ESLSCA
Business School
Paris

MBA 2 Trading et Finance de Marché
Projet final en informatique financière

Evaluation de produits dérivés (options, obligations et swaps)

Professeur

Monsieur MELLOUK

Élève

ZAETTA Paul

Année scolaire

2018 / 2019

Sommaire

Introduction

Ce projet porte sur la création d'un outil permettant à l'utilisateur d'évaluer certains produits dérivés tels que des options, des obligations et des swaps. Cet outil a été développé sous VBA (Visual Basic for Application). La programmation VBA est un langage orienté objet. Celui-ci nous a permis de réaliser une application aussi performante qu'ergonomique. L'outil développé se décompose en quatre applications.

Dans un premier temps, une application a été développée permettant à l'utilisateur de pouvoir valoriser de nombreuses options, telles que les options dites Vanilles c'est-à-dire les options classiques à l'aide du modèle de Black et Scholes ou du modèle Binomial (options européennes ou américaines). Il est également possible d'évaluer des options exotiques telles que les options à barrières, les options asiatiques et les options binaires. La volatilité implicite représentant la volatilité de l'option issue des conditions de marché (rentrées au préalable par l'utilisateur) peut être calculée. Cependant, cette volatilité implicite est uniquement quantifiable sous certaines conditions.
Dans un second temps, une autre application permet de réaliser des simulations de Monte Carlo. Ces simulations permettent de valoriser des options européennes selon le modèle log-normal et le modèle de diffusion à sauts de Merton.
Une troisième application évalue le prix des obligations en fonction de la périodicité des versements des intérêts, des taux d'actualisation, des taux de coupons, du nominal et de la maturité définis par l'utilisateur. D'autres caractéristiques à l'obligation sont aussi calculées telles que son taux de rendement actuariel, sa duration, sa sensibilité ainsi que sa convexité.
Enfin, une dernière application a été développée, celle-ci permettant de calculer la valeur d'un swap.

I. Evaluation des options

Cette première application permet d'évaluer des options (options d'achat ou options de vente) dont le sous-jacent peut être une action, une devise, un indice ou un future. L'évaluation des options consiste à déterminer la prime c'est-à-dire le prix à débourser pour acquérir l'option. L'application permet également d'évaluer les « grecques » associées à l'option. Les « grecques » représentent les sensibilités du prix de l'option par rapport au cours du sous-jacent, de la volatilité, du temps et du taux sans risque.

L'essentiel des modèles d'évaluation ont été développés pour un sous-jacent de type action. On trouve le modèle de Black et Scholes, le modèle binomial (pour les options européennes et américaines). Le pricer nous permet également de valoriser les options asiatiques, les options à barrières activantes (vers le haut) et désactivantes (vers le haut et vers le bas) ainsi que les options binaires (cash-or-nothing et asset-or-nothing). Cette application offre aussi la possibilité de valoriser via le modèle de Black et Scholes les options européennes avec comme sous-jacent une devise, un indice ou encore un future.

L'interface du pricer d'options se présente comme suit,

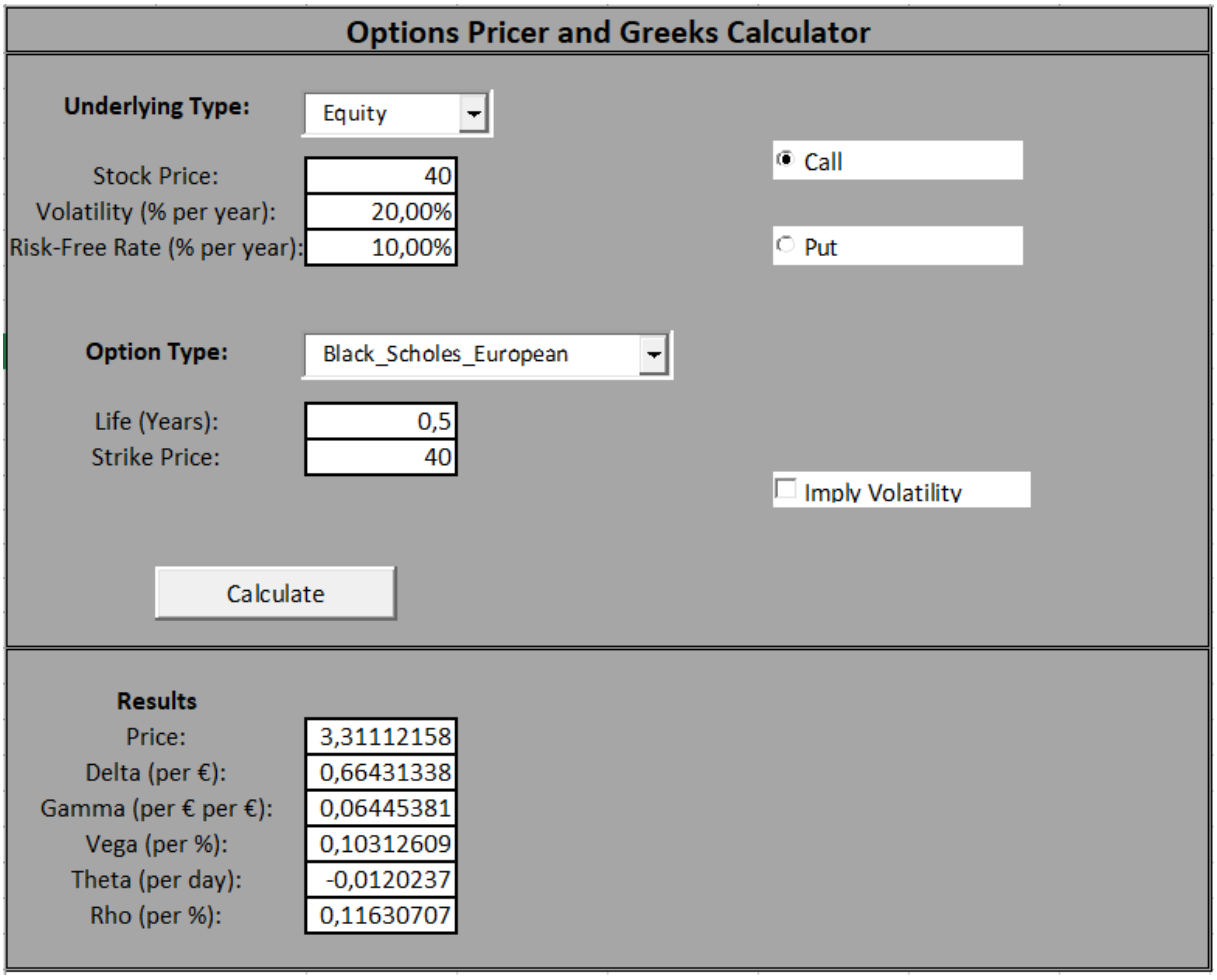

Illustration 1.

L'application offre donc la possibilité à l'utilisateur de choisir le type d'option, le type de sous-jacent et également le type de modèle. Il existe aussi la possibilité de calculer la volatilité implicite de l'option

(en cochant la CheckBox). Ce calcul est uniquement possible lorsque le sous-jacent est une action et que le modèle de Black et Scholes est sélectionné. Il faut remplir au préalable le prix de l'option, le prix du sous-jacent, le taux sans risque, la maturité et le prix d'exercice. La volatilité implicite calculée sera affichée dans la cellule C6. Dès lors, que nous sommes plus dans cette configuration la CheckBox n'est plus visible. Nous détaillons le code pour cette partie sur la volatilité implicite à la page 10.
Lorsque nous lançons le programme (en ayant au préalable rempli toutes les cellules nécessaires), le prix de l'option est affiché dans la cellule C21. Tandis que les « grecques » le delta, le gamma, le théta, le vega et le rho sont affichées respectivement dans les cellules C22, C23, C24, C25 et C26. Pour rappel, le delta et le gamma représentent respectivement la sensibilité et la vitesse de la variation du prix de l'option par rapport au prix du sous-jacent. Le vega correspond à la sensibilité du prix de l'option par rapport à la volatilité et le theta représente la sensibilité du prix de l'option par rapport à l'échéance. Et enfin, le rho indique la sensibilité du prix de l'option par rapport au taux sans risque.
Nous allons dans un premier temps expliquer le code qui permet à l'interface de s'ajuster automatiquement selon les choix de l'utilisateur.

```
' Cette procédure actualise la ComboBox pour "Option Type" quand on change l'input de la ComboBox associée à "Underlying Type"
Private Sub ComboBox1_Change()

Workbooks("vba_project").Sheets("options_calculator").Activate

X = ComboBox1.Value

Select Case X
       Case Is = "Equity"
            form11 'Appel de la procédure form11 située dans un module (MEF1_Pricer) si "Equity" est sélectionné
            ComboBox2.ListFillRange = "My_data!B2:B11"
            CheckBox1.Visible = True
       Case Is = "Currency"
            form12 'Appel de la procédure form12 située dans un module (MEF1_Pricer) si "Currency" est sélectionné
            ComboBox2.Value = "Black_Scholes_European"
            ComboBox2.ListFillRange = "My_data!B2"
            CheckBox1.Visible = False
       Case Is = "Index"
            form13 'Appel de la procédure form13 située dans un module (MEF1_Pricer) si "Index" est sélectionné
            ComboBox2.Value = "Black_Scholes_European"
            ComboBox2.ListFillRange = "My_data!B2"
            CheckBox1.Visible = False
       Case Is = "Futures"
            form14 'Appel de la procédure form14 située dans un module (MEF1_Pricer) si "Futures" est sélectionné
            ComboBox2.Value = "Black_Scholes_European"
            ComboBox2.ListFillRange = "My_data!B2"
            CheckBox1.Visible = False
End Select

End Sub
```

Le code suivant permet donc de modifier l'interface lorsque l'utilisateur change le type de sous-jacent. Il peut choisir entre un sous-jacent de type action, devise, indice ou future. Lorsque l'utilisateur choisit le sous-jacent de type action via la ComboBox1, le programme lance une procédure et deux instructions. La procédure appelée se nomme « form11 » (contenue dans le module « MEF1_Pricer ») et contient le code qui suit,

```
Sub form11() ' Forme initiale pour "Equity"

    Workbooks("vba_project").Sheets("options_calculator").Activate

    ActiveSheet.Range("A8:C8").Select
    Selection.Value = ""
    Selection.Interior.Color = RGB(166, 166, 166)
    For Each cl In Selection
         cl.Borders.LineStyle = xlLineStyleNone
    Next cl

    ActiveSheet.Range("C5").Select

End Sub
```

Ce code sélectionne dans un premier temps la plage de cellules A8 à C8 de l'onglet « options_calculator ». Ensuite, il vide ces cellules et rétablit la couleur d'origine en fond de cellule. En enfin, il oblige l'absence de contour sur ces cellules et sélectionne la cellule C5.

L'instruction qui suit l'appel de la procédure sert à fixer les modèles que l'utilisateur peut sélectionner. Pour cela nous allons dans la propriété « ListFillRange » de l'objet ComboBox2 puis rentrons les cellules qu'il doit sélectionner. Dans ce cas-là, la ComboBox2 affichera les valeurs de la plage de cellules B2 à B11 de la feuille « My_data ». La deuxième instruction permet d'afficher la CheckBox correspondant au calcul de la volatilité implicite.

Lorsque l'utilisateur choisit le sous-jacent de type devise via la ComboBox1, le programme lance une procédure et trois instructions. La procédure appelée se nomme « form12 » (contenue dans le module « MEF1_Pricer » et contient le code qui suit,

```
Sub form12() ' Forme initiale pour "Currency"

    Workbooks("vba_project").Sheets("options_calculator").Activate

    ActiveSheet.Range("A8").Select
    With Selection
         .Value = "Foreign RFR (% per year)"
         .Characters.Font.Size = 11
         .HorizontalAlignment = xlCenter
         .VerticalAlignment = xlCenter
         .Font.Color = RGB(0, 0, 0)
    End With

    ActiveSheet.Range("C8").Select
    With Selection
         .Font.Color = RGB(0, 0, 0)
         .Interior.Color = RGB(255, 255, 255)
         .BorderAround Weight:=xlMedium
    End With

End Sub
```

Ce code sélectionne dans un premier temps la cellule A8, et il y affecte une valeur via la propriété « Value », puis effectue une mise en forme (police et fond de cellule). Ensuite, il sélectionne la cellule C8 et travaille de nouveau sur la mise en forme.

Lorsque l'utilisateur choisit un sous-jacent de type indice ou future le programme lance des procédures et des instructions très similaires à celles déjà rencontrées concernant l'ajustement de l', c'est pour cela que nous n'allons pas les détailler.

