NTNU
Norwegian University of Science and Technology

Comparison between Matlab and @ RISK as a tool for risk analysis

Carlos Hernandez Ares

Submission date:June 2014

Supervisor: Reidar B Bratvold

Norwegian University of Science and Technology
Department of Petroleum Engineering and Applied Geophysics

Preface

This master's thesis is the concluding work for my degree in Mining Engineering from the University of Vigo. During the Spring 2014 and under the guidance of Pr. Bratvold, I have worked on a comparison between MATLAB and @RISK in risk analysis. This thesis is submitted at the Department of Petroleum Engineering and Applied Geophysics. The workload is twenty weeks, considered to be equivalent to one semester of full- term studies.

I would like to thank my supervisor Pr. Bratvold for useful guidance during my work. I would also like to thank the NTNU and the Erasmus program to give me the chance to finish my degree in a high level education university as the NTNU is.

Finally, I would like to thank my family for continuous support during my years of studying, my Erasmus friends, who helped me have a social life besides this thesis, and specially my girlfriend, Alba, who has always been there whenever I need it through the good and the bad times.

Carlos Hernandez Ares
June 9, 2013

Abstract

Risk derives from our inability to see into the future, and indicates a degree of uncertainty that is significant enough to make us notice it. The importance of risk analysis rests in the importance of evaluating certain risks in order to make decisions based on those results. Risk analysis is very important within the oil and gas industry, since there are a lot of variables that can be measured and that can influence whether a project should be or not be developed.

In order to measure the risk it is important to have powerful calculation tools that are fast and accurate. Computer programs are an obvious choice in order to obtain good results for the risk analysis. The purpose of this thesis is to address the differences and similarities of two useful programs, @RISK and MATLAB, in order to try to come up with a conclusion of which of them is a better tool in the oil industry.

The thesis compares MATLAB and @RISK at many different levels, from ease of implementation to graph display, passing through sampling algorithm or speed of calculation. In the end, a conclusion is reached, which will help decision makers whether to use one or the other program depending on the type of simulation that will be run.

Contents

Introduction

1. @RISK.

1.1. Introduction and brief history.

@RISK (pronounced 'at risk') is an add-in to Microsoft Excel, integrated completely with its spreadsheet. It was first released in 1987 by the Palisade Corporation, which is a company specialized in software adds in to Microsoft Excel like the mentioned @RISK and the Decision Tools Suite.

Palisade Corporation was founded in 1984 and its first product was PRISM, which gave PC users the ability to quantify risk by running Monte Carlo simulations. As said before, in 1987 @RISK was developed as the world's first Monte Carlos simulation add-in for spreadsheets. Building on the success of @RISK, Palisade has developed other programs to meet the growing professional demand for technically refined decision support software. Palisade is headquartered in Ithaca, New York, but it also maintains offices in London, Sydney, Rio de Janeiro and Tokyo.

@RISK performs risk analysis using Monte Carlo simulation to show the possible outcomes in the spreadsheet model. It mathematically and objectively computes and tracks many different possible scenarios, then tells you the probabilities and risks associated with each different one. This helps judging which risks to take and which ones to avoid.

@RISK is a true add-in to Microsoft Excel. All @RISK functions are true Excel functions and behave exactly as native Excel functions do. @RISK window are all liked directly to cells in your spreadsheet, so changes in one place are carried out in the other.

1.2.Working environment.

Since @RISK is a Microsoft Excel add-in, it works in a style very familiar in its menus and functions. @RISK adds risk analysis capabilities to a Excel spreadsheet, @RISK uses menus, toolbars and custom distribution functions into it.

Once the @RISK program is installed in your computer, an @RISK menu is added to Excel, allowing you to access all commands necessary for setting up and running simulations.

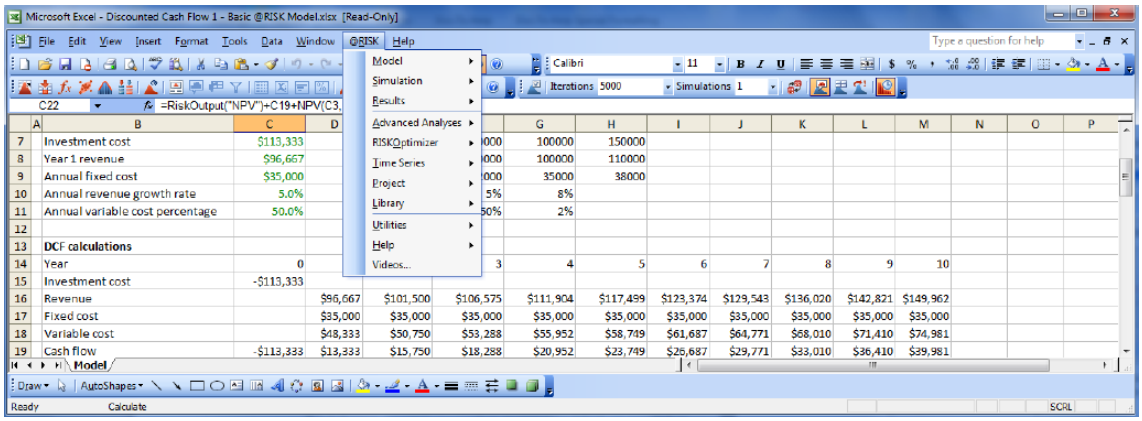


Figure 1.1.

Also a @RISK ribbon bar is also added to Excel. The icons and commands on these bars allow you to quickly access most @RISK options.

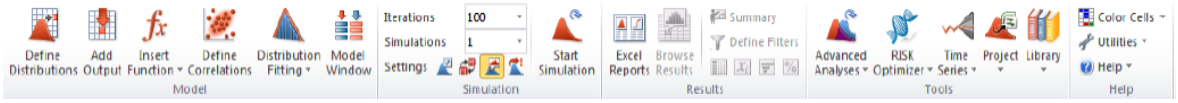


Figure 1.2.

Icons on this toolbar include:

- Simulation Settings: opens the Simulation Settings dialog box.
- Iterations: drop-down list , where the number of iterations to run can be quickly changed from the toolbar
- Simulations: drop-down list, where the number of simulations to run can be quickly changed from the toolbar.
- Random/static scale: flips @RISK between returning expected or static values from distributions, to returning Monte Carlo samples in a standard Excel recalculation.
- Show a Graph and Demo Mode: control what is shown on the screen during and after a simulation

1.3.How @RISK works.

Running an analysis with @RISK involves three simple steps:

- **Set up the model.**

In @RISK the uncertain values in the spreadsheet are replaced by probability distribution functions, such as Normal or Beta. When entering a distribution function both the

function name and the arguments which describe the shape and range of the distribution are entered. The distribution functions may be used just as any normal spreadsheet functions would be.

@RISK includes a Define Distribution window that allows to easily add probability distribution functions to spreadsheet formulas. By clicking the Define Distribution icon in the ribbon bar this window is displayed. This window graphically displays portability distributions which can be substituted for values in a spreadsheet formula. By changing the displayed distribution various distributions describe the range of possible values for an uncertain input in the model.

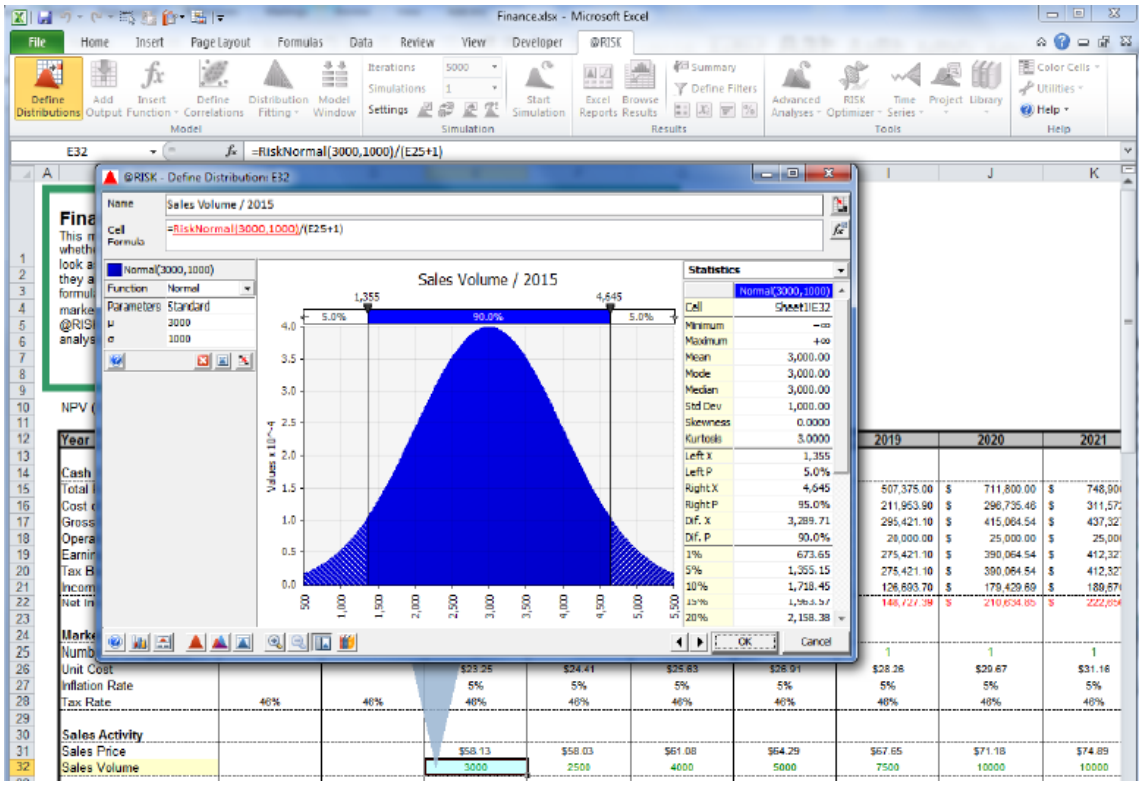


Figure 1.3.

After the inputs have been selected and each of them has a distribution linked to itself, the outputs must be selected. Typically , these output cells contain the results of the spreadsheet model but they can be any cells anywhere in the spreadsheet. To select outputs, simply highlight the cell or range of cells wanted as outputs and then click the Add Output icon in the ribbon bar.

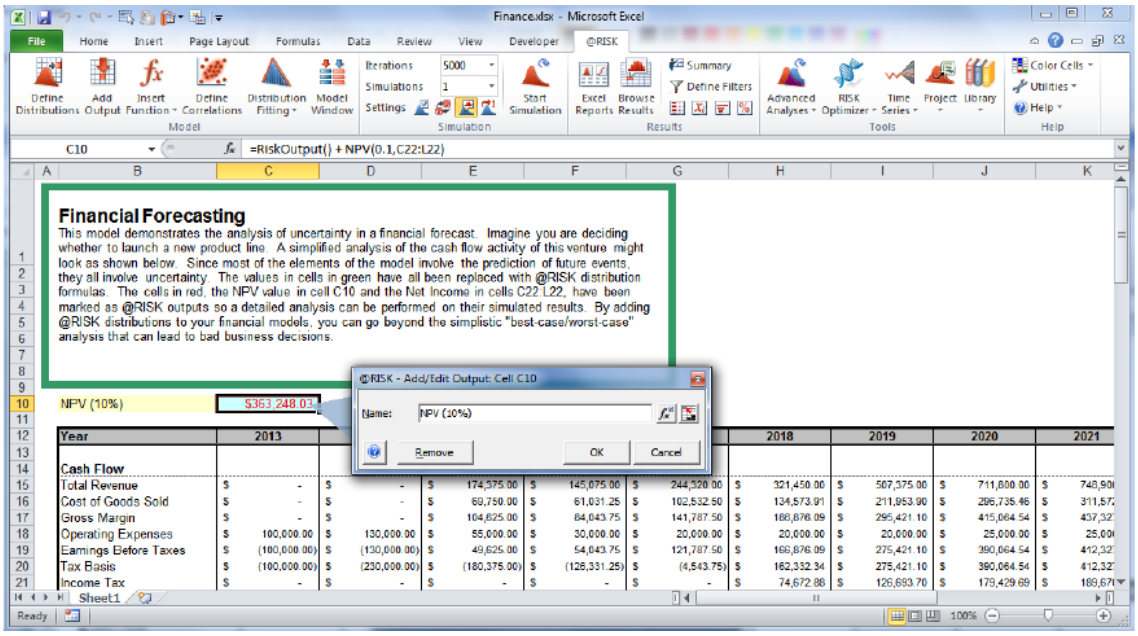


Figure 1.4.

An useful feature of @RISK that provides a complete table of all input probability distributions and simulation outputs described in the model. This window is called Model window. This window allows to:

- Edit any input distribution or output by simply typing in the table.
- Drag and drop any thumbnail graph to expand it to a full window.
- Quickly view thumbnail graphs of all defined inputs.

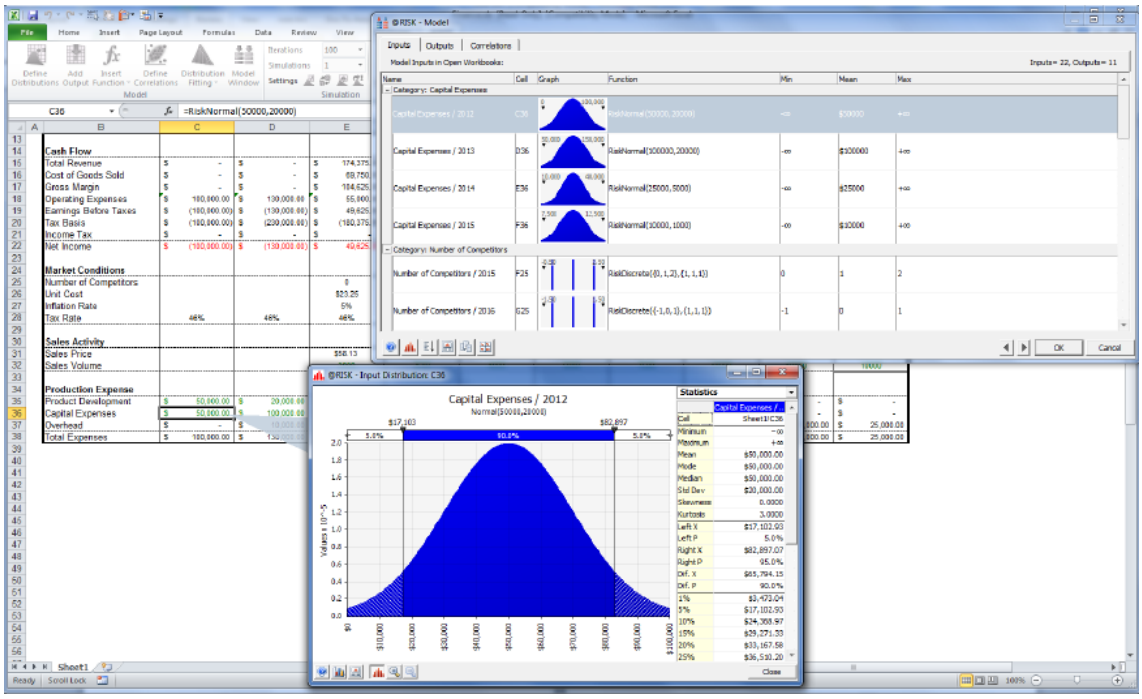


Figure 1.5.

- **Run the simulation.**

After all the input values have been set, as well as the output cells it is time to run the simulation to obtain the results. The simulation is run when you click the Start Simulation icon on the @RISK ribbon bar.

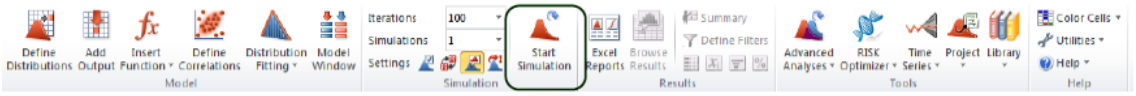


Figure 1.6.

When a simulation is run, the spreadsheet is calculated repeatedly with a set of new possible values sampled form each input distribution and each iteration. With each iteration the spreadsheet is recalculated with a new set of sampled values and a new possible result is generated for the output cells.

As the simulation progresses, new possible outcomes are generated from each iteration. @RISK keeps track of these output values and displays them on a popup graph which is displayed with an output.

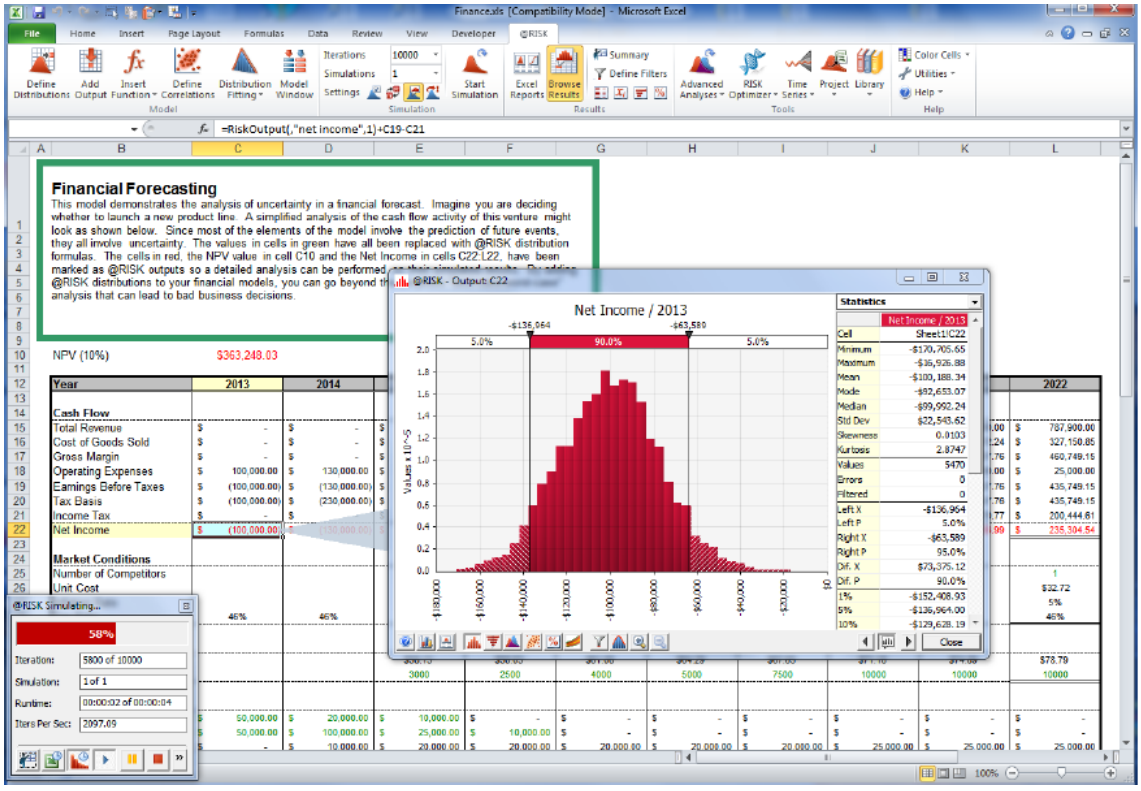


Figure 1.7.

This graph of the distribution of possible outcomes I created by taking all the possible output values generated, analyzing them and calculating statistics on how they are distributed across their minimum-maximum range.

When a simulation is running a progress window is displayed. The icons in this window allow to Run, Pause or Stop a simulation, as well as turn real-time updating of graphs and Excel recalculations on and off.

@RISK simulation results include distributions of possible results for the outputs. In addition, @RISK generates sensitivity and scenario analysis reports which identify the input distributions most critical to the results.

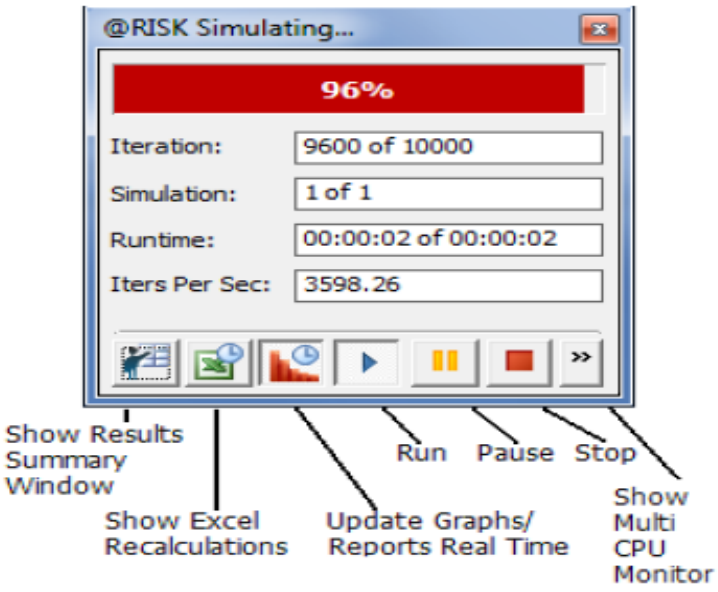


Figure 1.8.

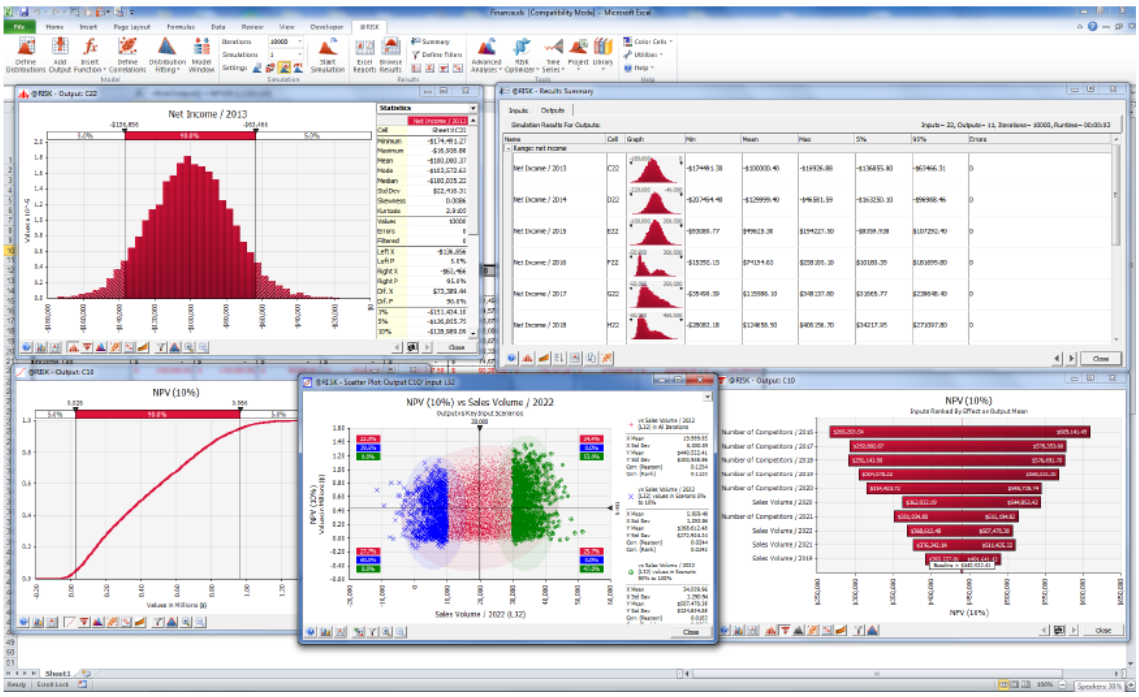


Figure 1.9.

1.4. Graphs.

In @RISK simulation results are easily expressed with graphs. The Results Summary Window shows thumbnail graphs of the simulation results for all your outputs and inputs. A graph of the results for an output shows the range of possible outcomes and their relative likelihood of occurrence. This type of graph may be displayed in standard histogram or frequency distribution form.

A very useful feature in @RIKS is that each graph created is displayed in conjunction with the statistics for the output or input graphed. In addition, by clicking the right mouse button on a graph window, a pop-up menu is displayed with commands that allow the changing the format, scaling, colours, titles...

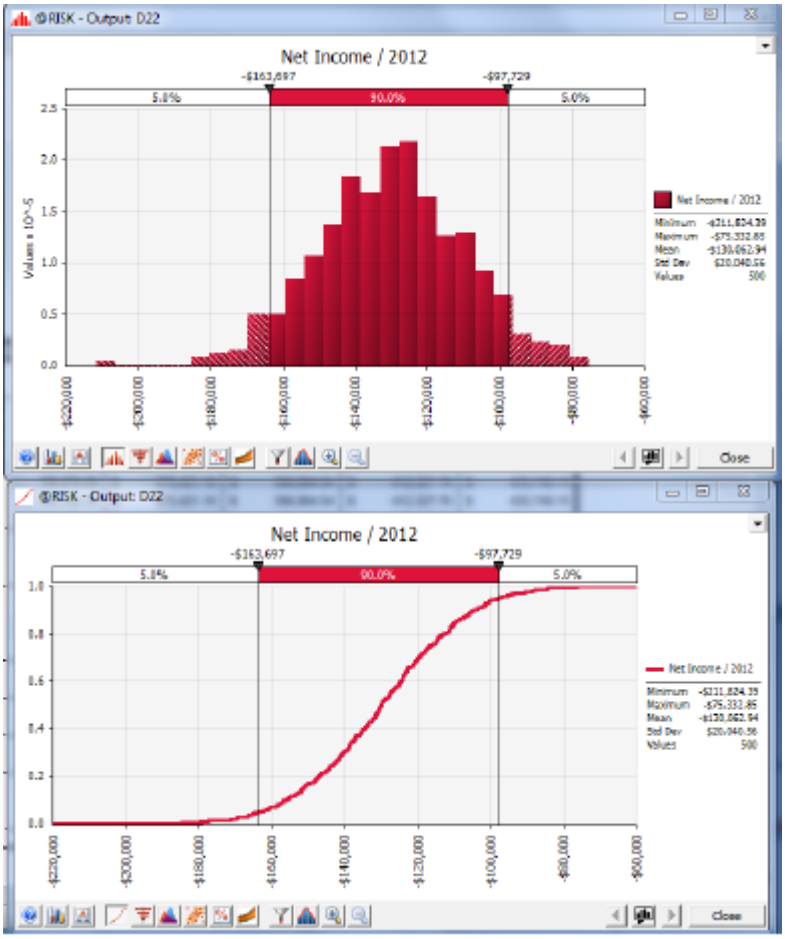


Figure 1.10.

