this part, would make another good term project.

**Everyone should bring their plots, not just the discussion leaders.**