Assignment #6

Due Friday 1 November 2024, at the start of class (*revised*)

From the textbook[1] please read sections 5.2–5.4 on the simplex method, but note that you can skip subsections 5.2.3, 5.2.4, and 5.4.2. (*That is, skip the stuff on tableaus and the "big-M" method.*) You can read section 5.3 lightly, and what you actually need from section 5.4 is on pages 149–150. To complete our coverage of linear programming, please read sections 6.1–6.2 on duality and sections 9.1–9.3 on computational complexity.

Regarding linear programming generally, consider spending some time with the linear programming Wikipedia page. Note the list of software. See also the revised simplex method page.

We are transitioning to nonlinear optimization. Please read sections 2.5 on rates of convergence and 2.7 on Newton's method for nonlinear systems.

DO THE FOLLOWING EXERCISES from section 2.7, pages 74–75:

- Exercise 7.1
- Exercise 7.10

DO THE FOLLOWING EXERCISES from section 6.2, pages 185–189:

- Exercise 2.4 *Hint. It is merely a corollary of weak duality (Theorem 6.4).*
- Exercise 2.11 *Hint. Use strong duality (Theorem 6.9) and then Corollary 6.6.*

Problem P11. I have posted `kleeminty.m`, `ezsimplex.m`, and `sfsimplex.m` at

`bueler.github.io/opt/codes.html`

Please download these codes; they are in `simplex.zip` at the same page.[2] The code `kleeminty.m`[3] sets up a Klee-Minty cube example in "EZ" form for any size n. Note that `kleeminty.m` calls `ezsimplex.m`, which calls `sfsimplex.m`, so they all need to be in the current directory to work.

Now, on your machine, for how big an n can you run `kleeminty(n)` in under 10 seconds? For this maximum n value, how many iterations does it take to find the optimal solution? What are the number of constraints and the number of variables

[1] Griva, Nash, and Sofer, *Linear and Nonlinear Optimization*, 2nd ed., SIAM Press 2009.

[2] For Python and Julia people I recommend just slumming it with the rest of us for **P11–P13**. Do them in Octave online, or Matlab online, etc.? Rewriting all my codes would be tedious.

[3] The specifics of the **P11** `kleeminty.m` example are from the Klee-Minty cube Wikipedia page. The version in section 9.3 of the textbook is essentially the same, but its coefficients grow unnecessarily fast.

if you were to put this maximum n problem in standard form? Note the exponential relation between the number of variables and the number of iterations.

Problem P12. Suppose we have an LP problem in standard form, with n variables and m equality constraints as usual, but suppose that we do *not* know a basic feasible solution. As explained on the first two pages of section 5.4, one can set up a *phase-1 problem* with one added artificial variable a_i for each equality constraint, $i = 1, \dots, m$, and then change the objective to

$$\text{minimize} \qquad z' = a_1 + a_2 + \dots + a_m.$$

This creates a new LP problem in standard form, now with $n + m$ variables, but with m equality constraints as before. However, now a basic feasible solution (BFS) is clear! (*Why? Note you can set the original variables to zero and use the constraints to solve for the artificial variables. Confirm that this is a BFS to the new problem.*)

Write a running code or complete pseudocode[4] which implements this phase-1 strategy and generates an initial BFS. Your (pseudo-)code should have signature

```
function x = phaseone(A,b)
```

Your code or pseudocode should call a standard-form simplex method once it sets-up the new LP problem; you are not expected to implement the simplex method here.[5] Your code or pseudocode should report failure when the original problem is not feasible; how is that detected?

Problem P13. Return to problem `salmon` on the 5 example optimization problems handout. Put it in standard form, figure out a BFS, and solve it using the simplex method.[6] Of course, you can refer to Assignment #1 for the correct answer.

Problem P14. *This problem replaces, simplifies, and clarifies Exercise 5.1 in section 2.5.*

For each of the following 3 sequences, determine the limit x_*. Then determine the rate of convergence; is it linear, superlinear, or quadratic? Specifically, recalling $e_k = x_k - x_*$, determine $r \geq 1$ and $0 < C < +\infty$ in the limit

$$\lim_{k\to\infty} \frac{\|e_{k+1}\|}{\|e_k\|^r} = C.$$

(i) The sequence with general term $x_k = 2^{-k}$, for $k = 1, 2, \dots$

(ii) The sequence $1.05, 1.0005, 1.000005, \dots$ with general term $x_k = 1 + 5 \times 10^{-2k}$, for $k = 1, 2, 3, \dots$

(iii) The sequence with general term $x_k = 2^{-(2^k)}$.

[4]Either code or pseudocode is fine but running a code is more fun!

[5]However, it is easy to use my implementation. Given how my code `sfsimplex.m` works, the line

```
>> [x, z] = sfsimplex(c,A,b,phaseone(A,b))
```

solves the standard form problem, from the data c, A, b, by the two-phase method, in a case where you don't know an initial BFS. Note `sfsimplex.m` gets called twice in this one-line command!

[6]You may do **P13** all by hand, or use my `sfsimplex.m`, or use even use the 2-phase method if you implemented that in **P12**. If you don't use 2-phase you'll have to find a BFS by hand.