@RISK also gives the possibility of overlay graphs for comparison by dragging one graph onto another.

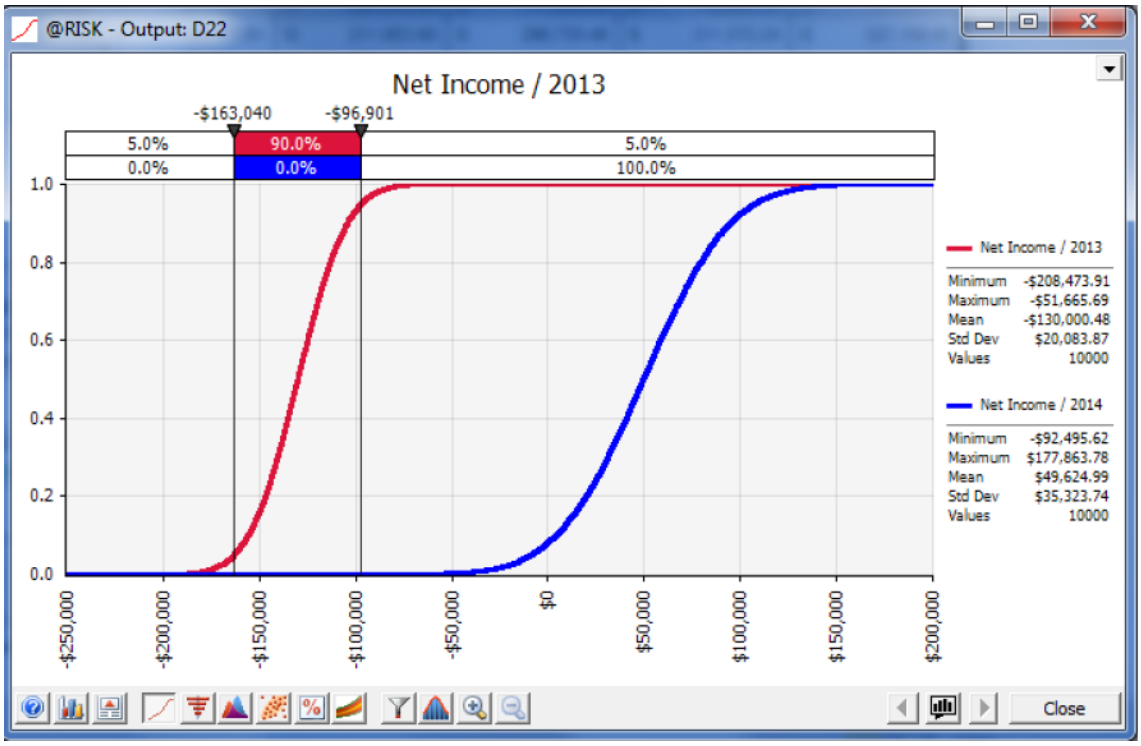


Figure 1.11.

@RISK has also the possibility of draw its graphs in Excel's native graph format. These graphs can be changed or customized just as with any Excel graph.

The interface in @RISK also allows to colour cells in the spreadsheet where @risk inputs, outputs, statistics functions and optimization variables are locates. This allows to easy identify the components of the model.

1.5.Using @RISK in risk analysis.

@RISK is the ultimate tool for risk analysis. This is a programs that was specifically created to make Monte Carlo simulations and to perform analyses.

A major strength of @RISK is that it allows you to work in a familiar and standard model building environment, Microsoft Excel.

All of the features that @RISK possesses are included in order to facilitate all types of risk analysis. It has a wide range of probability distributions for the user can choose from. Probably no other program has a library of functions as big as @RISK's, and certainly no other offers the possibilities of working with an Excel spreadsheet.

A big advantage for @RISK is the fact that tables of results can be seen directly into your spreadsheet. This allows the user for a very quick understanding of the analysis and helps to realize where mistakes could have been made.

The inclusion in later versions of the icons in the ribbon bar (Excel 2010) has helped the interface to be more intuitive and has eliminated the chance of getting lost into a high

number of menus and drop-down lists. This feature is especially helpful for the non-advanced users and for the newcomers in the risk analysis.

Simulations that have already been run may be saved as a document, making @RISK very good in comparing different simulation results.

1.6. Quick overview.

As it has been said before, @RISK is a very powerful tool for risk analysis. It has numerous features that can help to solve a lot of situations and it has an easy interface with the user.

@RISK is used to analyze risk and uncertainty in a wide variety of industries. Form the financial to the scientific, some examples are insurance, manufacturing or environment.

The oil and gas industry is not an exception, since @RISK is capable of helping the decision making in Oil reserves estimation, Pricing or Exploration and production.

For the purpose of this Thesis, @RISK is clearly a program to be counted on and also worth to seek which are its biggest advantages and disadvantages.

2. MATLAB.

2.1. Introduction and brief history.

MATLAB is an interactive system and programming language for general scientific and technical computation. Its developing started in the late 70s by Cleve Moler from the University of New Mexico and soon started to spread to other universities. In 1983, Moler joined Jack Little and Steve Bangert in order to launch a commercial program. They rewrote MATLAB in C and founded MathWorks to continue its development. Today, MATLAB is used in an educational level as well as in the investigation and industry in order to solve complex scientific problems.

MATLAB is an environment of technical integrated computations that mixes numeric computation, graphs and advanced visualization and a high level computational language. Whatever the objective, an algorithm, an analysis, graphs or simulations MATLAB can take it there. The flexible and interactive language of MATLAB allows engineers and scientist to express their ideas with simplicity. The powerful and vast methods of numeric calculus and the ability to make graphs allow the testing and exploration of alternative ideas while obtaining easy practical results.

MATLAB is a program to solve numeric calculus with vectors and matrices. One of its more useful features is the vast number of graphs in 2-D and 3-D that you can use, which makes it a very useful tool in so many applications.

2.2.Working environment.

MATLAB interact with the user through windows. The most important ones are the following:

- **Command Window**

The Command Window is the most important in MATLAB. In this window is where you run the instructions you have given to MATLAB and where the results are shown. Until the 6.5 version, the command window was the only one that existed. In further versions many improvements were made in order to facilitate the user experience:

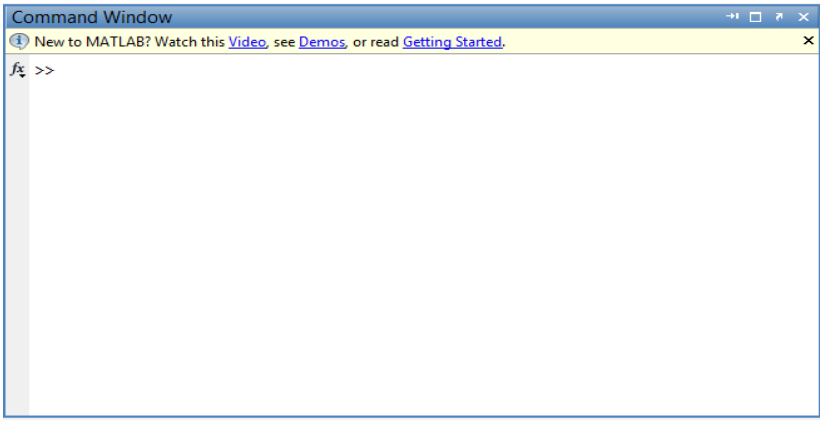


Figure 1.12.

- The command lines that are to long continue in the following line when they arrive to the end of the command window.

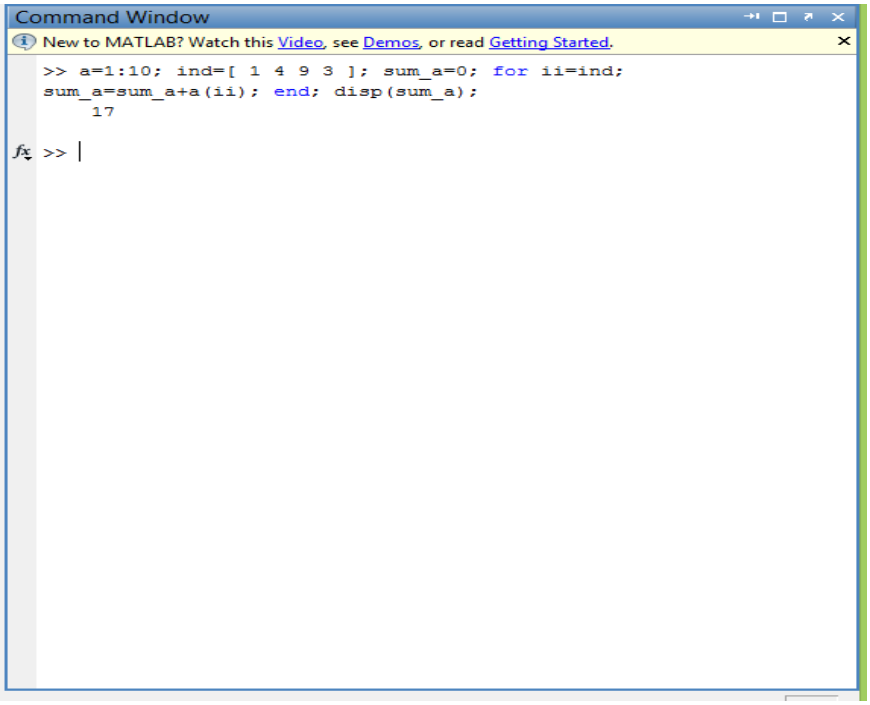


Figure 1.13.

- Clicking with the right mouse button on the name of a function you can access the Help page about that function. If the function has a code in a .m file you can also access the named file.

- While writing the name of a function and clicking the TAB button in your keyboard, MATLAB completes the name of the function automatically, or it shows you all of the possible functions that start with the letters written by the user.

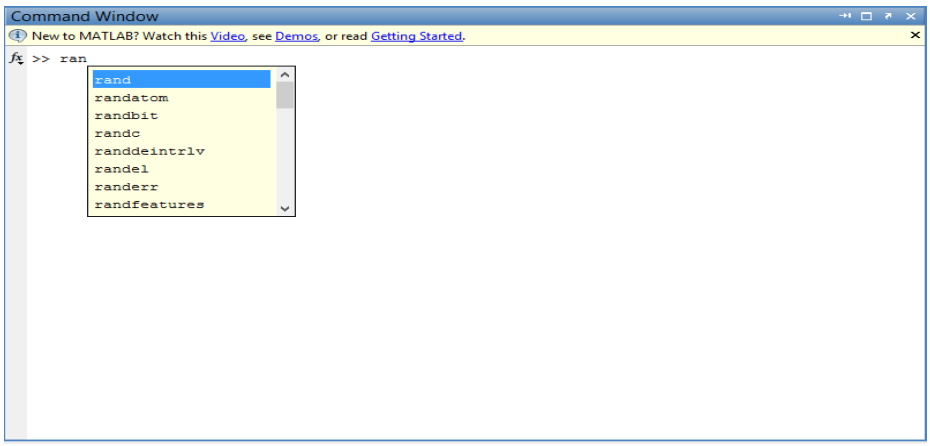


Figure 1.14.

- When an error occurs while running an .m file, in the command window it appears the correspondent message and it also shows the line where the mistake was made in the mentioned .m file. By clicking in the message you can access your .m file.

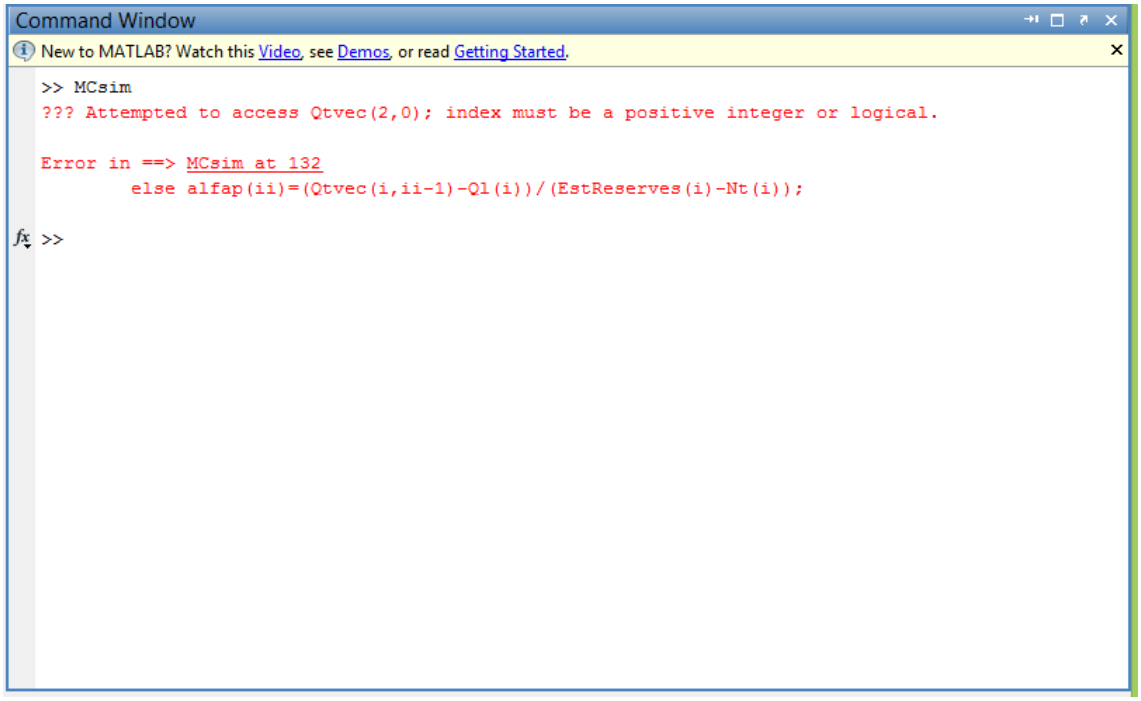


Figure 1.15.

- **Command History Browser**

The Command History Browser accesses the lines that have been run previously in the Command Window. You can also access to this sentences by pressing the arrows ↑ and ↓.

The previous sentences can be run again by double clicking over them. You can also copy the lines in to the Windows Clipboard. You can also delete some of all of the lines in this window.

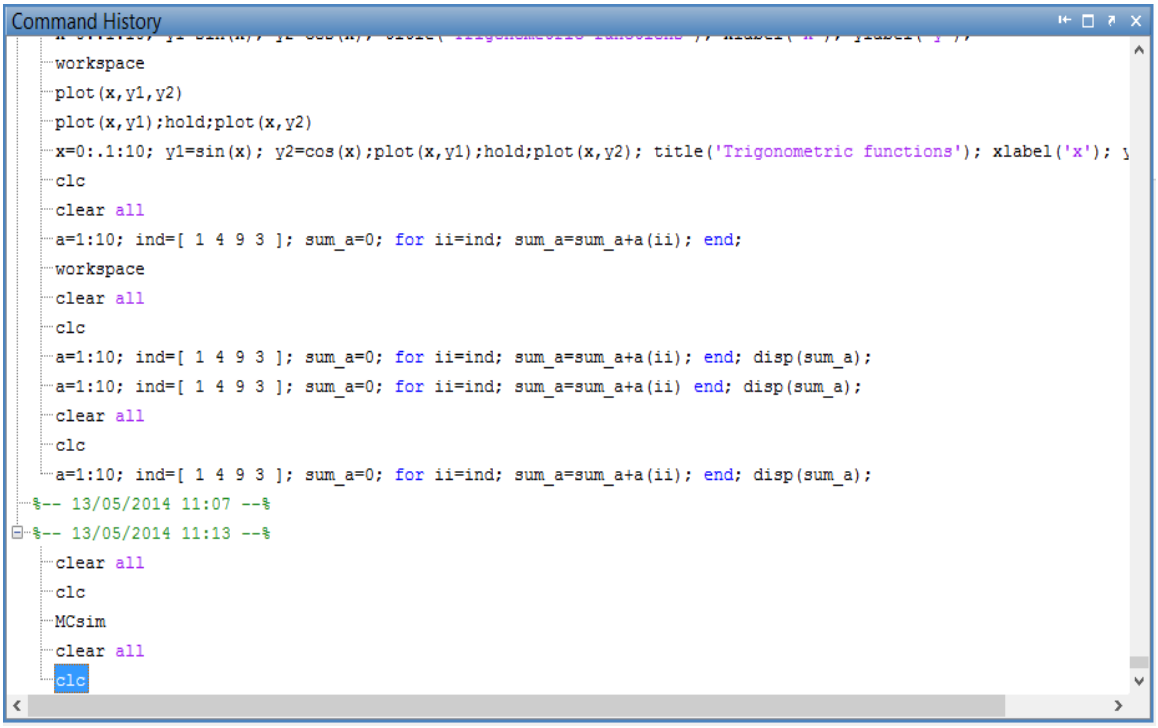


Figure 1.16.

- **Current Directory Browser**

The Current Directory Browser is very important in MATLAB. The programs of MATLAB are stored in files with the .m extension. This files are run by writing its name in the command window, but not all of the .m files that are stored in the hard drive or in a local network are accessible.

MATLAB has always one only active directory browser. This active directory is the first place where MATLAB searches is asked to run a file. If the name of the .m file is typed in the command window and such file is not placed in the active directory, an error message will appear and the .m file will not be run.

In the following image the list of the .m files that are currently stored in the active directory can be seen.

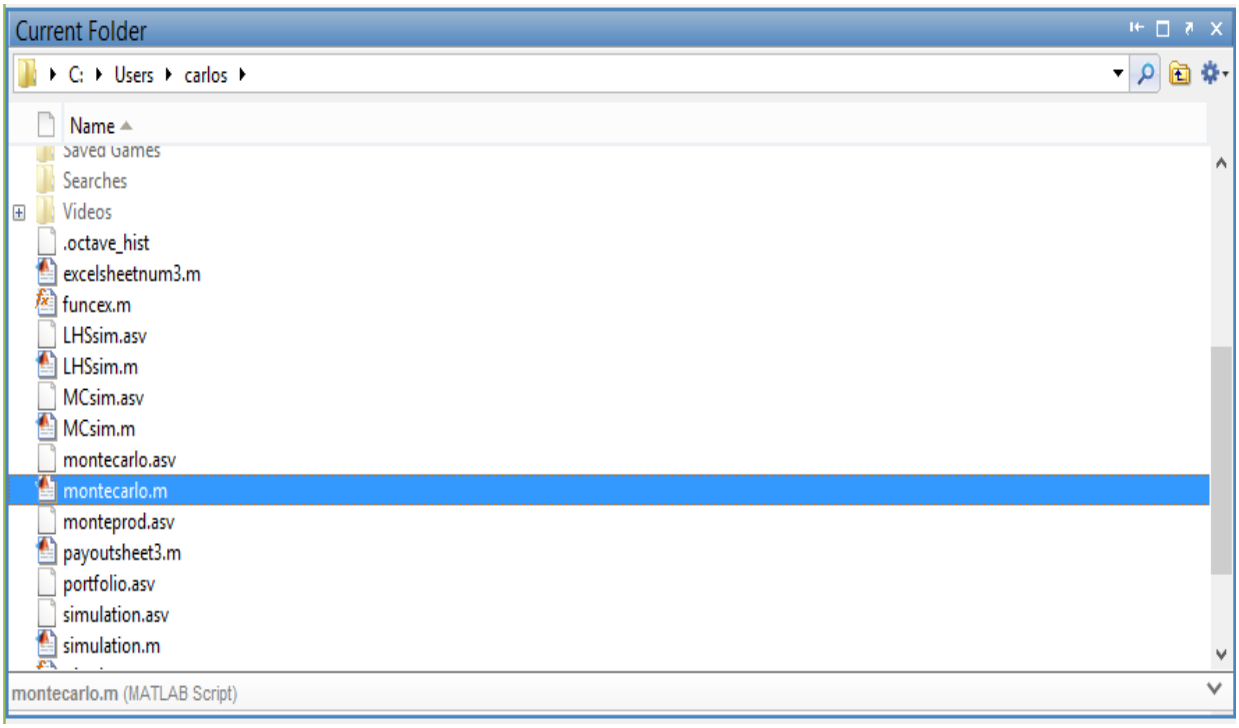


Figure 1.17.

The Current Directory Browser allows the user to explore the computer folder in a similar way as hw the File Explorer works.

- **Workspace Browser**

The Workspace is the group of variables and functions defined in the memory of the program. In order to obtain the information of the workspace form the command window 'workspace' must be typed.

In the workspace window the defined variables can be seen along with the following information of each variable:

- Name of the variable
- Size of the array
- Minimum and maximum values. (If the array is too big, this values may not be given).

By making double click in any of the variables you can access the values of each of the arrays.

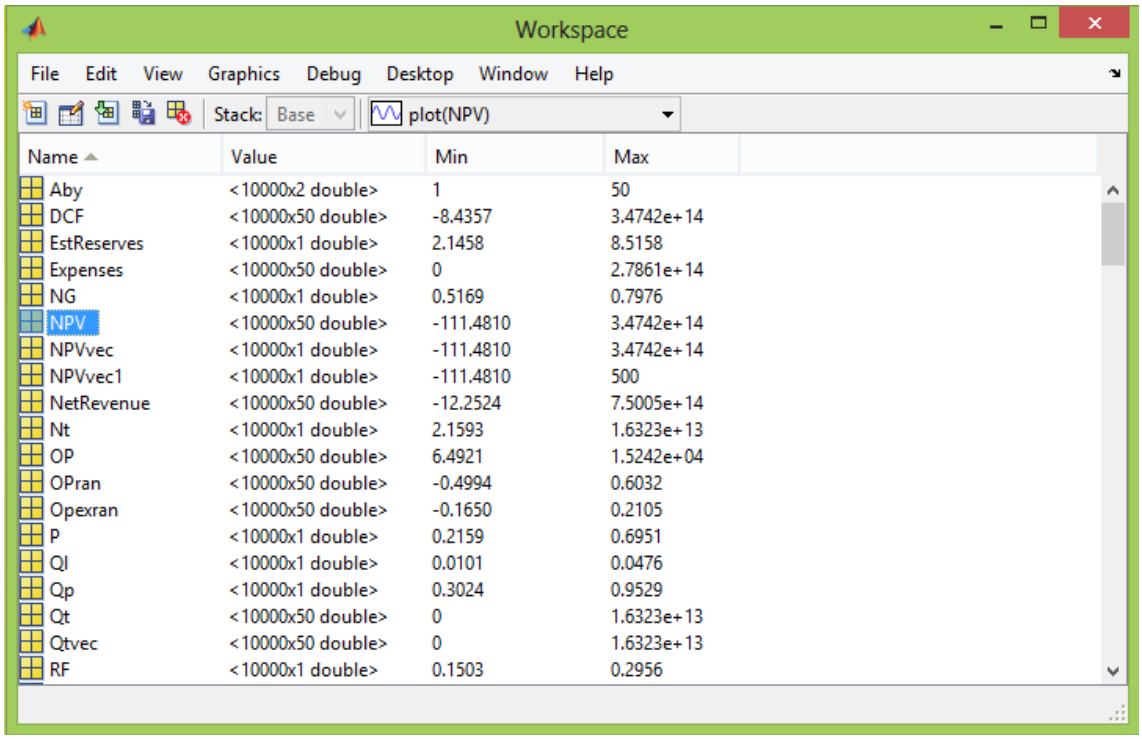


Figure 1.18.

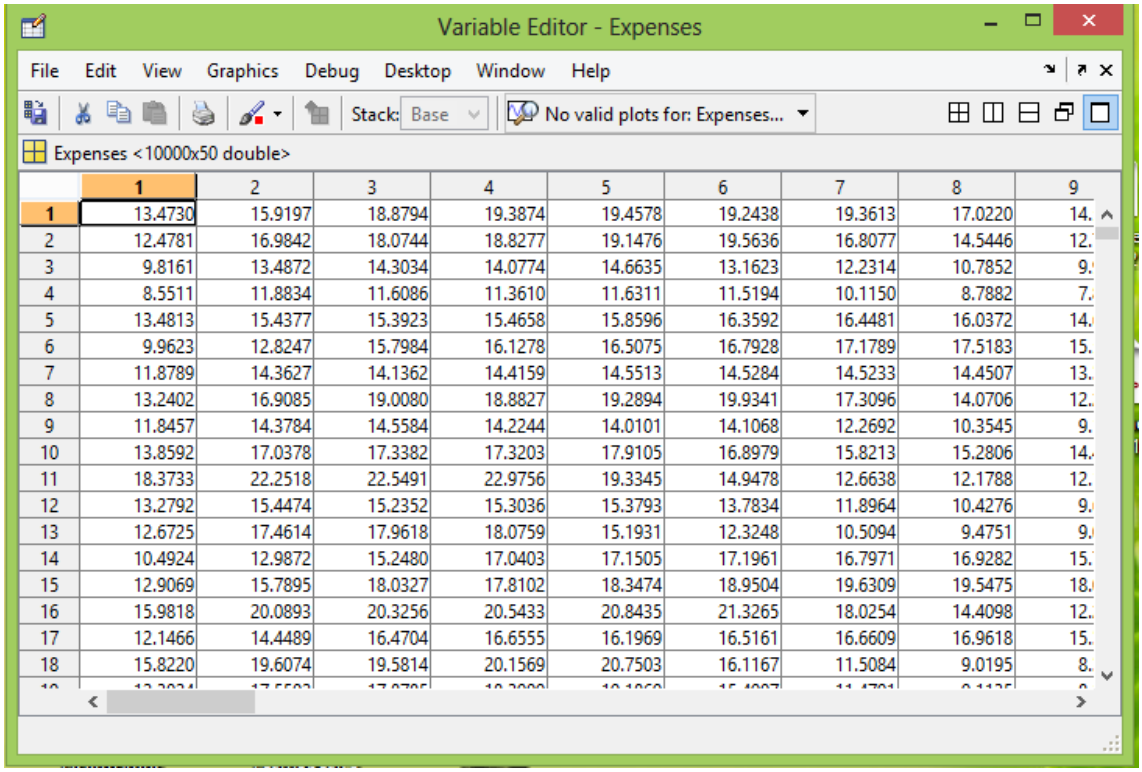


Figure 1.19.

