**Sensitivity (Post-optimality) Analysis of Linear Programming Problems**

**(**You should study this note along with the [example](file:///%5C%5Cacfiles%5Cakinc%5Cwww-home%5CFIN203%5CSensitivity.xls) problem distributed in class.)

In many applications or linear programming, the problem data either cannot be gathered very accurately or it may be too expensive to do so. Therefore, ex post, the decision maker may find (suspect) that the problem data was (might be) in error in some way. Seeing that the optimal solution is robust (i.e., it is not overly sensitive to reasonable deviations from what was inputted) is somewhat comforting. If an optimal solution is overly sensitive to even small changes in the problem data, the confidence in the optimal solution of the **model** to be also optimal to the **real world problem** will be low. Most linear programming computer packages, such as Excel's solver routinely create sensitivity reports to help decision maker to build confidence in the optimal solution. We will briefly review Excel Solver’s Sensitivity Report.

**Scope:** Although any problem data may be the subject of sensitivity analysis, we will confine our attention to:

• Objective function coefficients (*cj*)

• Right hand sides of constraints (*bi*)

• We will first allow only one parameter --one *cj* or one *bi* to change at a time, and then extend the analysis to simultaneous changes.

**Common terminology and conventions:**

• **OS**: Optimal solution (optimal values of the decision variables, *xj\**) *e.g., E =* 5*, F =* 6*, G =* 0

• **O**V: Optimal value of the objective function‑‑ a single real number, *e.g.,* $49,000.

1. **Objective function coefficients**

 The question that the sensitivity analysis answers here is this: By how much can the objective function coefficient of the *jth*variable, *cj* change (go up or down), before the current OS is no longer optimal? This range is called the *allowable range.* Therefore as *cj* changes within this range:

* OS stays the same (definition of allowable range)
* OV changes at a rate equal to *xj\**or *Δ*OV = *Δcj*, *xj*\* which means *ΔOV/* *Δcj* = *xj*\*.

In the example problem coefficient of *E*, currently $5,000*,* can decrease indefinitely and increase by as much as $3,000 (up to $8,000) before the current solution, OS : (*E* = 5*; F =* 6*, G =* 0)isno longer optimal, and some other solution becomes optimal. Also as long as the change stays within the allowable range , the OV will change by 5\* *Δcj.* If the coefficient of *E* increases by $1,000 (to $6,000) OV increases by $1,000 \* 5 = $5,000. Namely they will continue making 5 *E's* but will make a $1,000 more on each of those 5units, or extra $5,000profit.

In this part of the sensitivity table there is also a column called *reduce* cost.

• If *xj\** > 0 the reduced cost = 0

• If *xj\** = 0 reduced cost will generally not be zero and will mean

* + By how much *xj* must become more attractive (its *cj* must increase for a max problem, or decrease for a min problem) before *xj* can assume a positive value in the OS.
	+ Equivalently, if one insists that *xj\** > 0, the reduced cost tells by how much per unit of *xj* the OV will suffer (increase for a min problem, decrease for a max problem)

In the example problem *G*  = 0, its reduced cost is -$6,400. This means that unless the objective function coefficient of *G*, currently $2,000 is sweetened (increased) at least by $6,400 to $8,400, the optimal value of *G* will continue to be zero; alternatively, if *G* is forced to be > 0, the OV of $49,000 will decline at a marginal rate of $6,400 (loosely speaking, a unit of *G* will deduct $6,400 from the optimal profit.)

1. **Right hand sides of constraints**

 **Preliminaries:**

* Every inequality‑constrained problem can be represented as an equivalent equality constrained problem. Consider the example problem with equalities replacing the inequalities:

 *E+ F –S*1 *=* 5

 *E–* 3*F +S*2 *=* 0

20*E +* 10*F +* 15*G + S*3 *=* 160

10*E+*15*F+*12*G +S*4 *=* 150

30*E +*10*F +* 7*G –S*5= 210

Notice that we are adding a non‑negative quantity to the left hand side of a <= type constraint and subtracting a non‑negative quantity from the left hand side of a >= type constraint. Verify in your mind that all *Si* >= 0 implies that all the inequality constraints are satisfied, conversely if some *Si* < 0 then that constraint is violated. The *S* variables are called *slack* (if added) and *surplus (*if subtracted) variables.

* If an original inequality constraint holds as a strict equality (left hand side = RHS), the value of the associated slack/surplus will be zero. We call such a constraint as a *binding (active)* constraint. Plug in above the values of *E* = 5, *F* = 6, *G* = 0to calculate the values of the slack/surplus variables to identify the active and inactive constraints.
* **Metaphors for constraints:** Think of a ">=" type constraint as a *target* ora *hurdle* that must be met or exceeded and a " <= " type constraint as a limited *resource* constraint that the solution must be within.
* When you increase the RHS of a target (make the hurdle higher), or decrease the RHS of a resource (curtail the availability of it), the constraint becomes more difficult to satisfy. We call these types of changes in the RHS (*bi*) as *tightening* the constraint. Vice versa for *loosening.*
* Tightening a constraint (changing the RHS appropriately) can never help the *OV*. It may decrease the value ofa maximization objective function and may increase the value of minimization objective function. Likewise loosening a constraint cannot harm the OV.
* **Important new concept:** the *Shadow price of* a constraint is the rate of change in the *OV* as the RHS (*bi*)ofthe constraint changes *i.e., ΔOV/Δbi = ith* shadow price.
* *Allowable* range for *bi* is that range within which *bi* can change without altering the shadow prices (all of them).

