Ph 20.1 – Introduction to Python – Lissajous figures and beats

Due: Week 2 -v20170314-

Overview

Welcome to the first assignment of the physics computation sequence! Ph 20–22 is meant to give you a toolkit for approaching numerical and computational problems like a physicist. In this term you will learn the basic tools that every practicing physicist who works with data should be familar with:

- the Python programming language (we recommend Python 3.7),
- the Mathematica symbolic computation system,
- the LaTeX typesetting language, and
- the basics of the Linux operating system/Unix shell needed to work with these tools.

We'll focus not on learning these tools for their own sake but on using them to understand numerical results. The typical output of each assignment will be several figures which give insight into some topic, accompanied with text explaining each figure and its significance. This isn't very different from many scientific papers! If you continue with the rest of the sequence you will gain more tools for and experience with data analysis (in Ph 21) and numerical methods (in Ph 22).

This Assignment

The primary purpose of this assignment is to familiarize you with the edit/run/edit cycle typical of scripting languages, and of Python in particular. (By contrast, the cycle of compiled languages such as C or C++ could be described as edit/compile/debug/run.) Once you are done, you should have a working knowledge of the operating system (Linux), of basic shell operations (copying, moving, editing files), of the Python interpreter, and of graphing tools.

Your workflow on this assignment will probably consist of several phases:

1. Figuring out how to use the unix shell ("terminal") to open a text editor (e.g. `emacs`) to create and edit python programs and a python interpreter (e.g. `ipython`) to run them;
2. Writing a basic program to compute the trigonometric functions $X(t)$, $Y(t)$, and $Z(t)$ described below, using `for` loops and Python lists;
3. Learning how to use the `numpy` Python package to rewrite the program without `for` loops using `numpy` arrays;
4. Learning enough of the `pyplot`/`matplotlib` python package to plot these functions;
5. Figuring out how to pass arguments to those functions (using `sys.argv`) and output the sequences they produce to a file (using Python file objects and/or `numpy.savetxt()`);
6. Using your code to investigate Lissajous figures and beats, as described below, and produce plots illustrating the conclusions you draw;
7. Writing a report as a LaTeX document which contains both these figures and commentary on them, and using the `pdflatex` Unix utility to compile it into a PDF.

Your code and report should be sent to your TA by the deadline listed on the website. You'll probably want to be on or past the 4th step going into the second lab; if you haven't gotten to that point after the first lab you should plan to do some work outside of class before the second lab meets.

Your TA can help you with all of these steps, especially on getting started using Unix and the lab computers, but for the most part we intend for you to look things up and work things out for yourself—which is of course what you'll need to do as a praticing scientist! The hints section at the end of the assignment should get you started, but the nice thing about programming, especially in Python, is that most of your questions can be answered with a quick web search. In particular, documentation is your friend:

- The Python Tutorial: `https://docs.python.org/3.7/tutorial/` (if you've never seen Python before, skim the first 3–5 sections as needed)
- The Python Standard Library: `https://docs.python.org/3.7/library/index.html`. For this assignment you'll want to know about file objects and the `math` module.
- NumPy Manual: `https://docs.scipy.org/doc/numpy/index.html`. See in particular the User Guide and the sections of the reference on array creation and manipulation routines. Doing a web search for numpy and the thing you want to do will almost always take you to the function you're looking for in the documentation.
- PyPlot: Start with `http://matplotlib.org/api/pyplot_summary.html`. Note that this module goes by many names: although there are slight differences between them, for our purposes `pyplot`, `pylab`, and `matplotlib` are the same thing and they all share documentation.

In the `ipython` interpreter you can also access this documentation by typing a function name and putting a `?` after it. You may also find it helpful to browse other tutorials, e.g. the ones available on the website *Software Carpentry*.

Visualizing trigonometry: Lissajous figures and beats

Sine and cosine functions (sinusoids) are fundamental in the analysis of many physical systems, and in particular of those that undergo periodic motion. A reasonable (and venerable) introduction to the subject of periodic functions is given in Chapter 2 of A. P. French, *Vibrations and Waves* (Norton, New York, 1971). You will be studying (or must have studied) these topics exhaustively in Sophomore Physics.