2.3.Functions.

Functions are files that can accept input arguments and return output arguments. Functions operate on variables within their own workspace, separate from the workspace you access at the MATLAB command window.

MATLAB has a large number of incorporated functions. Some of them are already incorporated within the executable code of the program. This functions are extremely quick and efficient. There are also multiple functions defined in .m files that have been included in the program or that have been created by the users. The biggest difference between the incorporated functions and the other functions is that in the incorporated functions you are not allowed to see the code, while in the other functions you can see the .m file and even modify it. The function names are not reserved. It is possible to overwrite any of them with a new variable.

MATLAB has many different kinds of functions. Some of the most important are mentioned in the following list:

- Elemental mathematic functions: e.g.: sin(x), log(x), abs(x)...
- Vector functions: e.g.: min(x), prod(x), sum(x)...
- Elemental matrix functions: e.g.: size(A). trace(A), inv(A)...
- Specific matrix functions: e.g.: sqrtm(A), expm(A), logm(A)...
- Statistical functions: mean(x), std(x), betarnd(A,B)...
- Polynomial functions: e.g.: roots(pol), conv(pol), polyder(pol)...

The first line of a function starts with the keyword 'function'. It gives the function name and order of arguments. In this case, there are up to two input arguments and one output argument.

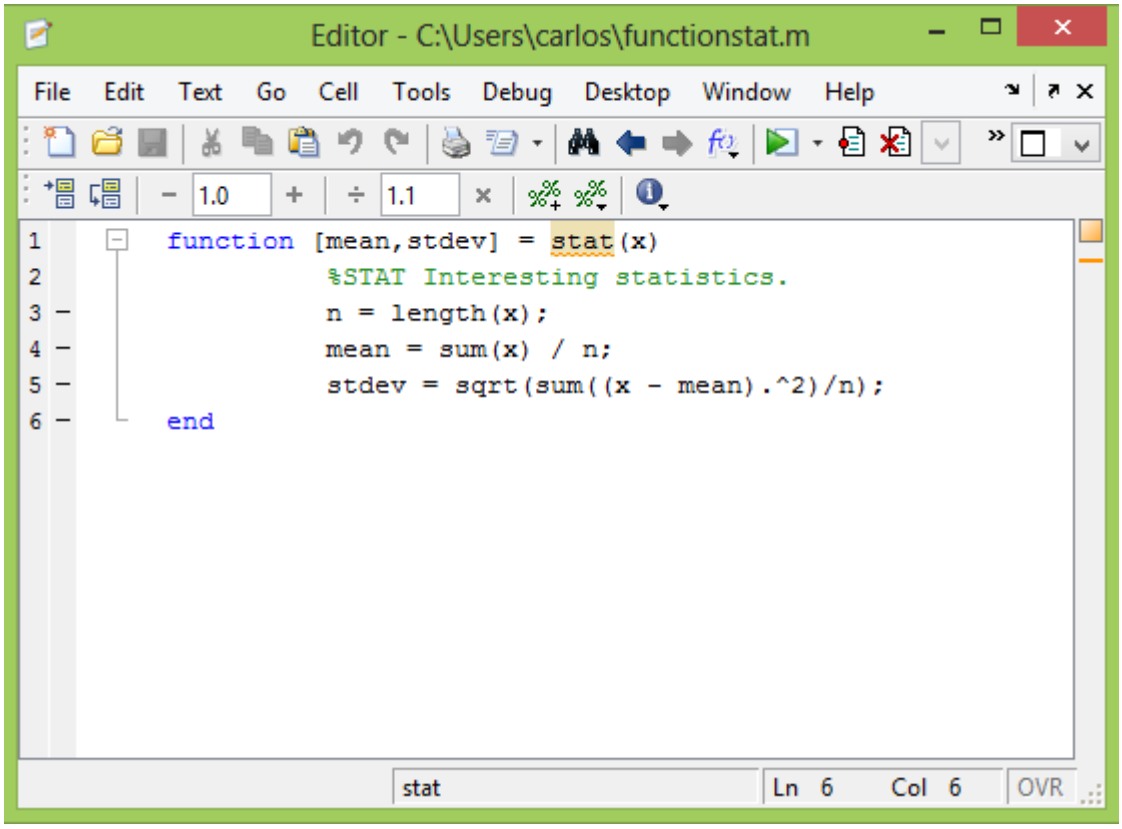


Figure 1.20.

The functions are a very useful tool that MATLAB possesses. By creating a function in a .m file, there is the possibility of recalling the code that has been made, and run the function with different inputs in order to obtain a new set of data that can be compared to different ones obtained by using another set of inputs. This feature makes MATLAB a very useful tool in very different applications.

2.4.Scripts

Scripts are files which do not accept input arguments on return output arguments. They operate on data in the workspace. Scripts are probably the most emblematic in MATLAB, and the most useful way to write a script is through a .m file.

- **.m files:**

The so mentioned .m files are text files with no format that constitute the centre of the MATLAB programming. This files can be created and modified with any text editor, but the most common is using the own MATLAB's text editor, which in the image below you can see how to access it.

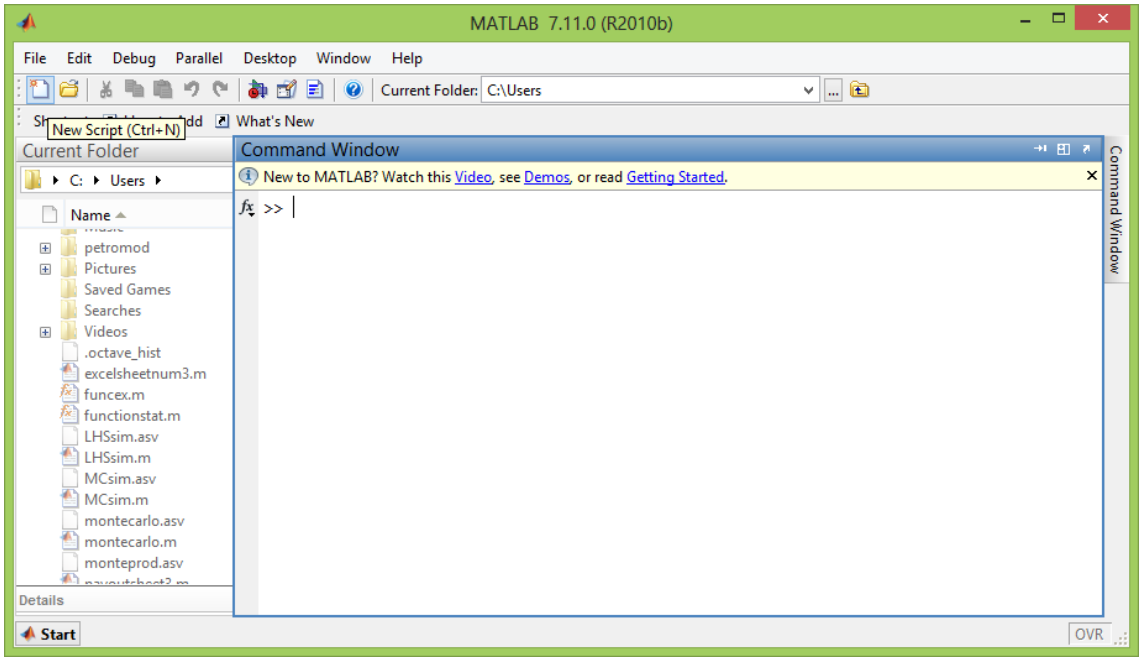


Figure 1.21.

One of the most important features that a .m file has is that it can call other .m files and even recall itself, which gives you a large number of possibilities of programming.

The scripts have a string of commands similar to the one that would be typed in the command window. These commands are run when the name of the file is typed, and the results and the parameters are stored in the workspace.

To summarize, the practical characteristic of a script is that there is the possibility of storing a series of commands as big as desired in a .m file that can be modified and run as many times as needed.

2.5. 2-D and 3-D plots

- **Two-dimensional graphs**

The 2-D graphs of MATLAB are fundamentally oriented to represent vectors and matrices. MATLAB uses a special type of windows to make graphic operations. Depending on the command used, some open a new graph window and other draw over an active one, either replacing the graph on it o adding new elements.

MATLAB possesses four basic functions in order to create two-dimensional graphs. This functions differ from each other in the scale that use in their axis.

- plot ( ): makes a graph from a vector and/or columns of matrices, with lineal scale in both axis.
- loglog ( ): same as plot but with a logarithmic scale in both axis.
- semilogx ( ): same as plot but with a lineal scale in the Y-axis and a logarithmic scale in the X-axis.
- semilogy ( ): same as plot but with a lineal scale in the X-axis and a logarithmic scale in the Y-axis.

The possibilities with the MATLAB graphs is very wide. MATLAB gives the opportunity to add graph titles, to label the X and Y axis, to introduce a legend or to activate a grid in the drawing.

As mentioned before, the key function in order to represent a 2-D graph in MATLAB is the 'plot' function. MATLAB can also define the color of the line, the type of line and the markers. The following picture illustrates an example of a 2-D graph using the plot function.

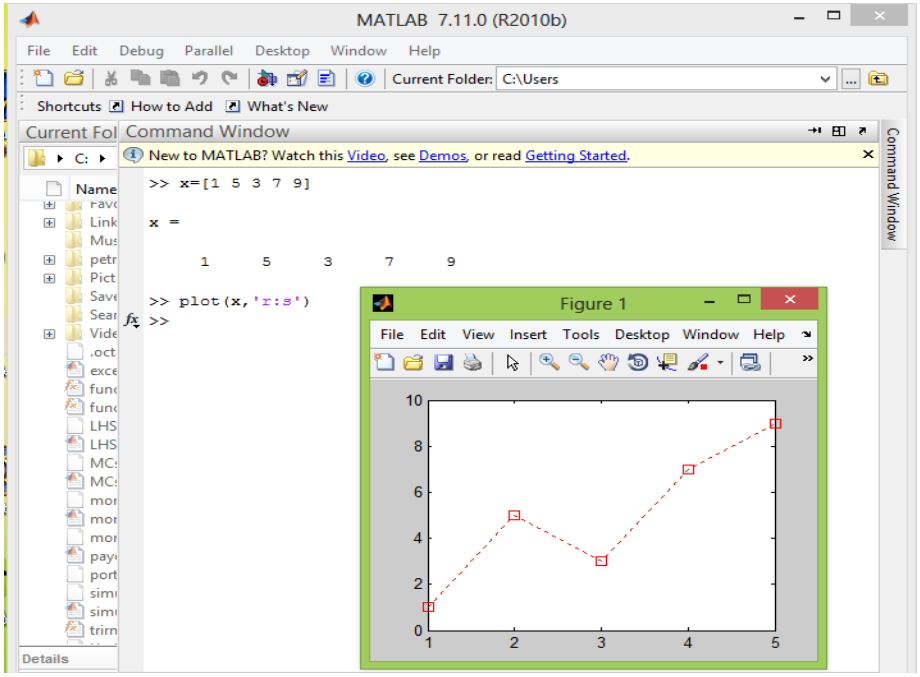


Figure 1.22.

A table of the possible combinations for the customization of the graph is also included here.

Type	Values	Meanings
Color	'c' 'm' 'y' 'r' 'g' 'b' 'w' 'k'	cyan magenta yellow red green blue white black
Line style	'-' '--' ':' '-.' no character	solid dashed dotted dash-dot no line
Marker type	'+' 'o' '*' 'x' 's' 'd' '^' 'v' '>' '<' 'p' 'h' no character	plus mark unfilled circle asterisk letter x filled square filled diamond filled upward triangle filled downward triangle filled right-pointing triangle filled left-pointing triangle filled pentagram filled hexagram no marker

Figure 1.23.

- **Three-dimensional graphs**

MATLAB includes the possibility of make different kind of 3-D graphs. The most useful function to create a 3-D graph is plot3 function, which is the tridimensional analog of the plot function. This function draws point which coordinates are stored in three vector. The following example draws an spiral line.

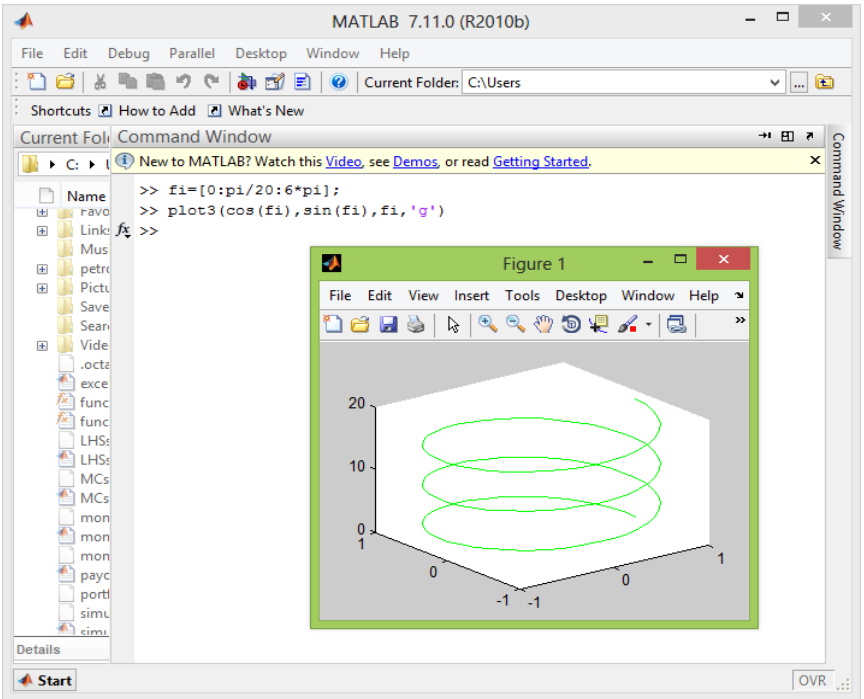


Figure 1.24.

As in the plot function, the plot3 function can also be customized within the line color, type of line and the marker type.

The three dimensional graph has multiples possibilities. The explanation of them would take too much time and for the purpose of this Thesis makes no sense to dig deep in to it, therefore a simple mention is just presented.

2.6. Using MATLAB for Risk Analysis.

MATLAB can be a very useful tool in order to use in risk analysis, since it has a very wide number of features that can be used ranging from statistical functions to graph design.

MATLAB has few limits in mathematic calculus and that means that it can solve a lot of the problems that may come with the risk analysis.

MATLAB has a very wide range of statistical functions. Some of the most important are the following.

- pdf: computes the probability density function for the one-parameter family of distributions specified by name. It can be used in so many statistical distributions like Beta, Normal or Uniform.
- cdf: returns the cumulative distribution function. As well as the pdf, it can be used in with so many statistical distributions like Beta, Normal or Lognormal
- random number generators: generates a random number from a wide range of statistical distributions. This function is very useful in order to obtain sampling for the risk analysis. e.g: betarnd, lognrnd, normrnd…
- stat: it gives the mean and variance of a very large number of statistical distributions. e.g: betastat, normstat, tstat… The name of the statistical value wanted form a vector with no statistical distribution can also be typed, therefore obtaining the mean ('mean(x)'), standard deviation ('std(x)'), mode ('mode(x)')…

A really helpful feature from MATLAB is the possibility of creating functions. This feature is key in risk analysis since if one statistical function does not exist in the MATLAB library it can be created and therefore be used as a common function.

The interactivity features that MATLAB possesses is also very useful in risk analysis. As we have seen before, MATLAB has multiple possibilities to show graphic results. It can show different graphs in the same window or in separate ones, it can tag the graphs including legends, it can change the axis range, it can zoom in or out in a given graph…

MATLAB also has a series of functions that represent statistical graphs, which are really important in risk analysis. Some of this specific graphs are:

- cdfplot: plot of empirical cumulative distribution function.
- normplot: Normal probability plot.
- hist: Histogram.

The use of matrices is also very useful in order to code a MATLAB program thought to be used in risk analysis. Loops need to be used in order to run a simulation in MATLAB, and the statement used to create a loop is 'for'. Since MATLAB uses matrices, there is the possibility of changing the size of the matrix en each iteration of the loop and therefore it is easy to change the number of iterations for the simulation. This characteristic helps to compare different results from simulations with different number of iterations.

2.7.Quick review of the program.

MATLAB is a high-level language and interactive environment for numerical computation, visualization, and programming. By using MATLAB, analyzing data, developing algorithms, and creating models and application is possible without the addition of any other program. The language, tool, and built-in functions enable the user to explore multiple approaches and reach a solution faster than with spreadsheets or traditional programming languages, such as C/C++ or Java.

For the purpose of this Thesis, it makes total sense to use a powerful calculation and useful visualization tool to represent the multiples possibilities and ranges into the risk analysis in the petroleum and oil industry.

Case of study

1. Description.

The purpose of this Thesis is to compare the efficiency, ease of implementation, the display of results as well as many other features between @RISK and MATLAB in the risk analysis applied to the petroleum and oil industry. Difference and similarities, as well as specific features, will be illustrated and discussed through an example case.

The example case is an off-shore fault-block near to a existing field. The original 3D seismic interpretations that was used for the main field developments shows a strong bright spot in the fault-block, located in the same formation as the current reservoir. Data from the current reservoir can be used to estimate the properties of the opportunity, which can be exploited by drilling a single step-out well, with tie-back to the existing facilities. Although the probability of failure can be ignored, there is considerable uncertainty around resource volumes, reserves and economics.

The implementation will give back a couple of results that will be very useful to determine the characteristics of the oil well. This characteristics are the reserves, the abandonment year, the production, the revenue, the NPV... With the estimation of this parameters the evaluation of the oil well will be easier and the decisions that would have to be made will be helped by the model to see if it fullfills the goals of the company.

The implementation has three components:

- **Original oil in place (OOIP) and reserves model**

Using the data given, the objective is to obtain the Estimated Reserves through the Original Oil In Place equation.

$$\text{Estimated Reserves} = [A.t.\Phi.N{:}G.(1\text{-}S_w)/b_{oi}] * RF$$

- **Production model**

Generate a production profile using a tank model, starting at year 0 when the well is drilled and tied back.

The tank model describes a reservoir without spatial variations. Due to the homogeneity, the locations of the production wells are of no consequence for the productions. As a results is theoretically possible to deplete the reservoir from one single well, and the tank model can conveniently be conceived of as a ball filled with oil, where each production well is a straw that can be used to suck up the entire volume.

- **Simple before-tax economics model**

With all the results obtained in the two steps before this, the objective is to obtain the expected NPV of the oil field. In order to do so it is important to take into account yearly expenses as well as oil price variation and discount rate.

After these three models have been set up, there are a couple of aspects that it can be used for.

2. @RISK implementation.

After briefly describing the case, it is time to set the simulation in order to run it in @RISK. This implementations has three worksheets, one for each of the steps of the step-out well model implementation.

- **OOIP and Reserves model.**

In the first sheet, the one named OOIP is where all the data to calculate the Estimated reserves is given. In Figure 2.1 this data is shown.

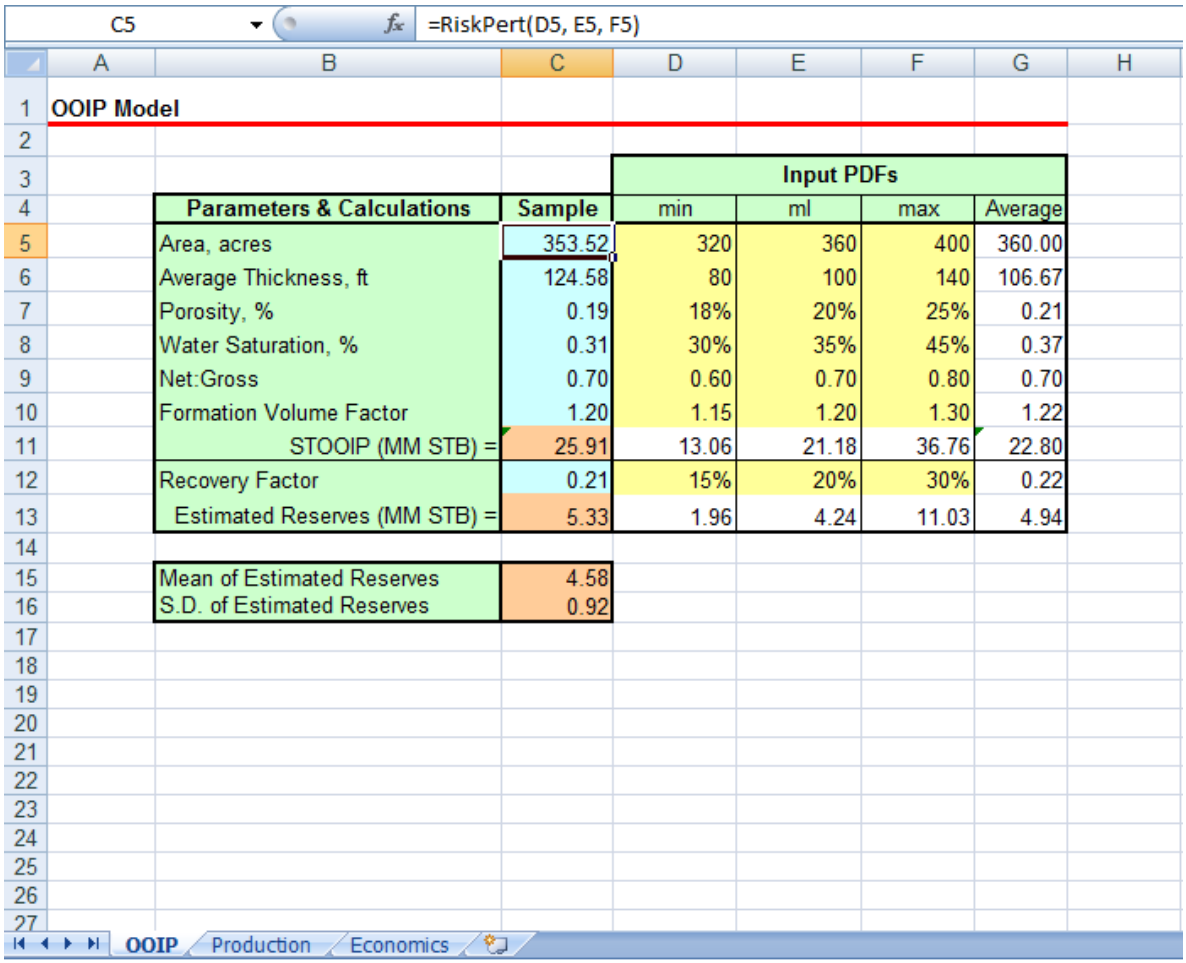


Figure 2.1.

There are six input parameters: Area, Average Thickness, Porosity, Water Saturation, Net Gross ratio, Formation Volume factor and Recovery Factor.

Three values are given for these parameters, the minimum value, the most likely value and the maximum value, and these three values are used to specify the PERT distribution (it can be seen in the formula bar as a description of cell C5).

The PERT distribution is a version of the Beta distribution, and requires the same three parameters as the Triangular distributions (minimum, most likely and maximum). The relation between the Beta distribution and the PERT distribution is the following:

$$\text{PERT}(a, b, c) = \text{Beta}(\alpha_1, \alpha_2) * (c - a) + a$$

Where:

$$\alpha_1 = \frac{(\mu - a) * (2b - a - c)}{(b - \mu) * (c - a)} \qquad \alpha_2 = \frac{\alpha_1 * (c - \mu)}{(\mu - a)}$$

One feature of the PERT distribution in opposition to the Triangular distribution is that the mean is four times more sensitive to the most likely value than to the minimum and maximum value, whether in the Triangular distribution the mean is equally sensitive to each parameter. This can be seen in the formula to calculate the PERT mean.

$$\mu = \frac{a + 4 * b + c}{6}$$

The standard deviation of a PERT distribution is also less sensitive to the estimate of the extremes.

The choice of the PERT distribution is very clear in order to give more importance to the most likely value to later run the simulation.

Following the OOIP sheet explanation, the Sample column will show the result of a random sample of the PERT distribution with the given parameters.

In this sheet there are two output values, STOOIP and Estimated Reserves. The STOOIP is calculated with the formula shown in Figure 2.2.

Figure 2.2.

The formula has the transformation of the parameters in to the International System of Units. The D11, E11 and F11 cells are calculated with the same formula but using the minimum, most likely and maximum values consequently.

The Estimated reserves are calculated with formula shown in Figure 2.3.