La prochaine partie de code que nous présentons ci-dessous correspond à l'agencement de l'interface quand l'utilisateur choisit un certain type d'option. Pour rappel l'utilisateur a le choix pour le sous-jacent de type action les modèles suivants : Black et Scholes, Binomial Européen, Binomial Américain, Asiatique, Barrière désactivante vers le haut, Barrière activante vers le bas, Barrière désactivante vers le bas, Binaire Cash-Or-Nothing et Binaire Asset-Or-Nothing.

```
' Cette procédure modifie l'agencement quand on change l'input de la ComboBox associée à "Option Type"
Private Sub ComboBox2_Change()

Workbooks("vba_project").Sheets("options_calculator").Activate

X = ComboBox2.Value

Select Case X
       Case Is = "Black_Scholes_European"
            form21 'Appel de la procédure form21 située dans un module le modèle "Black_Scholes_European" est sélectionné
            CheckBox1.Visible = True
       Case Is = "Binomial_European", "Binomial_American"
            form22 'Appel de la procédure form22 située dans un module si le modèle "Binomial" est sélectionné
            CheckBox1.Visible = False
       Case Is = "Asian"
            form23 'Appel de la procédure form23 située dans un module si le modèle "Asian" est sélectionné
            CheckBox1.Visible = False
       Case Is = "Barrier_Up_And_In", "Barrier_Up_And_Out", "Barrier_Down_And_In", "Barrier_Down_And_Out"
            form24 'Appel de la procédure form24 située dans un module si le modèle "Barrier" est sélectionné
            CheckBox1.Visible = False
       Case Is = "Binary_Cash_Or_Nothing"
            form25 'Appel de la procédure form25 située dans un module si le modèle "Binary_Cash_Or_Nothing" est sélectionné
            CheckBox1.Visible = False
       Case Is = "Binary_Asset_Or_Nothing"
            form26 'Appel de la procédure form26 située dans un module si le modèle "Binary_Asset_Or_Nothing" est sélectionné
            CheckBox1.Visible = False
End Select

End Sub
```

Lorsque l'utilisateur choisit l'un des modèles, une procédure et une instruction sont appelées. L'instruction va permettre d'afficher la CheckBox1 (liée au calcul de la volatilité implicite) lorsque que la propriété de celle-ci « Visible » est vraie, sinon elle ne sera pas affichée. Quand à la procédure, elle va charger l'interface selon le modèle de façon similaire à la procédure « ComboBox1_Change() ».

Les prochaines parties de code décrites ci-dessous concernent l'appel de la procédure adéquate aux choix de l'utilisateur permettant de calculer le prix et les « grecques » de l'option. La procédure se lance lorsque l'utilisateur clique sur le bouton « Calculate ».

```
' Cette procédure lance le programme qui calcule le prix de l'option et ses grecques selon le sous-jacent, le modèle
' et si on a sélectionné l'option d'achat ou l'option de vente lorsqu'on clique sur "Calculate".
Private Sub CommandButton1_Click()

Workbooks("vba_project").Sheets("options_calculator").Activate

Dim X$

X = ComboBox1.Value

Select Case X
       Case Is = "Equity" ' Underlying Type : Equity
            If ComboBox2.Value = "Black_Scholes_European" And OptionButton1.Value = True And CheckBox1.Value = False Then
               call_price_greeks_BS_model
            ElseIf ComboBox2.Value = "Black_Scholes_European" And OptionButton2.Value = True And CheckBox1.Value = False Then
               put_price_greeks_BS_model
            ElseIf ComboBox2.Value = "Black_Scholes_European" And OptionButton1.Value = True And CheckBox1.Value = True Then
               call_implied_volatility_BS_model
            ElseIf ComboBox2.Value = "Black_Scholes_European" And OptionButton2.Value = True And CheckBox1.Value = True Then
               put_implied_volatility_BS_model
            ElseIf ComboBox2.Value = "Binomial_European" And Range("C14").Value < 2 _
               Or ComboBox2.Value = "Binomial_American" And Range("C14") < 2 Then
               alertTreeNumberStep.Show 0
            ElseIf ComboBox2.Value = "Binomial_European" And Range("C14").Value > 500 Or _
               ComboBox2.Value = "Binomial_American" And Range("C14") > 500 Then
               alertTreeNumberStep2.Show 0
            ElseIf ComboBox2.Value = "Binomial_European" And OptionButton1.Value = True Then
               call_price_Binomial_European_model
               greeks_call_Binomial_European_model
            ElseIf ComboBox2.Value = "Binomial_European" And OptionButton2.Value = True Then
               put_price_Binomial_European_model
               greeks_put_Binomial_European_model
            ElseIf ComboBox2.Value = "Binomial_American" And OptionButton1.Value = True Then
               call_price_Binomial_American_model
               greeks_call_Binomial_American_model
            ElseIf ComboBox2.Value = "Binomial_American" And OptionButton2.Value = True Then
               put_price_Binomial_American_model
               greeks_put_Binomial_American_model
```

```
            ElseIf ComboBox2.Value = "Barrier_Down_And_Out" And Range("C5") - 1 < Range("C14") Then
                alertDAOBarrier.Show 0
            ElseIf ComboBox2.Value = "Barrier_Down_And_Out" And OptionButton1.Value = True Then
               call_price_Barrier_Down_And_Out_model
            ElseIf ComboBox2.Value = "Barrier_Down_And_Out" And OptionButton2.Value = True Then
               put_price_Barrier_Down_And_Out_model
            ElseIf ComboBox2.Value = "Barrier_Up_And_Out" And Range("C5") + 2 >= Range("C14") Then
               alertUAOBarrier.Show 0
            ElseIf ComboBox2.Value = "Barrier_Up_And_Out" And OptionButton1.Value = True Then
               call_price_Barrier_Up_And_Out_model
            ElseIf ComboBox2.Value = "Barrier_Up_And_Out" And OptionButton2.Value = True Then
               put_price_Barrier_Up_And_Out_model
            ElseIf ComboBox2.Value = "Asian" And OptionButton1.Value = True Then
               call_price_greeks_Asian_model
            ElseIf ComboBox2.Value = "Asian" And OptionButton2.Value = True Then
               put_price_greeks_Asian_model
            ElseIf ComboBox2.Value = "Barrier_Down_And_In" And OptionButton1.Value = True Then
               call_price_greeks_Barrier_Down_And_In_model
            ElseIf ComboBox2.Value = "Barrier_Down_And_In" And OptionButton2.Value = True Then
               put_price_greeks_Barrier_Down_And_In_model
            ElseIf ComboBox2.Value = "Barrier_Up_And_In" And OptionButton1.Value = True Then
               call_price_greeks_Barrier_Up_And_In_model
            ElseIf ComboBox2.Value = "Barrier_Up_And_In" And OptionButton2.Value = True Then
               put_price_greeks_Barrier_Up_And_In_model
            ElseIf ComboBox2.Value = "Binary_Cash_Or_Nothing" And OptionButton1.Value = True Then
               call_Binary_Cash_Or_Nothing_model
            ElseIf ComboBox2.Value = "Binary_Cash_Or_Nothing" And OptionButton2.Value = True Then
               put_Binary_Cash_Or_Nothing_model
            ElseIf ComboBox2.Value = "Binary_Asset_Or_Nothing" And OptionButton1.Value = True Then
               call_Binary_Asset_Or_Nothing_model
            ElseIf ComboBox2.Value = "Binary_Asset_Or_Nothing" And OptionButton2.Value = True Then
               put_Binary_Asset_Or_Nothing_model
            End If


        Case Is = "Currency" ' Underlying Type : Currency
             If ComboBox2.Value = "Black_Scholes_European" And OptionButton1.Value = True Then
                call_price_greeks_BS_Currency_model
             ElseIf ComboBox2.Value = "Black_Scholes_European" And OptionButton2.Value = True Then
                put_price_greeks_BS_Currency_model
             End If
        Case Is = "Index" ' Underlying Type : Index
             If ComboBox2.Value = "Black_Scholes_European" And OptionButton1.Value = True Then
                call_price_greeks_BS_Index_model
             ElseIf ComboBox2.Value = "Black_Scholes_European" And OptionButton2.Value = True Then
                call_price_greeks_BS_Index_model
             End If
        Case Is = "Futures" 'Underlying Type : Futures
             If ComboBox2.Value = "Black_Scholes_European" And OptionButton1.Value = True Then
                call_price_greeks_BS_Futures_model
             ElseIf ComboBox2.Value = "Black_Scholes_European" And OptionButton2.Value = True Then
                put_price_greeks_BS_Futures_model
             End If
End Select

End Sub
```

Ce programme va dans un premier temps regarder la valeur stockée dans la ComboBox1 référençant le type de sous-jacent. Selon la valeur contenue, il va directement aller dans la partie du code correspondante à sa valeur. Ceci se fait à l'aide d'un « Select Case ». Par la suite, il va à l'aide d'une instruction conditionnelle « If » trouver la procédure qui va permettre la valorisation de l'option selon les choix de l'utilisateur.
En résumé, il va attribuer à la variable X (de type string) la valeur associée à la ComBox1 c'est-à-dire le type de sous-jacent. Le « Select Case » va ensuite regarder si la variable X est égale à « Equity » si c'est le cas il rentre dans cette partie de code, sinon, il la quitte et va comparer X à « Currency » et ainsi de suite avec « Index » et « Futures ». Lorsque la variable X correspond au type de sous-jacent dans le « Select Case », il exécute le code de cette « Case » puis va directement au « End Select ». Lorsqu'il rentre dans une « Case » il a face à lui une instruction conditionnelle « If ».
Prenons l'exemple que la variable X est égale à « Equity ». Le code va donc rentrer dans le premier « Case » et procéder à des comparaisons. La première procédure « call_price_greeks_BS_model() » est lancée si les trois conditions suivantes sont respectées :

- le modèle sélectionné par l'utilisateur est Black et Scholes (ComboBox2)
- l'utilisateur souhaite valoriser une option d'achat (OptionButton1)
- le calcul de la volatilité implicite n'est pas sélectionné (CheckBox1)

Si l'une de ces conditions n'est pas respectée, alors le code va dans la prochaine instruction conditionnelle. Il va donc au prochain « Elself » et procède une nouvelle fois à la respectabilité des conditions. Dès lors que les conditions sont respectées, la procédure qui en découle est lancée.
Il est important de noter que lorsque l'utilisateur souhaite un sous-jacent de type « Equity » et le modèle binomial (Européen et Américain) il doit choisir le nombre d'itérations pour l'arbre binomial. Le nombre d'itérations admissible dans cette application est compris entre 2 et 500. Dans le cas où il ne choisit pas un nombre appartenant à cette intervalle, le programme va lancer une alerte sous forme d'un « UserForm », comme nous pouvons le voir ci-dessous,

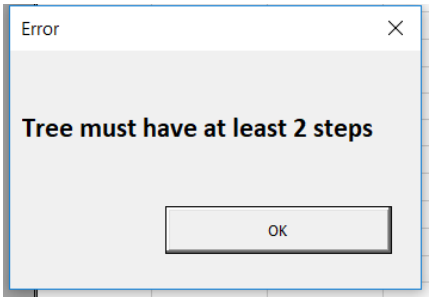

Illustration 2.

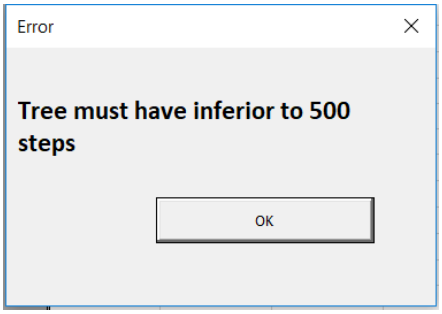

Illustration 3.

Une spécificité à ces « UserForms » a été rajoutée consistant à obliger l'utilisateur à cliquer sur le bouton « OK » et non sur la croix pour quitter le message d'alerte. Le code permettant cela est le suivant,

```
' Cette Sub oblige l'utilisateur à fermer l'Userform en cliquant sur la touche OK
Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)

If CloseMode = 0 Then
   MsgBox ("Click on OK to close the box")
   Cancel = 1
End If

End Sub
```

Par conséquent, tant que l'utilisateur ne rentre pas un nombre d'itérations admissible, le modèle binomial tant bien Européen ou Américain ne s'exécute pas.

Nous allons à présent décrire en détail pour une option d'achat chaque modèle lorsque le sous-jacent est de type « Equity ». Le code pour les options de vente n'est pas détaillé, en effet le code varie très peu par rapport à celui correspondant aux options d'achat. On note également, que plus nous avançons dans l'explication du code moins nous expliquons les instructions de bases et celles déjà vues.

A. Modèle de Black et Scholes sur option Vanille

```
Sub call_price_greeks_BS_model()

Dim spot#, volatility#, riskFreeRate#, life#, strike#, phi1#, d1#, d2#, cdf11#, cdf12#, Pi#

Workbooks("vba_project").Sheets("options_calculator").Activate

spot = ActiveSheet.Range("C5").Value 'Prix au comptant
volatility = ActiveSheet.Range("C6").Value 'Volatilité
riskFreeRate = ActiveSheet.Range("C7").Value 'Taux d'intérêt sans risque
life = ActiveSheet.Range("C12").Value 'Maturité
strike = ActiveSheet.Range("C13").Value 'Prix d'exercice

d1 = ((Log(spot / strike)) + ((riskFreeRate + ((volatility ^ 2) / 2)) * life)) / (volatility * Sqr(life))
d2 = d1 - (volatility * Sqr(life))

Pi = WorksheetFunction.Pi 'Nombre Pi
phi1 = Exp(-(d1 ^ 2 / 2)) / Sqr(2 * Pi)

cdf11 = Application.WorksheetFunction.Norm_Dist(d1, 0, 1, True) 'Fonction de répartition de la loi normale
cdf12 = Application.WorksheetFunction.Norm_Dist(d2, 0, 1, True) 'Fonction de répartition de la loi normale
```

Cette première partie du code déclare tout d'abord plusieurs variables de type « Double » grâce au caractère « # ». Ces variables prennent comme valeurs les données fixées par l'utilisateur dans la feuille Excel « options_calculator ». Il calcule ensuite les deux composantes « d1 » et « d2 » du modèle de Black et Scholes. La variable « Pi » est affectée de la constante d'Archimède via la fonction « .Pi ». La procédure affecte aussi une certaine valeur à la variable « phi1 » selon le modèle de Black et Scholes, puis calcule deux fonctions de répartition de la loi normale en les affectant aux variables « cdf1 » et « cdf2 ».

```
'Prix de l'option d'achat sur action
ActiveSheet.Range("C21").Select
Selection = (spot * cdf11) - (strike * Exp(-(riskFreeRate * life)) * cdf12)

' Delta
ActiveSheet.Range("C22").Select
Selection = cdf11

' Gamma
ActiveSheet.Range("C23").Select
Selection = phi1 / (spot * volatility * Sqr(life))

' Vega
ActiveSheet.Range("C24").Select
Selection = spot * phi1 * Sqr(life) / 100

' Theta
ActiveSheet.Range("C25").Select
Selection = (-((spot * phi1 * volatility) / (2 * Sqr(life))) - (riskFreeRate * strike * Exp(-riskFreeRate * life) * cdf12)) / 365

' Rho
ActiveSheet.Range("C26").Select
Selection = (strike * life * Exp(-riskFreeRate * life) * cdf12) / 100

End Sub
```

La seconde partie du code calcule le prix de l'option d'achat et ses « grecques ». Le prix de l'option est ainsi affecté à la cellule C21 de la feuille Excel « options_calculator ». Les « grecques » sont respectivement affectées aux cellules C22, C23, C24, C25 et C26 de la présente feuille.

