Computer Laboratory 1
LINEAR AND COMBINATORIAL OPTIMIZATION

LUND INSTITUTE OF TECHNOLOGY
DEPARTMENT OF MATHEMATICS

February 1, 2019

Preparation for the labs

Read through the whole manual and do the preparation exercises before the lab. Download the Matlab files needed (lab1.zip) for the exercise from the course webpage.

Some useful Matlab functions

If you have never used Matlab before, prepare yourself by following some of the tutorials on the page
`http://se.mathworks.com/help/matlab/getting-started-with-matlab.html`

To get help on a specific command, write `help` followed by the command that you want to read about.

The following commands, may be particularly useful: `find`, `max`, `sum`. Try the help command on these to see what they do.

Another way to find information of a specific command is to right click on the command in the matlab script or in the command window and select "Help on Selection" in the appearing menu. After you have tested the help files, try the following to get started:

1. In the command window, write
   ```
   b = [3 5 -2 0 9 -3 -4 8]
   A = [3 5 4 2 0; 7 6 3 5 1; 8 6 9 2 0]
   ```
   a) Using the information that you found in the help files, find the indices of the positive entries of `b` and `A`.

   b) Find the maximum value and the corresponding index of `b`.

   c) What does
   `[maxv,index]=max(A)` do? (Try it!)

   d) What is the difference between `sum(A)` and `sum(sum(A))`?

2. Write `B = [1 1 1 1 1;0 0 0 0 0;2 2 2 2 2]` in the command window.

   Note the difference between elementwise operations and matrix operations, and try for example (for multiplication) `A.*B`, `A*B'`, `A'*B` and note that `A*B` gives an error message.

When doing the exercises of the labs, continue to take help from the built in help files, and try the commands on simple examples in the command line, so that you know what they do before using them in your script.

Read this before you come to the lab session

This lab is about linear programming. The first task is to run an available Matlab program that demonstrates the simplex method. When running the program, you will manually choose the pivot element on every iteration. The program will call your own function `checkbasic1.m` that you made in Handin Exercise 1. Therefore you need to have access to this file at the lab. The second task is to modify the program so that the pivot elements are chosen automatically. Finally, you should test your program on some examples.

You should also make a version that produces the result as fast as possible, and investigate how the execution time increases with the number of constraints and variables.

All this will be done at the lab session. Prepare yourself by reading the lecture notes and the book about the simplex method.

Cycling can be prevented with Bland's rule:

1. If there is more than one variable which can enter the basis (that is, columns with negative reduced costs), then the one with the lower index should be chosen.

2. If there is more than one variable to leave the basis, then the one with the lowest possible index should be chosen.

The problem to solve in this session is

$$\text{maximize } z = \mathbf{c}^{\mathbf{T}}\mathbf{x}$$
$$\text{subject to } \begin{cases} \mathbf{A}\mathbf{x} = \mathbf{b} \\ \mathbf{x} \geq \mathbf{0}. \end{cases}$$

It is assumed that a feasible choice of basic variables is given. This is easy if the problem has been converted from its standard form with $\mathbf{b} \geq \mathbf{0}$. Make sure that you understand how, by looking in the book or lecture notes if necessary.

The simplex method in Matlab

The function `simpmovie` requires that the user assigns the correct variables to enter and leave on every iteration. Observe that the tableau is constructed with the function `checkbasic1` from Handin Exercise 1. Open the file `simpmovie.m` in Matlab and see what it does. Use the `help` command to find information about unknown commands in the file.

To run the program, you need to copy your function file `checkbasic1.m` into the directory `lab1`, and go to that directory. Then generate a random problem by typing `init` in the command window (look in the code to see what it does), and solve this problem by writing
`[y,basicvars,steps] = simpmovie(A,b,c,basicvars)` and then choosing the entering and leaving variables until an optimum is reached.

With this preparation, you will be ready for the lab session.

At the lab session

1. Go to the sub directory `lab1`.

2. Copy the script `simpmovie.m` to a new file `simp.m` using the command
   `!cp simpmovie.m simp.m` (Unix) or `!copy simpmovie.m simp.m` (Windows).

3. Modify the program `simp.m` so that the pivot elements are chosen automatically. The Matlab functions `min` and `find` are useful. Unbounded problems should be detected. The problem with cycling can be solved using Bland's rule, but this is not compulsory. The number of steps of the simplex algorithm is returned from the function.

4. Test your program on Example 1 in Section 2.1. of the course book by Kolman and Beck (tableau 2.1, 2.3 and 2.4). Test random matrices (created by `init` of sizes from 10×10 to 100×100 and make a table of the elapsed time. How does the number of steps grow with m and n? Exponentially or polynomially?

5. To study the phenomenon cycling, write `chvatal` to create matrices for an LP problem with this property. Choose the pivot elements according to "the most negative reduced cost should enter the basis" and "if more than one variable can leave the basis, then choose the one with the lowest index". (The cycling indices are $(q, p) = (1, 5), (2, 6), (3, 1), (4, 2), (5, 3), (6, 4)$).)

6. For every rule on how to choose the pivot element, it is possible to construct examples that makes the simplex method very slow. Take a look at `simpb` (write `edit simpb` in the Matlab command window), which is a simplex method using a 'Bland'- like rule for selecting the incoming and outgoing variables. The script `badexample` creates a difficult example for this simplex algorithm. The variable d sets the size of the matrix A of the problem. How does the number of steps grow with d? Is the simplex method a polynomial time algorithm for the linear programming problem?