ASTRO 530 **Spring 2026**

Homework Set #2 **due *Friday* January 23**

Do *not* just keep one big notebook or py file and keep adding to it each week. Save last week's assignment as `hw1.py` or some such, and put the pieces you need either in a library you import or, if you won't need them a lot, you can copy pieces over to `hw2.py`. You won't regret it.

Problems

2. **Limb darkening basics.**

   (a) Expand the source function in a MacLaurin series by writing: $S_\nu(\tau_\nu) = \sum_{n=0}^{\infty} a_n \tau_\nu^n$. Note that the coefficients a_n are functions of frequency. Find an expression for the emergent specific intensity $I_\nu^+(\tau_\nu = 0, \mu)$ in terms of a_n, and μ ($\mu = \cos\theta$).

   (b) In the case of a linear source function ($a_n = 0$ for $n \geq 2$), show that you recover the Eddington-Barbier value for emergent intensity.

   (c) Repeat, retaining the quadratic term ($a_2 \neq 0$).

   (d) Let's say you observe the emergent intensity at two points on the sun with emergent angles μ_1 and μ_2 and find them to be $I_{\nu,1}$ and $I_{\nu,2}$. Use the Eddington-Barbier relation to infer the functional form of $S(\tau_\nu)$.

   (e) Starting from scratch: what observational measurements would you need to determine the actual functional form of $S(\tau_\nu)$ for the Sun over some range, without any assumptions like Eddington-Barbier? *(Hint: how many observations at different μ did you need to determine the linear functional form? What if it were quadratic? Higher order?)*

3. **Integrals of B_ν.**

   (a) Using your work from Problem 1, numerically integrate $B_\nu(T = 7500 \text{ K})$ over the interval $0 < \tilde{\nu} < \infty$. Your result will have the units of specific intensity times wavenumber, which seems strange, but these sorts of "unreduced" units are not foreign to astronomy (think about the Hubble constant or the radio astronomy unit of intensity K km/s).

   For this exercise, use a box or trapezoid rule integrator. You may use a canned one but, again, you need to at least check that you understand its behavior in weird cases with one of your own design.

   In the future, you can use a fancier blackbox integrator that works to higher order, but you need to have your own "brute force" integrator you understand to check its output when it does things you don't expect (which it will later in the course, I promise you!).

Your box or trapezoid function should accept an array of x and an array of y points, and should not assume the points are evenly spaced (but you can assume they're ordered in x).

Then, wrap this integrator in another function that lets you specify the the two limits of integration and the density of points in the integration of a given function (that is, specify $(\tilde{\nu}_{\max} - \tilde{\nu}_{\min})/d\tilde{\nu}$, not the number of points).

Now you're ready to explore.

(b) Determine the precision of your calculation through comparison with the analytic result for the area under the Planck function. Precision is measured as $|(\text{truth} - \text{calculation})|/\text{truth} = |1 - \text{calculation}/\text{truth}|$, and is most usefully plotted logarithmically.

Your integrator has three parameters you can vary: which one(s) is(are) limiting your precision? Can you get to machine precision? Plot your precision vs. the 3 parameters with the others held constant. Interpret these.

(c) Deliberately add a very small error to your calculation by making it overshoot the right answer (just artificially add a constant to the output). What is the signature of such an error, as you plot your precision vs. the three parameters, given the known answer?

(d) In most situations, you will not have an analytic solution to guide you in whether your integrator has converged. In such cases, you can't tell if your calculation has an error as you did in part (c) (that is, you can't prove it's *accurate)*, but you can still tell what its *precision* is.

Imagine you did *not* have the analytic answer to guide you in part (b). What tests can you do to determine the precision of your integration, and to demonstrate that the values for your three parameters are sufficient to achieve that precision? Show the results of such a test.

Some hints:

- To do a quick box integration, you can use `numpy.diff` on the x axis, and `(y[:-1]+y[1:])/2` on the y, then use `numpy.sum` on the product (or, if you want to get fancy, use `numpy.dot`).
- Instead of doing a calculation in $\tilde{\nu}$ you may want to evaluate the integral over $\log \tilde{\nu}$. This will let you explore the upper and lower limits independently of your density of points. In this case your function will be $\tilde{\nu} B_\nu$ and your independent variable will be $d \ln \tilde{\nu}$. Be careful with units! If you do this you will reach machine precision with *many* fewer points.

- Be careful with units here! Make sure you are using the correct analytic form for your version of the blackbody function (flux or intensity). Also using these mixed units will introduce a factor of c compared to the usual integral over ν.
- When you don't know the right answer, you can still tell if your answer has *converged* by finding the asymptote of the function, or by examining the magnitude of the change you get when improving one of the parameters.
- Get used to using `astropy`'s `constants` and `units` modules to track units.
- Astropy quantities don't always play nicely with lists. You can wrap a list of quantities in `u.Quantity()` to turn it into a nice numpy array.
- Get to know the `decompose()` and `to()` methods for manipulating units.