Nous avons cité au début de ce chapitre la possibilité à l'utilisateur de calculer la volatilité implicite de l'option lorsque le sous-jacent est une action et que le modèle de Black et Scholes est choisi. Pour lancer ce code il est nécessaire que l'utilisateur spécifie le prix de l'option ainsi que le prix au comptant de l'action, le taux sans risque, la maturité et le prix d'exercice. La fonction permettant ce calcul se situe dans le module « Functions3 », le code de cette fonction se définit comme suit,

```
Sub call_implied_volatility_BS_model()

Dim spot#, riskFreeRate#, life#, optionValue#, strike#, impliedVolatility#, guess#

Workbooks("vba_project").Sheets("options_calculator").Activate

spot = ActiveSheet.Range("C5").Value 'Prix au comptant
riskFreeRate = ActiveSheet.Range("C7").Value 'Taux d'intérêt sans risque
life = ActiveSheet.Range("C12").Value 'Maturité
strike = ActiveSheet.Range("C13").Value 'Prix d'exercice
optionValue = ActiveSheet.Range("C21").Value 'Prix de l'option

guess = 0.2 'On assume une volatilité égale à 20% sur le marché action

While (optionValue < callPriceBlackScholes(spot, strike, 0.001, riskFreeRate, life))
MsgBox ("The option value is to small, you have to select a higher price.")
optionValue = InputBox("New option value: ")
Wend

ActiveSheet.Range("C21").Value = optionValue

' Volatilité implicite de l'option d'achat avec le modèle Black-Scholes
impliedVolatility = callImpliedVolatilityBlackScholes(spot, strike, riskFreeRate, life, optionValue, guess)
ActiveSheet.Range("C6").Select
Selection = impliedVolatility

End Sub
```

Cette procédure récupère dans un premier temps les valeurs que l'utilisateur a entré. Elle spécifie également une valeur « guess » représentant une valeur moyenne de la volatilité implicite sur le marché actions. Cette valeur subjective est fixée à 20%, elle a été fixée selon les données historiques et sert de point de départ à l'algorithme (voir page 12). Une boucle « While » est ensuite utilisée pour vérifier la cohérence des données entrées par l'utilisateur.
Le prix de l'option doit bien évidemment être supérieur à la valeur intrinsèque de l'option c'est-à-dire la différence entre le prix au comptant et le prix d'exercice dans le cas d'une option d'achat et inversement pour une option de vente. De plus, le prix de l'option ne peut pas être inférieur au prix d'une option équivalente avec une volatilité quasiment nulle. La boucle « While » veille à ce que ces conditions soit respectées. Si ce n'est pas le cas, on demande à l'utilisateur de rentrer un prix de l'option plus élevé.

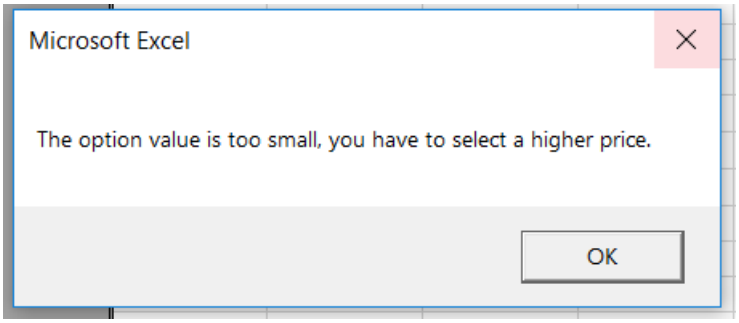

Illustration 4.

Suite à cette alerte, une autre fenêtre s'ouvre via un « InputBox ». Celle-ci demande à l'utilisateur de rentrer un nouveau prix d'option (voir Illustration 5). Ce nouveau prix sera ensuite affecté à la cellule C21.

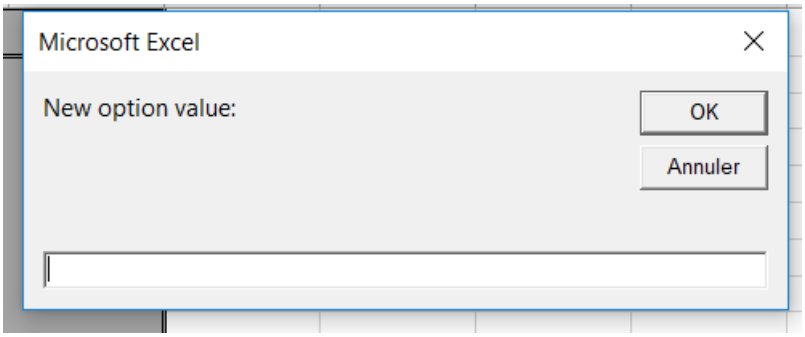

Illustration 5.

La suite de la procédure permet d'évaluer la volatilité implicite de l'option. La valeur calculée est stockée dans la variable « impliedVolatility ». Pour réaliser ce calcul, la fonction « callImpliedVolatilityBlackScholes » est appelée. Cette fonction est basée sur la méthode de Newton-Raphson. Cette fonction essaye de déterminer le niveau de volatilité correspondant au prix de l'option d'achat selon le modèle de Black et Scholes (cf code ci-dessous). Cette fonction contient une boucle de type « Do…Loop », le bloc d'instruction est donc répété jusqu'à ce que son objectif soit atteint. Dans notre cas, il faut que soit la variable « dx » soit inférieure à un certain seuil, soit si le nombre d'itérations maximal est atteint. En conclusion, cette fonction va déterminer la volatilité implicite via une méthode d'itération c'est-à-dire par tâtonnement.

```
Function callImpliedVolatilityBlackScholes(v1#, v2#, v3#, v4#, v5#, v6#) As Double

' Cette fonction calcule la volatilité implicite d'une option d'achat avec le modèle Black-Scholes
'
' INPUTS
'---------------------------------------------------------------
' v1 : prix au comptant
' v2 : prix d'exercice
' v3 : taux d'intérêt sans risque
' v4 : maturité
' v5 : prix de l'option
' v6 : valeur de départ de la volatilité
'
' OUTPUT
'---------------------------------------------------------------
'
' volatilité implicite de l'option d'achat
'
'---------------------------------------------------------------

Dim epsilon, dVol, vol_1#, i%, maxIter#, Value_1#, vol_2#, Value_2#, dx#

dVol = 0.0000001
epsilon = 0.0000001
maxIter = 1000
vol_1 = v6
i = 1

Do
   Value_1 = callPriceBlackScholes(v1, v2, vol_1, v3, v4)
   vol_2 = vol_1 - dVol
   Value_2 = callPriceBlackScholes(v1, v2, vol_2, v3, v4)
   dx = (Value_2 - Value_1) / dVol
   If Abs(dx) < epsilon Or i = maxIter Then Exit Do
   vol_1 = vol_1 - (v5 - Value_1) / dx
   i = i + 1
Loop
callImpliedVolatilityBlackScholes = vol_1
End Function
```

B. Modèle binomial sur option Vanille européenne

La procédure « call_price_Binomial_European_model() » ci-dessous permet de calculer le prix d'une option d'achat de type européenne selon le modèle binomial. Le début de la procédure déclare plusieurs variables de type « Double » et « Integer » grâce au caractère « # » et « % ». Les probabilités de hausse et de baisse du prix du sous-jacent sont calculées et sont respectivement affectées aux variables « prob » et « prob_ ». Une boucle « For » qui dépend de la taille de l'arbre binomial (définie par l'utilisateur) est ensuite utilisée pour calculer tous les prix du sous-jacent de l'arbre binomial. Ces prix sont stockés dans le tableau nommé « tableau ».

```
Sub call_price_Binomial_European_model()

Dim spot#, strike#, volatility#, riskFreeRate#, life#, up#, down#, prob#, prob_#, timeStep#, treeSteps%

Workbooks("vba_project").Sheets("options_calculator").Activate

spot = ActiveSheet.Range("C5").Value 'Prix au comptant
volatility = ActiveSheet.Range("C6").Value 'Volatilité
riskFreeRate = ActiveSheet.Range("C7").Value 'Taux d'intérêt sans risque
life = ActiveSheet.Range("C12").Value 'Maturité
strike = ActiveSheet.Range("C13").Value 'Prix d'exercice
treeSteps = ActiveSheet.Range("C14").Value 'Nombre d'itérations de l'arbre binomial

up = Exp(volatility * Sqr(life / treeSteps))
down = 1 / up

prob = (Exp(riskFreeRate * life / treeSteps) - down) / (up - down) 'Probabilité d'un mouvement de hausse
prob_ = 1 - prob 'Probabilité d'un mouvement de baisse

Dim tableau() As Double
ReDim tableau(0 To treeSteps)

For i = 0 To treeSteps Step 1 'Boucle qui stocke les valeurs finales de l'arbre binomial dans 'tableau'
    tableau(i) = spot * (down ^ i) * (up ^ (treeSteps - i))
    If tableau(i) < strike Then
    tableau(i) = 0
    Else
    tableau(i) = tableau(i) - strike
    End If
Next i

timeStep = life / treeSteps

For i = 1 To treeSteps Step 1 'Boucle qui détermine le prix initial de l'option d'achat (backward-looking)
    For j = 0 To treeSteps - i Step 1
        tableau(j) = Exp((-riskFreeRate) * timeStep) * (prob * tableau(j) + prob_ * tableau(j + 1))
    Next j
Next i

ActiveSheet.Range("C21").Select
Selection = tableau(0)

End Sub
```

Une fois tous les prix du sous-jacent déterminés pour chaque « branche » de l'arbre binomial, une instruction conditionnelle est appliquée. Cette instruction va tester que chaque prix est supérieur au prix d'exercice. Si c'est le cas, le prix de l'option à maturité est égal à la différence de ces deux valeurs, sinon le prix est égal à zéro. Une fois cette étape faite, la procédure calcule le prix actuel de l'option. Pour cela, une méthode dite « backward-looking » est appliquée. Cette méthode fonctionne par itération. L'algorithme part des prix finaux de l'arbre binomial pour déterminer le prix de l'option pour chaque période précédente, en appliquant une actualisation avec le taux sans risque des pay-offs pondérés par leurs probabilités d'occurrences. Cette méthode boucle jusqu'à la date initiale afin de calculer le prix actuel de l'option. La valeur de l'option d'achat européenne est affectée à la cellule « C21 » de la feuille Excel « options_calculator ».

C. Modèle binomial sur option Vanille américaine

La procédure « call_price_Binomial_American_model() » ci-dessous permet de calculer le prix d'une option d'achat de type américaine selon le modèle binomial. Le code est très proche de celui présenté dans la partie précédente. La différence ici avec une option américaine est que l'utilisateur inclut la possibilité d'exercer l'option durant le cours de sa vie. Cette procédure prend en compte cette caractéristique dans le calcul de la valeur de l'option. Le code de cette procédure se décrit comme suit,

```
Sub call_price_Binomial_American_model()

Dim spot#, strike#, volatility#, riskFreeRate#, life#, up#, down#, prob#, prob_#, timeStep#, treeSteps%

Workbooks("vba_project").Sheets("options_calculator").Activate

spot = ActiveSheet.Range("C5").Value 'Prix au comptant
volatility = ActiveSheet.Range("C6").Value 'Volatilité
riskFreeRate = ActiveSheet.Range("C7").Value 'Taux d'intérêt sans risque
life = ActiveSheet.Range("C12").Value 'Maturité
strike = ActiveSheet.Range("C13").Value 'Prix d'exercice
treeSteps = ActiveSheet.Range("C14").Value 'Nombre d'itérations de l'arbre binomial

up = Exp(volatility * Sqr(life / treeSteps))
down = 1 / up

prob = (Exp(riskFreeRate * life / treeSteps) - down) / (up - down) 'Probabilité d'un mouvement de hausse
prob_ = 1 - prob 'Probabilité d'un mouvement de baisse

Dim tableau() As Double
ReDim tableau(0 To treeSteps)

For i = 0 To treeSteps Step 1 'Boucle qui stocke les valeurs finales de l'arbre binomial dans 'tableau'
    tableau(i) = spot * (down ^ i) * (up ^ (treeSteps - i))
    If tableau(i) < strike Then
    tableau(i) = 0
    Else
    tableau(i) = tableau(i) - strike
    End If
Next i


timeStep = life / treeSteps

For i = 1 To treeSteps Step 1 'Boucle qui détermine le prix initial de l'option d'achat (backward-looking)
    For j = 0 To treeSteps - i Step 1
        tableau(j) = Exp((-riskFreeRate) * timeStep) * (prob * tableau(j) + prob_ * tableau(j + 1))
        If tableau(j) < (spot * (up ^ (treeSteps - i - j)) * (down ^ j)) - strike Then
        tableau(j) = (spot * (up ^ (treeSteps - i - j)) * (down ^ j)) - strike
        End If
    Next j
Next i

ActiveSheet.Range("C21").Select
Selection = tableau(0)

End Sub
```

La différence par rapport à l'option européenne intervient dans la dernière partie du code. La boucle « For » calcule le prix actuel de l'option via la méthode « backward-looking » qui fonctionne par itération comme vu dans la partie précédente. Une instruction conditionnelle est rajoutée par rapport au calcul de l'option européenne. Elle vérifie à chaque nœud de l'arbre binomial l'importance du gain, c'est-à-dire s'il est préférable d'exercer l'option au nœud, si c'est le cas la valeur de l'option à ce nœud doit être modifiée. La valeur de l'option d'achat américaine via le modèle binomial est affectée à la cellule « C21 » de la feuille Excel « options_calculator ».

