Physics 750: Exercise 1

Thursday, August 24, 2017

1. Log in to the account on your computer. Open a terminal window to access the command line interface. See what's in your home directory, and check that your default shell is set to BASH.

```
$ ls -F
Desktop/   public_html/
$ mkdir phys750
$ ls -F
Desktop/   phys750/   public_html/
$ cd phys750
$ env | grep SHELL
SHELL=/bin/bash
```

2. Download the Exercise 1 instructions and source code from the class website. You can either do this from the terminal as follows.

```
$ WEBPATH=http://www.phy.olemiss.edu/~kbeach/
$ curl $WEBPATH/courses/fall2017/phys750/src/exercise1.tgz -O
$ tar xzf exercise1.tgz
$ cd exercise1
```

3. Inside the `exercise1` directory is a C++ source file `gaussian.cpp` and a `makefile` containing instructions to compile the program.

```
$ ls
gaussian.cpp   makefile
$ head -n4 gaussian.cpp
// read in required header files
#include <iostream>
using std::cout;
using std::endl;
$ make gaussian
g++ -o gaussian gaussian.cpp
$ ls -F
gaussian*  gaussian.cpp   makefile
```

4. The program `gaussian` defines the function $f(x) = Ce^{-ax^2}$ and outputs a three-column table of values

$$x \qquad f(x-1)|_{C=2,a=1} \qquad f(x-2)|_{C=1.5,a=2}$$

with x ranging in discrete steps over the interval $[-1.5, 4.5]$.

```
$ ./gaussian
           -1.5      0.00386091      3.4346e-11
         -1.494      0.00397835      3.7353e-11
         -1.488      0.00409906     4.06175e-11
            .            .              .
            .            .              .
          4.488     1.04074e-05     6.30087e-06
          4.494     9.98039e-06     5.93522e-06
            4.5     9.57023e-06     5.58998e-06
```

5. The program output can be redirected to a file (using >) and viewed with `gnuplot`.

```
$ ./gaussian > curves.dat
$ gnuplot
> plot[-1.5:4.5] "curves.dat" using 1:2 with lines
> replot "curves.dat" using 1:3 with lines
```

6. Superimpose the arithmetic and geometric means of the two curves.

```
> replot "curves.dat" using 1:(0.5*($2+$3)) with lines
> replot "curves.dat" using 1:(sqrt($2*$3)) with lines
```

7. See if you can figure out what's going on here.

```
> a1=1.0; C1=2.0; x1=1.0;
> a2=2.0; C2=1.5; x2=2.0;
> plot "curves.dat" using 2:3
> replot C2*exp(-a2*(x1+sqrt(-log(x/C1)/a1)-x2)**2)
> replot C2*exp(-a2*(x1-sqrt(-log(x/C1)/a1)-x2)**2)
> quit
```

8. Use `emacs` (or your favourite text editor) to modify the `gaussian.cpp` program file. Change the function to $f(x) = Ce^{-a|x|}$. (You might want to use the `fabs` function.†) Recompile, and plot everything again.

```
$ emacs gaussian.cpp &
$ make
```

9. A *Lissajous figure*‡ refers to a planar trajectory that is harmonic in two orthogonal directions. This is something you might have seen traced out on an oscilloscope.

   Write a C++ program that computes the quantities

$$x(t) = A\cos(at)$$
$$y(t) = B\cos(bt + \delta)$$

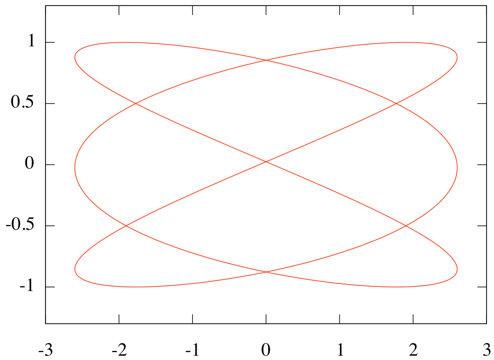
   at $100N$ equally spaced points in the range $0 < t < 2\pi N \times \max(1/a, 1/b)$ and outputs the results in three-column format $t, x(t), y(t)$ to the standard output stream (stdout, referred to in C++ as `cout`). Have your program require six command line arguments: the first five interpreted as floating-point numbers (with the `atof` function, say) and used to set the values of A, B, a, b, δ; the sixth interpreted as an integer (with `atoi`) and assigned to N. The program output can then be written to a file via redirection (>) and viewed with `gnuplot`.

