

Geosci 232 Problem Set 3  
Fall Quarter 2009  
Due Wed. Nov 11

October 29, 2009

### 3 Problem Set: Python, greenhouse effect and ice-albedo feedback

*Note:* The Chapter Scripts referred to in this problem set can be downloaded from the ClimateBook web site. Scroll down to the ChapterScripts section, and look under the appropriate chapter. You should download a copy of the script you want, put it into your own directory on `geoflop` (the easiest way to do that is to copy and then paste into an `idle` editor window, then save), and then run/modify it as per your requirements. The Workbook Datasets referred to here are in the directory `/home/rtp1/WorkbookDatasets/Chapter1Data`. To read data from a file `data.txt` in subdirectory `GlobalWarming`, you could first put

```
datadir = '/home/rtp1/WorkbookDatasets/Chapter1Data/'}
```

in your script then do

```
{\tt myData = readTable(datadir+'GlobalWarming/data.txt')}
```

This puts your data in a `Curve` object called `myData` for plotting or further manipulation. Remember, you need to first do `from ClimateUtilities import *`.

**Problem 3.1** *Arithmetic on columns*

One of the most basic forms of computation is the performance of arithmetic on rows or columns of tabular data. This kind of operation is very familiar from spreadsheets and commercial graphing software. All modern computer languages allow the performance of such operations on *arrays* or *vectors* without the need to write a loop dealing with each entry separately. An array or vector can be thought of as a row or column of data.

To exercise this skill, make up a column of data corresponding to pressure levels in an atmosphere. The pressure will be measured in millibars (*mb*), but the units are immaterial for this problem. The top of the atmosphere should start at 10 *mb*, and the data should extend to 1000 *mb*, which is a typical surface pressure. Next compute a column giving temperature on the *dry adiabat*, which you will learn about in Chapter 2. The dry adiabat for Earth air is defined by the formula  $T(p) = T_g \cdot (p/p_s)^{2/7}$ , where  $T_g$  is the ground temperature and  $p_s$  is the surface pressure. Use  $T_g = 300K$ . Use the data in this column to make a third column which gives the corresponding temperature in degrees Fahrenheit (just hold your nose and pretend you are making a plot to be used in a television weather show).

To do a somewhat more interesting application of these skills, first read in the instrumental surface temperature record in the file `GISSTemp.txt`, located in the `GlobalWarming` subdirectory of the `Workbook datasets` directory for this chapter. These are estimates of global mean temperature. The first column gives the year, and the remaining columns give the temperature for each month, and for selected annual and seasonal averages. The temperatures are given as deviations from a baseline temperature, measured in hundredths of a degree *C*. Divide the annual average column (labeled "J-D") by 100 and add 14 *C*, to convert to actual temperature. Plot this curve and take a look at it to make sure it is reasonable (apart from the fact that it is going up at an alarming rate).

Now take the difference of the June-August column ("JJA") and the December-February column ("DJF"). These correspond to Northern Hemisphere summer, but remember that these are global means, so that Northern Summer is being averaged with Southern Winter. Global mean JJA still tends to be warmer than global mean DJF, because the seasonal cycle is strongest in the Northern hemisphere. However, note that the columns give the *deviation* of the temperature from the baseline average for these months, so the difference you computed gives the *change* in the strength of the global mean seasonal cycle, not its absolute magnitude. Divide your result by 100

to convert the deviation to degrees. Do you notice any interannual variability in the strength of the seasonal cycle? What is the typical time scale? Do you notice an trend?

*Python tips:* The `Curve` object supports arithmetic on columns. Specifically, if `'x'` is a data column in the curve object `c`, then `c['x']` returns an array on which arithmetic can be done just as if it were a regular number. For example, `2*c['x']` is an array in which the entries are all doubled, `c['x']**4` returns an array consisting of the fourth powers of entries, etc. You can even do things like `c['T'] = c['T'] + 273.15` to convert a temperature from Celsius to Kelvin in-place, and can do similar things to create new columns that don't initially exist. You can compute the average of a column using constructions like `sum(c['T'])/len(c['T'])`. Many other array arithmetic operations can be done using the array functions available through `numpy` or its older version `Numeric`.

**Problem 3.2** *Plotting a function*

Make a graph of Eq. 1.1 showing the evolution of Solar luminosity over time. As the vertical axis, you may use the luminosity relative to its present value. Show values out to twice the current age of the Sun. On the same plot show a straight-line approximation to the function, which yields the same values as Eq. 1.1 at  $t = 0$  and  $t = t_{\odot}$ .

There are two different ways to go about making the plot. One alternative is to generate the data to be plotted in the same software package you use for plotting, and then carry out the operations (commands, menu choices, etc.) needed to carry out the plot. The second alternative is to generate the data in one programming environment, write it out to a text file, and then read the file into the plotting function of your choice. This can be handy if you want to use commercial plotting software that produces professional-looking figures.

*Python Tips:* Using the courseware in Python, you do this by making a list of arguments to the function, making a list of corresponding function values, installing each list in a `Curve` object, and then plotting. If the curve object is `MyCurve`, you plot using `plot(MyCurve)`. Check the examples in the supplementary material to see how to set plot options such as axis labels, titles, etc. As an alternative to plotting within Python, you can use the `dump` method of the `Curve` object to write out the data to a text file, which you can then read into the plotting software of your choice. For example, `MyCurve.dump('out.txt')` writes the data in `MyCurve` to a text file called `out.txt`.