D. Évaluation des options asiatiques

```
Sub call_price_greeks_Asian_model()

Dim spot#, spot2#, spot3#, volatility#, volatility2#, riskFreeRate#
Dim riskFreeRate2#, life#, life2#, strike#, timeSinceInception#, timeSinceInception2#
Dim currentAverage#, currentAverage2#, price#, price2#, price3#, delta#, delta2#

Workbooks("vba_project").Sheets("options_calculator").Activate

spot = ActiveSheet.Range("C5").Value 'Prix au comptant
spot2 = spot + 1 'Prix au comptant + 1
spot3 = spot + 2 'Prix au comptant + 2
volatility = ActiveSheet.Range("C6").Value 'Volatilité
volatility2 = volatility + 0.01 'Volatilité + 1%
riskFreeRate = ActiveSheet.Range("C7").Value 'Taux d'intérêt sans risque
riskFreeRate2 = riskFreeRate + 0.01 'Taux d'intérêt sans risque + 1%
life = ActiveSheet.Range("C12").Value 'Maturité
life2 = life * (364 / 365) 'Maturité moins 1 jour
strike = ActiveSheet.Range("C13").Value 'Prix d'exercice
timeSinceInception = ActiveSheet.Range("C14").Value 'Temps écoulé depuis la création
timeSinceInception2 = timeSinceInception * (366 / 365) 'Temps écoulé depuis la création + 1 jour
currentAverage = ActiveSheet.Range("C15").Value 'Moyenne actuelle du cours
currentAverage2 = ActiveSheet.Range("C15").Value 'Moyenne actuelle du cours en jour + 1

' Prix du call au spot :
price = callPriceAsian(spot, strike, volatility, riskFreeRate, life, timeSinceInception, currentAverage)
' Prix du call au spot + 1 :
price2 = callPriceAsian(spot2, strike, volatility, riskFreeRate, life, timeSinceInception, currentAverage)
' Prix du call au spot + 2 :
price3 = callPriceAsian(spot3, strike, volatility, riskFreeRate, life, timeSinceInception, currentAverage)

' Delta du call au niveau du spot :
delta = price2 - price
' Delta du call au niveau du spot + 1 :
delta2 = price3 - price2

' Prix de l'option d'achat asiatique sur action
ActiveSheet.Range("C21").Select
Selection = price
' Delta
ActiveSheet.Range("C22").Select
Selection = delta

' Gamma
ActiveSheet.Range("C23").Select
Selection = delta2 - delta

' Vega
ActiveSheet.Range("C24").Select
Selection = callPriceAsian(spot, strike, volatility2, riskFreeRate, life, timeSinceInception, currentAverage) - price

' Theta
ActiveSheet.Range("C25").Select
Selection = callPriceAsian(spot, strike, volatility, riskFreeRate, life2, timeSinceInception2, currentAverage2) - price

' Rho
ActiveSheet.Range("C26").Select
Selection = callPriceAsian(spot, strike, volatility, riskFreeRate2, life, timeSinceInception, currentAverage) - price

End Sub
```

Cette procédure permet de calculer le prix de l'option d'achat asiatique et les grecques associées à cette option. Le prix de l'option est affecté à la cellule « C21 » de la feuille Excel « options_calculator ». Tandis que, les grecques c'est-à-dire le delta, le gamma, le vega, le theta et le rho sont respectivement affectées aux cellules « C22 », « C23 », « C24 », « C25 » et « C26 ».

La fonction « callPriceAsian », permettant de déterminer le prix de l'option d'achat asiatique, est appelée plusieurs fois. Il est donc important de la présenter plus en détail. Cette fonction prend en paramètre sept variables : le prix au comptant du sous-jacent, le prix d'exercice, le niveau de volatilité, le taux d'intérêt sans risque, la maturité, le temps passé depuis la création de l'option et la moyenne du cours du sous-jacent entre la date de création et le jour de la valorisation. Le code de la fonction est présenté ci-dessous.

```
Function callPriceAsian(v1#, v2#, v3#, v4#, v5#, v6#, v7#) As Double

' Cette fonction calcule le prix d'une option d'achat asiatique
'
' INPUTS
'-----------------------------------------------------------------
' v1 : prix au comptant
' v2 : prix d'exercice
' v3 : volatilité
' v4 : taux d'intérêt sans risque
' v5 : maturité
' v6 : temps depuis la création de l'option
' v7 : moyenne actuelle du cours du sous-jacent
'
' OUTPUT
'-----------------------------------------------------------------
'
' Prix de l'option d'achat asiatique
'
'-----------------------------------------------------------------

Dim timeStep#, u#, numberSteps%, numberSimulations%, randomSeed%

numberSteps = 50
numberSimulations = 500

timeStep = (v5 - v6) / numberSteps

' Fixe le générateur de nombre aléatoire
randomSeed = 366
Randomize ([randomSeed])
u = Rnd(-randomSeed)

Dim tableauSpots() As Double
ReDim tableauSpots(1 To numberSimulations, 0 To numberSteps)
' Simulation de Monte Carlo sous l'hypothèse que le sous-jacent suit une distribution log-normale
For i = 1 To numberSimulations
    tableauSpots(i, 0) = v1
    For j = 1 To numberSteps
        u = Rnd()
        z = Application.WorksheetFunction.Norm_Inv(u, 0, 1)
        tableauSpots(i, j) = tableauSpots(i, j - 1) * Exp((v4 - ((v3 ^ 2) / 2)) * timeStep + (v3 * Sqr(timeStep) * z))
        u = Rnd
    Next j
Next i

Dim tableauResults() As Double
ReDim tableauResults(1 To numberSimulations)

For i = 1 To numberSimulations
   For j = 1 To numberSteps
       tableauResults(i) = tableauResults(i) + tableauSpots(i, j)
   Next j
   tableauResults(i) = ((v6 / v5) * v7) + (((v5 - v6) / v5) * (tableauResults(i) / numberSteps))
   tableauResults(i) = Application.WorksheetFunction.Max(0, tableauResults(i) - v2) * Exp(-v4 * v5)
   result = result + (tableauResults(i) / numberSimulations)
Next i

callPriceAsian = result

End Function
```

Cette fonction passe par une méthode de Monte-Carlo pour évaluer le prix de l'option asiatique. Le nombre de simulations et le nombre d'itérations sont délibérément fixés à 500 et 50 respectivement. Le générateur de nombre aléatoire est également fixé afin que l'utilisateur retombe toujours sur le même prix de l'option s'il relance le calcul. En effet, sans cette fixation, à chaque fois que l'utilisateur lance le code, la loi normal peut prendre un chemin différent et donc le résultat final est différent. Un tableau a deux dimensions est ensuite déclaré « tableauSpots ». Il va stocker les évolutions du prix de l'action sous-jacente à l'issue de chaque simulation. Le point de départ pour chaque simulation est le prix au comptant actuel (fixé par l'utilisateur). La dernière partie de cette fonction va déterminer le prix de l'option d'achat asiatique sur action. Un tableau à une dimension est d'abord créé. Ce tableau calcule dans un premier temps pour chaque simulation la somme des cours de l'action (pour les 50 itérations). Ensuite, pour chaque simulation on fait la moyenne du cours de l'action qui est égale à la moyenne pondérée entre la moyenne actuelle du cours du sous-jacent et la moyenne simulée future.

E. Évaluation des options à barrières activantes vers le bas

Le procédure appelée lorsque l'utilisateur fait appel à l'évaluation d'une option à barrière activante vers le bas est le suivant,

```
Sub call_price_greeks_Barrier_Down_And_In_model()

Dim spot#, vol#, rf#, life#, strike#, barrier#, spot2#, spot3#
Dim vol2#, rf2#, life2#, price#, price2#, price3#

Workbooks("vba_project").Sheets("options_calculator").Activate

spot = ActiveSheet.Range("C5").Value 'Prix au comptant
vol = ActiveSheet.Range("C6").Value 'Volatilité
rf = ActiveSheet.Range("C7").Value 'Taux d'intérêt sans risque
life = ActiveSheet.Range("C12").Value 'Maturité
strike = ActiveSheet.Range("C13").Value 'Prix d'exercice
barrier = ActiveSheet.Range("C14").Value 'Barrière

spot2 = spot + 1 'Spot augmentant de 1€
spot3 = spot2 + 1 'Spot augmentant de 2€
vol2 = vol + 0.01 'Volatilité augmentant de 1%
rf2 = rf + 0.01 'Taux d'intérêt sans risque augmentant de 1%
life2 = life * (364 / 365) 'Maturité moins 1 jour

' Prix du call au spot :
price = callPriceBarrierDownAndIn(spot, strike, vol, rf, life, barrier)
' Prix du call au spot + 1 :
price2 = callPriceBarrierDownAndIn(spot2, strike, vol, rf, life, barrier)
' Prix du call au spot + 2 :
price3 = callPriceBarrierDownAndIn(spot3, strike, vol, rf, life, barrier)

' Price
ActiveSheet.Range("C21").Select
Selection = price

' Delta
ActiveSheet.Range("C22").Select
Selection = price2 - price

' Gamma
ActiveSheet.Range("C23").Select
Selection = (price3 - price2) - (price2 - price)

' Vega
ActiveSheet.Range("C24").Select
Selection = callPriceBarrierDownAndIn(spot, strike, vol2, rf, life, barrier) - price

' Theta
ActiveSheet.Range("C25").Select
Selection = callPriceBarrierDownAndIn(spot, strike, vol, rf, life2, barrier) - price

' Rho
ActiveSheet.Range("C26").Select
Selection = callPriceBarrierDownAndIn(spot, strike, vol, rf2, life, barrier) - price

End Sub
```

Cette procédure permet de calculer le prix de l'option ainsi que ses grecques. Les résultats sont ensuite affichés sur l'interface en les affectant dans les cellules correspondantes sur le feuille Excel « options_calculator ». La valorisation de l'option avec des cours au comptant majorées permettent

l'évaluation du delta et du gamma. Pour réaliser ces évaluations cette procédure fait appel à la fonction « callPriceBarrierDownAndIn() ». Cette fonction permet de calculer le prix d'une option d'achat à barrière désactivante vers le bas. Elle prend en compte le cours au comptant, le prix d'exercice, la volatilité, le taux sans risque, la maturité et la barrière. Le code de cette fonction se situe dans la module « Functions2 » et se définit comme suit,

```
Function callPriceBarrierDownAndIn(v1#, v2#, v3#, v4#, v5#, v6#) As Double

' Cette fonction calcule le prix d'une option d'achat Barrier Down And In
'
' INPUTS
'---------------------------------------------------------------
' v1 : prix au comptant
' v2 : prix d'exercice
' v3 : volatilité
' v4 : taux d'intérêt sans risque
' v5 : maturité
' v6 : barrière
'
' OUTPUT
'---------------------------------------------------------------
'
' Prix de l'option d'achat Barrier Down And In
'
'---------------------------------------------------------------

Dim a#, b#, price#, lambda#, y#

lambda = (v4 + ((v3 ^ 2) / 2)) / (v3 ^ 2)
y = Log((v6 ^ 2) / (v1 * v2)) / (v3 * Sqr(v5)) + lambda * v3 * Sqr(v5)

a = v1 * ((v6 / v1) ^ (2 * lambda)) _
    * Application.WorksheetFunction.Norm_Dist(y, 0, 1, True)
b = (v2 * Exp(-v4 * v5)) * ((v6 / v1) ^ (2 * lambda - 2)) _
    * Application.WorksheetFunction.Norm_Dist(y - v3 * Sqr(v5), 0, 1, True)

price = a - b

callPriceBarrierDownAndIn = price

End Function
```