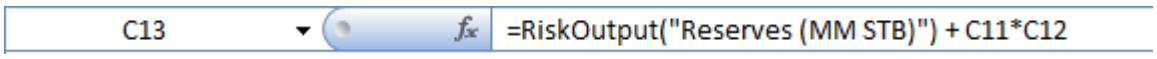


Figure 2.3.

It can be seen that the formula simply multiplies the value of the STOOIP and the Recovery Factor.

It can also be seen in the picture, the Mean of Estimated Reserves and the Standard Deviation of the Estimated Reserves. This two values are calculated in the OOIP sheet with the RiskMean and RiskStdDev applied to the cell C13. These commands just give back the mean and the standard deviation from the values generated in the simulation.

- **Production model**

The second sheet is called Production. In this sheet production curve parameters and constraints are given and field life is also given . This sheet can be divided into three areas.

The first one is a table where the data is given. (Figure 2.4.)

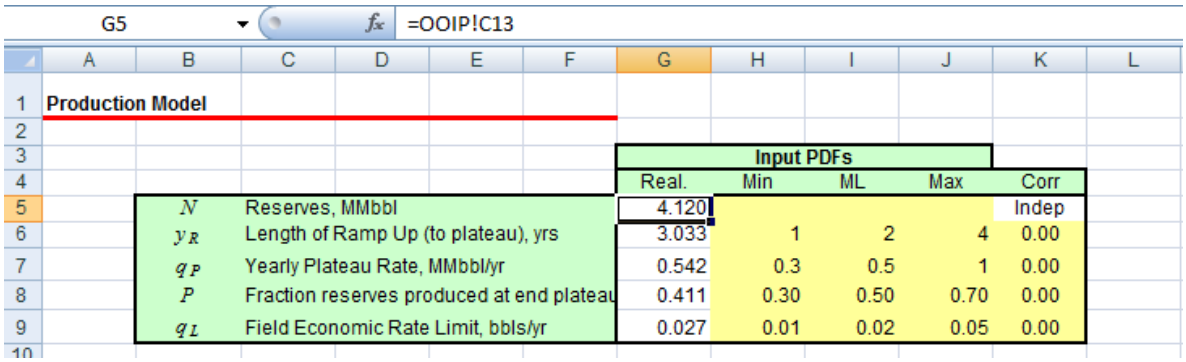


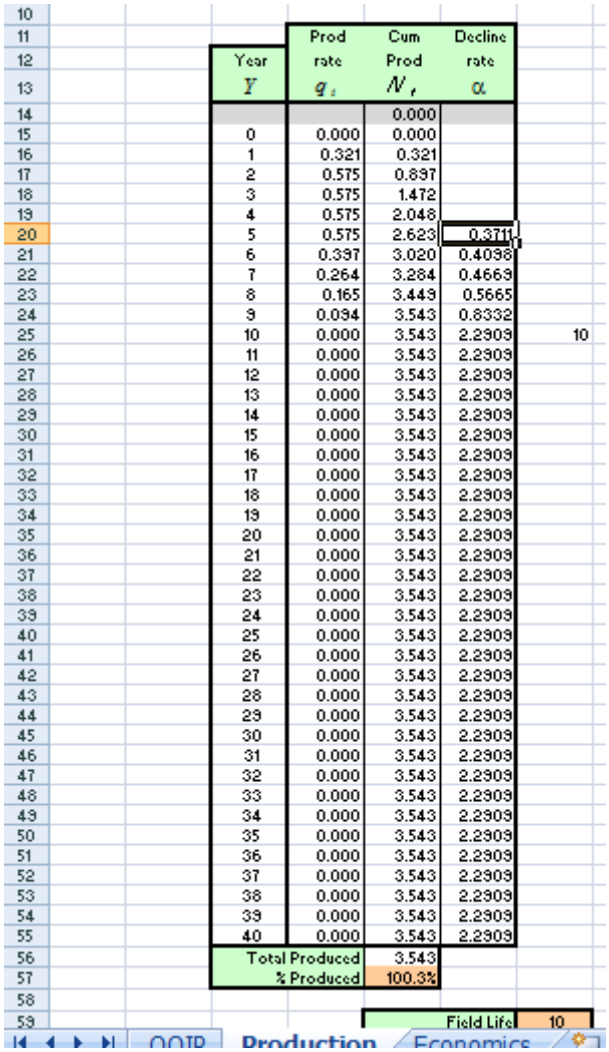
Figure 2.4.

The data given are Reserves, Length of Ramp Up to plateau, Yearly Plateau Rate, Fraction reserves produced at end of plateau and Field Economic Rate Limit.

As it can be seen in the picture, the Reserves are obtained from the results in the OOIP sheet. The other parameters also follow the PERT distribution.

The Corr column shows the correlation between variables, and they have been set as independent from one another, therefore the value given is a zero.

The next area of interest in this sheet is a table where the production year to year can be seen.(Figure 2.5.).



Year Y	Prod rate q_t	Cum Prod N_p	Decline rate α	
		0.000		
0	0.000	0.000		
1	0.321	0.321		
2	0.575	0.897		
3	0.575	1.472		
4	0.575	2.048		
5	0.575	2.623	0.3711	
6	0.397	3.020	0.4038	
7	0.264	3.284	0.4663	
8	0.165	3.449	0.5665	
9	0.094	3.543	0.8332	
10	0.000	3.543	2.2303	10
11	0.000	3.543	2.2303	
12	0.000	3.543	2.2303	
13	0.000	3.543	2.2303	
14	0.000	3.543	2.2303	
15	0.000	3.543	2.2303	
16	0.000	3.543	2.2303	
17	0.000	3.543	2.2303	
18	0.000	3.543	2.2303	
19	0.000	3.543	2.2303	
20	0.000	3.543	2.2303	
21	0.000	3.543	2.2303	
22	0.000	3.543	2.2303	
23	0.000	3.543	2.2303	
24	0.000	3.543	2.2303	
25	0.000	3.543	2.2303	
26	0.000	3.543	2.2303	
27	0.000	3.543	2.2303	
28	0.000	3.543	2.2303	
29	0.000	3.543	2.2303	
30	0.000	3.543	2.2303	
31	0.000	3.543	2.2303	
32	0.000	3.543	2.2303	
33	0.000	3.543	2.2303	
34	0.000	3.543	2.2303	
35	0.000	3.543	2.2303	
36	0.000	3.543	2.2303	
37	0.000	3.543	2.2303	
38	0.000	3.543	2.2303	
39	0.000	3.543	2.2303	
40	0.000	3.543	2.2303	
Total Produced		3.543		
% Produced		100.3%		

Field Life 10

Figure 2.5.

This table is filled following a series of formulas to determine the production rate year to year.

The column Year signals the year when a given production is going to be had. This is the simplest one to understand. The other three require a little bit more of effort.

The first column that we should look at is de column named Decline Rate (α). The Decline Rate it is defined as the negative relative change of production over a time period. The formula is given in Figure 2.6.:

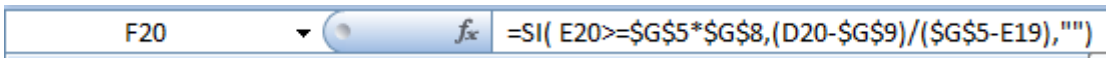


Figure 2.6.

We can see that this formula follows a conditional statement that would only apply if the column E (Cumulative production) is bigger than the Reserves multiplied by the Fraction of reserves produced at end of plateau. In our case, the decline rate will start when the

reserves that have been produced exceed the fraction of reserves produced at end of plateau. When that situation has been reached, the decline rate is given by the following formula:

$$\alpha = \frac{q_t - q_l}{N - N_{t-1}},$$

where q_t is the production rate in year t, q_l is the field economic rate limit, N are the Estimated Reserves and N_{t-1} are the cumulative production until year t.

The following column is the Cumulative Production, the one in the middle. This is probably the most simple one, since it only adds the production of each year to the production the years before. The formula can be seen in Figure 2.7.

Figure 2.7.

The last column is the Production rate. This is the most complicated formula since is the one that gives the values of the production rate each year. The production rate in this model follows the plateau model, in which we have three areas: ramp up, plateau and decline, and the column has to account for that in order to be correct. The formula is given in Figure 2.8.:

Figure 2.8.

This formula is once again an if statement, and each of the conditions evaluated must follow the plateau model named before.

If the year we are in is smaller than the length of the ramp up, it means that we are still in the ramp up phase and therefore, the production rate will follow this formula:

$$q_t = \frac{t * q_p}{Y_r};$$

where q_t is the production rate, t is the current year, q_p is the production rate at plateau and Y_r is the length of the ramp up phase.

If the statement mentioned before is false, it means that the stage is either plateau or decline. If the stage is plateau, the Cumulative Production in year t is smaller than the Estimated Reserves multiplied by the Fraction reserves produced at end of plateau, and therefore, the value of q_t will be q_p.

If this is not true either, the decline stage is the one left. The possibility of the well to be out of reserves must be taken into consideration, therefore the right way to put the statement is to make a new statement into the negation of the statement explained before. Notice that in this case, the statement if starts with N_t<N, but since this is the

value for the false statement mentioned before, this N_t will always be greater than N*P. If we are in this situation, then the sheet will execute the following formula:

$$q_t = q_{t-1} * e^{-\alpha},$$

where q_t is the production rate on the year t, q_{t-1} is the production year the year after and α is the decline rate for the year t.

If none of the statements mentioned before are true, then the production rate will be 0, which will mean that the well is out of reserves.

It is worth mentioning that the table commented during this last couple of paragraphs has a 'hidden' column, which is situated next to the decline rate column. This column has the following formula (Figure 2.9.).

Figure 2.9

This column is an if statement that ask to print the year when the production rate is zero only if the year after it was greater than zero. By doing this we obtain the year where the decline rate has hit zero and, therefore, the field life has come to an end.

After this has been explained it is time to explain the output cells in this sheet. One of them is the %produced (cell E57) and the other is the field life (cell G59).

The %produced is the ratio between the Estimated reserves calculated in the OOIP sheet and the Total oil produced during the field life. The reason why the %produced can be more than 100% it is because the program does not take into consideration the possibility of finishing the production before the year has ended, and thus, if the production in reality has finished when half the year has passed , the table in this sheet would keep calculating until the end of the year.

The field life is obtained from the 'hidden' column. As it can be seen, this column only has one value, which is represented in its corresponding cell.

Finally, the third and last area of this sheet is the representation of the production rate and the years, which is shown in Figure 2.10.

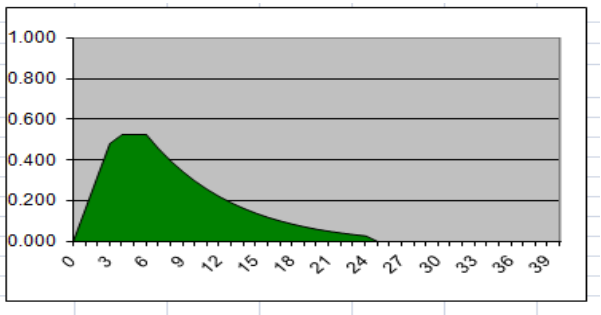


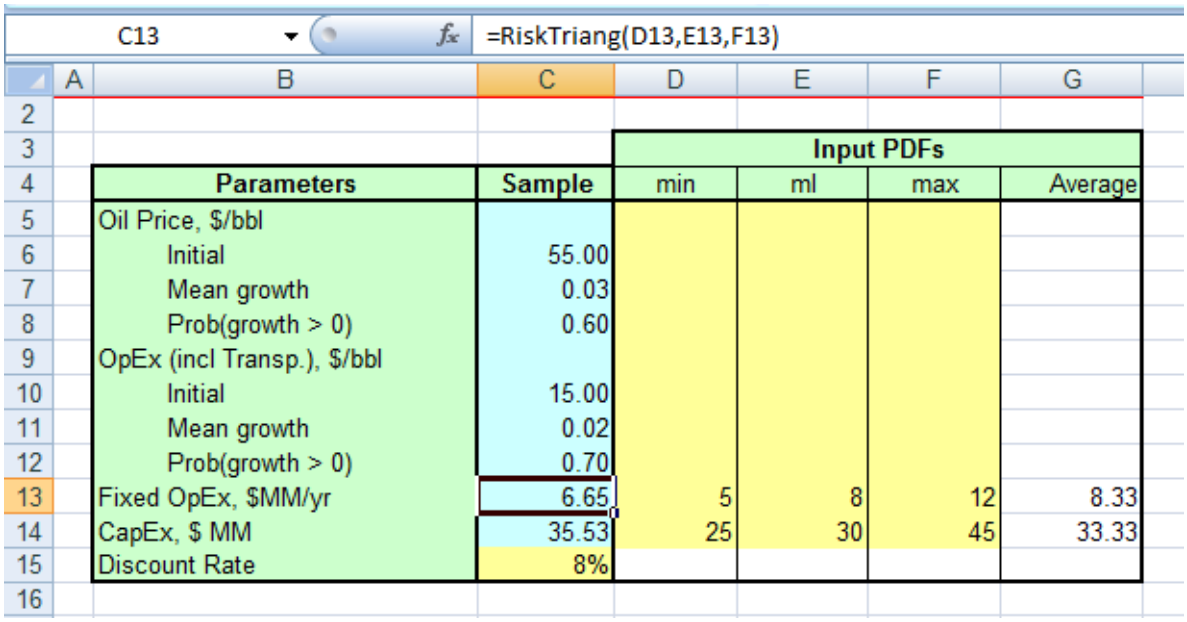
Figure 2.10

This graph is obtained by plotting the Year column and the Production rate column. It shows perfectly the three stages named before: the ramp up, a couple of year of plateau where the production rate is at its peak, and the decline rate, where the production is decreasing year by year until the end of the field life.

- **Before-tax economic model.**

This is the last step of the implementation before running the simulation. This sheet is called Economics and as well as the Production one, you can subdivided in different areas, in this case there are two significant areas.

The first one is where the input values are given. Figure 2.11. shows how it looks like:



	Parameters	Sample	Input PDFs min	ml	max	Average
5	Oil Price, $/bbl					
6	Initial	55.00				
7	Mean growth	0.03				
8	Prob(growth > 0)	0.60				
9	OpEx (incl Transp.), $/bbl					
10	Initial	15.00				
11	Mean growth	0.02				
12	Prob(growth > 0)	0.70				
13	Fixed OpEx, $MM/yr	6.65	5	8	12	8.33
14	CapEx, $ MM	35.53	25	30	45	33.33
15	Discount Rate	8%				

Figure 2.11.

A series of important parameter can be seen in this table.

First there is the oil price. It has three components:

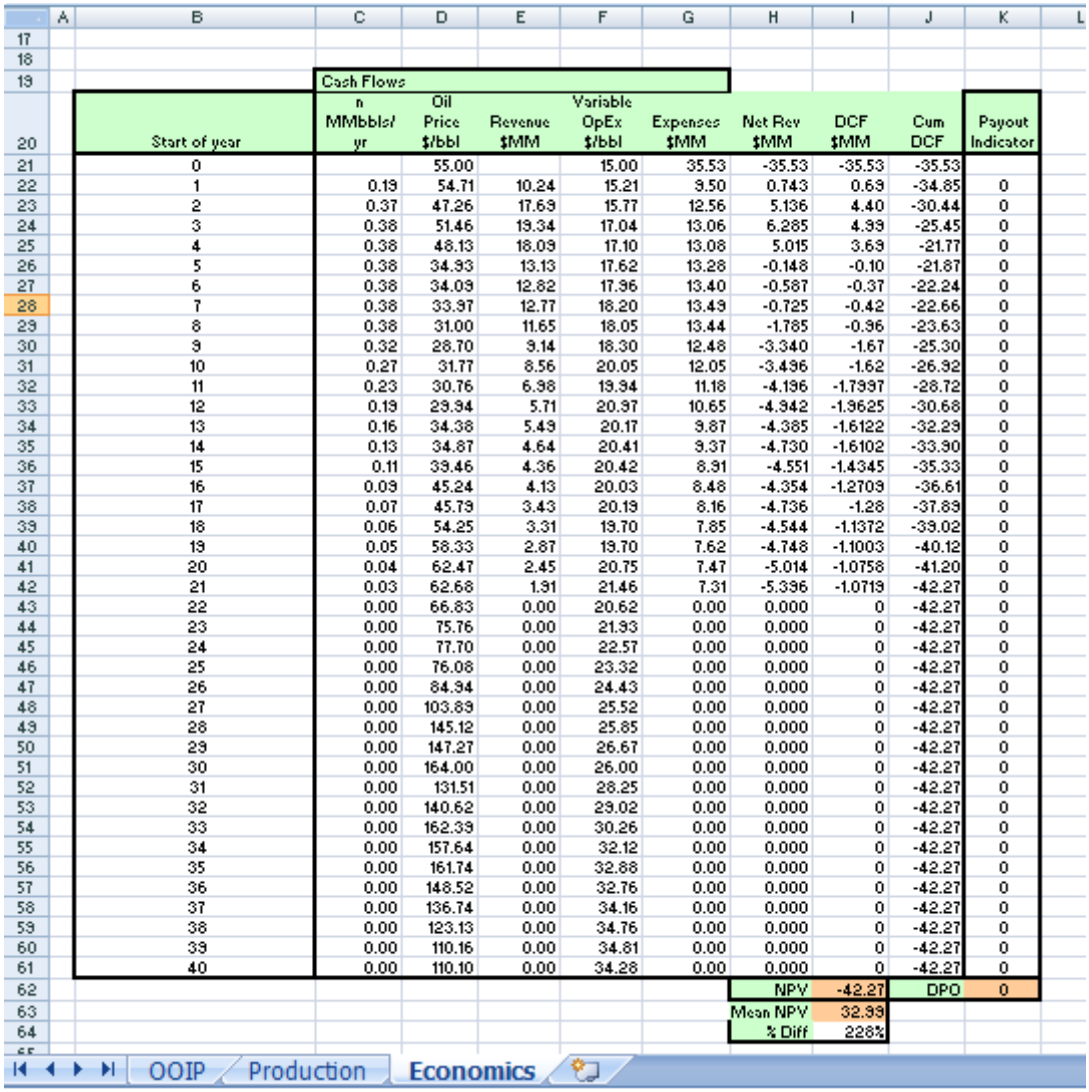
- Initial: it is the price set the year that we are in.
- Mean growth: it is the average price raise that the oil price is going to suffer each year.
- Probability of growth: it is the probability that the oil price increase its value each year.

It can be seen that the next step of values follow the same pattern, this time referring to the Operational Expenditures.

The next to values are the Fixed Operational Expenditures and the Capital Expenditures. This two values follow a statistical distribution, but this time it is not the mentioned PERT distribution but a regular Triangular distribution.

Finally, a discount rate of 8% is the one used in order to calculate the cash flows.

After this table it comes a new one called Cash Flows, which is shown in Figure 2.12.



	Start of year	n MMbbls/ yr	Oil Price $/bbl	Revenue $MM	Variable OpEx $/bbl	Expenses $MM	Net Rev $MM	DCF $MM	Cum DCF	Payout Indicator
		Cash Flows								
21	0		55.00		15.00	35.53	-35.53	-35.53	-35.53	
22	1	0.13	54.71	10.24	15.21	3.50	0.743	0.63	-34.85	0
23	2	0.37	47.26	17.69	15.77	12.56	5.136	4.40	-30.44	0
24	3	0.38	51.46	19.34	17.04	13.06	6.285	4.99	-25.45	0
25	4	0.38	48.13	18.09	17.10	13.08	5.015	3.69	-21.77	0
26	5	0.38	34.93	13.13	17.62	13.28	-0.148	-0.10	-21.87	0
27	6	0.38	34.09	12.82	17.96	13.40	-0.587	-0.37	-22.24	0
28	7	0.38	33.97	12.77	18.20	13.49	-0.725	-0.42	-22.66	0
29	8	0.38	31.00	11.65	18.05	13.44	-1.785	-0.96	-23.63	0
30	9	0.32	28.70	9.14	18.30	12.48	-3.340	-1.67	-25.30	0
31	10	0.27	31.77	8.56	20.05	12.05	-3.496	-1.62	-26.92	0
32	11	0.23	30.76	6.98	19.94	11.18	-4.196	-1.7997	-28.72	0
33	12	0.19	29.94	5.71	20.97	10.65	-4.942	-1.9625	-30.68	0
34	13	0.16	34.38	5.49	20.17	9.87	-4.385	-1.6122	-32.29	0
35	14	0.13	34.87	4.64	20.41	9.37	-4.730	-1.6102	-33.90	0
36	15	0.11	39.46	4.36	20.42	8.91	-4.551	-1.4345	-35.33	0
37	16	0.09	45.24	4.13	20.03	8.48	-4.354	-1.2709	-36.61	0
38	17	0.07	45.79	3.43	20.19	8.16	-4.736	-1.28	-37.89	0
39	18	0.06	54.25	3.31	19.70	7.85	-4.544	-1.1372	-39.02	0
40	19	0.05	58.33	2.87	19.70	7.62	-4.748	-1.1003	-40.12	0
41	20	0.04	62.47	2.45	20.75	7.47	-5.014	-1.0758	-41.20	0
42	21	0.03	62.68	1.31	21.46	7.31	-5.396	-1.0719	-42.27	0
43	22	0.00	66.83	0.00	20.62	0.00	0.000	0	-42.27	0
44	23	0.00	75.76	0.00	21.93	0.00	0.000	0	-42.27	0
45	24	0.00	77.70	0.00	22.57	0.00	0.000	0	-42.27	0
46	25	0.00	76.08	0.00	23.32	0.00	0.000	0	-42.27	0
47	26	0.00	84.94	0.00	24.43	0.00	0.000	0	-42.27	0
48	27	0.00	103.89	0.00	25.52	0.00	0.000	0	-42.27	0
49	28	0.00	145.12	0.00	25.85	0.00	0.000	0	-42.27	0
50	29	0.00	147.27	0.00	26.67	0.00	0.000	0	-42.27	0
51	30	0.00	164.00	0.00	26.00	0.00	0.000	0	-42.27	0
52	31	0.00	131.51	0.00	28.25	0.00	0.000	0	-42.27	0
53	32	0.00	140.62	0.00	29.02	0.00	0.000	0	-42.27	0
54	33	0.00	162.39	0.00	30.26	0.00	0.000	0	-42.27	0
55	34	0.00	157.64	0.00	32.12	0.00	0.000	0	-42.27	0
56	35	0.00	161.74	0.00	32.88	0.00	0.000	0	-42.27	0
57	36	0.00	148.52	0.00	32.76	0.00	0.000	0	-42.27	0
58	37	0.00	136.74	0.00	34.16	0.00	0.000	0	-42.27	0
59	38	0.00	123.13	0.00	34.76	0.00	0.000	0	-42.27	0
60	39	0.00	110.16	0.00	34.81	0.00	0.000	0	-42.27	0
61	40	0.00	110.10	0.00	34.28	0.00	0.000	0	-42.27	0
62							NPV	-42.27	DPO	0
63							Mean NPV	32.33		
64							% Diff	228%		

Figure 2.12.

The first column is the Production, which recalls the production rate from the Production sheet.

The second column is the oil price. The formula is shown in Figure 2.13.

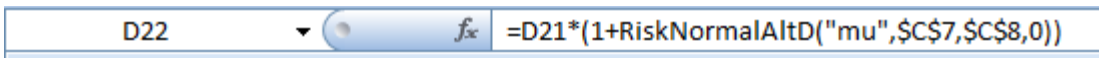


Figure 2.13

The RiskNormalAltD specifies a normal distribution with a descending percentile, where the mean is the Mean Growth mentioned previously, and where there is a 60% probability of having a 0 percent chance of the oil price to go down.

The revenue column is just the multiplication of the production and the oil price each year.

The fourth column represents the Variable Operational Expenditures. This column is very similar to the Oil price column as it can be seen in Figure 2.14.

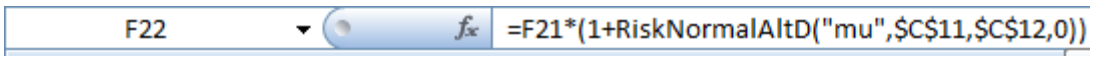


Figure 2.14.

The next column represents the Expenses. The formula for this column is shown in Figure 2.15.

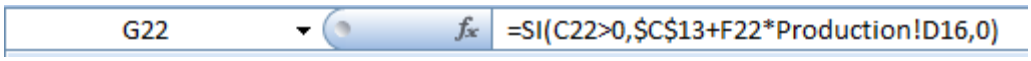


Figure 2.15.

This formula is a conditional statement. It says that when the production rate is greater than zero, the expenses will be the Fixed OpEx named before, and the value of the Variable OpEx multiplied by the production rate. By making this a conditional statement, it assures the user that when the production rate is zero, the Expenses will be also zero, which facilitates the calculation for the following columns.

After the Expenses column there is the Net Revenue, which it only subtracts the Expenses to the Revenue.

The next column is the discounted cash flow, where the discount rate must be taken into account. The formula is the following:

$$DCF = \frac{NetRevenue}{(1+d)^t},$$

where d is the discount rate and t is the current year.

Next to the DCF column is the Cumulative DCF, where the value for each year is added to the value of the years before. At the end of this column there is the output value, which is the Net Present Value.

The payout column is a similar column to the one named in the Production sheet called as the 'hidden' column. It is just a way to signal when the Cumulative DCF has reached a positive value, and therefore, it signals the year when the investments has paid itself back.

There are two more quick values that are worth to mention. One is the Mean NPV, which just gives the mean value of the NPV after a simulation. The other is the % differential between the mean and the NPV value in the table, which can vary a lot depending on the simulation run that is shown in the @RISK sheet.