**Problem 3.3** The physicist Freeman Dyson has speculated that an advanced civilizations might enclose its star within a spherical shell of material, and live on the inner surface. This gives them more living space, and avoids wasting all that starlight that would ordinarily escape to space. Consider a red dwarf star with photospheric temperature  $4000K$  and radius  $200,000km$ . What should the radius of the Dyson sphere be in order to have an inner-surface temperature of  $300K$ ? Assume that the sphere material is a perfect blackbody with zero albedo at all wavelengths, and that it is a good conductor of heat so that the inner surface and outer surface temperature are identical. You may also assume that the temperature of the photosphere remains unchanged after the Dyson sphere is built. If you were an astronomer looking for such a Dyson sphere, in what part of the electromagnetic spectrum should you look? What radius should the Dyson sphere have if it is instead made of a good thermal insulator, so that the outer surface is much, much colder than the inner surface?

*Challenge question:* In either case, is it actually reasonable to assume the photosphere temperature stays unchanged? If you think it should change, what value should it take on assuming: (a) the fusion energy output of the star remains unchanged, and (b) the radius of the Dyson sphere is adjusted so that the inner temperature is  $300K$  as before.

**Problem 3.4** For Jupiter, the observed  $OLR$  is  $14.3W/m^2$ . Compute the effective radiating temperature. Referring to the Jupiter temperature profile given in Chapter 2, estimate the effective radiating pressure of Jupiter. Is there more than one possible value? Which of these do you consider more likely?

**Problem 3.5** *Background for this problem:* On the "dry adiabat" the temperature vs. pressure is given by  $T(p) = T_s \cdot (p/p_s)^{R/c_p}$ .

Consider an atmosphere whose temperature profile is on the dry adiabat with  $R/c_p = \frac{2}{7}$ , all the way down to zero pressure (infinite altitude). The surface temperature is  $300K$  and the surface pressure is  $10^5 Pa$ . The gravity is the same as Earth's gravity,  $9.8m/s^2$ . We mix in a hypothetical greenhouse gas with mass concentration  $q$ . The greenhouse gas has the property that its absorption coefficient  $\kappa$  is  $1m^2/kg$  for wavelengths between  $15 \text{ microns}$  and  $18 \text{ microns}$ , and is zero elsewhere. Mixing in the greenhouse gas does not change the temperature profile, and you may assume that  $q$  is small enough that the change in surface pressure may be neglected. The radiating pressure for any given  $\kappa$  is estimated using  $\kappa q p_{rad}/g = 1$ , and the radiating temperature is the

temperature at this pressure level. Using this estimate, answer the following questions.

The radiating temperature, or *brightness temperature*, for any give frequency  $\nu$  is the temperature  $T_{rad}$  such the actual emission to space is equal to the blackbody emission  $\pi B(\nu, T_{rad})$ . Sketch a graph of the brightness temperature vs frequency, showing curves for a range of  $q$  from zero to 0.1. Compute a graph of the way the spectrum of outgoing radiation (as observed from space) depends on frequency, for the same range of  $q$ . Finally, graph the behavior of the *net* outgoing radiation (i.e. the integral of the curves in the previous question) as a function of  $q$ .

*Hint:* You can answer the last question using the fact that the integral of  $\pi B(\nu, T)$  over all  $\nu$  is  $\sigma T^4$ , together with a numerical evaluation of the integral over the 15 to 18 *micron* band. When the frequency interval is small enough that  $B$  is essentially constant over the frequency range, then the numerical integral should be approximately  $\pi B(\nu_o, T)\Delta\nu$ , where  $\nu_o$  is the frequency at the center of the band. How well does this approximation match the numerically computed value?

**Problem 3.6** *Computing the Ice-Albedo Hysteresis Diagram*

Calculations with a complete real-gas radiation simulation indicate that, for a  $CO_2$  concentration of 300ppmv, and with an atmosphere on the moist adiabat, a reasonable fit to the actual *OLR* curve in the range of 220K to 310K is the linear fit  $OLR(T) = a + b \cdot (T - 220)$  where  $a = 113W/m^2$  and  $b = 2.177W/m^2K$ . Compute the ice-albedo hysteresis diagram giving the set of equilibrium temperatures as a function of the solar constant  $L_\odot$ . Note that there is a simple trick for getting the bifurcation plot. The equation determining the equilibrium is  $\frac{1}{4}L_\odot(1 - \alpha(T)) = OLR(T)$  Instead of specifying  $L_\odot$  and finding the  $T$  that satisfy the equation, we can re-write the equation as

$$L_\odot = 4 \frac{OLR(T)}{1 - \alpha(T)} \tag{1}$$

Now, if we call the right hand side  $G(T)$ , then  $G(T)$  gives the unique value of  $L_\odot$  which supports the temperature  $T$ . Hence, to get the bifurcation diagram, you can just plot  $G(T)$  and then turn it sideways.

Use the same albedo-temperature function defined in Chapter 3. Assume that the albedo for an ice-free Earth is .2 and for an ice-covered Earth is .6.

Based on your calculations, if  $CO_2$  were held constant how much would  $L_\odot$  have to be reduced from its modern value before Earth was forced to fall

into an inevitable snowball state? Using the inverse-square law and assuming a circular orbit, compute how far out from the Sun the Earth would have to be displaced (relative to its present orbit) to achieve this solar constant. Conversely, how close to the Sun would you have to place the Earth before a Snowball state became impossible? *Note:* The assumption of fixed  $CO_2$  is very unrealistic, since tectonically active planets with water have a way of adjusting  $CO_2$  in response to changes in the solar constant. This will be discussed in Chapter ??.

*Python Tips:* The script `IceAlbedoZeroD.py`, found in the Chapter Scripts collection for this chapter, calculates and plots hysteresis diagrams using this method, and also makes various other plots useful in understanding bifurcations due to ice-albedo feedback. You can easily modify this script to solve this and similar problems, by redefining the `OLR(T)` function, and (in other cases) also the albedo function.