Practical 4: Submit solutions in Matlab Grader before Fri 20th Sept at 5pm.

Bring your lecture notes with you for reference.

The aim of this practical is to see some of the difficulties of Simplex computationally and numerically.

Simplex is an *exponential* algorithm, in the sense that its worst case performance takes an exponential number of steps. The Klee-Minty example below demonstrates this.

Note that, MATLAB's LP solver `linprog` doesn't use Simplex, and even when it does notionally use Simplex, it does a few smart things to the data first. Hence, we will use **your** Simplex code to perform many of the tests in this practical. You should have completed this code by now, but if not, or if your code worked badly, there is a *protected* set of Simplex code available for you to use on MyUni.

You will need to download the files

```
simplex.p
at_end.p
choose.p
construct_i.p
pivot.p
output_simplex.p
```

and place them in your working directory. Then you should be able to use these to perform Simplex. Test them as follows:

Tasks

1. Input the following problem into **your** Simplex code, or the example code described above (don't use MATLAB's `linprog`).

$$\begin{array}{rcccccccl} \max z & = & x_1 & + & x_2 & + & x_3 & & \\ \text{such that} & & x_1 & + & x_2 & & & \leq & 1 \\ & & & - & x_2 & + & x_3 & \leq & 0 \\ & & & & & & x_i & \geq & 0 \quad \forall i \end{array}$$

   (a) Carefully examine the trajectory (the series of vertices), and the objective function.
   Hint: You may have to modify your Simplex code to output the vertex point (each basic solution) after each pivot.

   (b) Draw a 3D picture of the problem. You can do this with MATLAB or on paper.
   Hint: MATLAB can draw in 3D. Try finding out about `plot3` using MATLAB's help. Once you draw the points on the plot, you can use the GUI's tools to rotate it and look at it from different perspectives.

   (c) What do you notice about the trajectory?

2. The Klee-Minty LP[1] is given by

$$\begin{array}{rrcl} \max & z = \sum_{i=1}^{n} 2^{n-i} x_i & & \\ \text{such that} & 2\sum_{j=1}^{i-1} 2^{i-j} x_j + x_i & \leq & 5^i, \quad \text{for } i = 1, \ldots n \\ \text{and} & \mathbf{x} & \geq & 0 \end{array}$$

(a) Write a MATLAB function that takes as input the size of the problem n, and returns as output the appropriate matrices and vectors A, $\mathbf{b}$ and $\mathbf{c}$ for this LP.

Hints:

- Start by constructing the matrices for $n = 3$ or 4 by hand to see what they look like.
- Now look at the coefficients, as if it were a typical LP on paper, and consider how to write these simple cases in MATLAB .
- Then generalise your code to work for any n.
- You can construct vectors like $[5, 25, 125, \ldots]$ by taking

```
b = 5.^[1:n]
```

- Your A matrix can be constructed using a Toeplitz matrix. MATLAB has a function to generate these matrices.

(b) **Test your function is correct in Matlab Grader.**

Hints:

- Make sure your vectors are output as **column** vectors.
- Your c vector should have minus signs for the test (and in the questions below you should be consistent to make sure that you are doing maximisation).

Please remember to submit your solution to Matlab Grader before the due date.

[1] This LP was constructed in "How good is the Simplex Algorithm?", Klee and Minty, 1972 J. Inequalities III, pp.159-175.

3. Now solve the Klee-Minty LP for $n = 3, 4, \ldots, 14$ using

   (a) **your** Simplex code, or my example (protected) code available on MyUni; and compare to

   (b) MATLAB's `linprog`.

   Hints:

   - You can pass MATLAB's `linprog` some `options` created[2] using

     ```
     options = optimoptions(@linprog, 'Display', 'iter');
     ```

     and this will ensure that MATLAB shows you each step of its process.
   - Make sure to include lower bounds on variables.
   - You can get more output from `linprog` by using

     ```
     [x,fval,exitflag,output] = linprog(f,A,b,Aeq,beq,lb,ub,x0,options)
     ```

     The variable `output` will contain a *structure*, and one of the elements of this is `output.iterations`, which contains the number of steps it needed (see `help linprog` for more details).

     You'll also need to make sure your Simplex code outputs information about each step.

   Plot and compare the results, in particular, how many steps (pivots) are required as a function of n, and comment on the computational complexity of the Simplex algorithm.

   Note the difference between your algorithm (in Simplex) and MATLAB, which doesn't use Simplex (any more).

[2] This only works for recent versions of MATLAB – older version use a different syntax which you can find using the help function.