# Executive summary

The herein report aims at discussing the advantages of adopting a VBA approach instead of a normal Excel sheet to a range of financial tasks varying from portfolio management to saving accounts and option pricing. Further each code is explained as far as its enhanced functionalities are concerned.

Precise code parts are analyzed in order to explain the backbone of the VBA functionalities to the user.

For each question the differences between a standard and a VBA approach are outlined, in particular attention is brought to the increased easiness in modifying sub procedure and functions for the final output.

Table of Contents

[Executive summary 1](#_Toc45701059)

[Question 3 1](#_Toc45701060)

[Introduction 1](#_Toc45701061)

[Code 1](#_Toc45701062)

[Question 4 3](#_Toc45701063)

[Overall structure 3](#_Toc45701064)

[Improved functionality 4](#_Toc45701065)

[Code 4](#_Toc45701066)

[Testing 5](#_Toc45701067)

[Question 7 5](#_Toc45701068)

[Declaring Variables 5](#_Toc45701069)

[Developing Binomial Option Tree 6](#_Toc45701070)

[Binomial Option Pricing Tree Calculations 7](#_Toc45701071)

[Conclusion 7](#_Toc45701072)

[Question 8 8](#_Toc45701073)

[VBA advantages 8](#_Toc45701074)

[Conclusion and recommendations 8](#_Toc45701075)

Question 3

## Introduction

Assume that you invest in equity. You will wonder how much You'll wonder how much profit you can make, and how much loss we'll lose if things get bad.

The purpose of this model is to visualize the price path of a stock using the Monte Carlo style simulation. Monte Carlo simulation refers to a methodology that solves a problem by generating random situations with random numbers. The price of the equity used in this model is based on Brownian motion.

## Code

(1)

S(T) is the price after T period, in this case a year after which is 250 business in this case. S(0) is the current price, r is expected return, σ is the volatility, and Z is the standard normal random variable with mean 0 and variance 1.

With this model, the investors can presume the future stock price based on given current price, expected return, volatility and the simulation step size. It will show the price path from now to a year later. Users can easily get the current price of the stock. By Using historical data such as ROE, Users can calculate the expected return, and can get a volatility as well from the notion of the standard deviation. Users can get other variables from the market data, but they have to choose themselves how many times they will simulate the model.

If the user input the simulation step size 10, it will operate the simulation 10 times. I used For/Next loop to build the 10 times simulation. I used simulation step size variable t and outer loop in the code to take a simulation step size that the users write.

Each simulation runs 250 times calculation in inner loop. In first iteration, the price S(T) would be the current price so I used If/Then function to use current price in the first iteration in the inner loop. From the second iteration, the loop calculates two factors. First, it calculates Z-value.

[1] rdraw = Application.NormSInv(rnd)

Using above code, it is able to draw a random number from the inverse of cumulative standardized normal distribution. Within inner loop, this statement inputs the random number between 0 and 1 in the rdraw variable which can be used as Z-value. To show the expected price in each day, I used the below code.

[2] Cells(28 + i, 2 + m) = Cells(27 + i, 2 + m) \* Exp((r - 0.5 \* (v ^ 2)) \* (1 / 250) + v \*

\_ Sqr(1 / 250) \* rdraw)

I converted the Brownian motion statement to VBA code. This code estimates the price of day ‘t’ using the price of the day ‘t-1’. The time increment of the code is one day, so I used 1/250 of period T in above Brownian motion (1). The old price which is the price of day ‘t-1’ is saved in the (27 + i, 2 + m) cell. The new price will be saved in the (28 + i, 2 + m) cell which is one cell below the previous price. After operating the inner loop for 250 times, it will input new simulation results in the next column, if the user’s step simulation variable is larger than 1.

To make a model, I set default value for the variables that I mentioned above but let users can change the variables on the spreadsheet. When the user click the ‘simulation’ button, the model will operate simulations according to the simulation step size that the users enter and it will show the each simulation results. The user can see the figures are changing according to the dates. It is hard to presume how the price will change only with the numbers, so I draw the line chart.

[3] Worksheets(“Question3”).Shapes.AddChart.Select

[4] ActiveChart.SetSourceData Source:=Range("B28").CurrentRegion

In code [3] Addchart function is used to draw the chart in the “Question3” sheet. (John GreenStephen, 2011) Drawing the chart, it is important to choose the source. In code [4], It choose the region that the cell(B28) belong to. I choose the cell(B28) because it is the cell which the price of first day of first simulation saved, so it can be recognized as the region regardless of what simulation step size the user have chosen, if it is not zero.

