Introduction to scientific computing

August 2017

In this assignment, we focus on loading data and creating plots. This includes adding titles, legends, axis labels, and error bars.

Exercise 1

Some experimental data may be stored across multiple files. For example, one file for each run of an experiment. Create a script that

- loads data from each file in the directory `to_combine`;
- combines that data into one `array`;
- outputs the mean and standard deviation of the combined data.

Exercise 2

A common task is plotting a function to visualize it. This may be just to have a quick look, or as a comparison against another data source. In numerical computing, we also need to provide which points to sample the function. When choosing the number of points, use enough to keep the curve smooth, but not too many or too much memory will be used.

For this question we'll plot the Lennard-Jones (12-6) potential, which approximates the interaction between two neutral atoms. The dimensionless form is

$$V(r) = \left(\frac{r_m}{r}\right)^{12} - 2\left(\frac{r_m}{r}\right)^{6}$$

where $r_m = 2^{1/6}$ is the minimum of the potential well.

Create a program to do the following:

1. import the `matplotlib.pyplot` and `numpy` modules;
2. create the r data with `numpy.linspace()`;
3. create the V data as in defined in the equation;
4. draw the plot with `pyplot.plot()`;
5. label the x and y axes with `xlabel()` and `ylabel()`;
6. rescale the axes as appropriate with `xlim()` and `ylim()`
7. add a title "Lennard-Jones potential" using `title()`;
8. save the plot as a `.pdf` file using `savefig()`.

Choose a domain in x and number of samples that you consider reasonable when calling `linspace()`. *Hint: there is a singularity at $r = 0$, so the domain should start slightly above zero.*

The reason we save in `.pdf` is to have vectorized graphics, which is typically better for printing/publishing. You can also save to `.png`, `.eps` or `.svg`.

Exercise 3

This is a more involved plotting exercise. You will plot multiple curves on one set of axes with a legend, title, and axis labels.

Having two plots on the same axes requires a legend. A good way to do this is labeling the individual plots, then calling the `legend()` function with no arguments. To label plots, we use the 'keyword argument' `label=`. For example,

```
plot(x, y, label="a data set")
```

In statistical mechanics, the Maxwell-Boltzmann distribution describes the distribution of speeds of particles in an ideal gas at a given temperature (T). Its probability density function (PDF) is

$$f(v;T) = \sqrt{\left(\frac{m}{2\pi kT}\right)^3} 4\pi v^2 e^{-\frac{mv^2}{2kT}},$$

where $k \approx 1.38 \times 10^{-23}$J/K is the Boltzmann constant, m is the particle mass, and v is the particle speed.

Use `pyplot` to plot the PDF for Nitrogen ($m \approx 4.652 \times 10^{-23}$ g) at the temperatures 80 K, 233 K, and 298 K. Put all three plots on one set of axes, and label them with their temperatures. Be sure to include a title, axis labels, and a legend, and scale the axes appropriately. Save the figure as a PDF, and submit it with the source code.

Suggestion: define a function to calculate the PDF instead of typing/copying the expression for each temperature.

Exercise 4

Sometimes you will only want part of an array. 'Slicing' arrays is the way to do it in python. For example, the expression `data[a:b]` gives the subarray with elements from `a` to `b-1` from `data`. If either `a` or `b` are omitted, the slice goes from the beginning or to the end. With multi-dimensional data, slices are still separated by commas, e.g. `data[a:b,c:d]`.

Create a program that adds the first and third rows of the following data, then prints the result (*There should be 4 numbers in the result*).

```
import numpy as np
data = np.array([[ 2,  5, 6,  2],
                 [ 0,  1,  0,  0],
                 [1,  1, -1,  1],
                 [ 20,  20,  1,  0],
                 [ 9, 1.6,  4.2, 2],
                 ])
```

Exercise 5

In this question, we supply you with a data file which you will use to make a scatter plot.

Data files are particularly useful for loading measurements – whether taken by hand or from software like LabView. With measurements, you should also include measurement errors on your plots in the form of error bars. The `pyplot.errorbar()` function should be used when plotting measured data.

Load the file, `sample_data.txt`, and create a scatter plot with error bars. Submit the plot along with your code. *Hint: you can either slice the array of data or use the unpack=True option to extract the x, y, and σ_y values.*