Final Project: Newton Fractals

NUMA01: Computational Programming with Python

Claus Führer, Malin Christersson, Robert Klöfkorn

This assignment has 8 tasks.

The goal of the final project is to experience programming in a group. Proceed in the following way

- discuss the mathematical part of the project in the group first until you fully understand the problem
- divide the problem in subproblems
- discuss how the different parts should be tested
- prepare a max 20 min presentation which should present the mathematical background, the organization of your work, your solutions and maybe alternative attempts, which you decided to reject.

Bring your Laptop or an USB stick for the presentation.

We will construct a fractal figure using Newton's method.

Task 1

Write a class fractal2D that is initialized with one function $f : \mathbb{R}^2 \to \mathbb{R}^2$ and, possibly, its derivative.

Task 2

Write Newton's method as a method of this class. This method should take an initial guess as its only argument (and `self`, of course).

Task 3

Write a method that given an initial point $x_0 \in \mathbb{R}^2$ applies Newton's algorithm (from the previous task). The resulting value will be added to an attribute of the class called `zeroes` (of type list) which contains all the zeroes that you've found

so far. You must allow for a tolerance when you compare the newly found value with the already found zeroes. This function should return the *number* of the zero in the list, or a special value if the algorithm has not converged.

Each time you run this method you may possibly increase the list stored in the attribute `zeroes`.

Task 4

Write a method `plot` that checks the dependence of Newton's method on several initial vectors x_0. This method should plot what is described in the following steps:

- Use the `meshgrid` command to set up a grid of N^2 points in the set $G = [a, b] \times [c, d]$ (the parameters N, a, b, c and d are parameters of the methods). You obtain two matrices X and Y where a specific grid point is defined as $p_{ij} = (X_{ij}, Y_{ij})^T$.
- Build a matrix A whose entries contain the index of the zero to which Newton's algorithm converged from the point p_{ij}.
- Visualize the resulting matrix, A, to observe the resulting pattern. Use for this the command `pcolor`.

Try this method when the object was initialised with the function

$$F(x) = \begin{pmatrix} x_1^3 - 3x_1x_2^2 - 1 \\ 3x_1^2x_2 - x_2^3 \end{pmatrix}$$

Why is this figure called a fractal? (Make sure that you used a sufficiently high resolution N.)

Task 5

Create a new method that implements the *simplified Newton's method*: the Jacobian is only computed once, on the first step, and is then used for all the remaining steps.

Modify the previous method `plot` by adding a Boolean parameter to allow to choose between the standard and simplified Newton method.

Task 6

Set up the derivative numerically by using finite differences with an increment h. Is the size of the increment significantly changing the plot?

Modify the code so that the derivative argument of the `__init__` method is now optional. If it is given then the object should use it. If it is not given, the object should use an approximation with finite differences.

Task 7

Add a method which allows to make a plot in which the number of iterations is depicted in dependence of the starting values.

Task 8

Try your code also with the functions

$$F(x) = \begin{pmatrix} x_1^3 - 3x_1x_2^2 - 2x_1 - 2 \\ 3x_1^2x_2 - x_2^3 - 2x_2 \end{pmatrix}$$

and

$$F(x) = \begin{pmatrix} x_1^8 - 28x_1^6x_2^2 + 70x_1^4x_2^4 + 15x_1^4 - 28x_1^2x_2^6 - 90x_1^2x_2^2 + x_2^8 + 15x_2^4 - 16 \\ 8x_1^7x_2 - 56x_1^5x_2^3 + 56x_1^3x_2^5 + 60x_1^3x_2 - 8x_1x_2^7 - 60x_1x_2^3 \end{pmatrix}$$