F. Évaluation des options à barrières désactivantes vers le haut

La procédure appelée pour l'évaluation d'une option d'achat à barrière désactivante vers le haut est le suivant,

```
Sub call_price_Barrier_Up_And_Out_model()

Dim spot#, vol#, rf#, life#, strike#, barrier#, spot2#, spot3#
Dim vol2#, rf2#, life2#, price#, price2#, price3#, step%

Workbooks("vba_project").Sheets("options_calculator").Activate

spot = ActiveSheet.Range("C5").Value 'Prix au comptant
vol = ActiveSheet.Range("C6").Value 'Volatilité
rf = ActiveSheet.Range("C7").Value 'Taux d'intérêt sans risque
life = ActiveSheet.Range("C12").Value 'Maturité
strike = ActiveSheet.Range("C13").Value 'Prix d'exercice
barrier = ActiveSheet.Range("C14").Value 'Barrière

step = 2000 'Nombre d'itérations

spot2 = spot + 1 'Spot augmentant de 1€
spot3 = spot2 + 1 'Spot augmentant de 2€
vol2 = vol + 0.01 'Volatilité augmentant de 1%
rf2 = rf + 0.01 'Taux d'intérêt sans risque augmentant de 1%
life2 = life * (364 / 365) 'Maturité moins 1 jour

' Prix du call au spot :
price = callPriceBarrierUpAndOut(spot, strike, vol, rf, life, barrier, step)
' Prix du call au spot + 1 :
price2 = callPriceBarrierUpAndOut(spot2, strike, vol, rf, life, barrier, step)
' Prix du call au spot + 2 :
price3 = callPriceBarrierUpAndOut(spot3, strike, vol, rf, life, barrier, step)
' Price
ActiveSheet.Range("C21").Select
Selection = price

' Delta
ActiveSheet.Range("C22").Select
If price2 - price < 0 Then 'Si on a un delta non hedge
   Selection = -price / (barrier - spot)
Else
Selection = price2 - price
End If

' Gamma
ActiveSheet.Range("C23").Select
Selection = (price3 - price2) - (price2 - price)

' Vega
ActiveSheet.Range("C24").Select
Selection = callPriceBarrierUpAndOut(spot, strike, vol2, rf, life, barrier, step) - price

' Theta
ActiveSheet.Range("C25").Select
Selection = callPriceBarrierUpAndOut(spot, strike, vol, rf, life2, barrier, step) - price

' Rho
ActiveSheet.Range("C26").Select
Selection = callPriceBarrierUpAndOut(spot, strike, vol, rf2, life, barrier, step) - price

End Sub
```

Cette procédure permet de calculer le prix d'une option d'achat à barrière désactivante vers le haut ainsi que ses grecques. Les résultats sont ensuite affichés sur l'interface en les affectant dans les cellules correspondantes sur le feuille Excel « options_calculator ». De la même manière que dans la section précédente, l'option est aussi valorisée avec des cours au comptant majorées, ceci permet l'évaluation du delta et du gamma. Afin de réaliser ces évaluations la procédure fait appel à la fonction « callPriceBarrierUpAndOut », cette fonction présentée plus bas se situe dans le module « Function2 ». On note également une instruction conditionnelle au niveau du delta. Cette instruction applique un calcul arithmétique évaluant ce qu'on appelle le « delta non hedge ». Cette notion de « delta non hedge » représente un delta inversé. Il correspond au spot que l'on doit acheter ou vendre pour récupérer la prime de l'option que l'on a payé.

La fonction « callPriceBarrierUpAndOut » décrite ci-dessous prend en paramètre le cours au comptant, le prix d'exercice, la volatilité, le taux d'intérêt sans risque, la maturité, la barrière et un nombre d'itérations.

Pour réaliser ces évaluations cette procédure fait appel à la fonction « callPriceBarrierDownAndIn() ». Cette fonction permet de calculer le prix d'une option d'achat à barrière désactivante vers le bas. Elle prend en paramètre le cours au comptant, le prix d'exercice, la volatilité, le taux sans risque, la maturité et la barrière. Le code de cette fonction se situe dans la module « Functions2 » et se définit comme suit,

```
Function callPriceBarrierUpAndOut(v1#, v2#, v3#, v4#, v5#, v6#, v7%) As Double

' Cette fonction calcule le prix d'une option d'achat Barrier Up And Out
' Utilisant la "Stretch Technique"
'
' INPUTS
'---------------------------------------------------------------
' v1 : prix au comptant
' v2 : prix d'exercice
' v3 : volatilité
' v4 : taux d'intérêt sans risque
' v5 : maturité
' v6 : barrière
' v7 : nombre d'itérations
'
' OUTPUT
'---------------------------------------------------------------
'
' Prix de l'option d'achat Barrier Up And Out
'
'---------------------------------------------------------------

Dim timeStep#, n#, nn#, dx#, discount#, u#, pu#, pd#, pm#, p_u#, p_d#, p_m#, exp_dx#
Dim i%, T%, know%, knext%

timeStep = v5 / v7

' Work out lambda (nn)
n = Log(v6 / v1) / (v3 * Sqr(timeStep))
If n > 2 Then
   nn = n / Int(n)
Else
   nn = n
End If
```

```
' Precompute invariant quantities
dx = nn * v3 * Sqr(timeStep)
discount = Exp(-v4 * timeStep)
u = v4 - ((v3 ^ 2) / 2)

pu = (1 / (2 * (nn ^ 2))) + ((u * Sqr(timeStep)) / (2 * nn * v3))
pd = (1 / (2 * (nn ^ 2))) - ((u * Sqr(timeStep)) / (2 * nn * v3))
pm = 1 - (1 / (nn ^ 2))

p_u = discount * pu
p_d = discount * pd
p_m = discount * pm

' Work out stock price
Dim Stree() As Double
ReDim Stree(1 To 2 * v7 + 1)

Stree(1) = v1 * Exp(-v7 * dx)
exp_dx = Exp(dx)

For i = 2 To 2 * v7 + 1
    Stree(i) = exp_dx * Stree(i - 1)
Next i


'Work out option price
Dim OptionValues() As Double
ReDim OptionValues(1 To 2 * v7 + 1, 1 To 2)

T = (v7 Mod 2) + 1

For i = 1 To 2 * v7 + 1
    If Stree(i) >= v6 Then
       OptionValues(i, T) = 0
    ElseIf Stree(i) - v2 > 0 Then
       OptionValues(i, T) = Stree(i) - v2
    Else
       OptionValues(i, T) = 0
    End If
Next i

For T = v7 - 1 To 0 Step -1
    know = (T Mod 2) + 1
    knext = ((T + 1) Mod 2) + 1
    For i = v7 - T + 1 To v7 + T + 1
        If Stree(i) >= v6 Then
           OptionValues(i, know) = 0
        Else
           OptionValues(i, know) = p_d * OptionValues(i - 1, knext) + p_m _
           * OptionValues(i, knext) + p_u * OptionValues(i + 1, knext)
        End If
    Next i
Next T

callPriceBarrierUpAndOut = OptionValues(v7 + 1, 1)

End Function
```

On note également l'apparition d'un message d'alerte si l'utilisateur fixe une barrière inférieure au prix du sous-jacent. Plus précisément, la valeur de la barrière doit au moins être supérieure de trois euros au prix spot pour que la valorisation de l'option à barrière désactivante vers le haut fonctionne. Si cette condition n'est pas respectée, l'alerte sous la forme d'un userform nommée « alertUAOBarrier » est appelée lorsque l'utilisateur clique sur « Calculate » via la procédure « CommandButton1_Click() ». Le d'erreur message est le suivant,

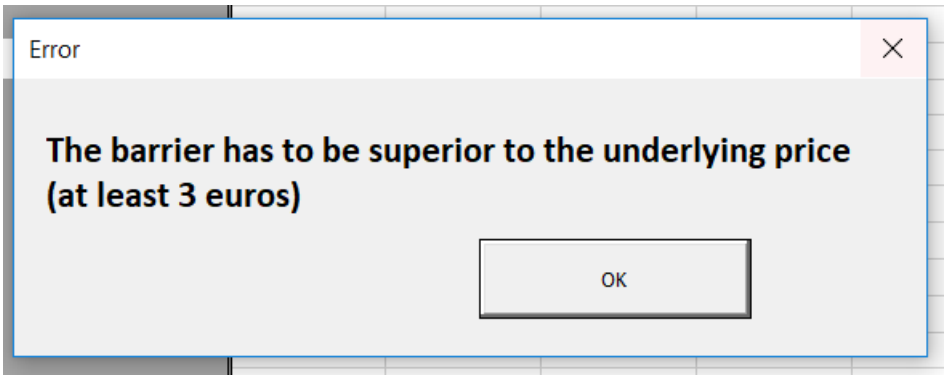

Illustration 6.

G. Évaluation des options à barrières désactivantes vers le bas

La procédure appelée pour l'évaluation d'une option d'achat à barrière désactivante vers le bas est le suivant,

```
Sub call_price_Barrier_Down_And_Out_model() 'Utilisant la "Stretch Technique"

Dim spot#, vol#, rf#, life#, strike#, barrier#, spot2#, spot3#
Dim vol2#, rf2#, life2#, price#, price2#, price3#, step%

Workbooks("vba_project").Sheets("options_calculator").Activate

spot = ActiveSheet.Range("C5").Value 'Prix au comptant
vol = ActiveSheet.Range("C6").Value 'Volatilité
rf = ActiveSheet.Range("C7").Value 'Taux d'intérêt sans risque
life = ActiveSheet.Range("C12").Value 'Maturité
strike = ActiveSheet.Range("C13").Value 'Prix d'exercice
barrier = ActiveSheet.Range("C14").Value 'Barrière

step = 2000 'Nombre d'itérations

spot2 = spot + 1 'Spot augmentant de 1€
spot3 = spot2 + 1 'Spot augmentant de 2€
vol2 = vol + 0.01 'Volatilité augmentant de 1%
rf2 = rf + 0.01 'Taux d'intérêt sans risque augmentant de 1%
life2 = life * (364 / 365) 'Maturité moins 1 jour

' Prix du call au spot :
price = callPriceBarrierDownAndOut(spot, strike, vol, rf, life, barrier, step)
' Prix du call au spot + 1 :
price2 = callPriceBarrierDownAndOut(spot2, strike, vol, rf, life, barrier, step)
' Prix du call au spot + 2 :
price3 = callPriceBarrierDownAndOut(spot3, strike, vol, rf, life, barrier, step)
' Price
ActiveSheet.Range("C21").Select
Selection = price

' Delta
ActiveSheet.Range("C22").Select
Selection = price2 - price

' Gamma
ActiveSheet.Range("C23").Select
Selection = (price3 - price2) - (price2 - price)

' Vega
ActiveSheet.Range("C24").Select
Selection = callPriceBarrierDownAndOut(spot, strike, vol2, rf, life, barrier, step) - price

' Theta
ActiveSheet.Range("C25").Select
Selection = callPriceBarrierDownAndOut(spot, strike, vol, rf, life2, barrier, step) - price

' Rho
ActiveSheet.Range("C26").Select
Selection = callPriceBarrierDownAndOut(spot, strike, vol, rf2, life, barrier, step) - price

End Sub
```

Le code de cette procédure est identique au code de l'option d'achat à barrière désactivante vers le haut à une exception près. Dans cette procédure, c'est évidemment une autre fonction qui est appelée pour calculer le prix de l'option d'achat à barrière désactivante vers le bas. La fonction en question est

la fonction « callPriceBarrierDownAndOut() », elle se situe aussi dans le module « Fonction2 ». Le code de cette fonction est le suivant,

```
Function callPriceBarrierDownAndOut(v1#, v2#, v3#, v4#, v5#, v6#, v7%) As Double

' Cette fonction calcule le prix d'une option d'achat Barrier Down And Out
'
' INPUTS
'---------------------------------------------------------------
' v1 : prix au comptant
' v2 : prix d'exercice
' v3 : volatilité
' v4 : taux d'intérêt sans risque
' v5 : maturité
' v6 : barrière
' v7 : nombre d'itérations
'
' OUTPUT
'---------------------------------------------------------------
'
' Prix de l'option d'achat Barrier Down And Out
'
'---------------------------------------------------------------

Dim timeStep#, n#, nn#, dx#, discount#, u#, pu#, pd#, pm#, p_u#, p_d#, p_m#, exp_dx#
Dim i%, T%, know%, knext%

timeStep = v5 / v7

' Work out lambda (nn)
n = Log(v1 / v6) / (v3 * Sqr(timeStep))
If n > 2 Then
   nn = n / Int(n)
Else
   nn = n
End If


' Precompute invariant quantities
dx = nn * v3 * Sqr(timeStep)
discount = Exp(-v4 * timeStep)
u = v4 - ((v3 ^ 2) / 2)

pu = (1 / (2 * (nn ^ 2))) + ((u * Sqr(timeStep)) / (2 * nn * v3))
pd = (1 / (2 * (nn ^ 2))) - ((u * Sqr(timeStep)) / (2 * nn * v3))
pm = 1 - (1 / (nn ^ 2))

p_u = discount * pu
p_d = discount * pd
p_m = discount * pm

' Work out stock price
Dim Stree() As Double
ReDim Stree(1 To 2 * v7 + 1)

Stree(1) = v1 * Exp(-v7 * dx)
exp_dx = Exp(dx)

For i = 2 To 2 * v7 + 1
    Stree(i) = exp_dx * Stree(i - 1)
Next i
```

```
'Work out option price
Dim OptionValues() As Double
ReDim OptionValues(1 To 2 * v7 + 1, 1 To 2)

T = (v7 Mod 2) + 1

For i = 1 To 2 * v7 + 1
    If Stree(i) <= v6 Then
       OptionValues(i, T) = 0
    ElseIf Stree(i) - v2 > 0 Then
       OptionValues(i, T) = Stree(i) - v2
    Else
       OptionValues(i, T) = 0
    End If
Next i

For T = v7 - 1 To 0 Step -1
    know = (T Mod 2) + 1
    knext = ((T + 1) Mod 2) + 1
    For i = v7 - T + 1 To v7 + T + 1
        If Stree(i) <= v6 Then
           OptionValues(i, know) = 0
        Else
           OptionValues(i, know) = p_d * OptionValues(i - 1, knext) + p_m _
           * OptionValues(i, knext) + p_u * OptionValues(i + 1, knext)
        End If
    Next i
Next T

callPriceBarrierDownAndOut = OptionValues(v7 + 1, 1)

End Function
```

Le code de cette fonction « callPriceBarrierDownAndOut() » est très proche de la fonction vue précédemment pour l'option d'achat à barrière désactivante vers le haut. La seule différence intervient au niveau de la partie « Work out option price » où pour chaque prix du sous-jacent inférieur à la barrière, le prix de l'option devient nul.

