Mathematics 2, Homework 3 part 1 **April 12th 2024**

Nelder-Mead method

1. Implement the two and three dimensional versions of Nelder-Mead method. Use a programming language of your choice.

One step beyond: Why fix the dimension? Surely you can pass the dimension as a parameter or better still compute it on the fly.

2. Qualitatively compare Nelder-Mead method with the methods you have implemented in HW2/4 (GD, Polyak GD, Nesterov GD, AdaGrad GD, Newton, BFGS) on

$$f(x,y,z) = (x-z)^2 + (2y+z)^2 + (4x-2y+z)^2 + x + y,$$

$$f(x,y,z) = (x-1)^2 + (y-1)^2 + 100(y-x^2)^2 + 100(z-y^2)^2,$$

and

$$f(x,y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2.$$

These functions are taken from HW2/6a,b,c. For the Nelder-Mead method choose starting samples of different diameters, one of the starting sample points should match the one taken from HW 2.

Black box optimization

Let *xxxxxxxx* be your student Id. The three functions

$$f_{xxxxxxxx,i} : \mathbb{R}^3 \to \mathbb{R},$$

$i \in \{1,2,3\}$ are given in a black-box setting.

The archive `hw3_1_executables.zip` (contained in the HW3 part1 dropbox) contains three (equivalent) command line executables `hw3_2024_linux`, `hw3_2024_mac`, and `hw3_2024_win.exe`, running on three different OS families. Each executable expects five parameters on the standard input and, after a time consuming computation, outputs a real number to the standard output. Below is a sample call.

```
$ ./hw3_2024_mac xxxxxxxx i 3.17 0.71 -1.55
```

Your student Id is the first parameter *xxxxxxxx*, the second parameter is an integer $i \in \{1,2,3\}$, and the remaining three parameters are real numbers.

Try to test the appropriate executable as fast as possible. Please report any problems should the above program not work as intended.

3. Find minima of functions

$$f_{xxxxxxxx,1}, \qquad f_{xxxxxxxx,2}, \qquad f_{xxxxxxxx,3}.$$

Use sufficiently high precision (ten significant digits or more).

These problems are in theory unconstrained, you are guaranteed that none of the calls results in an overflow if real parameters lie in $[-10, 10]$.

One step beyond: How would one use a gradient-descent based method in such a case. Which one is best suitable. Can you beat Nelder-Mead?

Upload your solution in a single .zip archive which contains the source-code (I would be most pleased with a Jupyter notebook in Python - nevertheless you can use any programming platform/language according to your personal preferences) and a .pdf.