OPTIMIZATION & NONLIN EQS (MA 784) — HW 4

Linear optimization

It is almost never a good idea to implement your own linear optimization algorithm, given the vast array of available solvers and how capable they are. In that spirit, this HW will only require that you use an existing LP solver (in the second problem) to solve an optimization problem (that cannot be written as a linear program, because that would be too simple, wouldn't it?) As always, hints are available (also for the first, theoretical, problem).

1. (Reverse-engineering a linear program from its solution.) An absent-minded professor has solved a linear program in standard form, and obtained the optimal solution $\mathbf{x}^*$. However, he completely forgot what the cost vector $\mathbf{c}$ was! Fortunately, he still remembers $\mathbf{A}$ and $\mathbf{b}$ in the constraints $\mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}$. Help the professor find a cost vector $\mathbf{c}$ for which the vector $\mathbf{x}^*$ is indeed optimal. (This problem has more serious applications than shown in this exercise. For example, extensions of this problem can be used to quantify decision makers' unknown preferences, aka. *utility function*, based on their perception of what appears to be an optimal solution to the problem that is being modeled.)

2. Let $\mathbf{a}_1, \ldots, \mathbf{a}_m, \mathbf{c}_1, \ldots, \mathbf{c}_m \in \mathbb{R}^n$ and $b_1, \ldots, b_m, d_1, \ldots, d_m \in \mathbb{R}$ be given, and consider the optimization problem

$$\min_{\mathbf{x} \in [0,1]^n} \max_{i=1,\ldots,m} \frac{\mathbf{a}_i^{\mathrm{T}}\mathbf{x} + b_i}{\mathbf{c}_i^{\mathrm{T}}\mathbf{x} + d_i}. \tag{1}$$

You may assume that each denominator is strictly positive on the domain $[0,1]^n$.

(a) Show using a simple example that this problem need not be a convex optimization problem even if $m = 1$. (Give some problem data that results in a non-convex problem, and demonstrate that it is indeed non-convex.)

(b) Denoting the objective function by f, show that the closed lower level set

$$L_z := \{\mathbf{x} \in [0,1]^n \mid f(\mathbf{x}) \leq z\}$$

is a convex polyhedron for every $z \in \mathbb{R}$.

(c) Prove that (in spite of the problem not being convex), every local minimizer of (1) is a global minimizer.

(d) Using the previous observations, propose an algorithm that computes a global optimal solution to (1) (within a given $\varepsilon > 0$ tolerance) using a linear optimizer as a subroutine. You may solve as many linear programs as you want.

(e) Find an upper bound (as a function of the problem data and ε) on the number of linear programs you need to solve.

(f) Implement your algorithm in Matlab using the built-in `linprog` function. Alternatively, you may use another programming language and its LP solver. Either way, I strongly recommend using a **modeling language** (such as YALMIP[1] for Matlab or JuMP[2] for Julia) to specify the LP, as manual conversion can be tedious and error-prone. These modeling tools have a very shallow learning curve and can save you a lot of time.

Your function `hwfour` should have five arguments, the input data (coefficients) stored in the matrices and vectors $\mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{b} \in \mathbb{R}^m, \mathbf{C} \in \mathbb{R}^{m \times n}, \mathbf{d} \in \mathbb{R}^m$ and the tolerance ε. The output should be a vector $\mathbf{x}$ for which the objective function is no more than ε away from the minimum.

Report your results (optimal solution, function value, time, number of iterations, etc.) when using your code with the following instance (try to get the optimal value within $\varepsilon = 10^{-10}$ if running times permit–it need not take more than a fraction of a second):

$$\min_{\mathbf{x} \in [0,1]^2} \max \left\{ \frac{2 - 3x_1}{x_1 + 2}, \frac{3x_1 + 2}{x_1 + 36x_2 + 2}, \frac{2}{x_1 + 9x_2 + 2}, x_1 + x_2 - 1 \right\}.$$

As before, please show your code of the function `hwfour`, the code you use to call your function (setting up the above instance), and the relevant parts of the output in your PDF.

[1] `https://yalmip.github.io/tutorial/basics/`

[2] `https://jump.dev/JuMP.jl/stable/tutorials/getting_started/getting_started_with_JuMP/#Getting-started-with-JuMP`