After all this has been set, the @RISK sheet is ready to perform the simulation. The most important results that will be obtained and that will be commented in the results and conclusions section will be:

- Standard Original Oil In Place
- Estimated reserves
- Abandonment Year
- Net Present Value

3. MATLAB implementation.

This chapter discuss the MATLAB implementation of the model. The MATLAB code consist in almost 300 lines of commands that will be explained in depth.

The first 70 lines describes the procedure in the first @RISK sheet, the one called OOIP. The code is explained in the following:

```
%Monte Carlo simulation of Excel sheet #1

n=10000;

%area
a=300;b=360;c=400;
mu=(a+4*b+c)/6;
alfa1=(mu-a)*(2*b-a-c)/((b-mu)*(c-a));
alfa2=alfa1*(c-mu)/(mu-a);
y=betarnd(alfa1,alfa2,[n,1])
area=a+(c-a)*y;
```

The n represents the number of simulations that MATLAB will run 10,000 is the initial number.

The next lines represent the PERT distribution for the area. The parameters a, b and c represent respectively the minimum, most likely and maximum values.

The next line is the mean in the PERT distribution and after that there are the two alpha values that are needed in the PERT distribution. Notice that the minimum value set for the area in the MATLAB code is 300 instead of 320. This is because if the value was set at 320, the mean value would be the same as the most likely value and that will result that the alfa1 parameter would be infinite and the simulation could not be run.

The y parameter represents a set of random values from the PERT distribution that has been described in the lines before that. The result in the workspace will be a matrix with 1 column and n rows with its values between 0 and 1.

Finally, the last line of the code represents the value of the area for the whole simulation space. What it does is re-escalate the beta function and the result is a matrix with one column and n rows with its values between the maximum and the minimum given.

The other parameters follow the same code pattern:

```
%thickness
a1=80;b1=100;c1=140;
mu1=(a1+4*b1+c1)/6;
alfa11=(mu1-a1)*(2*b1-a1-c1)/((b1-mu1)*(c1-a1));
alfa12=alfa11*(c1-mu1)/(mu1-a1);
y1=betarnd(alfa11,alfa12,[n,1]);

thickness=a1+(c1-a1)*y1;

%porosity
a2=0.18;b2=0.2;c2=0.25;
mu2=(a2+4*b2+c2)/6;
alfa21=(mu2-a2)*(2*b2-a2-c2)/((b2-mu2)*(c2-a2));
alfa22=alfa21*(c2-mu2)/(mu2-a2);
y2=betarnd(alfa21,alfa22,[n,1]);
porosity=a2+(c2-a2)*y2;

%Water Saturation
a3=0.3;b3=0.35;c3=0.45;
mu3=(a3+4*b3+c3)/6;
alfa31=(mu3-a3)*(2*b3-a3-c3)/((b3-mu3)*(c3-a3));
alfa32=alfa31*(c3-mu3)/(mu3-a3);
y3=betarnd(alfa31,alfa32,[n,1]);

WS=a3+(c3-a3)*y3;

%NetGross
a4=0.5;b4=0.7;c4=0.8;
mu4=(a4+4*b4+c4)/6;
alfa41=(mu4-a4)*(2*b4-a4-c4)/((b4-mu4)*(c4-a4));
alfa42=alfa41*(c4-mu4)/(mu4-a4);
y4=betarnd(alfa41,alfa42,[n,1]);
NG=a4+(c4-a4)*y4;

%Volume Factor
a5=1.15;b5=1.2;c5=1.3;
mu5=(a5+4*b5+c5)/6;
alfa51=(mu5-a5)*(2*b5-a5-c5)/((b5-mu5)*(c5-a5));
alfa52=alfa51*(c5-mu5)/(mu5-a5);
y5=betarnd(alfa51,alfa52,[n,1]);

VF=a5+(c5-a5)*y5;

%Recovery Factor
a6=0.15;b6=0.2;c6=0.3;
mu6=(a6+4*b6+c6)/6;
alfa61=(mu6-a6)*(2*b6-a6-c6)/((b6-mu6)*(c6-a6));
alfa62=alfa61*(c6-mu6)/(mu6-a6);
y6=betarnd(alfa61,alfa62,[n,1]);
```

The result of this are seven matrices with one column and n rows. One for each of the parameters needed for the calculation of the Estimated Reserves and the Standard Original Oil In Place. Notice that in the Net Gross ratio the minimum value had to be changed for the same reasons as the Area.

MATLAB has the possibility of importing the Excel data by using the xlsread command. This would give an array with all the input values that have been set in Excel sheet. Finally, this option was considered but rejected since it would not save as many lines in the code (the PERT distribution calculation had to be done anyways for each of the parameters) and would also have the data as a simple array, which could complicate the definition of future parameters.

The next step is to code the calculation of the STOOIP and the Estimated Reserves:

```
%STOOIP formula
STOOIP=((area*43560.*thickness/5.6148)/1000000).*porosity.*...
...((ones(1,n)')-WS).*NG./VF;

%Estimated reserves
EstReserves= STOOIP.*RF;
```

The STOOIP formula is the one described in the Introduction section. Notice the constant values in order to change the area and the thickness into the International System (originally the area is set in acres and the thickness in feet).

With this code, the OOIP model is implemented in MATLAB. The code is really simple for now, and the only 'irregular' thing is the introduction of the PERT distribution, because it does not exist in MATLAB, but, since the environment is very flexible it allows the user to implement the code for any distribution.

The codification for the Production sheet resulted to be much more challenging than the first one. This code needs to work with a few loops and complex matrices, which happens to make a complex code.

The code for this second sheet has approximately 100 lines:

```
%Calculation of the production EXCEL sheet#2

%Lenght of Ramp Up
a7=1;b7=2;c7=4;
mu7=(a7+4*b7+c7)/6;
alfa71=(mu7-a7)*(2*b7-a7-c7)/((b7-mu7)*(c7-a7));
alfa72=alfa71*(c7-mu7)/(mu7-a7);
y7=betarnd(alfa71,alfa72,[n,1]);

Yr=a7+(c7-a7)*y7;

%Yearly Plateau Rate
a8=0.3;b8=0.5;c8=1;
mu8=(a8+4*b8+c8)/6;
alfa81=(mu8-a8)*(2*b8-a8-c8)/((b8-mu8)*(c8-a8));
alfa82=alfa81*(c8-mu8)/(mu8-a8);
y8=betarnd(alfa81,alfa82,[n,1]);

Qp=a8+(c8-a8)*y8;

%Fraction reserves produced at end plateau
a9=0.2;b9=0.5;c9=0.7;
mu9=(a9+4*b9+c9)/6;
alfa91=(mu9-a9)*(2*b9-a9-c9)/((b9-mu9)*(c9-a9));
alfa92=alfa91*(c9-mu9)/(mu9-a9);
y9=betarnd(alfa91,alfa92,[n,1]);

P=a9+(c9-a9)*y9;

%Field economic rate limit
a10=0.01;b10=0.02;c10=0.05;
mu10=(a10+4*b10+c10)/6;
alfa101=(mu10-a10)*(2*b10-a10-c10)/((b10-mu10)*(c10-a10));
alfa102=alfa101*(c10-mu10)/(mu10-a10);
y10=betarnd(alfa101,alfa102,[n,1]);

Ql=a10+(c10-a10)*y10;
```

The beginning is similar to the code in the first sheet. They are the calculation of the parameters for the second sheet following the PERT distribution. Again, notice that the minimum value for the Fraction reserves produced at end of plateau has been changed to 0.2 instead of 0.3. The reasoning is the same as the one explained with the Area and the Net Gross ratio so the alfa91 parameter will not be zero.

After this, we have all the parameters as matrices with one column and n rows. Before the codification of the loop that will result in the calculation of the production rate table of the second sheet, it is necessary to introduce the following lines.

```
Year= 1:40;
Qtvec=zeros(n,length(Year));
alfapvec=zeros(n,length(Year));
Qt=zeros(n,length(Year));
Nt=zeros(n,1);
```

The purpose of the loop after this lines is to have a big vector that represents the production rate for each year and each simulation. This vector is the Qtvec. The reason you create a vector like this one with all zeros is that the loop will be substituting the zeros in the vector with the production rate for each year in each one of the simulations that are going to be. This same vector will be run over and over for each iteration and therefore it needs to have this huge dimensions so after the first iteration the second will not write over it.

The parameter Year is also a vector that has been set to 40, which is a high enough number to be sure that the field will not last to.

The same thinking put in to the Qtvec has been used in the alfapvec. The alfapvec is a vector where the decline rate will be stored for each iteration.

Qt and Nt are related to each other. Nt will represent the cumulative reserves and Qt will just store the production rate values for each of the iteration and add them into the cumulative reserves.

With no further addition, the following is the loop for the production rate, this parameters will explain themselves better if they are seen in action.

```
for i=1:n
   for ii=1:length(Year)
      if Nt(i)<EstReserves(i)*P(i);
         alfap(ii)=0;
      else alfap(ii)=(Qtvec(i,ii-1)-Ql(i))/(EstReserves(i)-Nt(i));
      end
       alfapvec(i,ii)=alfap(ii);
      if ii>Yr(i)
         if Nt(i)<EstReserves(i)*P(i);
            Qt(i,ii)=Qp(i);
         elseif Nt(i)>EstReserves(i)*P(i);
            Qt(i,ii)=Qtvec(i,ii-1)*exp(-alfapvec(i,ii-1));
         end
      else Qt(i,ii)=(ii)*(Qp(i))/Yr(i);

      end
      Nt(i)=Nt(i)+Qt(i,ii);
      Qtvec(i,ii)=Qt(i,ii);
      if Nt(i)>=EstReserves(i)
         break
      end
   end


end
```

The loop works with matrices that have n rows (one for each iteration) and 40 columns (one for each year). The way is set, first iterates for the first iteration, then for the first year of the first iteration and so on until the 40 years. Then iterates for the first year of the second iteration, and repeats the same procedure until the last year of the last iteration.

As it can be seen. This is a loop which has another loop inside. The first loop refers to each of the iterations needed in the simulation and the loop inside it solves for each of the years that each iteration has. The first two lines describe what it has been said.

After that, and inside the second loop, there is and if statement. The statement is only valid if the cumulative reserves are not greater that the Estimated Reserves times the Production rate at end of plateau, this refers to all of the stages before the decline. If this statement is set to be correct, the iteration will be in the decline stage, and therefore the decline rate would be calculated. After this is evaluated, the loop will store all of the alfap values into the alfapvec, which now will be a big vector with zeros in its columns and then the correspondent values of the decline rate when the iteration is in the stage of decline.

Following this, the code now identifies if the production profile is the ramp-up or after it (either plateau or decline). In order to set this, there is an if statement inside another if

statement. The first part of the if statement is that the year that it is being iterated is after the ramp up. There are two possibilities then: Plateau or Decline, therefore the existence of a new if statement to account for those possibilities. If the year that it is being iterated is during the ramp up, then it solves for it.

After doing this, the values that have been calculated is stored in to the vectors that have been described before and the loop will stop itself when the accumulated reserves (Nt) are greater than the Est. Reserves.

The next lines in this sheet are set to know which is the abandonment year for each simulation.

```
Aby=zeros(n,2);

for k=1:n
test=find(Qtvec(k,:));

Aby(k,:)=size(test);

end
```

The way the loop big loop is set, the Qtvec will have zeros when the field life is over. The purpose is to find when the Qtvec matrix changes the production to zero.

The first step is to create a matrix to evaluate the field life of each iteration, that is Aby. After this, a loop is run that will change create a new matrix called test, which will be a matrix with ones and zeros. This matrix will have a 1 in the position where the Qtvec has a nonzero value and a zero if the value is a zero. After this, the matrix Aby will print the size of each of the rows of the test matrix. The result is that the matrix Aby is a two column matrix, the first column are all ones while the second column will be the place where the test matrix is a zero, which coincides with the year where the production is zero, thus, the abandonment year.

After the two first sheets of the @RISK implementation have been coded in MATLAB, it is now time to explain the last sheet that was coded. This one presented a few changes worth to talk about.

The first lines are the following:

```
%Calculation of the excel sheet #3

%fixedopex
a12=5;b12=8;c12=12;
randfopex=trirnd(a12,b12,c12,n,1);

%capex
a13=25;b13=30;c13=45;
randcapex=trirnd(a13,b13,c13,n,1);

%Interest rate
d=0.08;

%oil price
ioilprice=55;
muop=0.03;
sdop=0.1184;

OPran=normrnd(muop,sdop,[n,length(Year)]);

%opex
iopex=15;
muopex=0.02;
sdopex=0.0381;

Opexran=normrnd(muopex,sdopex,[n,length(Year)]);
```

This code represent the calculation for the parameters needed for the third sheet. The difference now is that this parameters do not use the PERT distribution, and therefore a few changes had to be introduced.

The fixed opex and the capex work with a triangular distribution. Unfortunately, MATLBA does not have the possibility of obtaining a random number from a triangular distribution in its library. The solution to this problem was found in a code in the internet written by L. Cavin in 2003. This code is a function that gives a random number from a triangular distribution by putting the minimum value, the top value and the maximum value as inputs, and it generates a vector of random values.

The oil price and the opex present a different challenge in the MATLAB implementation. The RiskNormalAltD function does not exist in MATLAB, since the only way to describe a normal distribution is by having its mean and its standard deviation. Therefore, what it has been done is run the @RISK simulation and take the results of the mean and the standard deviation, and use those values to establish the

normal distribution for the oil price and the opex, and later obtain the random values from that distribution.

The interest rate was obviously the easiest one to set, since it is just a decimal number.

With the parameters already calculated, the next step is calculating all of the results in the table from the Economic sheet in the @RISK implementation.

First, the Oil price and the Revenue were calculated.

```
OP=zeros(n,length(Year));
for i=1:length(Year)
   if i==1
      OP(:,i)=ioilprice;
   else OP(:,i)=OP(:,i-1)+OP(:,i-1).*OPran(:,i);
   end

end

Revenue=Qtvec.*OP;
```

We define and oil price vector (OP) that will store the price for each year and each iteration. The first year the oil price is set at 55, and the year after it follows the formula given in the @RISK sheet.

The Revenue is just the multiplication of the Production rate of each year and the Oil price in its corresponding year.

After this, the Variable Operational Expenditures and the Expenses are calculated.

```
Vopex=zeros(n,length(Year));
for i=1:length(Year)
   if i==1
     Vopex(:,i)=iopex;
   else Vopex(:,i)=Vopex(:,i-1).*(1+Opexran(:,i));
   end

end
Expenses=zeros(n,length(Year));

meta=0<Qtvec;
for ii=1:length(Year)

      Expenses(:,ii)=meta(:,ii).*((randfopex)+(Vopex(:,ii)).*(Qtvec(:,ii)));
end
```

The Vopex follow the same pattern as the Oil Price. The Expenses follow a different pattern from the ones coded before.

First the Expenses vector is created to store the Expenses for each year. Then, a vector called meta is created, this vector is a matrix with a when in the place where the Qtvec has a nonzero value and a 0 where the Qtvec matrix has a zero. This will be used to obtain a matrix with the Fixed Opex that will be added with another matrix that represent the Vopex multiplied by the Production rate each year. The result of the Fixed Opex and the Variable Opex will be the Expenses.

Now that the expenses and the revenue has been calculated it is easy to calculate the Net Revenue.

NetRevenue=Revenue-Expenses;

With the Net Revenue calculated, the DCF and the NPV are the next and final steps in order to implement the third @RISK sheet.

```
DCF=zeros(n,length(Year));
for i=1:length(Year)

    DCF(:,i)=NetRevenue(:,i)./((1+d)^i);

end

NPV=zeros(n,length(Year));
for i=1:length(Year)
    if i==1
        NPV(:,i)=-randcapex;
    else
        NPV(:,i)=NPV(:,i-1)+DCF(:,i-1);
    end

  end
NPVvec=NPV(:,length(Year));


NPVvec1=NPVvec;
NPVvec1(NPVvec1>500)=[500]
```

The DCF is a simple loop that adjust the value of each of the Net Revenue amounts with the interest rate.

The NPV uses the capital expenditures as the value for the first year, and after that it adds the DCF for each year. This would give a vector that has the NPV in its last column, therefore, the creation of a vector called NPVvec is necessary. The NPVvec is a vector with one column and n rows that gives the NPV for each iteration.

The creation of a vector called NPVvec1 is necessary because for the comparison of results the mean and the standard variation is needed. The problem is that in some iterations, the NPV values skyrockets, making the mean to be to high because those few abnormal values (usually no more than 10 values have that behavior). Therefore, for the calculation of the mean and standard deviation the NPVvec1 is used. This vector is the same as the NPVvec, but when one of its values its bigger than 500 (which is way higher than P95) it stays as 500.

Now that the implementation is set, the next step is to represent the results in graphs. The code for the graphing is the following.

```
figure(1)
xRange1=0:60;
N1=hist(STOOIP,xRange1);
plot(xRange1,N1./numel(STOOIP));
title('PDF of STOOIP')
figure(2)
xRange2=0:25;
N2=hist(EstReserves,xRange2);
plot(xRange2,N2./numel(EstReserves));
title('PDF of Estimated Reserves')
figure(3)
cdfplot(STOOIP)
title('CDF of STOOIP')
figure(4)
cdfplot(EstReserves)
title('CDF of Estimated Reserves')
figure(5)
xRange3=0:50;
N3=hist(Aby(:,2),xRange3);
plot(xRange3,N3./numel(Aby(:,2)));
title('PDF of Abandonment Year')
figure(6)
cdfplot(Aby(:,2));
title('CDF of Abandonment Year')
figure(7)
xRange4=-200:400;
N4=hist(NPVvec,xRange4);
plot(xRange4,N4./numel(NPVvec));
title('PDF of NPV')
figure(8)
cdfplot(NPVvec);
axis([-200,+400,0,1]);
title ('CDF of NPV');
```

This code results on the display of eight different windows, one for each of the most meaningful results of the implementation. This are PDF and CDF graphs for the Standard Original Oil In place, Estimated Reserves, Abandonment Year and NPV. When the m-file is run MATLAB will pop-up those eight windows.

The last few lines of the code display the mean and the standard deviation of the results of the simulation. This results are sown in the command window since the code does not include any semicolon at the end of each line. The reason for that is because this result will be used to compare with the @RISK results, and therefore if they are shown in the command window will be easier to locate them than if they appear in the workspace, where all the variables are stored.

```
mSTOOIP=mean(STOOIP)
sdSTOOIP=std(STOOIP)
mEstReserves=mean(EstReserves)
sdEstReserves=std(EstReserves)
mAbYear=mean(Aby)
sdAbYear=std(Aby)
mNPV=mean(NPVvec1)
sdNPV=std(NPVvec1)
```

After this, the code is finished and the only step left is to go to the command window and type the name of the m-file, which will display all the results of the simulation. The results are analyzed and compared in the Results and Conclusion section.

Results and Conclusions

1. Introduction

After explaining both implementations in MATLAB and @RISK, it is time to run both simulations in order to obtain the results. This results will be evaluated and compared in the following pages.

The comparison will try to determine which is the better program in the evaluation of risk analysis in the oil and gas industry. In order to do so, the conclusion will take in to consideration speed, ease of implementation, the sampling of the input and the graph display.

2. Results in @RISK

In order to obtain the results in @RISK the simulation icon in the ribbon bar must be accessed (Figure 3.1.).

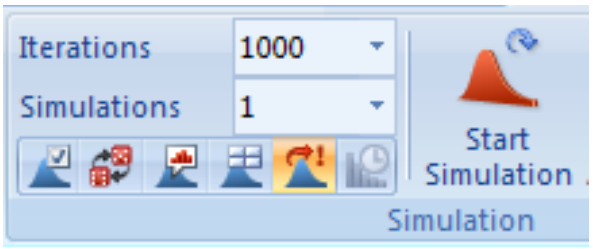


Figure 3.1

In the panel, the number of iterations and the number of simulations can be set. In this simulation the number of iterations that are going to be compared are 1000, 5000, 10000, 50000 and 100000.

The results of the simulation that are the most representative are: Standard Original Oil In Place, Estimated Reserves, Abandonment Year and Net Present Value. This results will be evaluated by four values each: P90,P10, Mean and Standard deviation. P90 represents a value which has a 90% chance of being greater than the real value, while the P10 represents a value which has a 10% chance of being greater than the real value. For the NPV one other value will be evaluated, which is the probability of having NPV<0. With each simulation we obtain also CDF graphs of each of the parameters.

The results of the simulation will also show the speed that the program has taken to run it.

- **Results with n=1000**

Time of simulation: Approximately 2 seconds

	P90	P10	Mean	St. Deviation
STOOIP	26.24	17.95	21.998	3.260
Reserves	5.80	3.50	4.580	0.902
Ab. Year	26	12	18.237	5.826
NPV	99.97	-27.57	32.92	50.78

Probability of NPV<0: 25.63%

The CDF graphs are shown in Figure 3.2. to 3.5.

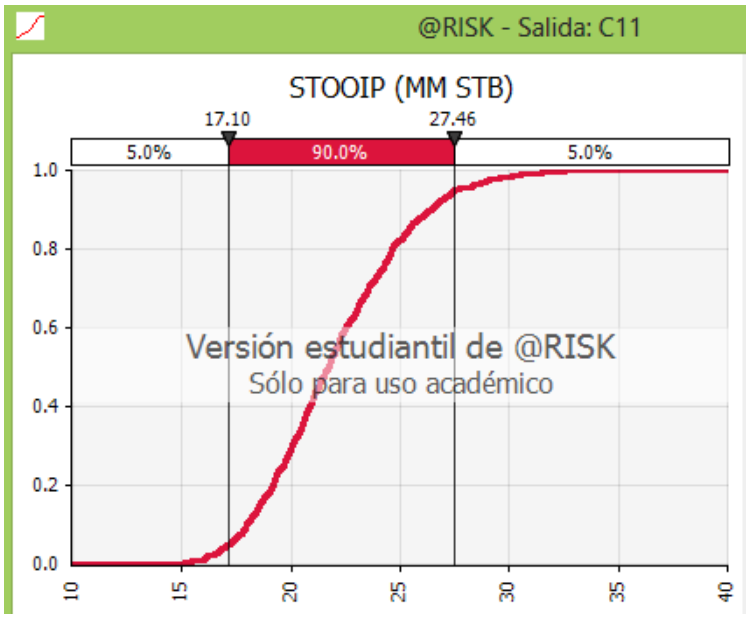


Figure 3.2.

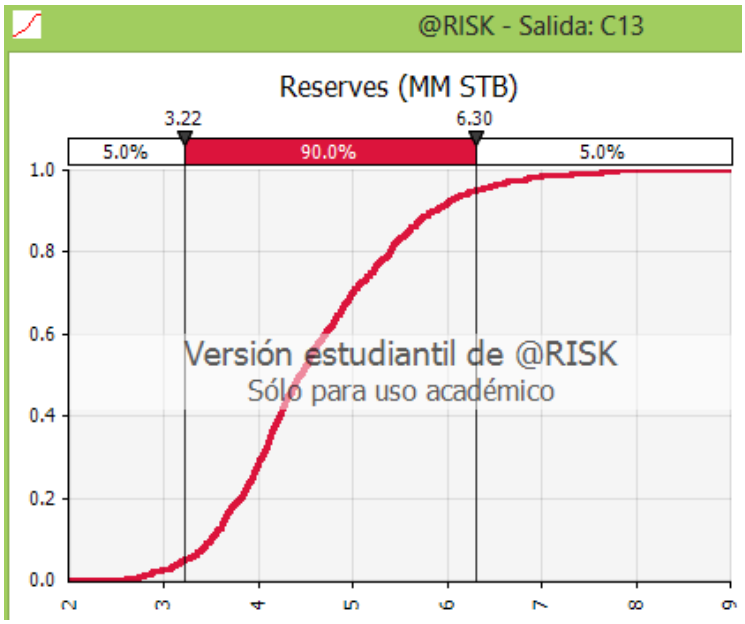


Figure 3.3.

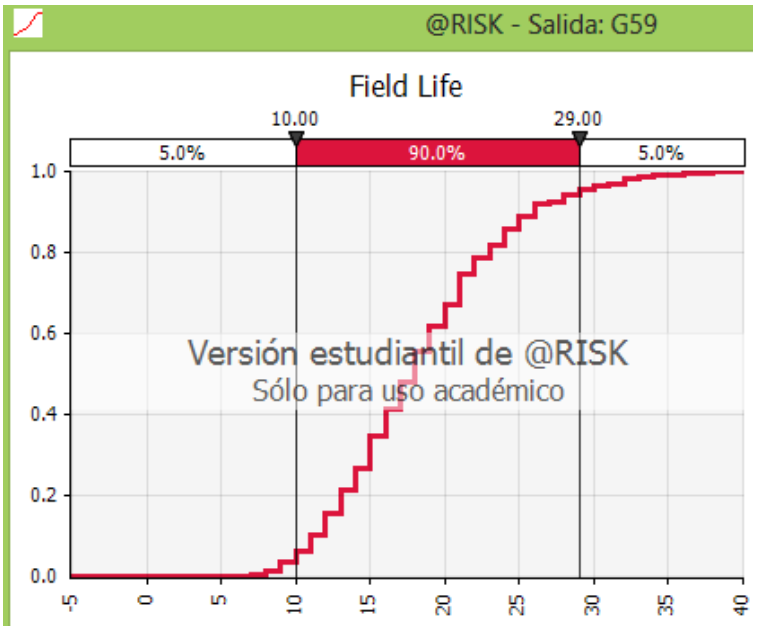


Figure 3.4.