Lissajous figures are named after the French physicist Jules Antoine Lissajous (1822–1880), who studied the mechanics of tuning forks,[1] and who developed a method to calibrate a new tuning fork against a reference tuning fork ["Étude Optique des Mouvements Vibratoires," *Annales de Chimie et de Physique* **51** (1857)]. Lissajous attached mirrors to the two forks, which were positioned orthogonally, and shone a beam of light on a screen, via the two mirrors (see illustration). On the screen, the vibrations of the two mirrors are translated into the horizontal and vertical motion of the point of light, and a closed elliptical pattern results if the frequencies of the two forks are in agreement. In modern laboratories, at least before the age of computers and

[1] Among other things, Lissajous recommended that the standard "A" note be set to 435 Hertz (periods per second). The modern value is 440 Hertz, at the end of an arms race where competing orchestras would tune their instruments higher and higher to sound sharper and more incisive.

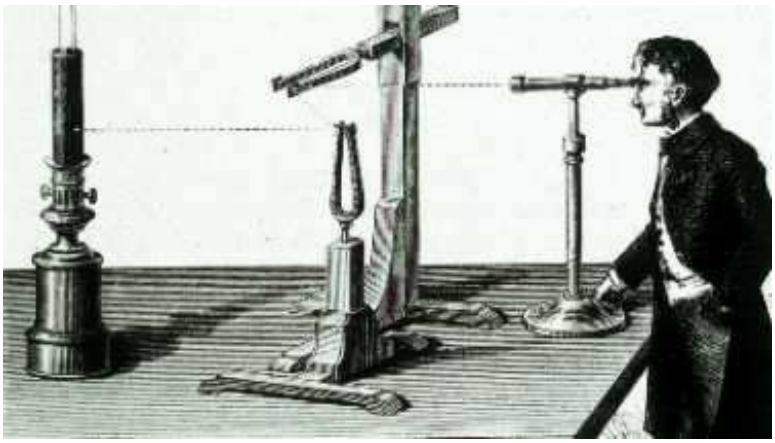

Figure 1: Lissajous' fork-tuning apparatus (from `http://physics.kenyon.edu/EarlyApparatus/index.html`).

acquisition boards, Lissajous figures were familiar sights on the screens of oscilloscopes, because they allowed quick comparisons of frequencies and phases.

The phenomenon of *beating* originates from the sum of two sinusoids of different frequencies. The resulting signal is a sinusoid of frequency given by the average of the two original frequencies, but whose amplitudes oscillates with a frequency given by the difference of the two original frequencies: in formulas,

$$\cos(\omega_1 t) + \cos(\omega_2 t) = 2\cos\Big(\frac{\omega_1 + \omega_2}{2}\,t\Big)\cos\Big(\frac{\omega_1 - \omega_2}{2}\,t\Big). \tag{1}$$

Beats have many applications in music, optics, and communication. In the technique of *amplitude modulation*, used for radio transmission in the AM band, a *carrier signal* is multiplied by the *content signal* (i.e., music and words) before being transmitted. Usually the carrier frequency is rather large (in the MHz range), while the content signal can be characterized as the superposition of many sinusoids with frequencies within the audible range, 20–20,000 Hz (we shall learn more about frequency components in Ph22, when we work with *Fourier transforms*). The amplitude-modulated signals would then contain a superposition of sinusoids with frequencies scattered around the carrier frequency, within a bandwidth of 2×20,000 Hz. Radio receivers use various schemes to *demodulate* AM signals: the simplest are based on tunable resonant circuits followed by diode rectifiers and low-pass filters.

The Assignment

1. Browse some of the resources listed above to familiarize yourself with their contents. Look in particular at the information about Python. The computation laboratory has various Python references for additional information.

2. Write a program to compute sequences of the simple trigonometric functions

$$X(t) = A_X \cos(2\pi f_X t), \tag{2}$$
$$Y(t) = A_Y \sin(2\pi f_Y t + \phi), \tag{3}$$
$$Z(t) = X(t) + Y(t), \tag{4}$$

at the equally spaced times $t = n\Delta t$, with $n = 0, \ldots, N$. The program should take as input the parameters f_X, f_Y, A_X, A_Y, ϕ, Δt, and N; it should output the X, Y, and Z sequences to one or more ASCII files.

Construct two versions of the program: one that uses `python` lists to store the X, Y, and Z sequences, and another version that uses `numpy` arrays.

You will then use the `pyplot` module, or another program of your choice to plot the three functions, as directed in the next two parts of the assigment. If desired you may work out how to plot the functions using `xmgrace`, `gnuplot`, *Mathematica*, or other utilities, but we will use `pyplot` throughout the Ph 20–22 sequence. Be mindful of Tufte's principles mentioned in this week's reading. As mentioned above, you can find plotting help by looking up `matplotlib`, looking among the "Resources" on the ph20 homepage, or asking your TA.