**End Preliminaries**

 **a)RHS of an inactive constraint**

Inactive means the associated *Si* > 0 (there is some slack or surplus in the constraint) and the constraint is not limiting the objective function. See that this means the shadow price, *ΔOV/Δbi =* 0. An inactive constraint can be loosened indefinitely and can be tightened at most by *Si* > 0. In the example the first constraint is " >= " type (a hurdle) and has a surplus of 6 units ( *E* + *F* is 11 while the RHS is 5). The RHS can be reduced (constraint be loosened) indefinitely and increased (tightened) at most by 6 (can lower the hurdle as much as you want, but can raise it by at most 6 after which *E =* 5*, F* = 6 will no longer be feasible. So, the allowable range for *b*1 extends from *-∞* to 11 (with Excel's terminology *allowable increase* = 6; *allowable decrease* = ∞). In the allowable range:

* OV stays the same (since *ΔOV* = 0 \* *Δbi = 0*)
* OS stays the same, except, the associated *Si*.
* Of course, by definition of the allowable range, all shadow prices stay the same.

In the example, DeptB constraint, a "<=" type is non‑binding; slack = 10 (10 units of the available capacity is idle, only 140 hours are needed). This constraint can be relaxed (the RHS increased) indefinitely but tightened at most by 10 units. Within the interval from 140to ∞, nothing changes except the value of the slack (unused capacity of DeptB).

**b) RHS of an active (binding) constraint**

Active means *S* = 0 (left hand side of the constraint evaluates to match exactly the RHS). In the allowable range (as reported by the software):

* Shadow prices will remain intact (definition of the allowable range)
* OV changes at the rate of shadow price (definition of shadow price), *i*. *e., ΔOV = Δbi* times theshadow price of the *ith* constraint (as reported by software).
* In general, the OS will change!

In the example, DepA constraint, a" <= " type (resource) is active (binding) and its shadow price is $700*.* The solution *E = 5, F =6* and G = 0 uses all the available 160 units of capacity (*S*3 *=* 0). Allowable range extends from 160 ‑ 13 *=* 147to 160 + 2.85 *=* 162.25*.* If *bi* stays in this interval, say

Δ*bi = ‑*6(a six unit decrease in the availability of the DeptA's capacity): All the shadow prices will stay at the same levels; the OV will change by Δ*bi \* shadow* price = ‑6 \* $700 *=$‑*4,200*; OS* (values of *E, F,* G and thus all the slack/surplus variables) will probably change.

**C**. **Simultaneous changes in *cj* or *bi***

So far, we have assumed that only one *cj* or *bi* changes, while every other problem data remains constant. The so‑called 100% rule allows simultaneous changes to *cj* or *bi* to be considered.

**100% rule:** If the sum of changes (up or down) in multiple *cj* or *bi's* as a % of their respective allowable increase or decrease does not exceed 100%, the above rules for single changes in the allowable range apply.

 In the example, say *E's* coefficient increases from *$*5,000to *$*6,000 *i. e., ΔcE =* $1,000and

at the same time *F's* coefficient decreases by *$*500 from *$*4,000to *$*3,500 *i. e., ΔcF =* $*‑*500.

Allowable increase of *E*'s coefficient is *$*3,000while the allowable decrease of *F's* coefficient is

*$*1,500. *E's* coefficient increased by 33.33*%* of the allowable increase and *F's* decreased by *33.33%*

of the allowable decrease. The sum is66.66%therefore within the 100% rule. Thus:

* *OS: (E =* 5*, F =* 6*,* G = 0) remains optimal
* OV: changes by +$1,000\*5 *-* $500\*6 *=* $2,000*.*

Again, since the solution does not change, with these simultaneous changes you will make $1000 more on each of the 5 *E's* and *$*500less on each of the *6 F', s* netting a *$*2,000increase in profit.

 Likewise, in the example, say the RHS of the first, third and fifth constraints change simultaneously by *+*2*,* ‑2and *+*5respectively. Allowable increase of the first constraint's RHS is 6*;* allowable decrease of the third constraint is 13and allowable increase of the fifth constraint is 18.57*.* The changes are respectively 2/6 = 33.33%;2/13 = 15.4%and 5/18.57 = 27%.They add up to 75.7*%.* Thus the rules of within allowable range changes apply:

* Shadow prices remain the same.
* *ΔOV = ∑ bi* \* shadow price of constraint *i* = 2 \* 0 +(-2) \* 700 + 5 \* (-300) = -8,400
* OS (*E* = 5, *F* = 6, *G* = 0) may however change.