# HomeworkG-numericalIntegration

July 11, 2022

**Due on Friday, July 15th, 5:00pm.**

In this homework, you're going to practice using functions from modules, writing functions to visualize data and perform tasks, accessing elements of dictionaries, and calculating numerical integrals. Read the assignment *carefully* and submit a `jupyter` notebook that responds to all steps of the Coding Assignment outlined below. Include all the code you write as "Code" cells and any written responses as "Markdown/Raw NBConvert" cells. Be sure to indicate at the start of a block of code which part of the assignment it is in response to.

In your notebook, please indicate clearly at the start of a block of code which part of the assignment it is in

**Be sure to use comments throughout your code to explain how it works, variables, etc. You don't need to include full docstrings for any functions you write, but it *is* good practice! (1 pt)**

## 1 Physics Background: Colors of Stars

When we look at a star or a galaxy or a quasar, that object will appear to our eyes to have some color. That color is a result of the spectrum of light the object emits. Your eyes integrate the many wavelengths of light in a spectrum into three coarse bins, which we often treat as red, green, blue (RGB). In this homework, you will estimate (roughly) the RGB colors you would see for looking at various different astronomical objects.

As you saw in HomeworkF, stars can be roughly approximated as a blackbody. (Although you also saw in HomeworkF that true stars will have additional absorption and emission lines on top of their blackbody spectra!) For this homework, we'll go back to approximating stars as blackbodies (and their spectra given by the Planck function) and perform numerical integrals to approxiate the colors you might see if you were to look at one of these objects with your eyes.

## 2 Coding Assignment

1. For this assignment, you need access to a tool for generating random blackbody curves. I have put a module containing a tool for this in `/home/hama2717/astr2600/shared/blackbody.py` on scorpius and on Canvas.

- To access it through `JupyterHub`, run the following code cell:

```
import sys
sys.path.append('/home/hama2717/astr2600/shared/')
```

- If you do your homework from your own Anaconda installation, you will need to download the `blackbody.py` module (and the `thermal.py` and `constants.py` it uses) from Canvas and save (or upload) them to the same directory as your jupyter notebook or save them in your own `$PYTHONPATH`.

2. Test that you have access to this module by running:

```
import blackbody
blackbody.testBlackbody()
```

This should generate a fake blackbody of a (semi-)random temperature and make a plot of it.

3. (5 pts) Write a Python function called `integrateSpectrum` that uses the Trapezoidal Method to numerically estimate the bounded integral. The first few lines of its definition are outlined here. It's up to you to fill in the rest of the function with code:

```
def integrateSpectrum(wavelengths, fluxes, w_lower, w_upper):
    """
    This function performs a numerical integral of a spectrum.

    Arguments:
        wavelengths = a numpy array of wavelengths, in nanometers
        fluxes = a numpy array of fluxes, corresponding to w
        w_lower = the lower wavelength limit of the integration
        w_upper = the upper wavelength limit of the integration

    Returns:
        the integral of fluxes, with wavelength limits from w_lower to w_upper
        (this should be one floating point number)
    """

    # now you write the rest!
    #HINT: Booleans arrays can be really useful for selecting ranges out of an array
```

4. (2 pts) Test your function on the following imaginary spectrum ($F(x) = x$), temporarily ignoring that the flux units do not make sense, and calculating the numerical integral from 500 to 600 nm.

```
imaginary_w = np.sort(np.random.uniform(300, 1000, 1000))
imaginary_f = imaginary_w
print integrateSpectrum(imaginary_w, imaginary_f, 500, 600)
```

Based on the analytic integral of this imaginary spectrum ($F'(x) = \frac{1}{2}x^2$), use Python (as a calculator) to calculate the *analytical* solution over those same limits. Notice the np.random.uniform() function. This generates a random set of (unevenly spaced) data (1000 points from 300 to 1000) for your wavelengths. Your numerical and analytic results are likely slightly different. Explain why this is the case in a Markdown/RawNBConvert cell.