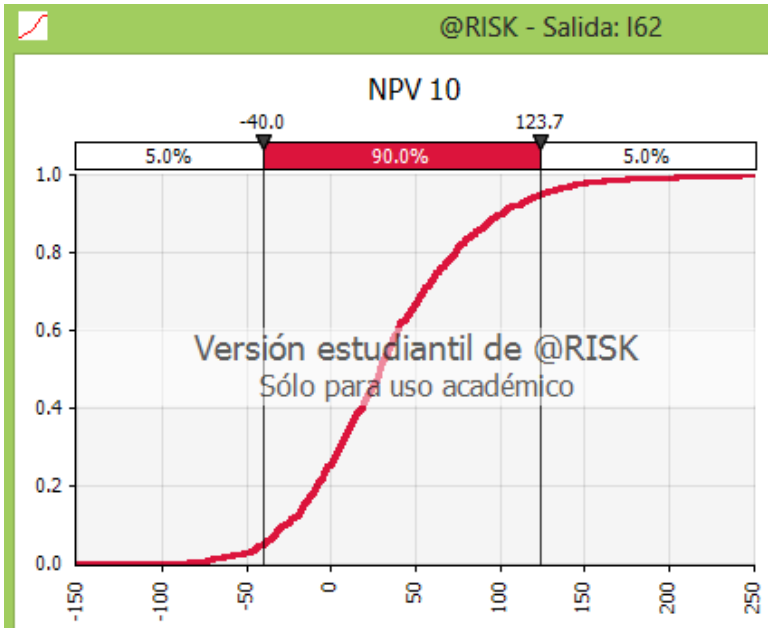


Figure 3.5.

- **Results with n=5000**

Time of simulation: Approximately 7 seconds.

	P90	P10	Mean	St. Deviation
STOOIP	26.41	17.99	22.002	3.291
Reserves	5.81	3.46	4.580	0.925
Ab. Year	26	12	18.286	5.747
NPV	97.75	-25.30	33.08	50.62

Probability of NPV<0: 26.77%

The CDF graphs are shown in Figure 3.6. to 3.9.

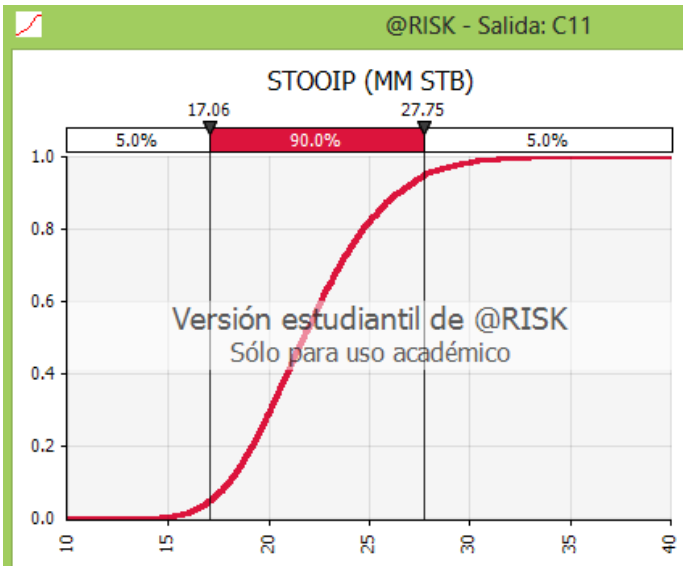


Figure 3.6.

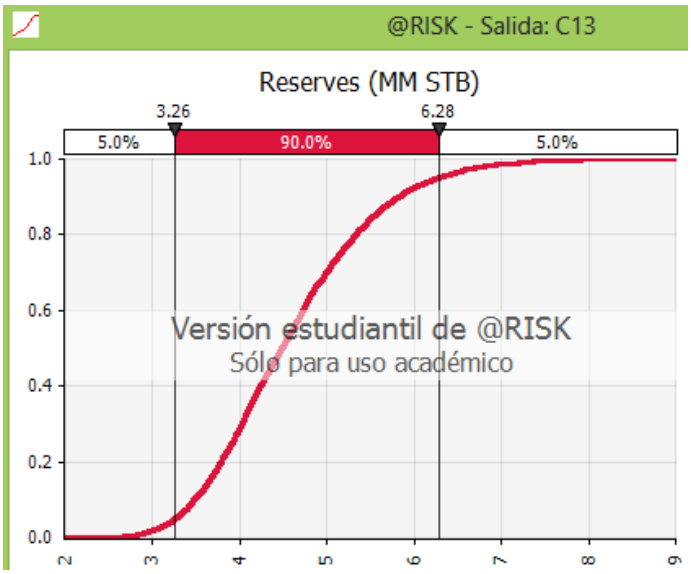


Figure 3.7.

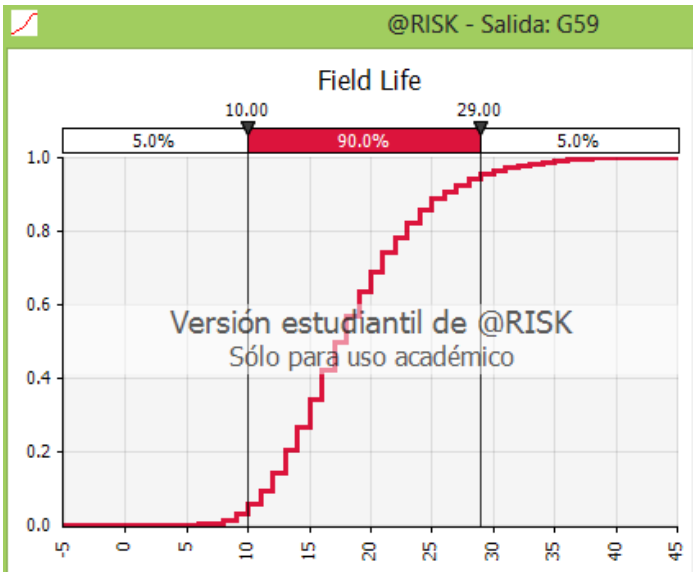


Figure 3.8.

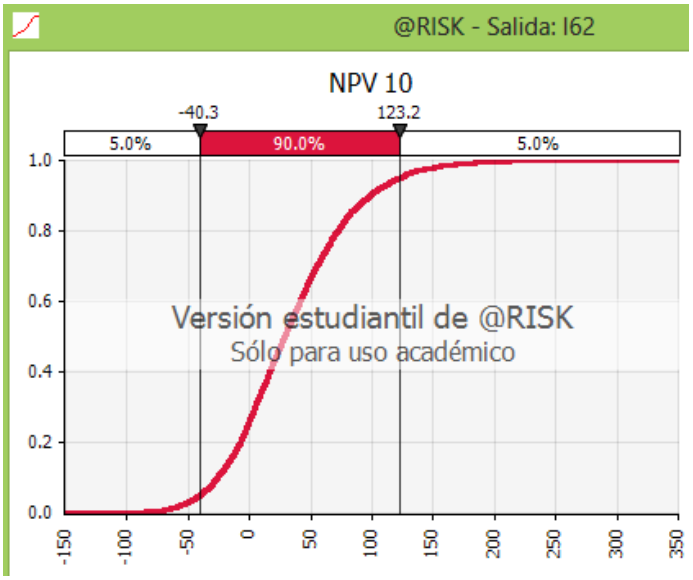


Figure 3.9.

- **Results with n=10000**

Time of simulation: Approximately 14 seconds

	P90	P10	Mean	St. Deviation
STOOIP	26.44	17.93	22.003	3.298
Reserves	5.82	3.47	4.584	0.920
Ab. Year	26	12	18.337	5.744
NPV	99.46	-26.02	33.28	50.42

Probability of NPV<0: 26.13%

The CDF graphs are shown in Figure 3.10. to 3.13.

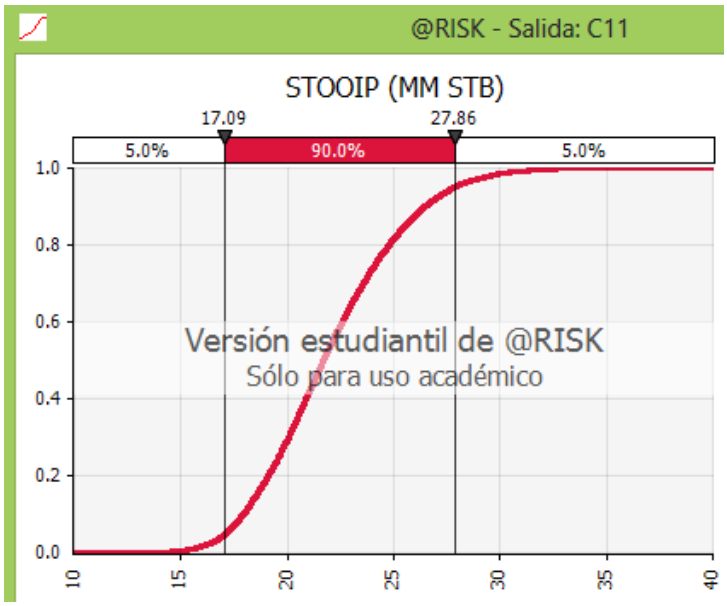


Figure 3.10.

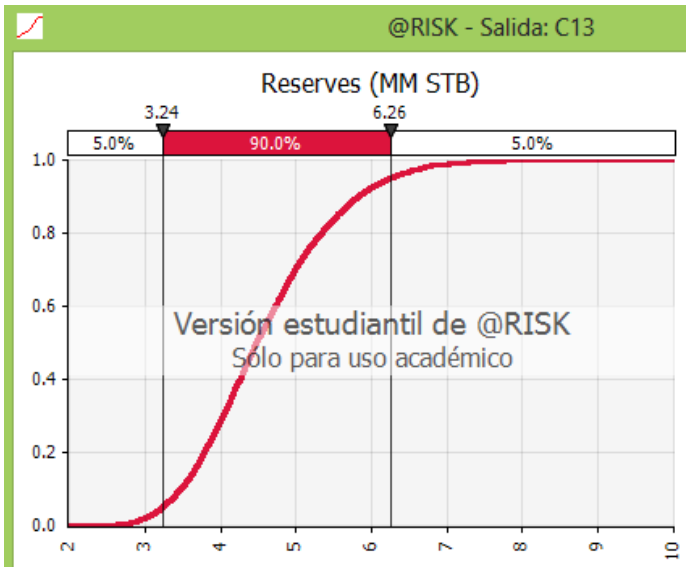


Figure 3.11.

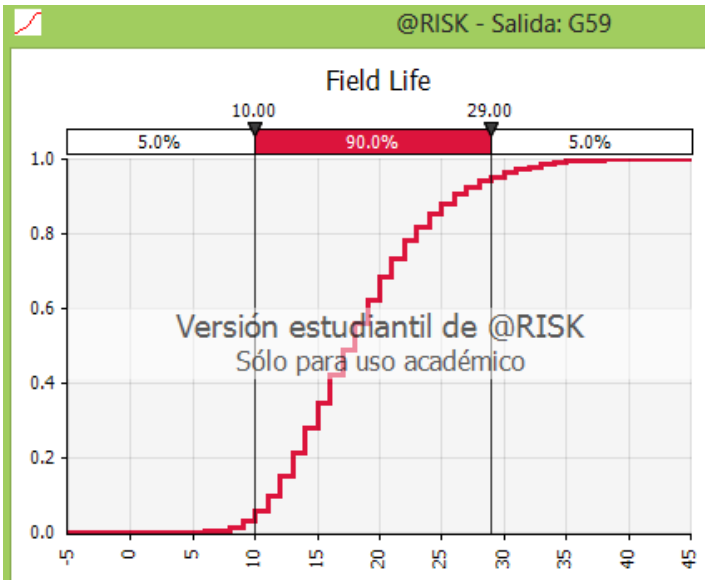


Figure 3.12.

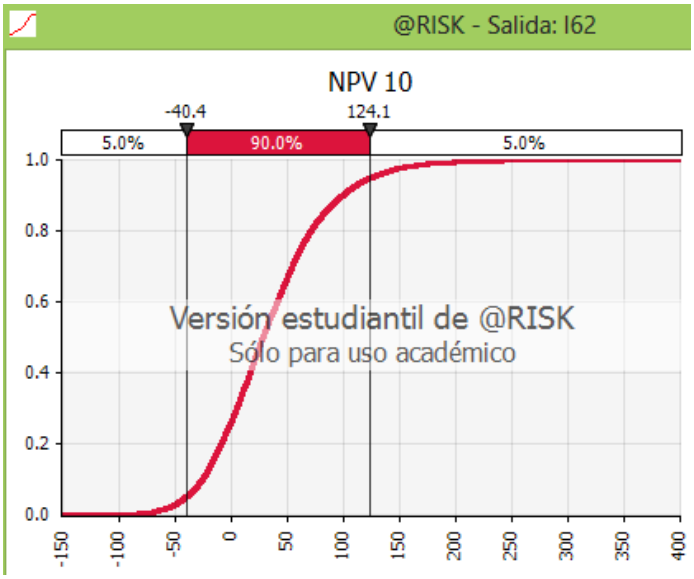


Figure 3.13.

- **Results with n=50000**

Time of simulation: Approximately 23 seconds

	P90	P10	Mean	St. Deviation
STOOIP	26.38	17.94	22.004	3.301
Reserves	5.82	3.47	4.585	0.92
Ab. Year	26	12	18.334	5.784
NPV	98.81	-25.95	33.27	50.47

Probability of NPV<0: 26.17%

The CDF graphs are shown in Figure 3.14. to 3.17.

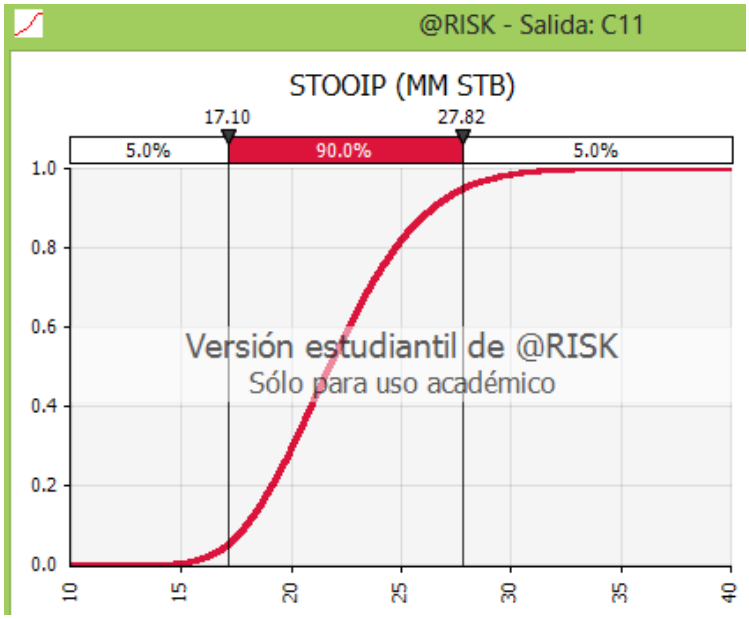


Figure 3.14.

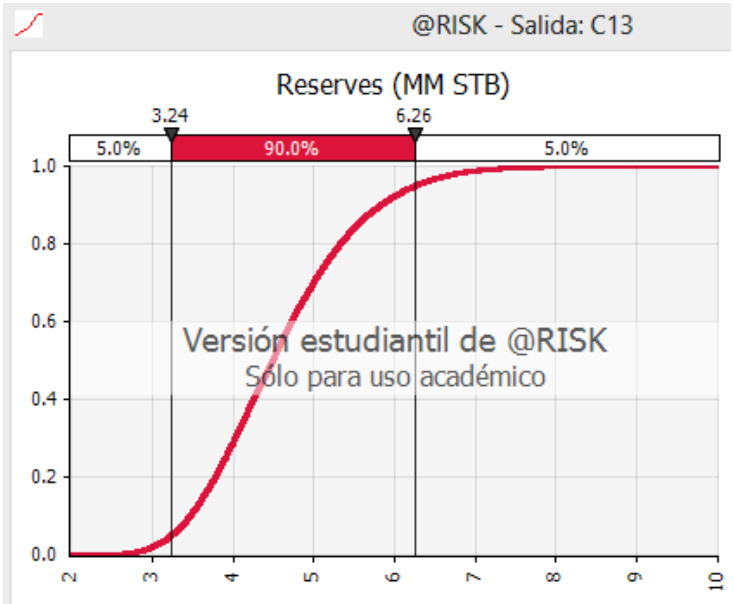


Figure 3.15.

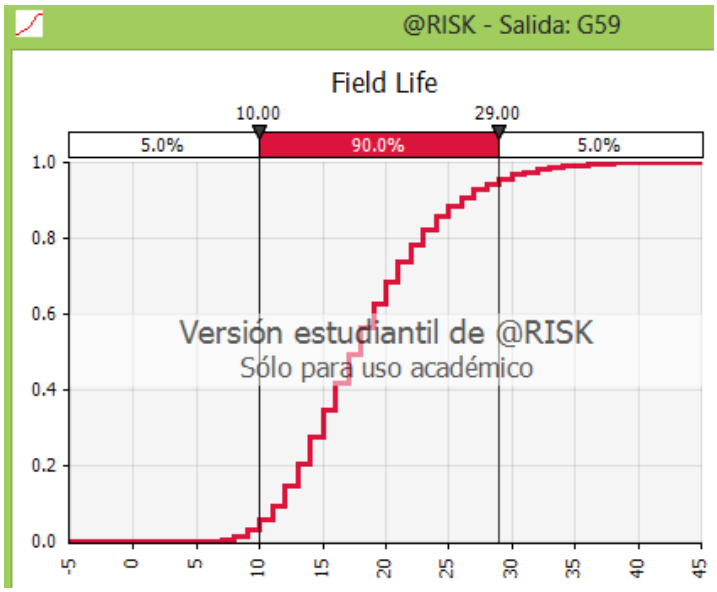


Figure 3.16.

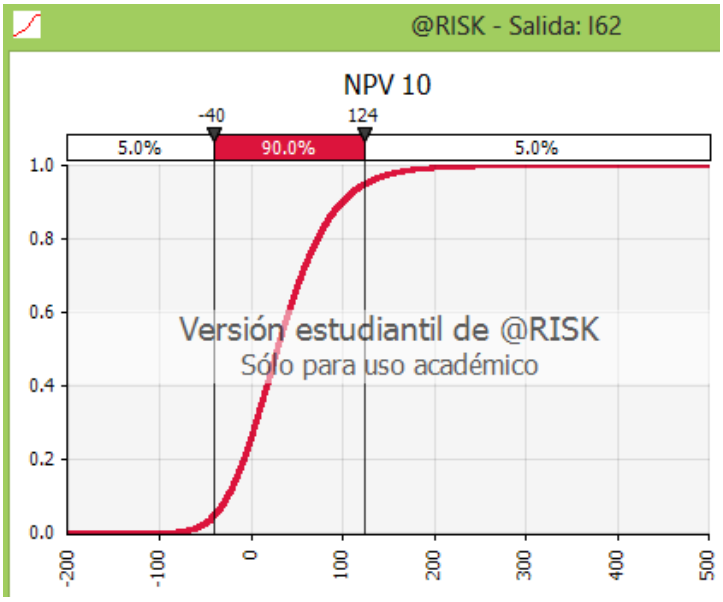


Figure 3.17.

- **Results with n=100000**

Time of simulation: Approximately 45 seconds.

	P90	P10	Mean	St. Deviation
STOOIP	26.42	17.95	22.003	3.295
Reserves	5.82	3.47	4.584	0.922
Ab. Year	26	12	18.314	5.773
NPV	97.85	-26.04	33.33	50.68

Probability of NPV<0: 26.23%

The CDF graphs are shown in Figure 3.18. to 3.21.

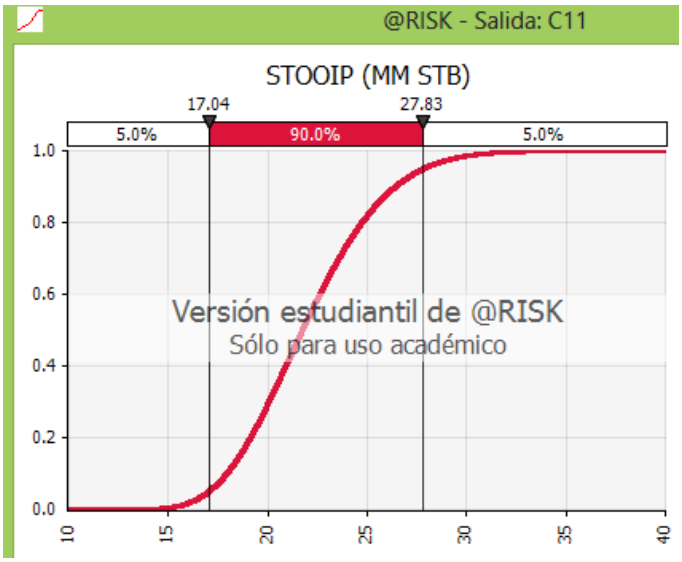


Figure 3.18.

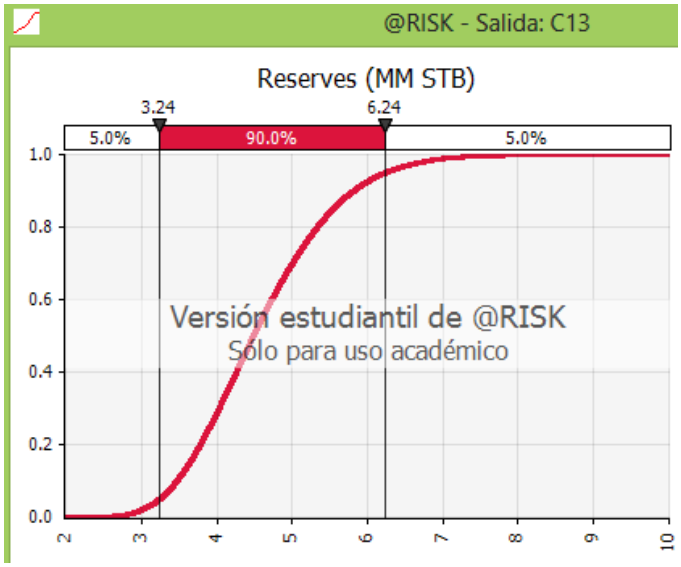


Figure 3.19.

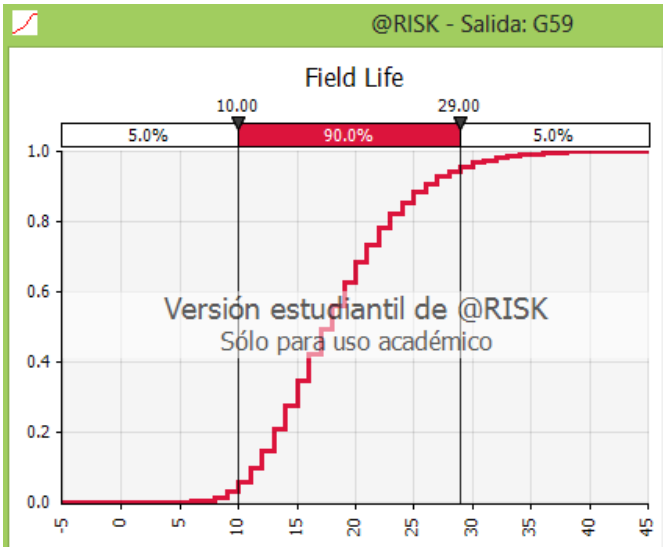


Figure 3.20.

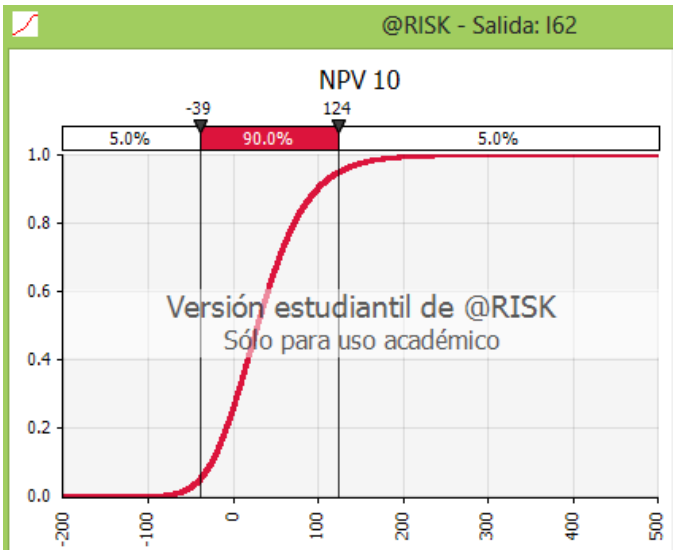


Figure 3.21.

3. Results in MATLAB.

In order to run the MATLAB simulation the name of the m-file which contains the code must be written in the command window, as it is shown in Figure 3.22.

