Computer Laboratory Exercises in Optimization 2016
Lab 2. Penalty and Barrier Function Method.

The lab is to be done in groups (2-3 persons in a group). You can bring your own laptop with an installed MATLAB package of the version 2014 or later, but make sure in advance that the lab m-file is working correctly.

Preparation Exercises

The preparation is to be done **before** the actual lab session. You may also download and run the m-file to learn how it works as a part of the preparation and test your penalty/barrier function implementations. However, the lab must be done in the class.

- Give a brief explanation to the penalty and the barrier function methods (pages 313–319 and 323–325) and to the linear programming (pages 157–163).
- Calculate the dual problem to the problem (3) below.
- Implement in MATLAB the penalty function of the type (for $S = \{g_k(x) \leq 0\}$)

$$\alpha(x) = \sum_{k=1}^{m} \max(0, g_k(x))$$

  for the problems (1), (2), (3) and the dual to (3). The function should take a *single* vector variable x and return the functional value at that point. Since you need penalty functions for several problems, but the program works only for the standard name `penalty.m`, you may write all cases in one file and comment/uncomment what is necessary during the lab (or write several m-files and rename the one used to `penalty.m`).
- Implement in MATLAB the barrier function of the type (again for $S = \{g_k(x) \leq 0\}$)

$$\beta(x) = -\sum_{k=1}^{m} \frac{1}{g_k(x)}.$$

  for the same problems (1), (2), (3) and the dual to (3). What is said for the penalty function above is valid here too. The standard name is `barrier.m`.
- Make the problem data m-file for the dual problem to (3). It should have the name `problem_data.m` and contain data for solving an optimization problem in the form

$$\min f(x) \qquad \text{subject to } Ax \geq b. \tag{$*$}$$

  Use the following file format

```
function [A,b,f] = problem_data

    A = <put your matrix A here>;
    b = <put your column vector b here>;

    f = @(x) <expression for f(x) in terms of x(1),x(2),...,x(n)>;
```

  Note that the constraints $Ax \geq b$ should contain *all* inequalities, even $x_k \geq 0$.

At the computer

♡ Download the file `lab2.m` from the course homepage

`http://www.maths.lth.se/matematiklth/personal/ghulchak/optimization/2016/`.

♡ Type `lab2` in the Matlab window to start the program. A new window appears where you select the first method by clicking the *Penalty method*. The first problem to solve is

$$\min(x_1 + 2x_2) \quad \text{subject to} \begin{cases} x_1 + x_2 & \geq 2, \\ x_2 - x_1 & \geq 0, \\ x_1 & \geq 0, \\ x_1 + 2x_2 & \leq 6. \end{cases} \tag{1}$$

To begin with, you can simply click the mouse left button once to pick a starting point and then continue clicking to see the sequence of approximations for some default values μ_k. Try different starting points. The optimization stops when the header lights yellow.

- Do you think the method finds the minimum? Why?
- Are you satisfied with the approximation? Does it have any drawback?

Now type an appropriate μ-value in the correponding μ-field for *Penalty method* and try to do minimization with a manual consequential increase of μ values.

- Provide a sequence μ_k that you think works well.
- Can you get fewer iterations than with the default μ?
- Try to start with a very large μ from the beginning like 10^6 and larger. Try different starting points. Is the approximation good? Does the result depend on a starting point, for example, if you start really close to the minimum?

♡ Now select the *Barrier method* and choose again a starting point and then simply click further on.

- Do you think the method finds the minimum? Is it the same minimum as for penalty method? If not, explain why not.
- Is there any difference in how you pick a starting point in the penalty method and now?
- How would you compare the barrier method to the penalty method for this problem? Is any of the approximations better in any sense?

Similar to the penalty method, you can try now your own ϵ values and sequentialy decrease them by hand.

- Can you find a sequence of ϵ_k that gives better convergence then the default one?
- Start with a very small ϵ like 10^{-6} and smaller. Try different starting points. Is the result satisfactory? Does it depend on a starting point?

♡ Choose the Problem 2 in the pop-up meny and repeat the same questions for the problem

$$\min(x_1 + 2x_2) \quad \text{subject to} \begin{cases} -x_1 + 3x_2 & \leq & 12, \\ x_1 - 3x_2 & \leq & 2, \\ x_1 + 2x_2 & \geq & 2, \\ 3x_1 + 5x_2 & \leq & 34, \\ x_1 & \geq & 0, \\ x_2 & \geq & 0. \end{cases} \tag{2}$$

♡ Choose the coresponding pop-up meny options and try to solve Problems 1 and 2 with your penalty/barrier functions. Note that the penalty and barrier functions must be m-files in the same directory as the file `lab2.m` and to be named `penalty.m` and `barrier.m`. They must take in a single parameter x which is the function (vector) argument and return the functional value at the point. Try different starting points and your own values for μ and ϵ.

- Test your penalty for μ-values between 1 and 5. Is the result satisfactory? What happens if you increase μ? Compare with the behaviour of the default penalty function.
- Is the result different for different starting points? Have you already seen something similar today for those two problems with the default penalty? Suggest an improvement of the penalty function that you implemented.
- Compare the default barrier function and yours — which one gives a faster convergence? Can you think of a possible reason for that?

♡ Now switch to the 3D problems and choose the *Penalty method* first. Repeat the same questions as above for the following problem

$$\min(-2x_1 - 4x_2 - 3x_3) \quad \text{subject to} \begin{cases} x_1 + 3x_2 + 2x_3 & \leq & 30, \\ x_1 + x_2 + x_3 & \leq & 24, \\ 3x_1 + 5x_2 + 3x_3 & \leq & 60, \\ x_1, x_2, x_3, & \geq & 0. \end{cases} \tag{3}$$

Note that now you have to type in the starting point coordinates manually in the red frame box instead of clicking with the mouse. Do the same questions for the *Barrier method* too as well as test your penalty/barrier functions on this problem.

♡ Load the dual problem to (3) by choosing *Your data* in the pop-up meny and your m-file `problem_data.m`. The file must be in the same directory with `lab2.m`. Pay attention to the data format in the file (see the Preparation section) in order for the program to work correctly. Solve the problem first by using the default penalty/barrier functions and then try your own functions.

- Compare the optimal values of the dual and the primal problems. Do you see any similarities?

♡ In case you have time left you may try solving your own 2D or 3D problem in the form (*), for example, with a nonlinear objective function. Note that it is your responsibility to ensure that the minimum exists and to provide a proper penalty/barrier function implementation for the problem. As a suggestion you may take the exercises 9.1 and 9.4 in the book.