3. *Lissajous figures.* Create these figures by plotting X against Y. Then:

   (a) convince yourself that the figures are closed curves if f_X/f_Y is a rational number, and produce plots showing that this is true;

   (b) investigate the relation between the ratio f_X/f_Y and the shape of the resulting curve, write your conclusions, and print some examples for your TA;

   (c) for $f_X = f_Y$, investigate the relation between the phase shift ϕ and the shape of the curve, write your conclusions, and print some examples (how can this relation be used to tune an electronic circuit, using an oscilloscope?).

   Always choose values of f_X, f_Y, and Δt such that your figures are not too cluttered, and that curves look... curvy.

4. *Beats.* Demonstrate the phenomenon of beats by setting nearby (but different) f_X and f_Y, and plotting $Z(t)$ as a function of t. Choose the frequencies so that your plots display many cycles at the carrier frequency $(\omega_1 + \omega_2)/2$, and a few modulation cycles at the frequency $(\omega_1 - \omega_2)$, where $\omega_1 = 2\pi f_X$ and $\omega_2 = 2\pi f_Y$. [question: why not $(\omega_1 - \omega_2)/2$?]

5. Write two paragraphs about this week's experience (perhaps your first) programming in Python. How does it compare to your experiences using other programming languages? Do you agree with Guido van Rossum (see this week's readings) in his evaluation of his own language? [If you prefer, instead of writing two paragraphs of prose, you may jot down a checklist and discuss it verbally with your TA.]

Python Hints

First of all, we suggest you use `emacs` to edit your Python programs. If correctly configured, it will load a Python mode that will give you syntax highlighting (meaning that your keywords, variables, etc. will be shown in different colors, making it easier to browse your code) and help you with indentation, which in Python is important for the execution of the code. See the reading for this week for `emacs` help.

It is a good idea to use `ipython` rather than the very basic `python`. You can add `-pylab` for useful numerical and plotting packages. Your output will look something like this:

```
% ipython -pylab
```

```
IPython 0.13 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

Welcome to pylab, a matplotlib-based Python environment [backend: MacOSX].
For more information, type 'help(pylab)'.
```

You might also want to try using python notebooks, the command line being (for example):

```
>>>  ipython notebook --pylab
```

You do not need much Python to get through this week's assignment. Here's what you should know: you can go find it in your Python tutorials and manuals.

- You should have an understanding of the assignment of variables, of writing simple mathematical expressions, and of coding a `for` loop using `range`.
- You should know that your Python script (say, `lissajous.py`) will run from the shell if its first line is

  ```
  #!/usr/bin/python
  ```

  and if you make it executable (`chmod u+x lissajous.py`).
- You should know that `cos`, `sin`, and `pi` reside in the `math` *module*, so you can access them as `math.cos`, etc. after doing `import math`. Alternatively (and preferably) use `np.cos`, etc. after doing e.g. `import numpy as np`. We encourage you to use numpy arrays because of the flexible indexing and slicing operations that are possible . See, for example, `http://wiki.scipy.org/Tentative_NumPy_Tutorial`.
- You should know how to get arguments into your script from the command line. If you import `sys`, then `sys.argv` is a *list* of strings giving, in order, the name with which your script was called, and its arguments. I said they're strings, so you will need to do things like

  ```
  N  = int(argv[1])
  dt = float(argv[2])
  ```

  to get numerical values. You can check how many strings you're passing from `len(sys.argv)`. This is primarily a `unix` feature, not `python`. Often, you will be using `python` in interpretive mode and not calling python code from the unix shell.
- You should know how to get numbers out of your script and into an ASCII file from which you can then plot. For the purpose of this assignment, you can use `print` (which will go to the console), and *redirect* the output to a file, as in

  `python lissajous.py` [*arguments...*] `> output.txt`

Finally, it is often somewhat tricky to correctly install python modules and ensure complementarity among different modules. If you are interested in installing python on one of your own computers, we suggest that you take a look at `https://www.anaconda.com/distribution/`. In most of the situations, it is a very useful tool to set up python, to manage python environments and to download modules.

LaTeX Hints

In Ph20, you are required to use LaTeX to write your report of classwork. The easiest way to start is looking at `https://www.overleaf.com` and going through some of the documentation at `https://www.overleaf.com/learn`. In this way, you'll be able to write LaTeX and compile online without installing the LaTeX distribution on your computer which is usually annoying. Of course, if you have installed LaTeX on your own computer, you can edit and compile your tex files locally. The lab computers also have LaTeX installed, you can compile there.