The model will generate the chart and data table every time the user clicks the ‘simulation’ button. I could insert the chart in the ‘insert’ tab in the Excel spreadsheet, but I decided to generate chart with VBA because the model can cope with various situation when I use VBA. For example, if I used an already-made-chart with the maximum band 20 and minimum band -10, the chart cannot show the price path properly, when the user input the current price larger than 20. In addition, if I used already-made-chart which has already-selected region, the chart cannot select the region properly after the simulation step size has changed. This method has the disadvantage of taking more time operating the model, but it is user friendly that it gives more choice to users and they don’t need to select the region every time they operate the model.

As the model, generate the chart and data table every time the user operating the model, it is needed to delete the exiting chart and data table. I used below codes.

[6] Rows(27 & ":" & 278).Select

[7] Selection.Clear

[8] Worksheets("Question3").ChartObjects(1).Delete

In code[6], it selects the rows from 27 to 278 where the data table belong, and delete the data in the selected rows. In code[8], it deletes the existing chart in the spread sheet named “Question3”.

Users can use this model before buying the equities, presuming the price after a year and the best or the worst scenario that users can encounter during the investment.

References

Glasserman, P. (2013). Monte Carlo Methods in Financial Engineering. In P. Glasserman, *Monte Carlo Methods in Financial Engineering* (p. 596).

John Green, S. B. (2011). Excel 2007 VBA Programmer's Reference. In S. B. John Green, *Excel 2007 VBA Programmer's Reference* (p. 1176).

# Question 4

## Overall structure

The aim of the herein code is that of giving the user control over his investment portfolio.

The investor will select its share of wealth to invest namely, the portfolio size, as well as category and asset class.

The user has control over the portion of monies to allocate to each category, the model of this particular code embodies: commodities, stock indices and bonds. The portion to the previous categories will be accessible through the worksheet ‘inputs’.

The commodity category is composed by four indices structured on rolling futures contracts, each composed by a different commodity: agricultures, copper, platinum and palladium. The stock indices category gives choice over three sectors of the economy: healthcare devices, consumer staples and utilities. The bonds category provides the user with two bond indices: one composed by treasury bills of intermediate maturity and the other by corporate fixed income.

The investor has also the freedom to allocate different portions of its wealth to every asset class described above, specifically in the worksheet ‘funds’.

Finally the code will run producing the output on the ‘portfolio’ worksheet building an investment portfolio according to the investor choices.

## Improved functionality

This model performs at best when used for structuring diverse portfolios. As the user will have more funds available to invest he will be able to simply add them to the initial input of the portfolio size, the same is also true for each category as well as asset class.

When compared to similar models programmed with Excel instead of VBA, the latter has the advantage of not showing the category or fund where monies were not allocated to the portfolio making output sheet clearer.

Further an Excel built model would most probably have all the inputs in a single worksheet to avoid changing it every time when building the portfolio, however, this will be confusing for the final user. On top of that when indeed the total funds are allocated to only few asset classes the output will result visually unstructured and dispersed.

## Code

In the first part the variables has been defined, those are needed to guide the entire flow of the coding throughout its sections. It is important in this section to clearly and concisely define a variable in order to be easily understandable from the user as well as easily typeable as so avoiding typo errors when repeating it in other sections.

Following are given instruction on how structuring the interface in the three different worksheets and the position of the inputs on the excel grid.

When certain inputs are to be used, the worksheet in which they are must be activated, later it is possible to proceed with the denomination and form of each variable to be used in that worksheet. Specifically, in the Read Inputs section, the percentages as of the portfolio value for each investment category are set as well as their value in monetary terms. For both percentage and dollar terms of the Bonds category a deductive approach has been adopted in attributing the percentage as to complete 100%.

When the portfolio sheet is activated specific cells are chosen for the output of related values as well as their precise format such as number of decimals, font and size.

We take the example of the commodity category to explain how a bridge is built between two worksheets and how to find a determined keyword, in a selected range of cells: typeR = Application.Match("Commodity", Range("A1:A300"), 0).

The if statement was chosen as method to condition the execution of the program flow.

In our specific case to find the keyword, in this case always commodity and its related value, indicating which worksheet to copy from and the one for its output: Sheets("Funds").Range("A" & r1 & ":B" & r1).Copy \_

Sheets("Portfolio").Range("A" & rOut)

The if conditions can be modelled with ‘else, true or false’ in order to stop them or go to another function, in the herein case ‘If Cells(r1, 1).Value = "" Then Exit For’ was used to end the loop and later on closed ‘End If, Next r1’.

In order to calculate, in this example the stock indices funds has been used another variations of the if flow: If Sheets("Inputs").Range("A" & r2).Value <> "" And \_

Sheets("Inputs").Range("C" & r2).Value <> 0 Then so as testing two statements and moving on to values and position determination.

