INVENTORY OPTIMIZER

**Executive Summary**

As a Supply Chain Major, one of the best tools I obtained in the last 8 months is a series of equations that allow any manager to optimize the level of inventory within a warehouse.

These equations take into consideration a series of parameters such as Demand, Cost, Standard deviation of demand, Lead Time, Standard Deviation of Lead Time, Margin.

In order to find an optimum equilibrium between Service Level (the percentage of demand that can be satisfied – Happy Customers) and profit it is necessary to plug all of these parameters in several formulas: the quickest way to come to a solution is through the use of SOLVER within Microsoft Excel.

The purpose of this project is to create a quick, automated way to find the right amount of inventory to be kept within a warehouse of n products.

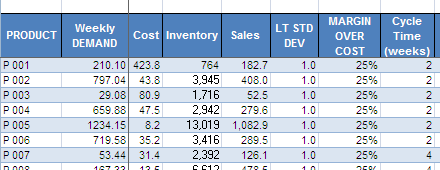
Solver nevertheless has a glitch, or a bug, that in some instances will not allow you to find the optimum.

I was able though to overcome this problem through the use of a FOR-NEXT cycle.

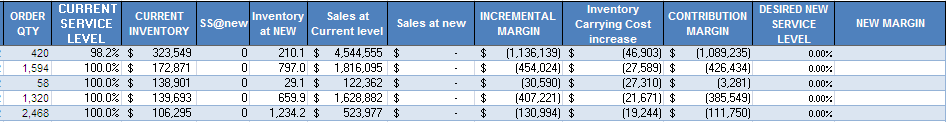
At the same time I implemented a system to handle and modify the user data through the use of forms.

**How I did it**

The first step is to create a spreadsheet where the user can add all the variables concerning the products present in the warehouse.



The following columns contain formulas (with a simple if function the columns don’t fill if the variables are not inserted).



The last two column have a special purpose: by changing the desired service level Solver finds the solution to maximize the new Margin, which is basically a function of the increase in Carrying Cost and of the Contribution Margin.

**Solver Bug**

As I started to run the code for Solver I realized that solver has a major glitch. It will not find the best Optimum point but it will only find a local optimum based on the starting value of the cell to be modified.

I found a solution to this by reiterating a cycle of Solvers for a series of values, starting from 95% up to 100%. Experience shows that most likely the optimum point falls in this range: furthermore companies want to satisfy at least 95% of their customers on a regular basis.

By using references to active cells, the program reads all the active lines and quits the optimization as soon as it finds an empty line

**Modifying Data**

Once the data have been inserted by the user, it is possible to modify them through the use of two user forms (associated to a macro and a button): it is much easier to modify them with a form rather than by working on a series of rows and columns. Any change applied to the form is then applied to the spreadsheet, given that it is a valid input (number).

**What I learned**

The project allowed me to master the use of Solver and of user forms. Solver especially has a major glitch that I was able to overcome with the use of a for cycle.

**Conclusion**

Instead of having to manually run Solver thousands of times, the code allows the user to find optimum points with a single click, even if the time required is still in the order of several minutes.

This code will allow smaller companies to handle their inventory in a simple but proven way, highly increasing their profits by reducing costs and finding the desired service level.