De la même manière que pour l'option à barrière désactivante vers le haut, un message d'alerte apparait si l'utilisateur fixe une barrière supérieure au prix du sous-jacent. Ce message d'alerte sous forme d'userform « alertDAOBarrier » est appelé lorsque l'utilisateur clique sur « Calculate » via la procédure « CommandButton1_Click() ». Le message d'erreur est le suivant,

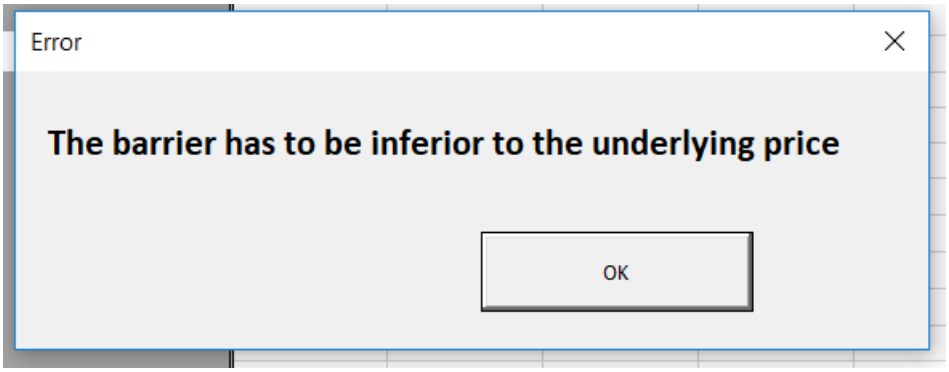

Illustration 7.

H. Évaluation des options binaires « Cash or Nothing »

La procédure permettant de calculer le prix et les grecques d'une option d'achat binaire « Cash or Nothing » est la procédure « call_Binary_Cash_Or_Nothing_model() ». Elle est de la forme suivante,

```
Sub call_Binary_Cash_Or_Nothing_model()

Dim spot#, volatility#, riskFreeRate#, life#, strike#, cash#, Pi#
Dim phi1#, d1#, d2#, cdf2#, volatility2#, d1Vol#, d2Vol#, pdf2Vol#
Dim riskFreeRate2#, d1RFR#, d2RFR#, cdf2RFR#, life2#, d1life#, d2life#, cdf2life#

Workbooks("vba_project").Sheets("options_calculator").Activate

spot = ActiveSheet.Range("C5").Value 'Prix au comptant
volatility = ActiveSheet.Range("C6").Value 'Volatilité
riskFreeRate = ActiveSheet.Range("C7").Value 'Taux d'intérêt sans risque
life = ActiveSheet.Range("C12").Value 'Maturité
strike = ActiveSheet.Range("C13").Value 'Prix d'exercice
cash = ActiveSheet.Range("C14").Value 'Montant de cash

d1 = ((Log(spot / strike)) + ((riskFreeRate + ((volatility ^ 2) / 2)) * life)) / (volatility * Sqr(life))
d2 = d1 - (volatility * Sqr(life))

Pi = WorksheetFunction.Pi 'Nombre Pi
phi1 = Exp(-(d1 ^ 2 / 2)) / Sqr(2 * Pi)

cdf2 = Application.WorksheetFunction.Norm_Dist(d2, 0, 1, True) 'Fonction de répartition de la loi normale

' Price
ActiveSheet.Range("C21").Select
Selection = Exp(-(riskFreeRate * life)) * cdf2 * cash

' Delta
ActiveSheet.Range("C22").Select
Selection = ((spot / strike) * (phi1 / (spot * volatility * Sqr(life)))) * cash

'Gamma
ActiveSheet.Range("C23").Select
Selection = (-((phi1 / (spot * volatility * Sqr(life))) / strike) * (d1 / (volatility * Sqr(life)))) * cash

'Vega
volatility2 = volatility + 0.01 'Lorsque la volatilité augmente de 1%
d1Vol = ((Log(spot / strike)) + ((riskFreeRate + ((volatility2 ^ 2) / 2)) * life)) / (volatility2 * Sqr(life))
d2Vol = d1Vol - (volatility2 * Sqr(life))
cdf2Vol = Application.WorksheetFunction.Norm_Dist(d2Vol, 0, 1, True)
ActiveSheet.Range("C24").Select
Selection = (Exp(-(riskFreeRate * life)) * cdf2Vol * cash) - (Exp(-(riskFreeRate * life)) * cdf2 * cash)

'Theta
life2 = life * 0.99726027 'Maturité - 1 jour
d1life = ((Log(spot / strike)) + ((riskFreeRate + ((volatility ^ 2) / 2)) * life2)) / (volatility * Sqr(life2))
d2life = d1life - (volatility * Sqr(life2))
cdf2life = Application.WorksheetFunction.Norm_Dist(d2life, 0, 1, True)
ActiveSheet.Range("C25").Select
Selection = (Exp(-(riskFreeRate * life2)) * cdf2life * cash) - (Exp(-(riskFreeRate * life)) * cdf2 * cash)

'Rho
riskFreeRate2 = riskFreeRate + 0.01 'Lorsque le taux sans risque augmente de 1%
d1RFR = ((Log(spot / strike)) + ((riskFreeRate2 + ((volatility ^ 2) / 2)) * life)) / (volatility * Sqr(life))
d2RFR = d1RFR - (volatility * Sqr(life))
cdf2RFR = Application.WorksheetFunction.Norm_Dist(d2RFR, 0, 1, True)
ActiveSheet.Range("C26").Select
Selection = (Exp(-(riskFreeRate2 * life)) * cdf2RFR * cash) - (Exp(-(riskFreeRate * life)) * cdf2 * cash)

End Sub
```

Dans un premier temps les données entrées par l'utilisateur sont affectées à des variables. La procédure calcule dans un deuxième temps le prix de l'option binaire « Asset or Nothing », le delta et le gamma. Le calcul des autres grecques nécessitent quelques opérations. L'estimation du vega se fait à l'aide d'une simulation du prix de l'option pour une volatilité majorée d'un pourcent. De la même manière, le theta avec une maturité minorée d'un jour et le rho avec un taux sans risque majoré d'un pourcent. Les résultats sont ensuite affectés aux cellules correspondantes de le feuille Excel « options_calculator ».

I. Évaluation des options binaires « Asset or Nothing »

La procédure permettant de calculer le prix et les grecques d'une option d'achat binaire « Asset or Nothing » est la procédure « call_Binary_Asset_Or_Nothing_model() ». Elle est de la forme suivante,

```
Sub call_Binary_Asset_Or_Nothing_model()

Dim spot#, volatility#, riskFreeRate#, life#, strike#, spot2#, volatility2#
Dim riskFreeRate2#, life2#, spot3#, price#, price2#, price3#, delta#, delta2#

Workbooks("vba_project").Sheets("options_calculator").Activate

spot = ActiveSheet.Range("C5").Value 'Prix au comptant
volatility = ActiveSheet.Range("C6").Value 'Volatilité
riskFreeRate = ActiveSheet.Range("C7").Value 'Taux d'intérêt sans risque
life = ActiveSheet.Range("C12").Value 'Maturité
strike = ActiveSheet.Range("C13").Value 'Prix d'exercice

spot2 = spot + 1 'Spot augmentant de 1€
spot3 = spot2 + 1 'Spot augmentant de 2€
volatility2 = volatility + 0.01 'Volatilité augmentant de 1%
riskFreeRate2 = riskFreeRate + 0.01 'Taux d'intérêt sans risque augmentant de 1%
life2 = life * (364 / 365) 'Maturité moins 1 jour

' Prix du call au spot :
price = callPriceBlackScholes(spot, strike, volatility, riskFreeRate, life) _
+ callPriceBinaryCashOrNothing(spot, strike, volatility, riskFreeRate, life, strike)
' Prix du call au spot + 1 :
price2 = callPriceBlackScholes(spot2, strike, volatility, riskFreeRate, life) _
+ callPriceBinaryCashOrNothing(spot2, strike, volatility, riskFreeRate, life, strike)
' Prix du call au spot + 2 :
price3 = callPriceBlackScholes(spot3, strike, volatility, riskFreeRate, life) _
+ callPriceBinaryCashOrNothing(spot3, strike, volatility, riskFreeRate, life, strike)
' Delta du call au niveau du spot :
delta = price2 - price
' Delta du call au niveau du spot + 1 :
delta2 = price3 - price2


' Price
ActiveSheet.Range("C21").Select
Selection = price

' Delta
ActiveSheet.Range("C22").Select
Selection = delta

' Gamma
ActiveSheet.Range("C23").Select
Selection = delta2 - delta

' Vega
ActiveSheet.Range("C24").Select
Selection = callPriceBlackScholes(spot, strike, volatility2, riskFreeRate, life) _
+ callPriceBinaryCashOrNothing(spot, strike, volatility2, riskFreeRate, life, strike) - price

' Theta
ActiveSheet.Range("C25").Select
Selection = callPriceBlackScholes(spot, strike, volatility, riskFreeRate, life2) _
+ callPriceBinaryCashOrNothing(spot, strike, volatility, riskFreeRate, life2, strike) - price

' Rho
ActiveSheet.Range("C26").Select
Selection = callPriceBlackScholes(spot, strike, volatility, riskFreeRate2, life) _
+ callPriceBinaryCashOrNothing(spot, strike, volatility, riskFreeRate2, life, strike) - price

End Sub
```

Dans un premier temps les données entrées par l'utilisateur sont affectées à des variables. La procédure calcule dans un deuxième temps le prix d'une option binaire « Asset or Nothing » pour trois niveaux de spot différents. La procédure calcule ensuite le delta et le gamma de l'option binaire pour le spot fixé par l'utilisateur.

La seconde partie de la procédure concerne le calcul des autres grecques et l'affectation des résultats aux cellules correspondantes de le feuille Excel « options_calculator ». Il est important de noter que cette procédure utilise deux fonctions dans son exécution. Ces fonctions se situent dans le module « Functions2 ». La première fonction utilisée est la fonction « callPriceBlackScholes ». Elle permet de calculer le prix d'une option d'achat avec le modèle de Black et Scholes. Cette fonction prend en paramètre le prix au comptant du sous-jacent, le prix d'exercice, le niveau de volatilité, le taux sans risque et la maturité. Le code de cette fonction est le suivant,

```
Function callPriceBlackScholes(v1#, v2#, v3#, v4#, v5#) As Double

' Cette fonction calcule le prix d'une option d'achat avec le modèle de Black-Scholes
'
' INPUTS
'-----------------------------------------------------------------
' v1 : prix au comptant
' v2 : prix d'exercice
' v3 : volatilité
' v4 : taux d'intérêt sans risque
' v5 : maturité
'
' OUTPUT
'-----------------------------------------------------------------
'
' Prix de l'option d'achat
'
'-----------------------------------------------------------------

Dim d1#, d2#, cdf1#, cdf2#

d1 = ((Log(v1 / v2)) + ((v4 + ((v3 ^ 2) / 2)) * v5)) / (v3 * Sqr(v5))
d2 = d1 - (v3 * Sqr(v5))

cdf1 = Application.WorksheetFunction.Norm_Dist(d1, 0, 1, True)
cdf2 = Application.WorksheetFunction.Norm_Dist(d2, 0, 1, True)

callPriceBlackScholes = (v1 * cdf1) - (v2 * Exp(-(v4 * v5)) * cdf2)

End Function
```

La seconde fonction utilisée est la fonction « callPriceBinaryCashOrNothing ». Cette fonction calcule le prix d'une option d'achat binaire « Cash Or Nothing ». Le code de cette fonction est le suivant,

```
Function callPriceBinaryCashOrNothing(v1#, v2#, v3#, v4#, v5#, v6#) As Double

' Cette fonction calcule le prix d'une option d'achat Binary Cash Or Nothing
'
' INPUTS
'-----------------------------------------------------------------
' v1 : prix au comptant
' v2 : prix d'exercice
' v3 : volatilité
' v4 : taux d'intérêt sans risque
' v5 : maturité
' v6 : montant de cash
'
' OUTPUT
'-----------------------------------------------------------------
'
' Prix de l'option d'achat Binary Cash Or Nothing
'
'-----------------------------------------------------------------

Dim d1#, d2#, cdf#


d1 = ((Log(v1 / v2)) + ((v4 + ((v3 ^ 2) / 2)) * v5)) / (v3 * Sqr(v5))
d2 = d1 - (v3 * Sqr(v5))

cdf = Application.WorksheetFunction.Norm_Dist(d2, 0, 1, True)

callPriceBinaryCashOrNothing = Exp(-(v4 * v5)) * cdf * v6

End Function
```

J. Autres modèles d'évaluation

Ce projet développe également le modèle de Black et Scholes pour d'autres sous-jacents. Le module « Black_Scholes_Currency » contient les procédures « call_price_greeks_BS_Currency_model() » et « put_price_greeks_BS_Currency_model() » permettant de calculer le prix et les grecques d'une option d'achat ou de vente pour une devise. Tout comme le module « Black_Scholes_Index » contenant les procédures permettant de calculer les prix et les grecques d'une option d'achat ou de vente « call_price_greeks_BS_Index_model() » et « put_price_greeks_BS_Index_model() » pour un indice. De la même manière, deux fonctions permettant de faire la même chose pour un sous-jacent de type future sont disponibles dans le module « Black_Scholes_Futures ». Ces fonctions ne sont pas présentées ici du fait qu'elles sont extrêmement proches aux fonctions décrites précédemment.