5. (5 pts) In Python, RGB colors are expressed in three-element arrays. For example, `[1.0, 0.0, 0.0]` is red, `[0.0, 1.0, 0.0]` is green, `[0.0, 0.5, 0.5]` is sort of bluish-green, and `[1.0, 1.0, 1.0]` is white. Let's estimate an RGB color for a spectrum by integrating the flux of wavelengths

corresponding to red, green, and blue. Those integrals will represent the brightness in each of the three RGB colors; when mixed together they would reproduce the visual appearance of the color an object would have. The fluxes integrated over the three RGB ranges would be:

$$R = \int_{565nm}^{660nm} f(w)dw$$

$$G = \int_{485nm}^{570nm} f(w)dw$$

$$B = \int_{400nm}^{490nm} f(w)dw$$

Write a Python function called `estimateRGB`, that accepts wavelength and flux arrays of a spectrum, uses your `integrateSpectrum` function (perhaps more than once), and returns an RGB color array. The start of this function is outlined here:

```
def estimateRGB(wavelengths, fluxes):
    """
    This function estimates the RGB color of an object, from its spectrum.

    It calculates the integrals the flux over wavelength ranges
    that correspond roughly to the red, green, and blue colors
    displayed on computer monitors.

    Arguments:
        wavelengths = a numpy array of wavelengths, in nanometers
        fluxes = a numpy array of fluxes, corresponding to w

    Returns:
        an RGB color, expressed as a three-element array,
        with R as the first element, G the second, B the third

        (In Python these values must span from 0.0 to 1.0, so you
        will need to renormalize your integrals by the maximum of
        the three values, to ensure this happens.)
    """

    # now you write the rest!
```

6. (1 pt) Test your function on that same imaginary spectrum as before

```
print(estimateRGB(imaginary_w, imaginary_f))
```

You should get an answer *close* to `array([ 1.0, 0.77, 0.69])`.

7. (3 pts) Using the `blackbody` module, generate a blackbody spectrum of a **randomly generated** temperature (*the function itself should choose the temperature, not you!*) and store it in a variable called `firstspectrum`. You will need to look over the `blackbody.py` code or use the jupyter help tools (`blackbody??`) on the `blackbody` module to figure out how to do that and to answer some of the following questions. Do the following:

- State the data type of the variable `firstspectrum`.

- Determine how to extract wavelengths and fluxes from this variable.
- State what units the wavelengths are in.
- From your `firstspectrum` object, create two new numpy arrays: one to contain the wavelengths and another to contain the fluxes.

8. (4 pts) Make a plot of this sample blackbody, with wavelength on the x-axis and flux on the y-axis. Use your `estimateRGB` function to calculate an RGB color for the spectrum, and use that RGB color to set the color of your plot. Your spectrum variable should contain an 'objectID' dictionary key that contains a string describing the spectrum. Use this string to set the title of the plot. Make similar plots for at least **ten** more randomly generated spectra. *Because of the large range of possible intensities, you should plot each spectrum on its own plot.*

9. (1 pt) What trends do you expect between the color and shape of the spectrum? What trends do you notice? Discuss!

10. (3 pts) Again using functions within the `blackbody` module, generate three blackbody spectra of specific temperatures: 5000K, 10,000K, 20,000K. Using your `integrateSpectrum` function, integrate each of these spectra across the *entire* visible spectrum (400 - 700nm). The resulting values can be used to generate a relative comparison (using ratios!) of how bright each of these objects would appear to your eyes. Separately, compare how bright the 20,000K and 10,000K object would look when compared to the 5,000K object. *(Since you only integrated across the visible spectrum, it may not be quite what you would expect from a certain S-B law you may have learned in your physics or astronomy classes!)*

## 3 Turn in your Assignment

Your final version of your assignment should run from top to bottom without errors. (You should comment out or delete code blocks that were just used for testing purposes.) To create a clean version, rerun your notebook using "Kernel|Restart & Run All." Be sure to save this final version (with output). To submit, click "File|Download As >|HTML (html)" inside jupyter notebook to convert your notebook into an HTML web page file. It should have a name like `homework#_{identikey}.html` (where # is the appropriate letter for this week's homework). Please upload this HTML file as your homework submission on Canvas.