Sensitivity Analysis. So far nearly everything we’ve talked about has had to do with just a single LP (which we put in standard form for purposes of solving)

- Maximize $\vec{c} \cdot \vec{x}$ subject to $A\vec{x} \leq \vec{b}$ and $\vec{x} \geq 0$.

We’ve generally treated the parameters $A$, $\vec{b}$, and $\vec{c}$ as set in stone. But how realistic is that in practice? Well, not all that much! Because think about where these things may come from in a real-world problem:

- The values showing up in any of $A$, $\vec{b}$, $\vec{c}$ might only be a best guess - we might get new information, or we might want to make sure that our solution doesn’t go horribly wrong if we put in more pessimistic estimates, or so on.
- Even if we know the values exactly they might change over time and we want to re-solve a slightly modified problem later on.
- New constraints might be imposed, or we might want to add a new variable (say representing a new product that could be produced, if we’re thinking of a factory production example).
- We might want to know how much you can move a parameter around before it makes a “significant” change to the optimal solution.
- Extending the last idea, maybe you don’t know an exact value of an important parameter and want the solution as a function of it. (This is called parametric analysis and we won’t really into it, but it is something that’s been extensively studied if you need to do it.)

Given what we’ve done so far, we’d just treat each instance of the first three things as a separate problem and go solve it from scratch using the simplex method. That isn’t a big issue for small examples, but might be if it’s a really big problem that took a lot of effort to solve, or we have a lot of modifications to consider. So the basic question of sensitivity analysis is:

- Suppose we have an optimal solution (and an optimal dictionary) for our original problem. How do we find how the optimal solution changes when we modify the problem?

The basic idea is pretty straightforward: keep the same list of basic/nonbasic variables as our original optimal solution, and write down the new dictionary for them. If it’s still optimal, great! If not, then just continue with the simplex method with that dictionary as a starting point, which is probably better than starting from scratch. (This is yet another reason the simplex method is so useful in practice: solving one problem puts us a long way towards solving any related problem, which may not be true or easy to do for other algorithms!)

An example and some changes we can make to it. Let me set up a simple example to illustrate the sort of thing I’m talking about. (This one will be a purely mathematical example, but on Friday or Monday we’ll look at an extended applied example)

- Maximize $2x_1 + x_2 + 2x_3$, subject to:
  - $x_1 + x_3 \leq 1$
  - $x_2 + 2x_3 \leq 2$
  - $x_1 + 2x_2 + 3x_3 \leq 3$
  - $x_1, x_2, x_3 \geq 0$. 

Fall 2017 Notes by Dan Collins
In matrix form:

\[
A_{aug} = \begin{bmatrix}
1 & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 2 & 0 & 1 & 0 \\
1 & 2 & 3 & 0 & 0 & 1
\end{bmatrix}, \quad \vec{b} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \quad \vec{c}_{aug} = \begin{bmatrix} 2 \\ 1 \\ 2 \\ 0 \\ 0 \\ 0 \end{bmatrix}.
\]

If you go to solve this it’s straightforward enough: the first pivot will be \( x_1 \) entering and \( x_4 \) leaving, and then \( x_2 \) entering and \( x_6 \) leaving, so we find an optimal dictionary has

\[
\vec{x}_B = \begin{bmatrix} x_1 \\ x_2 \\ x_5 \end{bmatrix}, \quad \vec{x}_N = \begin{bmatrix} x_3 \\ x_4 \\ x_6 \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}
\]

and corresponds to an optimum value of \( z = 3 \) occurring when \( x_1 = x_2 = 1 \) and \( x_3 = 0 \) (and \( x_4 = x_6 = 0 \) and \( x_5 = 1 \)).

So first question: what happens if I change the objective function? Say the coefficient \( c_3 \) of \( x_3 \) goes up to 2.1, or down to 1.9? You could always just rewrite the original problem and start from scratch, though you’d find that putting it up to 2.1 would change the order of pivots the standard method tells you to do and you’d basically be solving an entirely new problem. Better idea: let’s just look at the dictionary with the same \( \vec{x}_B \) and \( \vec{x}_N \) for our new problem! In fact, since \( x_3 \) is a non-basic variable and we’re taking \( x_3 = 0 \), it’s a reasonable conclusion that decreasing \( c_3 \) will not change the solution (we’re just decreasing the value of an option we’ve already decided wasn’t worth doing!). But for increasing it’s harder to say - at some point \( x_3 \) would become valuable enough to be worth doing, but we’d actually need to do some computations to see when.