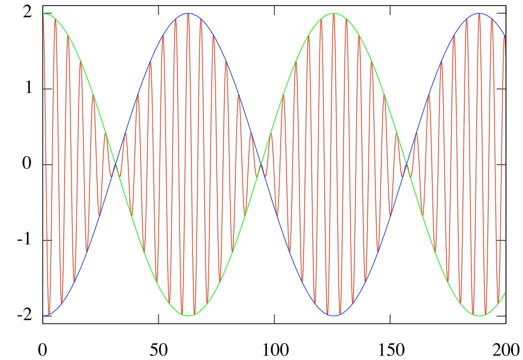
```
$ make lissajous
$ ./lissajous 2.6 1 3 2 0.5 2 > curve1.dat
$ ./lissajous 1 1 1.1 1.2 0 35 > curve2.dat
$ gnuplot
> plot "curve1.dat" using 2:3 with lines
> plot "curve2.dat" using 1:($2+$3) w l, 2*cos(0.05*x), -2*cos(0.05*x)
> quit
```

   If you've done everything correctly, you should see something like this:

† part of the `cmath` library, described in http://www.cplusplus.com/reference/clibrary/cmath/

‡ https://en.wikipedia.org/wiki/Lissajous_curve





(a) Convince yourself that a Lissajous figure is closed iff a/b is a rational number.

(b) How does the ratio a/b control the shape of the curve?

(c) In the case $a = b$, how does the phase shift δ effect the curve?

(d) Investigate the beats produced when the two sinusoidal components—with equal amplitudes and slightly different frequencies—are superimposed. In other words, plot $z(t) = x(t) + y(t)$ versus t for $A = B$ and $|a - b| \ll 1$. The result is a product of a slowly varying envelope function and a rapidly varying beat function:

$$\cos(\alpha t) + \cos(\beta t) = 2\cos\left[\tfrac{1}{2}(\alpha + \beta)t\right]\cos\left[\tfrac{1}{2}(\alpha - \beta)t\right]$$

10. The *Mandelbrot set*[§] consists of the bounded orbits of the complex-valued recurrence relation

$$z_{n+1} = z_n^2 + c, \qquad z_0 = c \equiv x + iy$$

The set is typically visualized as a plot in the x-y plane, with each point corresponding to an unbounded orbit coloured according to its rate of escape.

Write a C++ program that implements the following algorithm. Scan over a fine grid of c values such that its real and imaginary parts range over $x \in [-2, 1]$ and $y \in [-1, 1]$. At each point, run the recurrence relation until $|z_n| > R$ or $n > N$. I suggest the values $R = 3$ and $N = 500$. You'll have to make a decision whether to represent each complex number as two `double`s or as a single `complex<double>` class object.

Output the escape counts n as a rectangular table of values to stdout, and then plot the Mandelbrot set using `gnuplot`:

```
$ make mandelbrot
$ ./mandelbrot > mandelbrot.dat
$ gnuplot
gnuplot> set pm3d map
gnuplot> splot "mandelbrot.dat" matrix
```

11. Take a look at the file `mandelbrot-png.cpp`, which includes sample code for constructing RGB bitmaps in the png format. Modify it so that it draws a Mandelbrot set into the file `out.png`. Note that the mapping from escape counts to RGB values is arbitrary. Feel free to choose whatever transformation gives a compelling visualization.

[§] https://en.wikipedia.org/wiki/Mandelbrot_set

```
$ make mandelbrot-png
$ ./mandelbrot-png
$ display out.png
$ convert out.png out.pdf
$ evince out.pdf
```