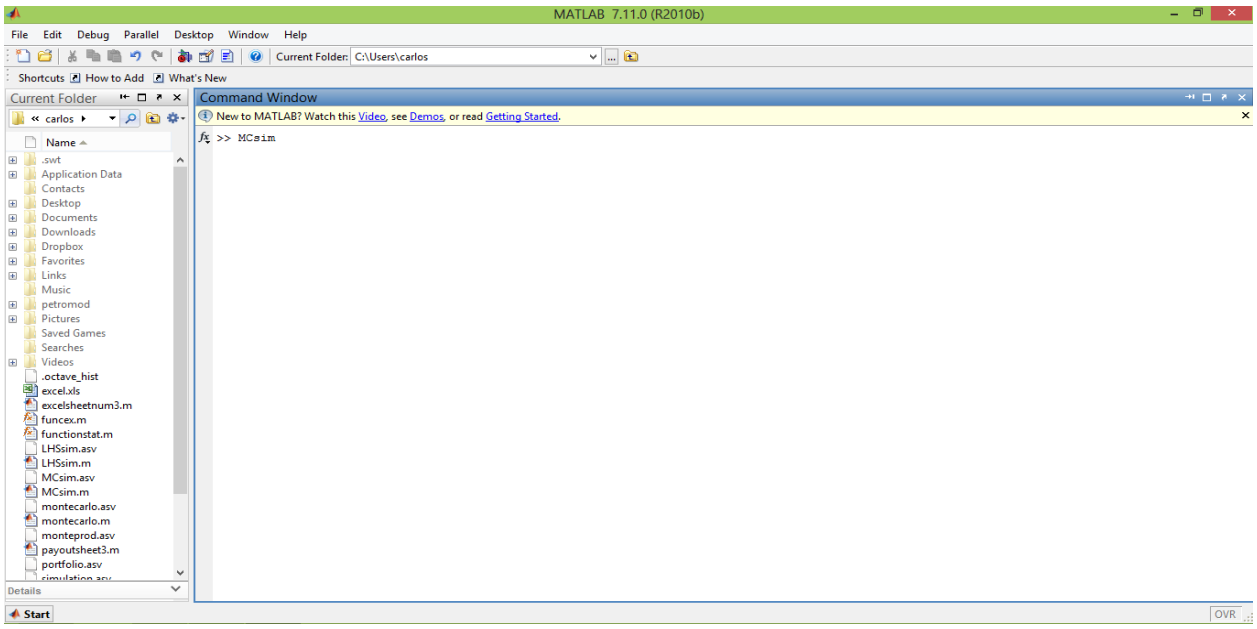


Figure 3.22.

As well as in the @RISK simulation, the number of iterations that have been run in MATLAB are 1000,5000,10000,50000 and 100000.

- **Results with n=1000.**

Time of simulation: 0.695 seconds

	P90	P10	Mean	St. Deviation
STOOIP	25.89	16.94	21.369	3.507
Reserves	5.71	3.31	4.440	0.930
Ab. Year	27	13	19.569	6.077
NPV	77.35	-31.60	22.065	46.074

Probability of NPV<0: 32.1%

The CDF graphs are shown in Figure 3.23. to 3.26.

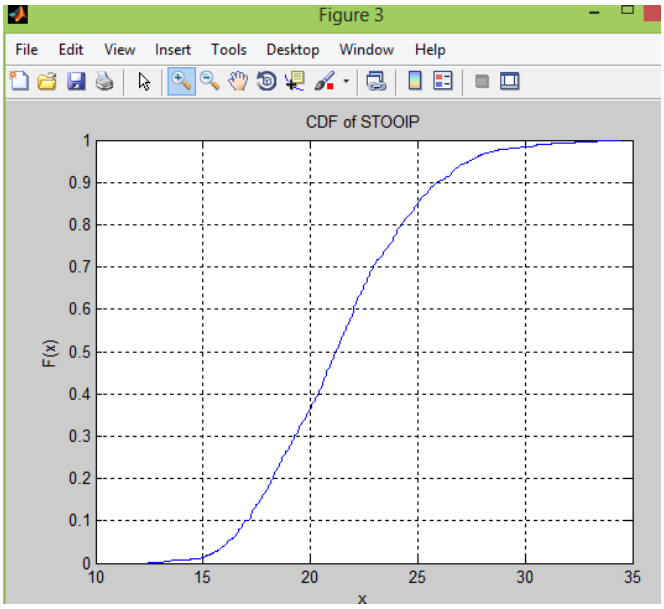


Figure 3.23.

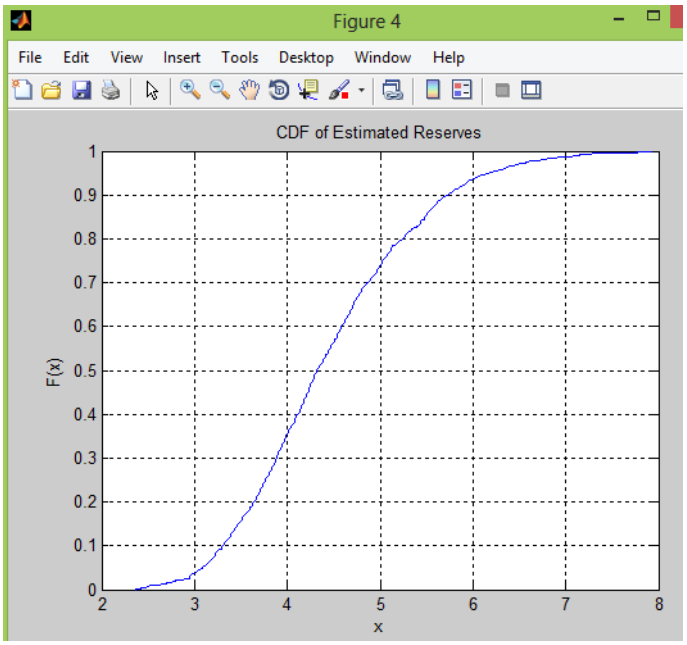


Figure 3.24.

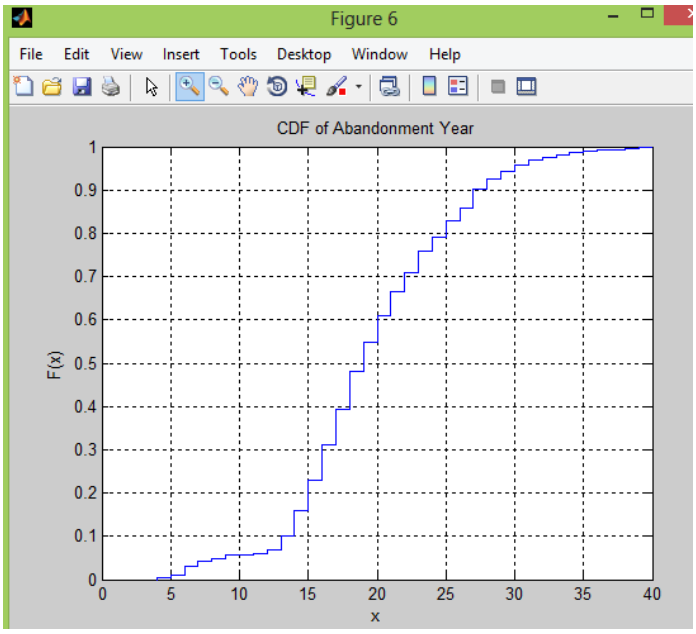


Figure 3.25.

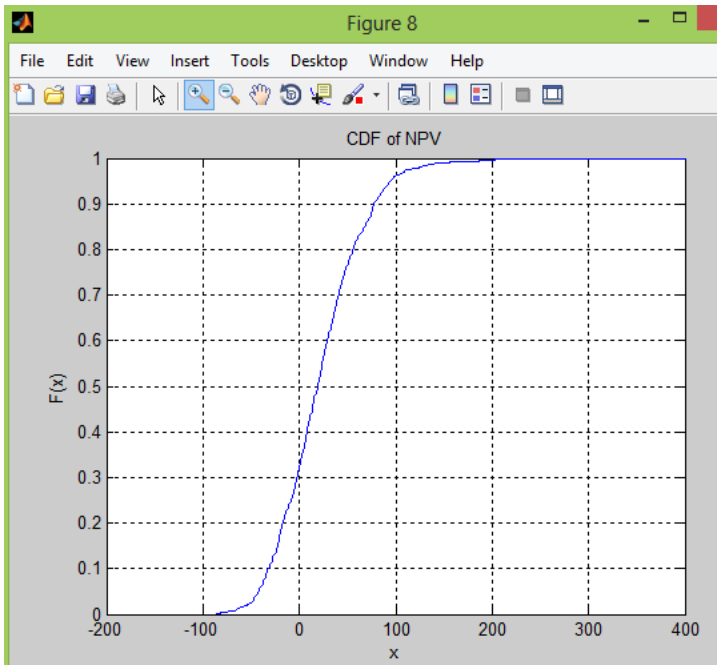


Figure 3.26.

- **Results with n=5000**

Time of simulation: 0.733 sec

	P90	P10	Mean	St. Deviation
STOOIP	25.98	16.95	21.278	3.509
Reserves	5.72	3.30	4.438	0.945
Ab. Year	27	14	19.561	6.071
NPV	79.07	-31.02	22.941	46.673

Probability of NPV<0: 31.96%
The CDF graphs are shown in Figure 3.27. to 3.30.

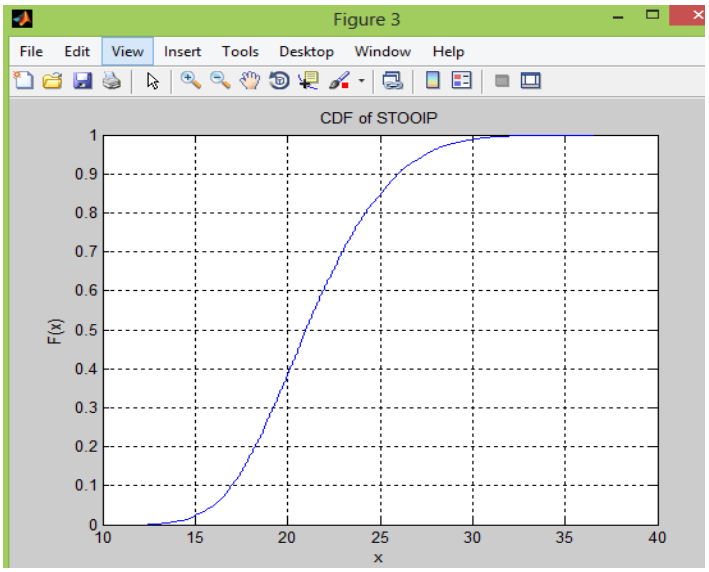


Figure 3.27.

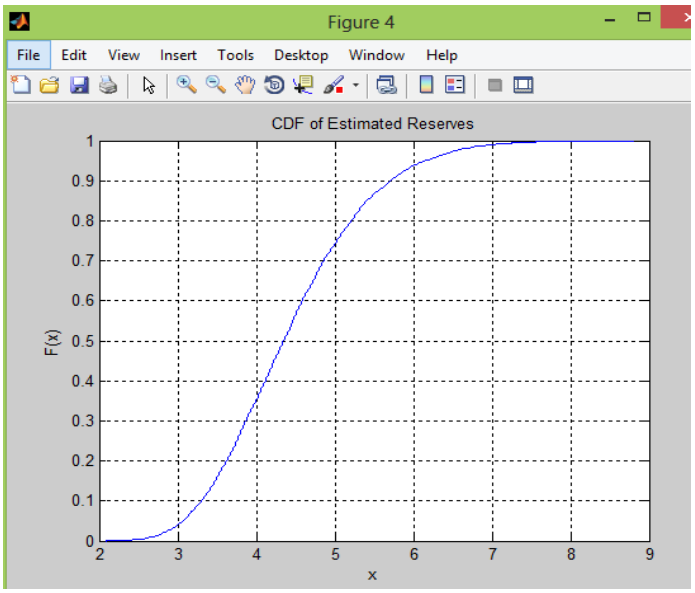


Figure 3.28.

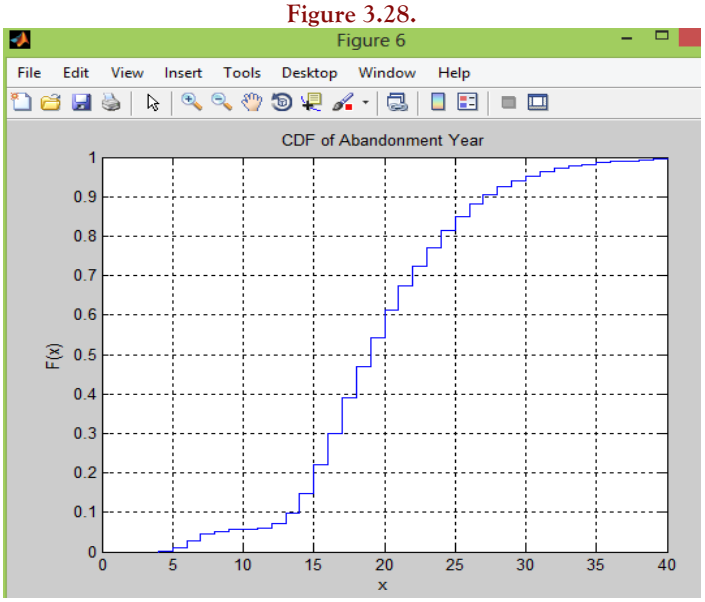


Figure 3.29.

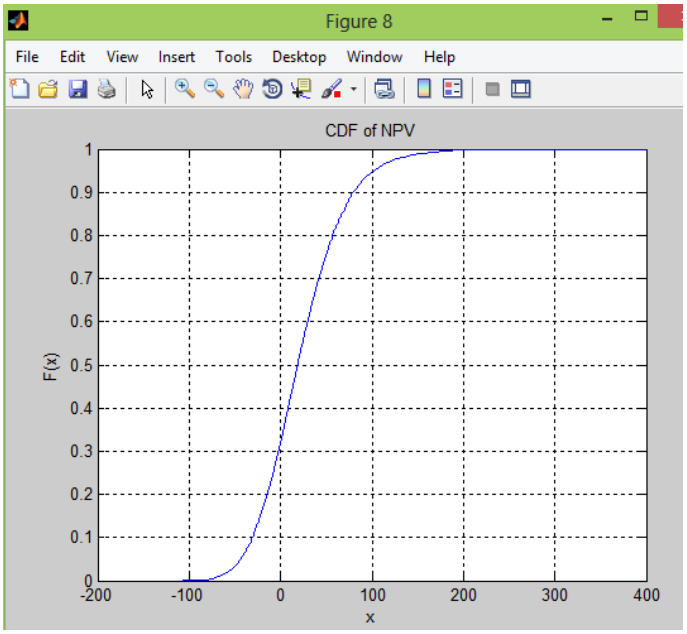


Figure 3.30.

- **Results with n=10000**

Time of simulation: 0.817 seconds

	P90	P10	Mean	St. Deviation
STOOIP	25.97	16.92	21.263	3.495
Reserves	5.69	3.29	4.433	0.944
Ab. Year	27	14	19.589	5.990
NPV	80.17	-29.52	23.437	46.049

Probability of NPV<0: 31.22%
The CDF graphs are shown in Figure 3.31. to 3.34.

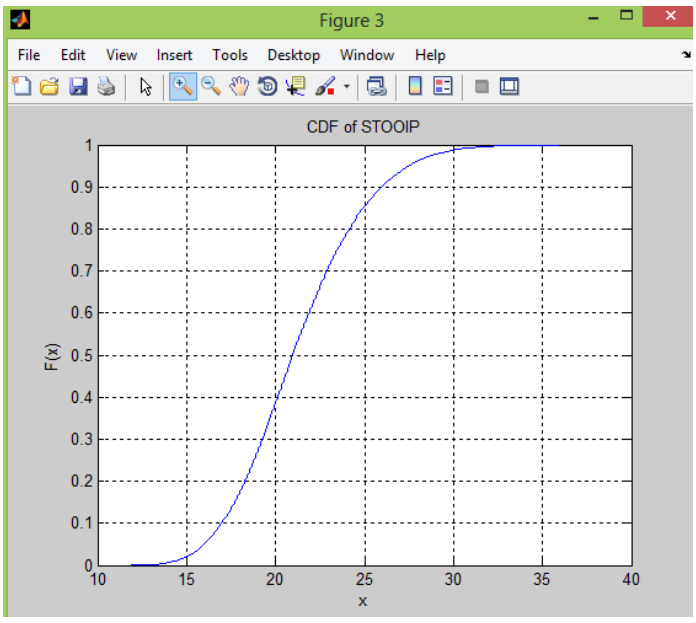


Figure 3.31

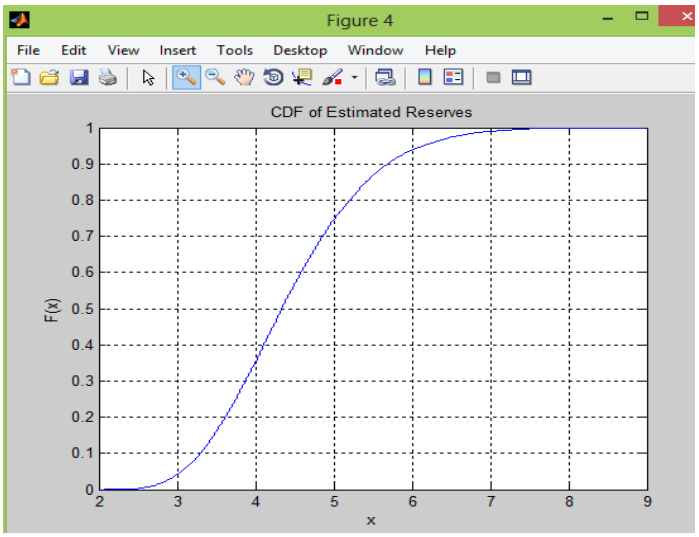


Figure 3.32.

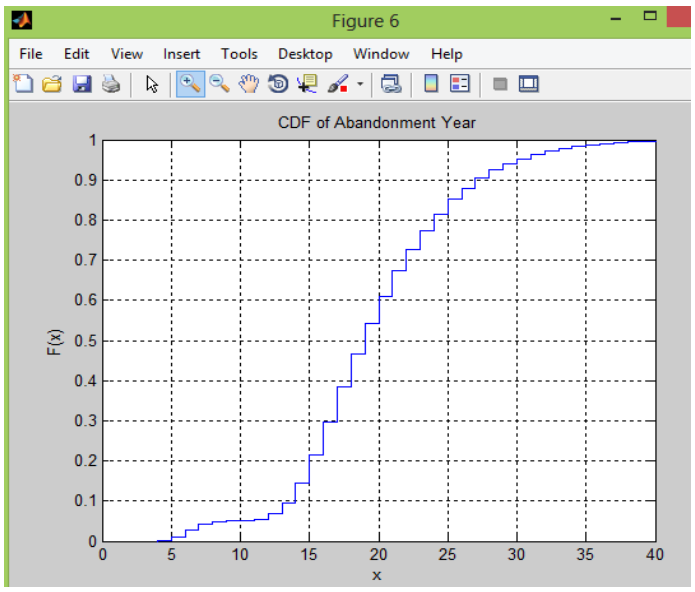


Figure 3.33.

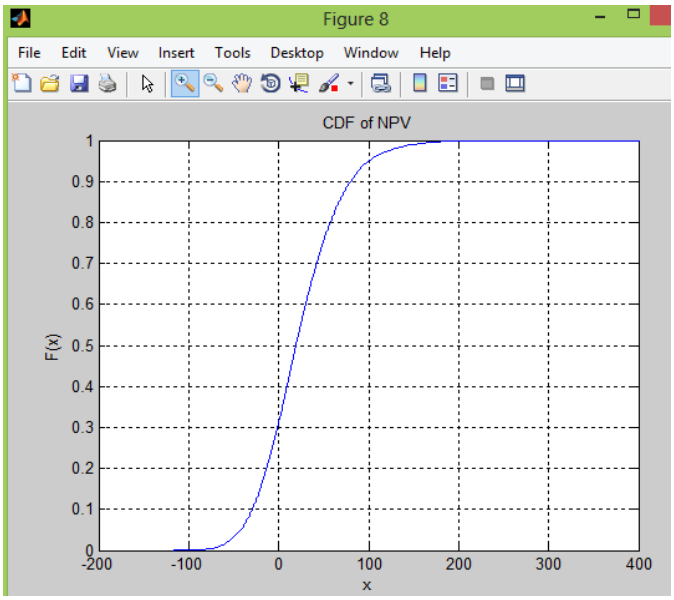


Figure 3.34.

- **Results with n=50000**

Time of simulation: 1.196 seconds

	P90	P10	Mean	St. Deviation
STOOIP	25.97	17.01	21.291	3.507
Reserves	5.70	3.30	4.432	0.944
Ab. Year	27	14	19.516	6.023
NPV	80.60	-30.46	23.155	46.727

Probability of NPV<0: 31.73%

The CDF graphs are shown in Figure 3.35. to 3.38.

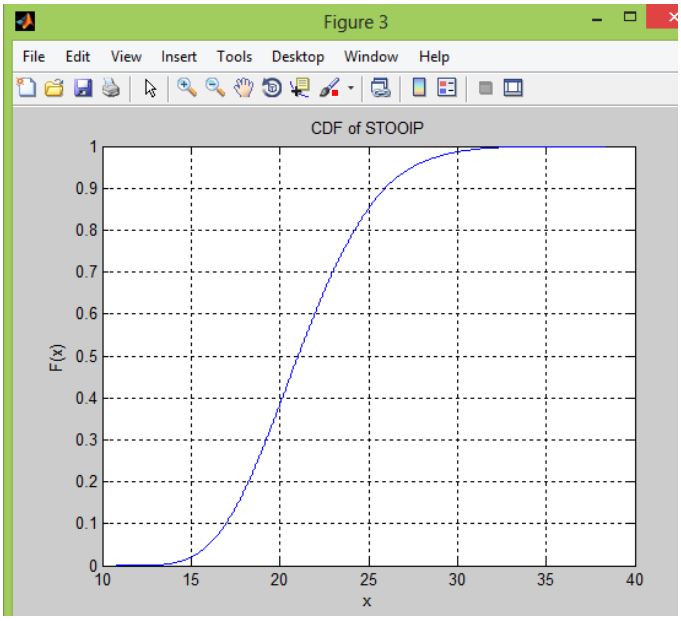


Figure 3.35.

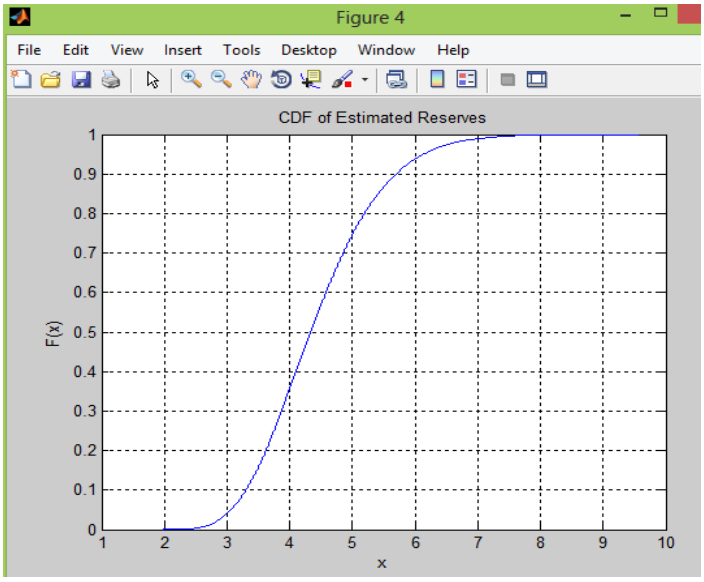


Figure 3.36.

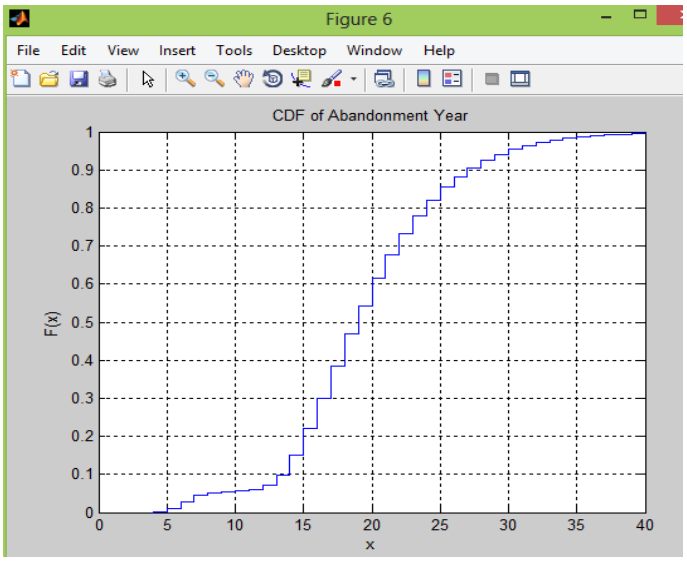


Figure 3.37.

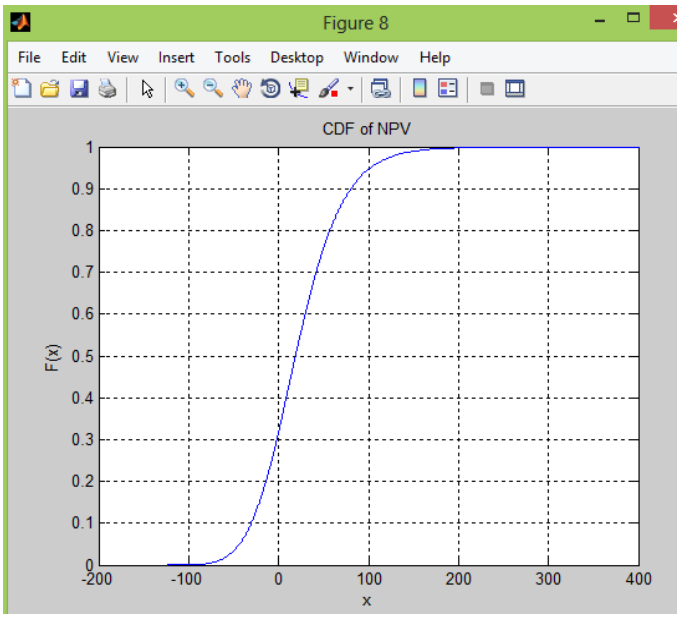


Figure 3.38.

