MAA251 – Numerical Optimization – Session 2

École Polytechnique

Practical Session #2

Instructions: The codes should be written in Python. The use of notebooks is encouraged and you may start from the given examples.

Exercise 1: Curve fitting methods

1. Play with the code `Illustration_Newton.ipynb` available on Moodle and understand all its components. In particular:

a) Observe how the algorithm behaves when you change the initialization.

b) Add new functions of your choice in order to test the necessity of all conditions involved in the quadratic convergence result.

c) Starting from this code, **implement the False Position or Secant method**. Verify the order of convergence and the dependence on the initialization.

2. **(Challenge)** We saw that basic line-search methods like the bisection method are very robust: they will always approximate the minimum of a unimodal function, but their rate of convergence is linear. Newton's method, on the other hand converges quadratically provided we have access to second derivatives, the minimizer is non-degenerate and we start close enough to the optimum.

Modify the given code implementing Newton's method so that it will converge regardless of the starting point, following the indications below:

- at each iteration consider two candidates for the new position: the one given by the bisection method (the midpoint of the current search interval) and the one given by Newton's method.
- decide what is the new interval bracketing the minimum and pass to the next iteration

For a valid solution you should check the following aspects:

a) Your algorithm should converge even in cases where Newton's method alone does not work.

b) Observe the convergence rate: you should notice that in the beginning the convergence is linear due to the eventual use of bisection steps, while towards the end the rate of convergence becomes the same as for the Newton method.

Exercise 2: Gradient descent with line-search in 1D

1. Write a code implementing the gradient descent with **fixed step** in 1D. Apply it for various test functions and observe the convergence rate. Observe also the behavior of the algorithm with respect to the initial condition and the size of the step.

Indication: If you are not sure how to start, just take the Goldstein-Price code and replace the line-search part by your choice of the fixed descent step t.

2. Implement the gradient descent with line-search based on **Armijo's rule**. You may start from the Notebook related to the Goldstein-Price line-search given on Moodle.

3. **(Challenge)** Implement the **Wolfe** line-search starting from the code given for the Goldstein-Price line-search.

4. Practical questions:

- Test the behavior of the algorithm for various functions and for various choices of the parameters $m_1, m_2 \in (0, 1)$.
- Check that for the function $f(x) = x^2$ choosing $m_1 > 0.5$ greatly increases the number of iterations because the optimal step for a quadratic function cannot be chosen.

Exercise 3: Find the closest point to a curve

Suppose $\gamma : [0, 2\pi] \to \mathbb{R}^2$ is a closed curve in the plane and $A = (a_1, a_2)$ is a given point (you may denote $\gamma(\theta) = (x(\theta), y(\theta))$). The objective of this exercise is to write an algorithm which allows you to find the minimal distance AM_θ, where M_θ is the point corresponding to $\gamma(\theta)$.

1. (Optimality condition) Suppose that A is not a point on γ. Prove that if γ is of class C^1 and M_0 is the point which realizes the minimal distance AM_θ then AM_0 is a normal vector to the curve γ.

2. Note the minimization of AM_θ or AM_θ^2 gives the same minimizer. If $\gamma(\theta) = (x(\theta), y(\theta))$ and A has coordinates (a, b) give a formula for AM_θ^2 and for the derivative $\frac{d}{d\theta}(AM_\theta^2)$.

3. Implement a numerical algorithm which can search for the point realizing the minimal distance using one of the methods in the previous two exercises. Verify numerically that the minimizer verifies the optimality conditions.

Note that you can use an algorithm implemented in previous exercises and just change the objective function.

You may try the following cases:

- the ellipse given by the parametrization $\gamma(\theta) = (2\cos\theta, \sin\theta)$ for various points A in the plane.
- a curve given in radial coordinates by the following parametrization

$$\gamma(\theta) = ((1 + 0.3\cos(3\theta))\cos\theta, (1 + 0.3\cos(3\theta))\sin\theta).$$

  and various points A in the plane.

Answers to theoretical questions:

1. The minimal distance is minimized at the same place where its square is minimized. The function to be minimized is, thus

$$\theta \mapsto AM_\theta^2 = (x(\theta) - a)^2 + (y(\theta) - a)^2.$$

At the minimum the derivative of this function is zero, which means that

$$2(x(\theta) - a_1)x'(\theta) + 2(y(\theta) - a_2)y'(\theta) = 0.$$

This can also be interpreted as the fact that the following scalar product is zero:

$$\overrightarrow{AM_\theta} \cdot (x'(\theta), y'(\theta)) = 0.$$

Since $(x'(\theta), y'(\theta))$ is a tangent vector to γ at θ it follows that AM_θ is a normal vector to the curve γ. 2. We already saw that the derivative of AM_θ^2 is

$$2(x(\theta) - a_1)x'(\theta) + 2(y(\theta) - a_2)y'(\theta).$$

3. Use one of the gradient descent codes implemented in the previous exercise.