Dans le prochain chapitre de ce projet, le code permettant de réaliser des simulations de Monte Carlo pour l'évaluation des options est présenté.

II. Méthode de Monte Carlo

Cette application permet d'évaluer des options européennes (options d'achat et options de vente) selon le modèle log-normal et le modèle de diffusion à sauts de Merton à l'aide de simulations de Monte Carlo. Elle se trouve dans le second onglet « Monte_Carlo ». De manière générale, la méthode de Monte Carlo est une méthode d'estimation d'une quantité numérique utilisant des nombres aléatoires. Dans notre cas, cette méthode nous permet de simuler le prix futur du sous-jacent, qui est ici une action. Cette projection de prix futurs nous permet de valoriser l'option. L'écart-type associé à cette simulation est aussi calculé.
L'interface de la simulation Monte Carlo pour le modèle log normal se présente comme suit,

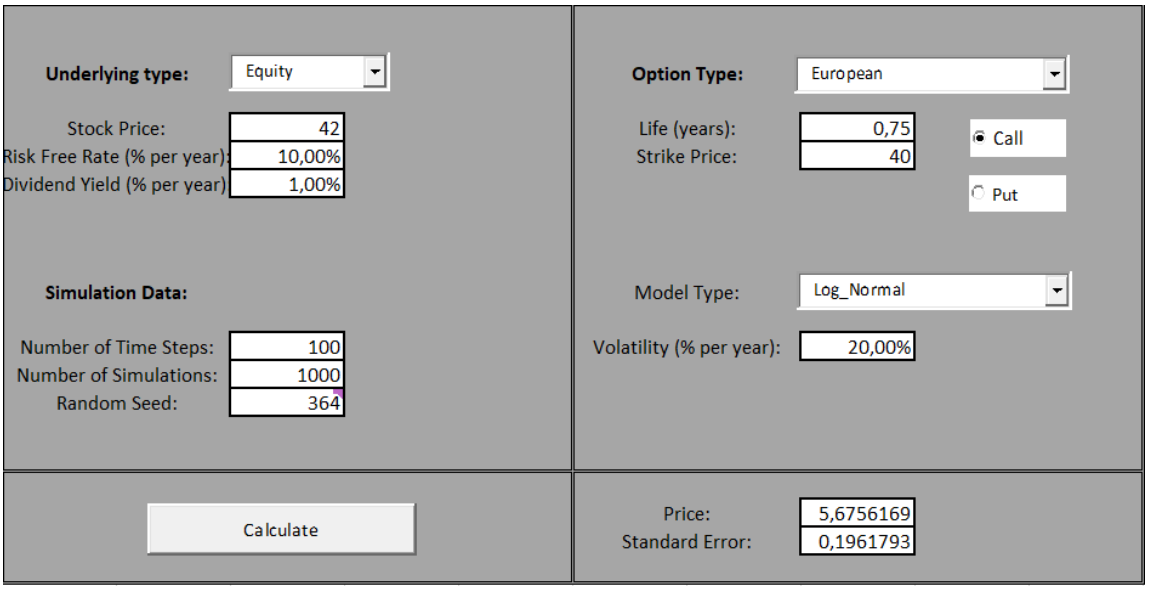

Illustration 8.

Les inputs à rentrer sont le prix de l'action, la volatilité de l'action (en % par année), le taux sans risque (en % par année) et le dividende (en % par année). Il faut également rentrer le nombre de simulations souhaité ainsi que le nombre d'itérations par simulation. Un message d'erreur apparaît si le nombre de simulations n'est pas compris entre 4 et 10000 (voir illustration 2) et quand le nombre d'itérations n'est pas compris entre 2 et 200 (voir illustration 3). Ces messages d'erreur apparaissent également avec le modèle de Merton si les mêmes conditions ne sont pas respectées. Par ailleurs, le « random seed » permet de fixer le chemin pris par les nombres aléatoires. Plus précisément, si on fixe comme ci-dessus le « random seed » égal à 364, alors à chaque fois que l'on va calculer la valeur de l'option, les chemins des prix simulés seront toujours les mêmes. Par conséquent, le prix de l'option sera toujours la même (avec des inputs inchangés) et ce même si on fait appel à chaque fois à la loi normale. L'application affiche également les dix premières simulations du prix de l'action, la valeur de l'option et l'écart type associés à ces dix premières simulations. Un graphique est aussi affiché permettant à l'utilisateur de voir les deux premières simulations à titre d'illustration.

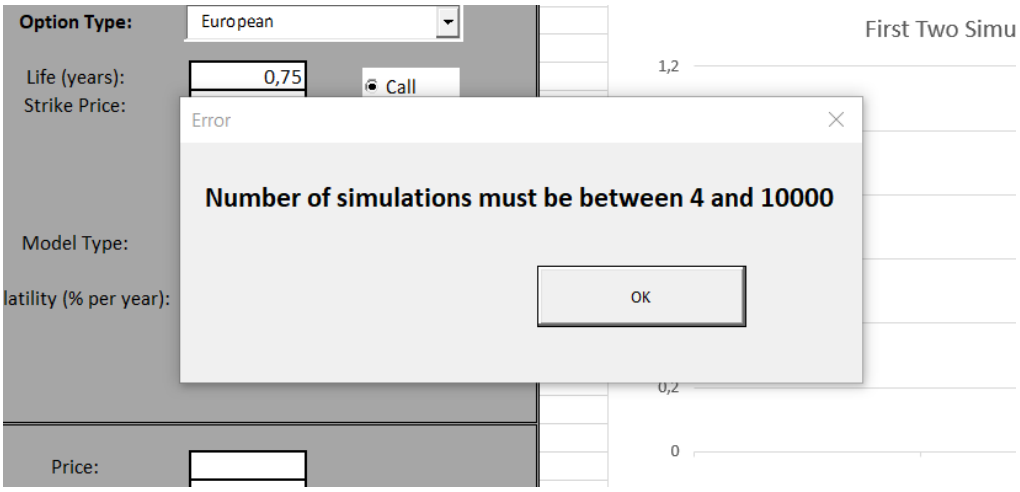

Illustration 9.

Ce message d'erreur est la résultante de l'userform « alertMCNumberSimulations ».

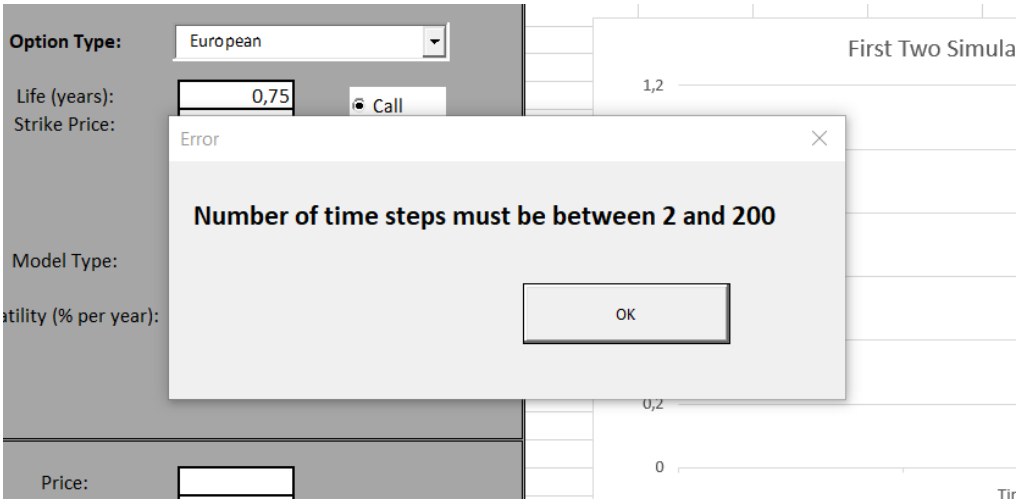

Illustration 10.

Ce message d'erreur est la résultante de l'userform « alertMCNumberSteps ».

Le code suivant permet de modifier l'interface lorsque l'utilisateur change le type de modèle. Il peut choisir entre le modèle log-normal et le modèle de diffusion à sauts de Merton. Lorsque l'un des modèles est choisi à travers la ComboBox3, le programme va automatiquement ajuster l'interface. Pour cela il va appeler la procédure qui lui est associée.

```
' Cette procédure modifie l'agencement quand on change l'input de la ComboBox associée à "Model Type"
Private Sub ComboBox3_Change()

' Appel de la procédure formEquityEuropeanLogNormal situé dans le module MEF3_MC si le modèle "Log_Normal" est sélectionné
If ComboBox3.Value = "Log_Normal" Then
   formEquityEuropeanLogNormal

' Appel de la procédure formEquityEuropeanMertonJump situé dans le module MEF3_MC si le modèle à sauts de Merton est sélectionné
ElseIf ComboBox3.Value = "Merton_Jump_Diffusion" Then
   formEquityEuropeanMertonJump
End If

End Sub
```

Lorsque l'utilisateur sélectionne le modèle log-normal, la procédure ci-dessous est appelée. Cette procédure adapte la plage de cellules F14:H16 pour ce modèle.

```
' Forme initiale pour le modèle LogNormal pour une "European Equity"
Sub formEquityEuropeanLogNormal()

    Workbooks("vba_project").Sheets("Monte_Carlo").Activate

    ActiveSheet.Range("F14:H16").Select
    With Selection
    .Value = ""
    .Interior.Color = RGB(166, 166, 166)
    End With
    For Each cl In Selection
         cl.Borders.LineStyle = xlLineStyleNone
    Next cl

    ActiveSheet.Range("H13").Select 'Selectionne la cellule H13 une fois l'ajustement de l'interface terminé
End Sub
```

Lorsque l'utilisateur sélectionne le modèle de Merton, la procédure qui suit est appelée et adapte la plage de cellules F14:H16 pour ce modèle. Ce modèle utilise trois valeurs d'inputs supplémentaires par rapport au modèle log-normal. C'est pour cela que trois nouvelles cellules s'affichent dans la feuille de calcul.

```
' Forme initiale pour le modèle de Merton pour une "European Equity"
Sub formEquityEuropeanMertonJump()

    Workbooks("vba_project").Sheets("Monte_Carlo").Activate

    ActiveSheet.Range("F14").Select
    With Selection
         .Value = "Jumps per Year"
         .Characters.Font.Size = 11
         .HorizontalAlignment = xlCenter
         .VerticalAlignment = xlCenter
         .Font.Color = RGB(0, 0, 0)
    End With

    ActiveSheet.Range("F15").Select
    With Selection
         .Value = "Average Jump Size (%)"
         .Characters.Font.Size = 11
         .HorizontalAlignment = xlCenter
         .VerticalAlignment = xlCenter
         .Font.Color = RGB(0, 0, 0)
    End With

    ActiveSheet.Range("F16").Select
    With Selection
         .Value = "Jump Std Deviation (%)"
         .Characters.Font.Size = 11
         .HorizontalAlignment = xlCenter
         .VerticalAlignment = xlCenter
         .Font.Color = RGB(0, 0, 0)
    End With
    ActiveSheet.Range("H14").Select
    With Selection
         .Font.Color = RGB(0, 0, 0)
         .Interior.Color = RGB(255, 255, 255)
         .BorderAround Weight:=xlMedium
    End With

    ActiveSheet.Range("H15").Select
    With Selection
         .Font.Color = RGB(0, 0, 0)
         .Interior.Color = RGB(255, 255, 255)
         .BorderAround Weight:=xlMedium
    End With

    ActiveSheet.Range("H16").Select
    With Selection
         .Font.Color = RGB(0, 0, 0)
         .Interior.Color = RGB(255, 255, 255)
         .BorderAround Weight:=xlMedium
    End With

    ActiveSheet.Range("H14").Select 'Selectionne la cellule H14 une fois l'ajustement de l'interface terminé
End Sub
```

