THE UNIVERSITY of ADELAIDE

Optimisation & Operations Research

Haide College, Spring Semester

Practical 2 (1%) See website for practical and due dates

Work through the below instructions in MATLAB and MATLAB Grader as indicated. Submit to MATLAB Grader by the due date.

Visualising Simplex, and Klee-Minty

Matt Roughan, Mike Chen

The aim of this practical is to highlight some computational and numerical difficulties with the Simplex algorithm.

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, `linprog` in MATLAB doesn't use Simplex, and even when it does notionally use Simplex, it does a few smart things to the data first. Hence, in this practical a *protected* set of Simplex code for problems in **standard inequality form** is available for you to use on Cloudcampus.

You will need to download the files (available as a `.zip` file):

```
simplex_sif.p
at_end.p
choose.p
construct.p
pivot.p
output_simplex.p
```

and place them in your working directory. Then you should be able to use these to perform Simplex as usual. You could also use the Simplex code you wrote in Assignment 2, however note the provided version gives some extra output that we will use here. Enter `help simplex_sif` to find out about these.

---

1. **Visualising Simplex trajectories**

   Input the following problem and solve using the provided Simplex code (don't use MATLAB's `linprog`).

$$\max \quad z = x_1 + x_2 + x_3$$

   such that

$$\begin{aligned} x_1 + x_2 &\leq 1 \\ -x_2 + x_3 &\leq 0 \\ x_i &\geq 0 \quad \forall i \end{aligned}$$

a) Examine the trajectory (the series of vertices), and the objective function to make sure you understand these outputs.

   Hint: The provided code outputs these as `X` and `Z`, respectively.

b) Draw a 3D picture of the problem with MATLAB .

Hint: MATLAB can draw in 3D. Try finding out about `plot3` in MATLAB's documentation (it is very similar to the 2D `plot` command). 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. **Klee-Minty setup**

The Klee-Minty LP[1] is given by

$$\max \quad z = \sum_{i=1}^{n} 2^{n-i} x_i,$$

such that

$$2\sum_{j=1}^{i-1} 2^{i-j} x_j + x_i \leq 5^i, \quad \text{for } i = 1, \ldots n,$$

$$\text{and} \quad \mathbf{x} \geq 0.$$

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]
```

- The A matrix can be constructed using a Toeplitz matrix. MATLAB has a function to generate these matrices for a given first row and first column.

b) Copy your function to MATLAB Grader to make sure it passes all the tests. **You must submit your fucntion by the due date to get the 1% course mark.**

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).

---

3. **Solve Klee-Minty LP with Simplex and `linprog`**

Now solve the Klee-Minty LP for $n = 3, 4, \ldots, 18$ using

a) the code available on Cloudcampus; and compare to

b) MATLAB's `linprog`, using three different built-in algrorithms:

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

- `dual-simplex`,
- `interior-point`,
- `interior-point-legacy`

Hints:

- You can get MATLAB's `linprog` to show you each step of it's process by settting an option with

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

  and then passing this as an extra input to `linprog`. Similarly, you can force MATLAB to use a particular algorithm with

  ```
  options = optimoptions(@linprog, 'Algorithm', 'dual-simplex');
  ```

  where can choose `interior-point` or `interior-point-legacy`.
- 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,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).

Plot and compare the results, in particular, how many steps (pivots) are required as a function of n? How about the computational time as a function of n?

Note the difference between the provided Simplex algorithm and MATLAB's which uses a more sophisticatd algorithms.