- **Results with n=100000**

Time of simulation: 1.374 seconds

	P90	P10	Mean	St. Deviation
STOOIP	26.00	16.99	21.304	3.502
Reserves	5.71	3.31	4.435	0.939
Ab. Year	27	14	19.546	5.978
NPV	79.73	-30.08	23.101	46.352

Probability of NPV<0: 31.70%

The CDF graphs are shown in Figure 3.39. to 3.42.

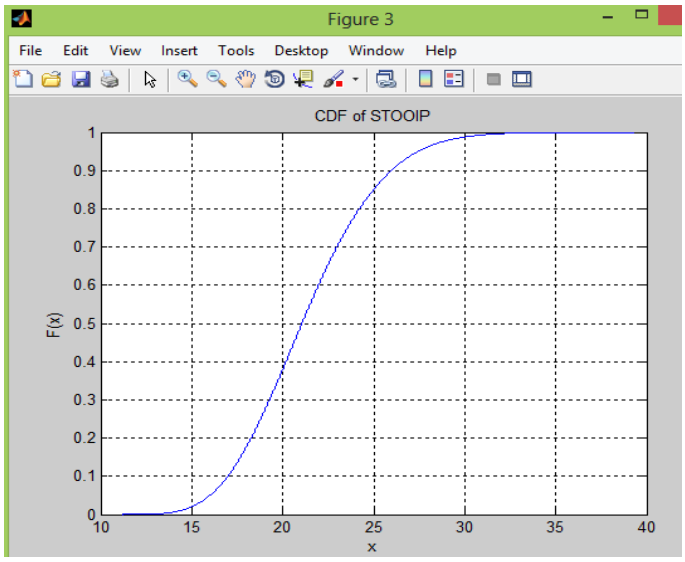


Figure 3.39.

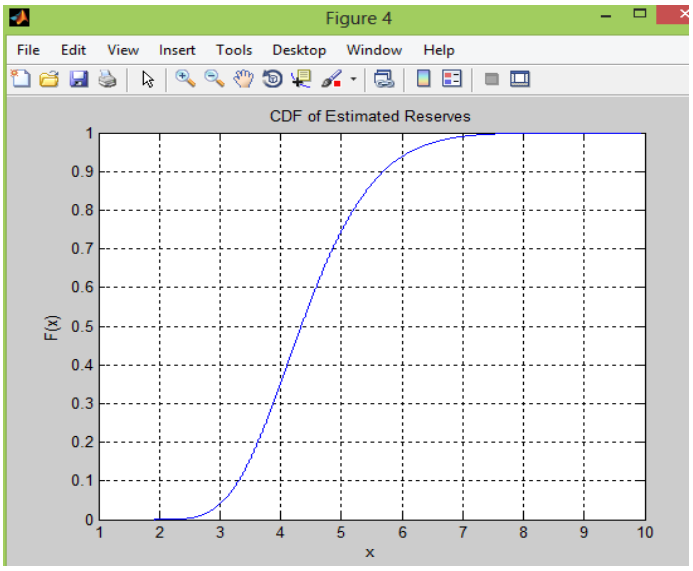


Figure 3.40.

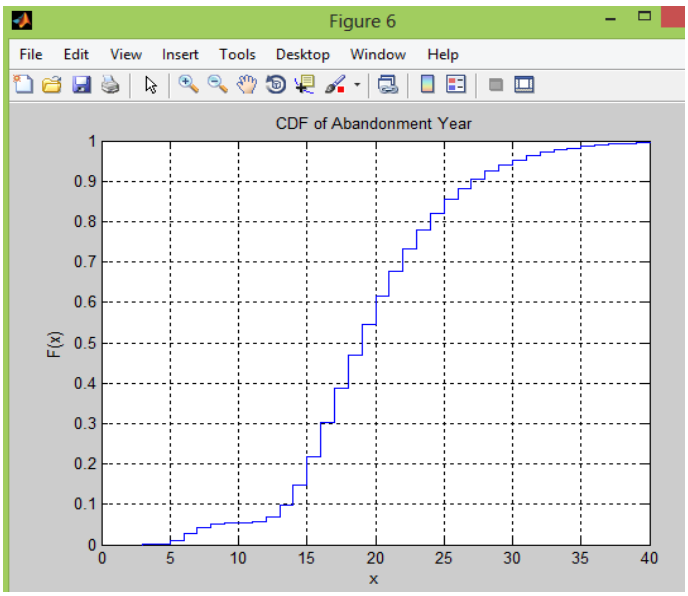


Figure 3.41.

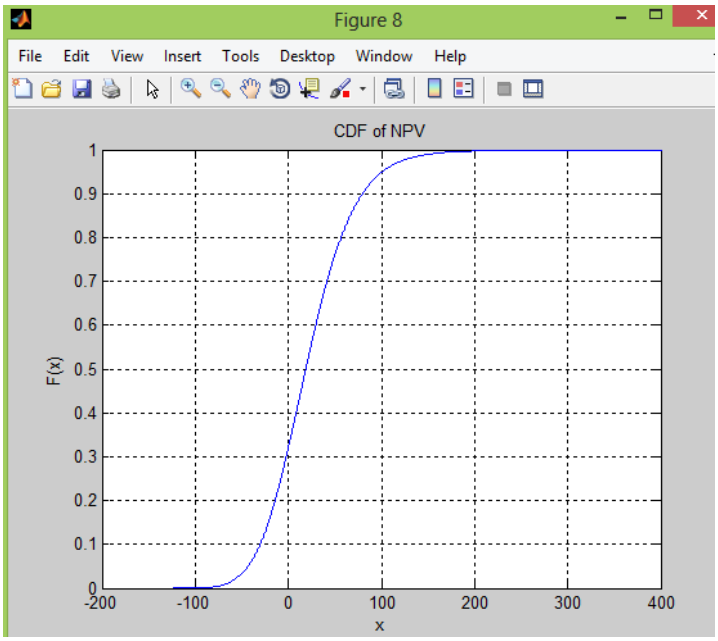


Figure 3.42.

4. Comparison and Analysis

After showing all the results from the simulations in MATLAB and @RISK it is now time to compare them. As it has been said, this comparison will prioritize speed, ease of implementation, sampling and graph display.

4.1.Speed

Both simulations had been run with a laptop ASUS A55VD, which runs a processor Intel Core i7-3630QM @ 2.40 GHz. The RAM memory is 8.00 GB and the system is Windows 8.

MATLAB is clearly faster at running the simulation. This is especially noticeable in the simulations with a lot of iterations, as it can be seen in Figure 3.43.

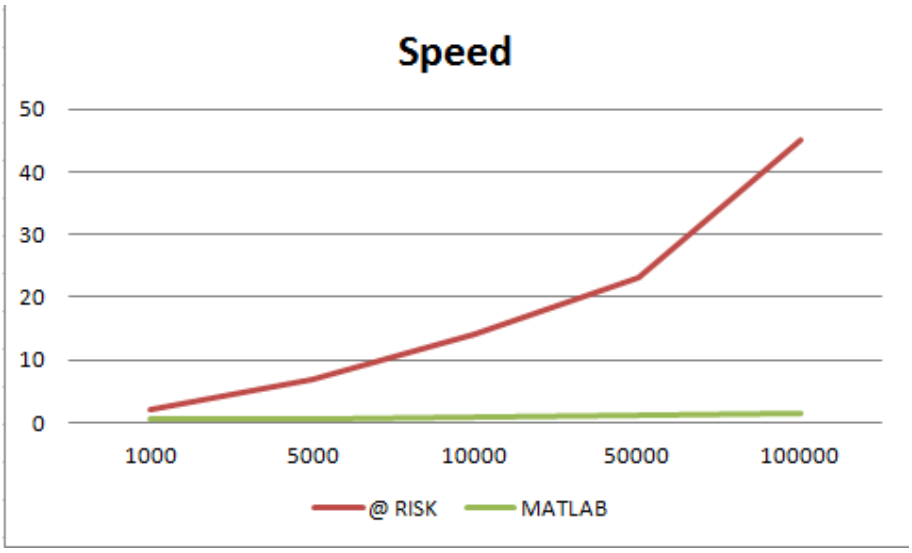


Figure 3.43.

4.2.Ease of implementation

As it has been mentioned before, @RISK and MATLAB have very different mechanics to set up simulations.

@RISK uses the same interface as Excel since it is an add-in for that program. This makes the implementations quite easy, since all of the graphs and inputs can be directly changed without having to introduce any code.

MATLAB uses its own interface, which is sometimes a little bit frustrating. In MATLAB is necessary to code all of the steps that the simulation must run to and it is quite common to commit a mistake without noticing it, and even though MATLAB indicates the line where the mistake is, it is not always easy to identify what is wrong.

Therefore, @RISK is slightly easier to use than MATLAB, since its interface is one that the user is probably more used to see than the one of MATLAB.

4.3.Graph display.

In the figures that have been shown in previous pages the difference between graph display can be seen.

Both program allow to zoom in and out of the graphs and they also allow the introduction of labels, change of the colour of the lines and the superposition of different graphs.

Even though both program allow for multiple variation in the graph display, @RISK has the slightly edge in graph display since it is easier to change the display of the graphs by directly accessing to the graph menu, while in MATLAB the code in the m-file needs to be changed in order to do so.

4.4.Sampling

It has been observed in the results of both programs that there is a slight difference. This difference is a result of the sampling method in both implementations.

@RISK uses the Latin Hypercube Sampling while the MATLAB code uses a Monte Carlo Sampling.

Latin Hypercube Sampling was first introduced in 1979 by McKay, Beckman and Conover in their paper “A comparison of three methods for selecting values of input variables in the analysis of output from a computer code”. In this paper the principles of LHS are explained.

Latin Hypercube Sampling is a similar method to the Stratified Sampling that tries to ensure that all portions of the sample space are sampled. One advantage of LHS appears when the output is dominated by only a few of the components of the input. This

method ensures that each of those components is represented in a fully stratified manner, no matter which components might turn out to be important.

In Figure 3.44 to 3.51 the mean and the standard deviation are compared in both methods and for each of the parameters that have been commented before.

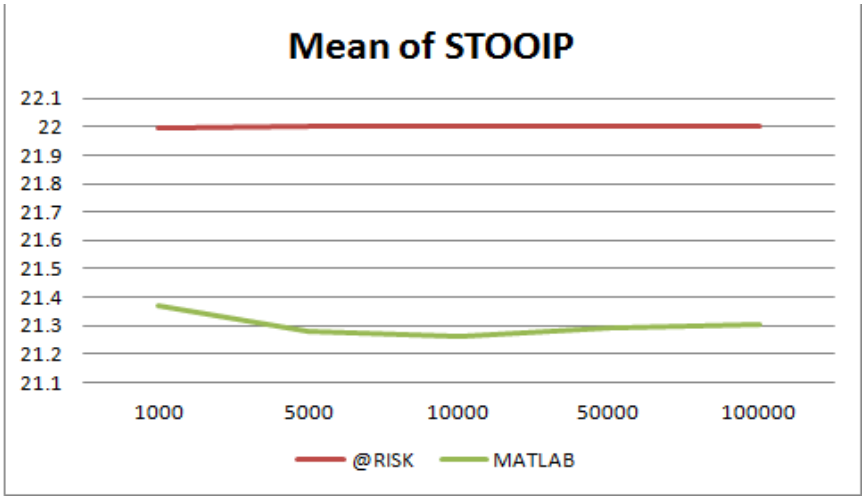


Figure 3.44.

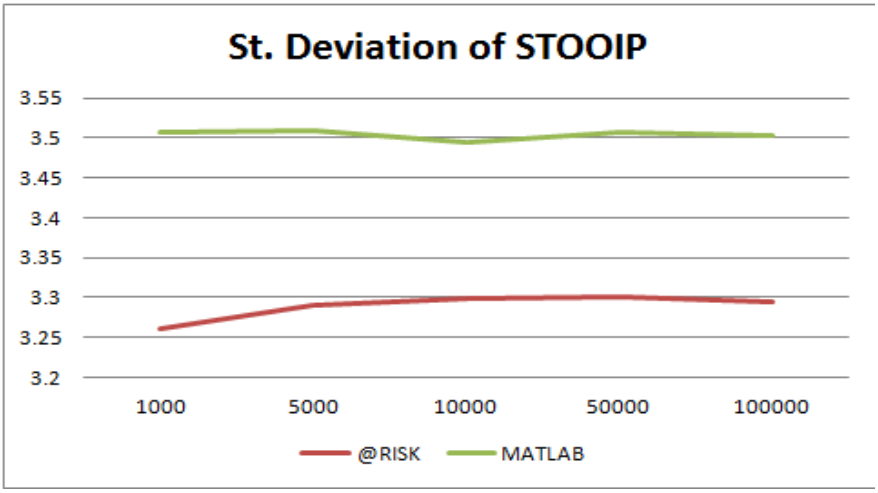


Figure 3.45.

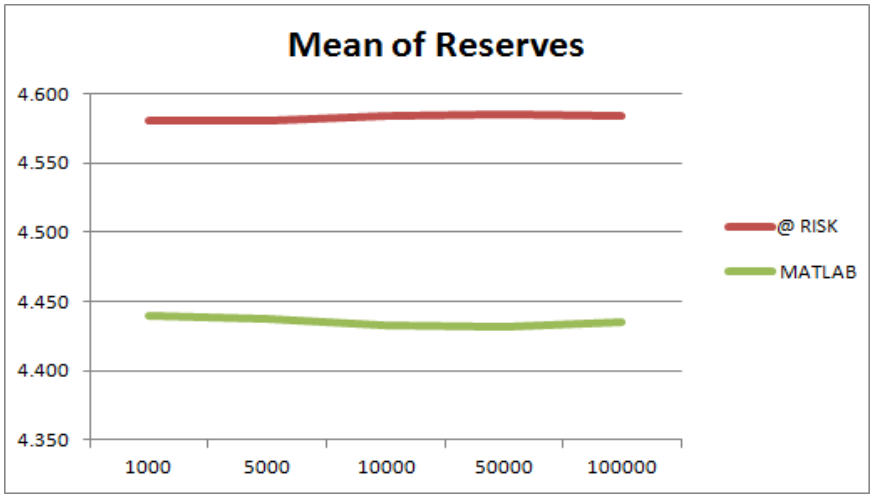


Figure 3.46.

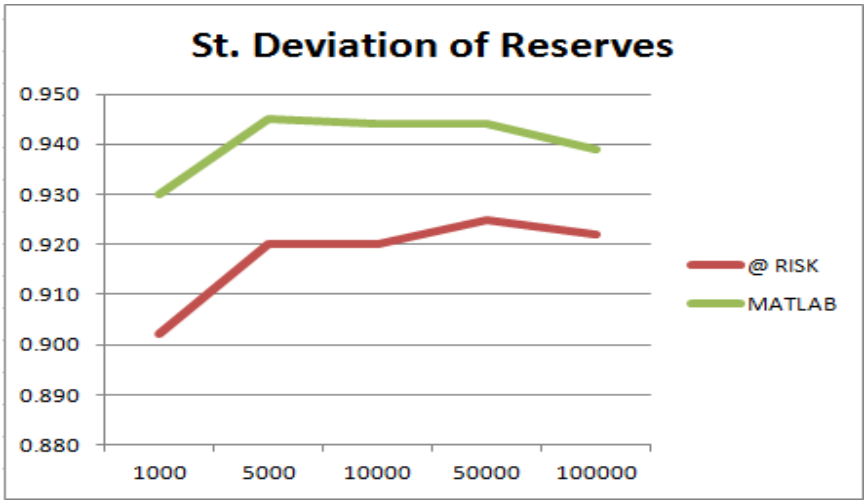


Figure 3.47.

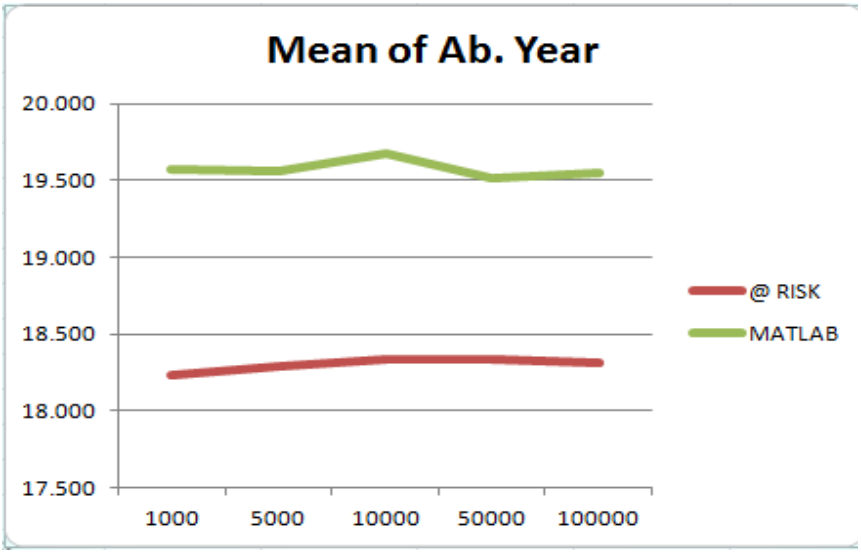


Figure 3.48.

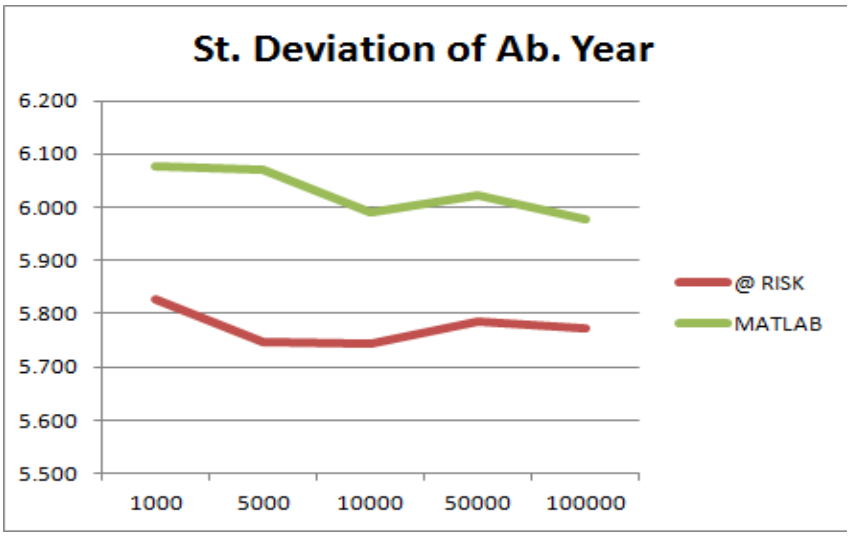


Figure 3.49.

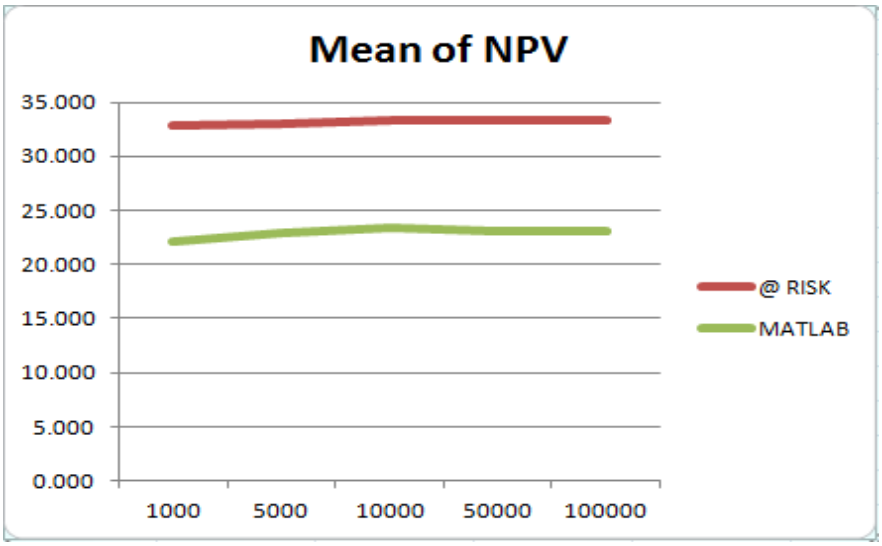


Figure 3.50.

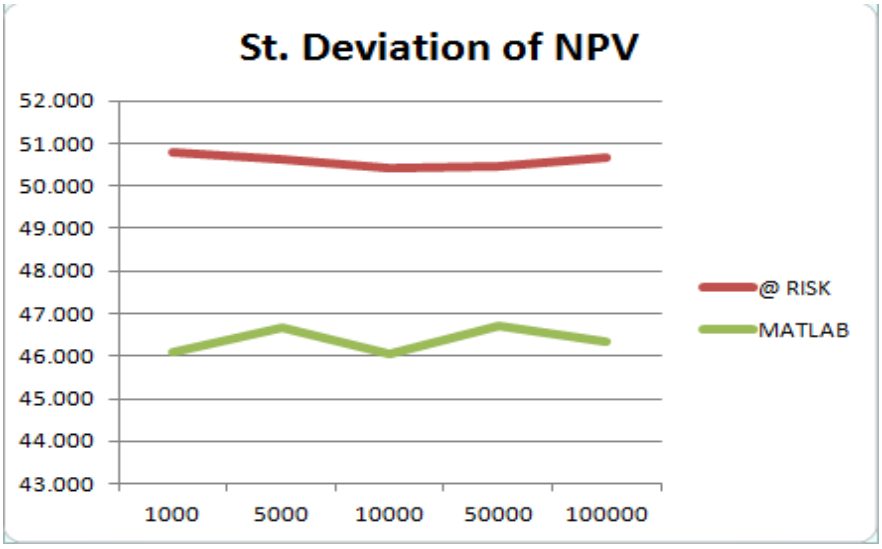


Figure 3.51.

It is observed that all of the parameters have a higher Standard Deviation with the MATLAB routine and all of them a higher Mean in the @RISK routine but in the Abandonment Year.

The answer for which of the sampling methods is better for the risk analysis in the oil industry can be found in Helton's paper form 2004 "A comparison of uncertainty and sensitivity analysis results with random and Latin Hypercube Sampling". In this paper, Helton affirms that Monte Carlo Sampling may have valid results when the sampling needed is small, since the goal is usually to determine general patterns of behaviour rather than likelihoods for specific, low probability behaviours. But in this case, the sample needed is quite big and there is also a large number of inputs and possible outputs. Therefore the importance of stratified sampling is big, since "it should decrease the likelihood of being mislead in assessing the relationships between individual inputs and outputs". Therefore, the sampling technique used in the @RISK routine is better than the one used in the MATLAB routine.

5. Conclusions

This Thesis has tried to compare @RISK and MATLAB into solving the problems that risk analysis brings in the oil and gas industry. It has being seen that both programs are really useful in order to do so. @RISK brings an easier implementation as well as a more consistent sampling method, but MATLAB is a more powerful calculation tool.

At first glance, it would seem that @RISK is better because even though it is not faster than MATLAB, the edge in the sampling method would make the difference.

MATLAB does not include the Latin Hypercube Sampling in its library but that does not mean that it cannot be used. Since MATLAB is so flexible that it would allow to introduce the Latin Hypercube Sampling and use it in the code. This advantage, that has been commented in previous pages, is what makes MATLAB a special program for any calculation, not just risk analysis. MATLAB is such a flexible program that allows the user to code anything needed. This makes MATLAB a far superior program than @RISK that also would allow more possibilities in risk analysis.

@RISK is useful when a simple well is tried to be analyzed, if the simulation that needs to be run has a lot of inputs and outputs, complex correlations etc, @RISK might not have the calculation power needed for that. While MATLAB, since it is a more powerful calculation tool allows the creation of more complex simulations.

One example where MATLAB might be more useful in risk analysis in the oil industry is the creation of a portfolio of fields or wells which would help decision-makers choose the optimal portfolio given the goals and constraints of the company. A realistic portfolio would have up to 50 fields/wells and with advantages that MATLAB has over @RISK, the development of such portfolio would me very efficient.

To sum it all up, the purpose of this Thesis has been to bring light about whether MATLAB or @RISK is the better program in order to assess risk analysis in the oil and gas industry. While @RISK is more accurate and easier to use, MATLAB is faster and more flexible. Therefore, the conclusion that has been reached is that @RISK is more appropriate for simple simulations where the number of inputs and outputs is not very high, and where @RISK will have better results because of its higher accuracy, but with a more complicated simulation, with a higher number of inputs and outputs, MATLAB has the edge because of its better calculation power.

Bibliography

[1] @RISK users guide Version 6

[2] "Aprenda Matlab 6.1", Escuela Técnica Superior de ingenieros Industriales (Universidad Politécnica de Madrid), Javier García de Jalón José Ignacio Rodríguez Alfonso Brazales, 2001

[3] http://www.epixanalytics.com/modelassist/AtRisk/Model_Assist.htm#@RISK_and_Excel_functions/@RISK_functions.htm

[4] "Tutorials for Matlab and Pro/Engineering Wildfire 2.0" Dept. of Mechanical and Manufacturing Engineering (University of Manitoba) Gary Wang

[5] MATLAB users guide version R2014a

[6] Mckay, Beckman & Conover – A comparison of three methods for selecting values of input variables in the analysis of output from a computer code.

[7] Startzman & Wattenbarger – An improved computation procedure for risk analysis problems with unusual probability functions.

[8] Glynn & Iglehart – Importance of sampling for stochastic simulations.

[9] Helton, Davis & Johnson – A comparison of uncertainty and sensivity analysis results with random and LHS.

[10] Huntington & Lyrintzis – Improvements and limitations of LHS

[11] http://www.palisade.com/risk/

[12] http://www.peakoil.net/