So, let’s do these calculations, going back to the revised simplex method: to determine optimality we needed to look at the objective function row of the dictionary which we could compute as

\[
\frac{\vec{x}_B}{z} = \frac{A_{B}^{-1} \vec{b}}{\vec{c}_{aug} A_{B}^{-1} \vec{b} - A_{B}^{-1} A_{N} \vec{x}_N}
\]

In our example, to check our dictionary was optimal we would have written out

\[
A_B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 2 & 0 \end{bmatrix}, \quad A_N = \begin{bmatrix} 1 & 1 & 0 \\ 2 & 0 & 0 \\ 3 & 1 & 0 \end{bmatrix}, \quad \vec{c}_B = \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}, \quad \vec{c}_N = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}.
\]

Thinking back to what the revised simplex method wants us to do, we need to find \( \vec{y} = (A_B^T)^{-1} \vec{c}_B \) by solving \( A_B^T \vec{y} = \vec{c}_B \):

\[
\begin{bmatrix} 1 & 0 & 1 & 2 \\ 0 & 1 & 0 & 1/2 \\ 0 & 1 & 0 & 0 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 1 & 2 \\ 0 & 0 & 1 & 1/2 \\ 0 & 0 & 1 & 0 \end{bmatrix} \Rightarrow \vec{y} = \begin{bmatrix} 3/2 \\ 0 \\ 1 \end{bmatrix}.
\]

Then we compute that \( z = \vec{y}^T \vec{b} + (\vec{c}_N^T - \vec{y}^T A_N) \vec{x}_N \) is

\[
\begin{bmatrix} 3/2 & 0 & 1/2 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + \left( \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 3 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 2 & 0 \\ 3 & 0 \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \\ x_6 \end{bmatrix} \right)
\]

which becomes \( z = 3 - x_3 - \frac{3}{2} x_4 - \frac{1}{2} x_6 \).

Now, that was our computation for the original problem, which confirms that our \( \vec{x}_B \) and \( \vec{x}_N \) lead to an optimal solution with \( z = 3 \). But what about the new one, changing the coefficient of \( x_3 \) in the objective

Fall 2017
Notes by Dan Collins
function to 1.9 or 2.1 or (more generally) some parameter \( c_3 \)? Well we just need to find where that 2 was and replace it with \( c_3 \) and we get

\[
z = \left[ \begin{array}{ccc} 3/2 & 0 & 1/2 \\ 2/3 & \end{array} \right] + \left[ \begin{array}{ccc} c_3 & 0 & 0 \\ 1/2 & \end{array} \right] - \left[ \begin{array}{ccc} 3/2 & 0 & 1/2 \\ 2 & 0 & 0 \\ 3 & 0 & 1 \\ \end{array} \right] \left[ \begin{array}{c} x_3 \\ x_4 \\ x_6 \end{array} \right].
\]

So \( z = 3 + (c_3 - 3)x_3 - \frac{3}{2}x_4 - \frac{1}{2}x_6 \). Looking at this, we see it will be optimal as long as \( c_3 \leq 3 \). In particular if \( c_3 = 2.1 \) or 1.9 our same solution (with our same objective value!) will still remain optimal. So we can conclude:

- Our optimal dictionary (and the corresponding optimal solution \( x_1 = x_2 = 1, x_3 = 0, z = 3 \)) remains optimal whenever \( c_3 \leq 3 \).
- If \( c_3 > 3 \), it’s not optimal any more, but that isn’t a huge deal: we just follow the simplex method and do another pivot to let \( x_3 \) enter and see where that takes us.

This sort of thing is what I mean by \textit{sensitivity analysis} - working out how changes to the initial problem do or don’t affect the optimal solution, and coming up with a range where things remain the same (at the very least, the same \( \vec{x}_B \)).

What if I’d changed the coefficient of \( x_1 \) to 2.1 or 1.9 (or more generally \( c_1 \)) instead of the coefficient of \( x_3 \)? This would change \( \vec{e}_B \), since \( x_1 \) is a basic variable, so we have to work a little harder to see how it affects things: \( \vec{y} \) depends in \( c_1 \) now, and we solve

\[
\left[ \begin{array}{ccc} 1 & 0 & 1 \\ 0 & 1 & 2 \\ 0 & 1 & 0 \end{array} \right] \sim \left[ \begin{array}{ccc} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array} \right] \sim \left[ \begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array} \right] \Rightarrow \vec{y} = \left[ \begin{array}{c} c_1 - 1/2 \\ 0 \\ 1/2 \end{array} \right].
\]

Thus the objective function for the optimal dictionary is

\[
\left[ \begin{array}{ccc} c_1 - 1/2 & 0 & 1/2 \\ 2/3 & \end{array} \right] + \left( \left[ \begin{array}{ccc} 2 & 0 & 0 \\ 1/2 & \end{array} \right] - \left[ \begin{array}{ccc} c_1 - 1/2 & 0 & 1/2 \\ 2 & 0 & 0 \\ 3 & 0 & 1 \end{array} \right] \right) \left[ \begin{array}{c} x_3 \\ x_4 \\ x_6 \end{array} \right].
\]

so

\[
z = (c_1 + 1) + (1 - c_1)x_3 + (1/2 - c_1)x_4 - 1/2x_6.
\]

So we conclude this stays optimal as long as \( c_1 \geq 1 \), and in that case the optimum value is \( z = c_1 + 1 \). If \( c_1 \) goes below 1 then \( x_3 \) gets a positive coefficient and we will need to let it enter.