Numerical Optimization with Python

Programming Assignment 02

In this exercise we will implement an interior point method solver for small constrained optimization problems.

Instructions:

1. To your `src` directory, add a new module, `constrained_min.py`
2. Implement the function (or as a method of a class):
   `interior_pt(func, ineq_constraints, eq_constraints_mat, eq_constraints_rhs, x0)` which minimizes the function `func` subject to the list of inequality constraints specified by the Python list of functions `ineq_constraints`, and to the affine equality constraints $Ax = b$ that are specified by the matrix `eq_constraints_mat`, and the right hand side vector `eq_constraints_rhs`. The outer iterations start at `x0`.
3. Use the log-barrier method studied in class, with the initial parameter $t = 1$ and increase it by a factor of $\mu = 10$ each outer iteration.
4. To your `tests` directory, add a module `test_constrained_min.py` and define, using the `unittest` framework as in HW01, the function `test_qp()`,`test_lp()` that will demonstrate solutions for a quadratic programming example and a linear programming example.

5. To your `examples.py` file, add the functions and the definition of the matrix and vector, to enable `test_qp()` use them for solving the following problem:

$$\min x^2 + y^2 + (z+1)^2$$

Subject to: $x + y + z = 1$

$$x \geq 0$$

$$y \geq 0$$

$$z \geq 0$$

Note: the problem finds the closest probability vector to the point $(0,0,-1)$. Choose an initial interior point $(0.1, 0.2, 0.7)$, and do not implement a phase I method for finding a strictly feasible point in this exercise.

6. To your `examples.py` file, add the functions to enable `test_lp()` use them for solving the following problem:

$$\max[x+y]$$

$$\text{Subject to: } y \geq -x+1$$

$$y \leq 1$$

$$x \leq 2$$

$$y \geq 0$$

   Note: the problem finds the upper right vertex of a planar polygon. You only have inequality constraints here, hence at each outer iteration you will solve an unconstrained problem. Choose an initial interior point $(0.5, 0.75)$, and do not implement a phase I method for finding a strictly feasible point in this exercise.

7. For both examples above, plot
   a. The final candidate
   b. Objective and constraint values at the final candidate
   c. Plot the feasible region and the path taken by the algorithm.
   d. The graph of objective value vs. outer iteration number.

   Note: in both cases the feasible region is a polygon, but in the first example it is a triangle to be plotted in 3D space, and the path is in 3D space, there are several options to do that, here:

   https://matplotlib.org/2.0.2/mpl_toolkits/mplot3d/tutorial.html

   Submit the required plots and final iterates in a PDF file to the course site, and your code should be sent over email as a link to a GitHub repo (do not send notebooks or Python files as email attachments).

Good luck!