Lastly, in the bottom part of the code we can find the instructions given by the code for different formatting of rows and columns alignment as well as font and sizes, this time the method employed was that of sub procedures. Its peculiarity is that of carrying out tasks and manipulating objects.

## Testing

It is really handy the verification that all the funds were allocated correctly since in

the “Portfolio” sheet is shown the portfolio value, that will correspond to the initial input.

Subsequently the user will easily change its inputs from both percentage and dollar allocation to the categories and funds being able to immediately see any wrongdoing given the fact that the final result would not correspond to what was originally planned as total investment.

# Question 7

Analysis of VBA macros:

## Declaring Variables

A screenshot of a cell phone

Description automatically generated

Two separate arrays were declared for stock prices and an array to store the time in each step. Other variables were declared and were assigned a value on the spreadsheet, therefore the cells changed to respond to the input data. Div and exdiv are input variables.

## Developing Binomial Option Tree

A screenshot of a cell phone

Description automatically generated

The code here describes the basis of the binomial option tree:

|  |  |
| --- | --- |
| Variable | Description |
| dt | The steps that will occur in years. |
| u | If the option goes up it will use the u multiplier. |
| d | If the option goes down it will use the d multiplier |
| emrdt | Discount factor to give the present value of option. |
| Su | Calculates the uncertain component. |
| P | The probability the tree will take an up (u) movement. |

The last line indicates that the code starts from time 0.

## Binomial Option Pricing Tree Calculations

A screenshot of a social media post

Description automatically generated

The For loops are used to create the stock price tree. At each step, calculations are made to find at what time is each step occurs and the present value of the dividends at this time. After this, the code calculates the values of the option for the tree of the uncertain part of the stock price. Then, the stock price is calculated. The four loops generate the option value tree which starts from maturity, this uses excels max function, the earlier nodes are calculated as present values of the expected values, this done for both American and European models. Then the code calculates the option value from the tree at time 0. Finally, the immediate exercise payoff is calculated this only occurs in the American Option Model, this is where the higher values are chosen at each node to decide whether the option should be exercised early or not.

## Conclusion

With a large number of steps the model can produce accurate estimates for the option value. The model can be changed to allow for other known dividends.

# Question 8

## VBA advantages

While VBA and Excel are similar in many ways, VBA represents a big advantage by automating procedures and therefore making financial models and analysis much faster and efficient. VBA automates usual Excel procedures enabling models to run faster, for instance by using shortcut keys.

Some differences in procedure include the use of loops, where the same variable is used to calculate outcomes under a certain condition: VBA wouldn’t store intermediate variables in its calculations. Moreover, VBA has to be given exact instructions for the order of the sequence to successfully implement the model. Thus, we’d have to make all values used in VBA readily available for calculations.

Perhaps some people may find Excel more convenient because there is no need of checking if all the variables used in calculations are correct and it is easy to overwrite variables to rectify calculations (Module handbook 2).

To answer question 8, we used ‘Sub’, a procedure, which contains most of VBA programs and is similar to a user-defined function (Module handbook 2). We used a VBA model which included Solver to optimise retirement plan calculations. Our approach and the order of procedures closely followed the ones which were used in class.

For this question, another alternative to a VBA model would be to input individual values in the cells, define functions for certain cells (for instance, Year Beginning Saving Balance for years 2 to 23), run Solver to find the desired future value of the Savings Balance, and from there find the required savings amount for year 1. Using VBA, however, we are able to run a program based on entered values and incorporate solver inside the code, so it is executed automatically when the macro is launched.

In the model, we used Option Explicit in order to specifically declare certain variables as Dim. Using this option, those items which were not explicitly declared as Dim are automatically treated as Variant type. This option also helps to avoid typing the name of an existing variable incorrectly. The advantage of using Dim at the module level is that Dim variables become available to all procedures within that module.

When creating the model, one problem which came up when we were trying the model for the first time was linked to successfully running Goal Seek. This was due to cell names which were not matching, and because Excel had issues renaming the relevant cells. However, it was then resolved by copying the module to a new worksheet.

# Conclusion and recommendations

VBA is critical in order to optimize Excel functions, also, if programmed efficiently it may provide real time updates and a much user-friendly interface.

A number of error can be avoided and detected immediately but most importantly modifying the function through the code is much quicker.

VBA is a language that everyone in the financial industry should know due to the sort of operations are carried which can be enhanced with this programming language.

At first learning and writing the code may be longer than a simple Excel however in the long term, using VBA a model can be applied to different dataset and kept in Excel for future usage.