Assignment #6 (*revised again*)

Due Wednesday, 26 October 2022, at the start of class

There is a lot to read from the textbook[1] for this assignment!

Regarding linear programming, 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. All you need from section 5.4 are pages 149–150.*) Then read sections 6.1–6.2 on duality and sections 9.1–9.3 on computational complexity.

Regarding unconstrained optimization, read section 2.5 on rates of convergence and then section 2.7 on Newton's method for scalar equations and systems of equations. Read sections 11.1–11.2 on "gradient equal zero" and Hessian concerns. Finally read section 11.3 on Newton's method for optimization.

DO THE FOLLOWING EXERCISES from section 2.7, pages 74–76:

- Exercise 7.1
- ~~Exercise 7.5~~ *Removed. I don't like the way this problem is written. I'll address it in class.*
- Exercise 7.10

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

- Exercise 2.4
- Exercise 2.8 *This is a canonical-form minimization problem, so its dual is a canonical-form maximization problem. You can guess at optimums for both problems. Then prove you are right using duality theorems.*
- Exercise 2.11

Problem P10. 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 constraint, $i = 1, \dots, m$, and then change the objective to

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

This creates a new LP problem in standard form, with $n+m$ variables and m equality constraints, and a basic feasible solution is clear. (*Why is a b.f.s. clear? Because you can set the original variables to zero, and then use the constraints to solve for the artificial variables. Do you see why this is a b.f.s. to the new problem?*)

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

Write a running code or pseudocode—either is fine but running a code is more fun!—which implements this phase-1 strategy and generates an initial basic feasible solution. Your code or pseudocode should have signature[2]

```
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. Your code or pseudocode should report failure when the original problem is not feasible; how is that detected?

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

https://bueler.github.io/opt/codes.html

Please download these codes.[3] Note that `kleeminty.m` sets up a Klee-Minty cube example in "EZ" form for any size n. (The specifics of this example are from the revised simplex method Wikipedia page, not section 9.3 of the textbook, but the idea is the same.) 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, how big an n can you run in under 20 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 if you were to put the problem in standard form?

Problem P12. *This problem replaces, simplifies, and clarifies Exercise 5.1 in section 2.5.*

For each of the following sequences, determine the limit x_*. Then determine the rate of convergence; is it linear, superlinear, or quadratic? Specifically, 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 $k = 1, 2, \ldots$

(ii) The sequence

$$1.05, 1.0005, 1.000005, \ldots$$

with general term $x_k = 1 + 5 \times 10^{-2k}$, for $k = 1, 2, \ldots$

(iii) The sequence with general term $x_k = 2^{-(2^k)}$. Noting $x_0 = 1/2$, how is this sequence is generated recursively? That is, give a function $f(z)$ so that the formula $x_{k+1} = f(x_k)$ generates this sequence.

---

[2] 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. Note `sfsimplex.m` gets called twice in this one-line command!

[3] For Python and Julia people I recommend just getting trashy and doing *this* problem in Octave online, or Matlab online, etc. Rewriting all my codes would be tedious.