Practical 1
Least Squares Fitting

Preliminaries

These practicals follow on from those in the TPOP course and they are in Python. Here we are not interested particularly in data-structures, classes or inheritance. Instead, we are concerned with the code which goes into the functions. In particular, you must code the numerical algorithms discussed in the lectures.

You must complete these practicals in the MS Windows 7 OS as the required modules are only installed there. You should begin by downloading the practicals from the course website and unzipping them into your home directory. Each practical has its own directory. Open Practical 1 now in an editor (IDLE for example).

If you examine the code for Practical 1, you will see a number of functions, some only partly coded. The most important functions are `PartA(), PartB(), ...` These correspond to the parts below. You will complete these functions by following the instructions below. Each part can then be completed by calling the appropriate function at the end and running the code. Sections of code which you must complete are marked

```
#------------------#
⋮
#------------------#
```

Modules

The practicals make extensive use of modules for linear algebra and plotting. These are *SciPy* for scientific computing and *matplotlib* for plotting.

A reference is provided at `http://www-module.cs.york.ac.uk/numa/PyReference.pdf` for the datatypes and functions you are most likely to need to complete these practicals. For more details see `http://www.scipy.org/` for SciPy and `http://matplotlib.org/` for matplotlib. There are extensive reference pages at these websites.

Part A

Simple linear least squares

In lecture 2, the idea of least squares fitting was introduced. The task in this part is to implement this method to fit the equation $y = ax + b$ to the given data:

x	y
1	1.1
2	2.05
3	2.9
4	4.1
5	5.02

Implement this method in the function `simpleLeastSquares`, returning the parameters a and b in the matrix $\mathbf{p}$[1]. The parameter set should be found using the inverse method from Lecture 5 (see the Python library primer on the course website for details of how to compute the inverse).

Call `partA`; the result should be plot where the line fits the points.

Part B

Three-variable linear least squares

In part B, we now extend the problem to one more variable. The task in this part is to implement this method to fit the equation $y = ax_1 + bx_2 + c$ to the given data:

[1] In the notation `[a; b]` the `[...]` signifies an array and `;` denotes the end of a line - `[a; b]` is therefore a 2 by 1 array. `[a b; c d; e f]` would be a 3 by 2 array

x_1	x_2	y
1	1	0.5
2	2	3
3	1	4
1	3	4.5
3	3	5.5
4	2	6
2	4	7

Implement this method in the function `threeDLeastSquares`, returning the parameters a, b and c in the vector $\mathbf{p}$. You will need to work out the formula for this least squares problem using the method described in Lecture 5.

Call `partB`; the result should be plot where the surface fits the points.

Part C

Complex least squares

Not all fitting functions are linear, but as long as the function is linear in the *parameters* we can use the same method. This time you should fit the equation $y = ax^2 + bx + c$ to the given data:

x	y
0	3.1
1	5.8
2	7.1
3	6.1
4	2.7
5	-2.0

Implement this method in the function `complexLeastSquares`, returning the parameters a, b and c in the vector $\mathbf{p}$. You will need to work out the formula for this least squares problem using the method described in Lecture 2.

Call `partC`; the result should be plot where the line fits the points.