Cette partie du code ci-dessous sert de lancement à la simulation de Monte Carlo. Elle se lance lorsque l'utilisateur clique sur le bouton « Calculate ». La première étape de ce code est d'effacer les cellules illustrant les dix premiers chemins de l'ancienne simulation afin de pouvoir y mettre ceux de la nouvelle simulation. Il en vient ensuite les appels aux UserForms en guise d'alerte si l'utilisateur ne rentre pas un nombre de simulations ou d'itérations admissibles. Les instructions suivantes appellent la procédure qui va calculer l'écart-type et la valeur de l'option selon le sous-jacent, le type d'option, le modèle sélectionné et si c'est une option d'achat ou de vente. Nous aurions pu optimiser le code en supprimant le type de sous-jacent et le type d'option de la condition, puisqu'ici seules les actions européennes peuvent être calculées. Cependant, comme nous avons pour objectif futur de compléter ce pricer nous avons décidé de le laisser tel quel.

```
' Cette procédure lance la simulation de Monte Carlo lorsqu'on clique sur "Calculate".
Private Sub CommandButton1_Click()

Workbooks("vba_project").Sheets("Monte_Carlo").Activate

' On vide les cellules des anciens résultats stockés avant chaque nouvelle simulation
ActiveSheet.Range("B32:L232").Value = ""
ActiveSheet.Range("D25").Value = ""
ActiveSheet.Range("D26").Value = ""
ActiveSheet.Range("H19").Value = ""
ActiveSheet.Range("H20").Value = ""
ActiveSheet.Range("C29:L29").Value = ""

' Alerte via un Userform si le nombre de Step est inférieur à 2 ou supérieur à 200
If ComboBox2.Value = "European" And Range("C13") < 2 Or ComboBox2.Value = "European" And Range("C13") > 200 Then
   alertMcNumberSteps.Show 0

' Alerte via un Userform si le nombre de Simulation est inférieur à 4 ou supérieur à 10000
ElseIf ComboBox2.Value = "European" And Range("C14") < 4 Or ComboBox2.Value = "European" And Range("C14") > 10000 Then
   alertMcNumberSimulations.Show 0

' Lancement de la simulation Monte-Carlo (modèle : Log-Normal) pour un Call
ElseIf ComboBox1.Value = "Equity" And ComboBox2.Value = "European" And ComboBox3 = "Log_Normal" And OptionButton1.Value = True Then
   monteCarloEuropeanLogNormalCall
   formMCDynamicChart ' Appel de la procédure formMCDynamicChart située le module "Dynamic Chart" pour l'ajustement du graphique

' Lancement de la simulation Monte-Carlo (modèle : Log-Normal) pour un Put
ElseIf ComboBox1.Value = "Equity" And ComboBox2.Value = "European" And ComboBox3 = "Log_Normal" And OptionButton2.Value = True Then
   monteCarloEuropeanLogNormalPut
   formMCDynamicChart ' Appel de la procédure formMCDynamicChart située le module "Dynamic Chart" pour l'ajustement du graphique

' Lancement de la simulation Monte-Carlo (modèle : Merton Jump Diffusion) pour un Call
ElseIf ComboBox1.Value = "Equity" And ComboBox2.Value = "European" And ComboBox3 = "Merton_Jump_Diffusion" And OptionButton1.Value = True Then
   monteCarloEuropeanMertonJumpDiffusionCall
   formMCDynamicChart ' Appel de la procédure formMCDynamicChart située le module "Dynamic Chart" pour l'ajustement du graphique

' Lancement de la simulation Monte-Carlo (modèle : Merton Jump Diffusion) pour un Put
ElseIf ComboBox1.Value = "Equity" And ComboBox2.Value = "European" And ComboBox3 = "Merton_Jump_Diffusion" And OptionButton2.Value = True Then
   monteCarloEuropeanMertonJumpDiffusionPut
   formMCDynamicChart ' Appel de la procédure formMCDynamicChart située le module "Dynamic Chart" pour l'ajustement du graphique

End If
End Sub
```

La procédure « formMCDynamiqueChart » est appelée lors de chaque lancement de simulation. Elle permet l'ajustement de façon automatique de l'axe des X du graphique, illustrant les deux premières simulations. En effet, cet axe doit s'ajuster selon le nombre d'itérations et la durée de vie de l'option sélectionnée par l'utilisateur. L'axe des abscisses du graphique prend les valeurs de la plage de données « plageX ». Cette plage de données sélectionne les cellules allant de B32 à B(32 + le nombre d'itérations) de la feuille « Monte_Carlo ».

```
Sub formMCDynamicChart() 'Ajustement du tableau dynamique pour la simulation Monte-Carlo

Dim plageX As Range

Workbooks("vba_project").Sheets("Monte_Carlo").Activate
ActiveSheet.ChartObjects("Graphique 1").Activate
ActiveChart.SetSourceData Source:=Range("Monte_Carlo!chart_plage")

Set plageX = Range("Monte_Carlo!x_plage")
ActiveChart.SeriesCollection(1).XValues = plageX

ActiveSheet.Range("C22").Select 'Selectionne la cellule C22 une fois l'ajustement du tableau terminé
End Sub
```

A titre d'exemple, par rapport aux données utilisées dans l'illustration, la plage de données « plageX » prend ses valeurs de B32 à B132. Le graphique résultant est comme suit :

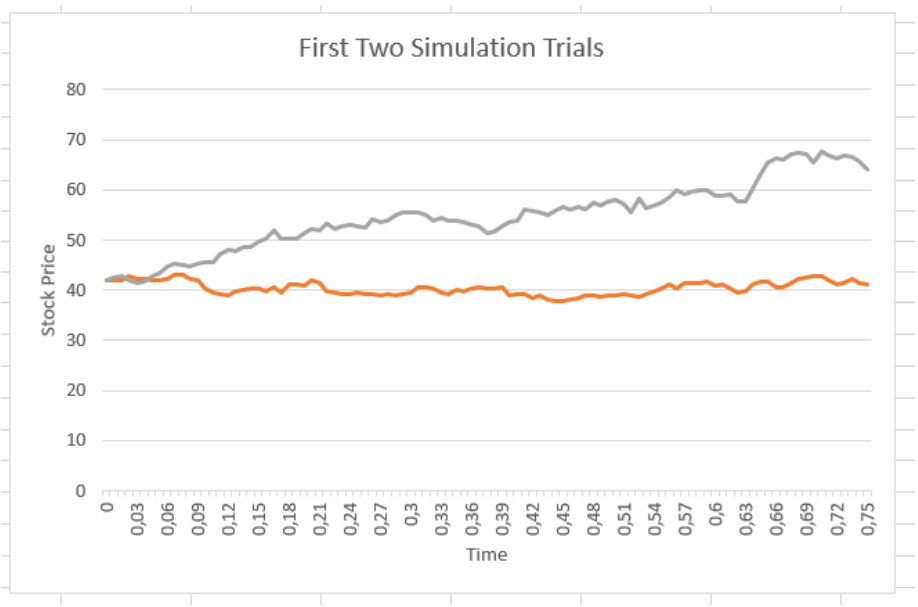

Illustration 11.

Les procédures « monteCarloEuropeanLogNormalCall » et « monteCarloEuropeanLogNormalPut » évaluent par simulation de Monte Carlo une option d'achat ou de vente respectivement (de type européenne et avec comme sous-jacent une action) utilisant un modèle log-normal. Ces deux procédures sont extrêmement proches en terme de code. Nous allons donc décrire ici uniquement la première des deux procédures citées précédemment.

```
Sub monteCarloEuropeanLogNormalCall()

Dim numberSteps%, numberSimulations%, randomSeed%, i%, j%
Dim spot#, strike#, volatility#, riskFreeRate#, life#, timeStep#, z#, u#, priceCum#, dividend#

Workbooks("vba_project").Sheets("Monte_Carlo").Activate

numberSteps = ActiveSheet.Range("C13").Value 'Nombre d'itérations
numberSimulations = ActiveSheet.Range("C14").Value 'Nombre de simulations
spot = ActiveSheet.Range("C5").Value 'Prix au comptant
volatility = ActiveSheet.Range("H13").Value 'Volatilité
riskFreeRate = ActiveSheet.Range("C6").Value 'Taux sans risque
life = ActiveSheet.Range("H5").Value 'Maturité
strike = ActiveSheet.Range("H6").Value 'Prix d'exercice
randomSeed = ActiveSheet.Range("C15").Value 'Fixation de l'aléa
dividend = ActiveSheet.Range("C7") 'Dividende

timeStep = life / numberSteps 'Le pas d'itération

Dim tableauSpots() As Double
ReDim tableauSpots(1 To numberSimulations, 0 To numberSteps)

' Fixe le générateur de nombre aléatoire (optionnel)
If randomSeed <> 0 Then
   Randomize ([randomSeed])
Else
   Randomize
End If
u = Rnd(-randomSeed)

' Tableau contenant le passage du temps
Dim tableauTime() As Double
ReDim tableauTime(0 To numberSteps)

tableauTime(0) = 0 'Initialisation
For i = 1 To numberSteps
    tableauTime(i) = tableauTime(i - 1) + timeStep
Next i
```

Dans cette première partie du code, les variables déclarées prennent les valeurs associées. Nous déclarons ensuite un tableau qui contient l'évolution du prix pour chaque simulation. La fonction « Randomize » permet de gérer le générateur de nombres aléatoires. L'utilisateur peut décider ou pas de fixer ce générateur. S'il décide de le fixer, il faudra qu'il remplisse la cellule correspondant à « Random Seed » sur l'interface. Le tableau déclaré en-dessous contient le passage du temps à chaque itération.

```
' Simulation de Monte Carlo sous l'hypothèse que le sous-jacent suit une distribution log-normale
For i = 1 To numberSimulations
    tableauSpots(i, 0) = spot
    For j = 1 To numberSteps
        u = Rnd()
        z = Application.WorksheetFunction.Norm_Inv(u, 0, 1)
        tableauSpots(i, j) = tableauSpots(i, j - 1) * Exp((riskFreeRate - dividend - ((volatility ^ 2) / 2)) * timeStep + (volatility * Sqr(timeStep) * z))
        u = Rnd
    Next j
Next i

' Affiche le passage du temps sur la feuille Excel
For i = 0 To numberSteps
    ActiveSheet.Cells(i + 32, 2) = tableauTime(i)
Next i

' Affiche les valeurs sur la feuille Excel jusqu'à 10 simulations si le nombre de simulations est inférieur ou égale à 10
' Affiche le prix et l'écart type des simulations affichées sur la feuille Excel
If numberSimulations <= 10 Then
   For i = 1 To numberSimulations
       ActiveSheet.Cells(32, i + 2) = tableauSpots(i, 0)
       For j = 1 To numberSteps
           ActiveSheet.Cells(j + 32, i + 2) = tableauSpots(i, j)
       Next j
       ActiveSheet.Cells(29, i + 2) = Application.WorksheetFunction.Max(tableauSpots(i, j - 1) - strike, 0) * Exp(-riskFreeRate * life)
       ActiveSheet.Range("D25") = ActiveSheet.Range("D25") + ActiveSheet.Cells(29, i + 2)
   Next i
ActiveSheet.Range("D25") = ActiveSheet.Range("D25") / numberSimulations
ActiveSheet.Range("D26") = Application.WorksheetFunction.StDev(Range(Cells(29, 3), Cells(29, 2 + numberSimulations))) / Sqr(numberSimulations)
ActiveSheet.Range("H19") = ActiveSheet.Range("D25")
ActiveSheet.Range("H20") = ActiveSheet.Range("D26")
End If
```

Dans un premier temps, le code ci-dessus simule l'évolution du prix de l'action pour chaque simulation en partant du prix au comptant de l'action. Toutes les évolutions du prix de l'action sont stockées dans le tableau « tableauSpots ». Ensuite, le passage du temps à chaque itération est affiché dans la feuille

Excel. La dernière partie de cet extrait de code affiche sur la feuille les dix premières simulations si le nombre de simulations est inférieur à dix. Si c'est le cas, il affiche également le prix de l'option pour chacune des dix ou moins premières simulations, ainsi que l'écart type associé à ces simulations.

```
' Affiche les 10 premières valeurs sur la feuille Excel si le nombre de simulations est supérieure à 10
' Affiche le prix et l'écart-type des 10 premières simulations affichées sur la feuille Excel
If numberSimulations > 10 Then
   For i = 1 To 10
       ActiveSheet.Cells(32, i + 2) = tableauSpots(i, 0)
       For j = 1 To numberSteps
           ActiveSheet.Cells(j + 32, i + 2) = tableauSpots(i, j)
       Next j
       ActiveSheet.Cells(29, i + 2) = Application.WorksheetFunction.Max(tableauSpots(i, j - 1) - strike, 0) * Exp(-riskFreeRate * life)
       ActiveSheet.Range("D25") = ActiveSheet.Range("D25") + ActiveSheet.Cells(29, i + 2)
   Next i
ActiveSheet.Range("D25") = ActiveSheet.Range("D25") / 10
ActiveSheet.Range("D26") = Application.WorksheetFunction.StDev(Range(Cells(29, 3), Cells(29, 2 + 10))) / Sqr(10)
End If

' Calcul et affiche sur la feuille Excel le prix final de l'option et l'écart-type de la simulation
If numberSimulations > 10 Then
   Dim tableauResults() As Double
   ReDim tableauResults(1 To numberSimulations)
   For i = 1 To numberSimulations
       tableauResults(i) = Application.WorksheetFunction.Max(tableauSpots(i, numberSteps) - strike, 0) * Exp(-riskFreeRate * life)
       priceCum = priceCum + tableauResults(i)
   Next i
   priceCum = priceCum / numberSimulations
   ActiveSheet.Range("H19") = priceCum
   ActiveSheet.Range("H20") = Application.WorksheetFunction.StDev(tableauResults) / Sqr(numberSimulations)
End If
End Sub
```

Le code ci-dessus correspond à la dernière partie de la procédure que nous décrivons « monteCarloEuropeanLogNormalCall ». Ce code fait la même chose que le code précédent mais dans le cas où le nombre de simulations est supérieur à dix. Il va afficher les dix premières simulations, le prix de l'option pour chacune des dix premières simulations ainsi que l'écart type associé à ces simulations. Il affiche également le prix de l'option et l'écart type de la simulation complète.

A présent, les deux procédures suivantes « monteCarloEuropeanMertonJumpDiffusionCall » et « monteCarloEuropeanMertonJumpDiffusionPut» évaluent par simulation de Monte Carlo une option d'achat ou de vente respectivement (de type européenne et avec comme sous-jacent une action) utilisant le modèle de diffusion à sauts de Merton. Ces deux procédures sont extrêmement proches en terme de code. Comme précédemment, nous allons donc décrire ici uniquement la première des deux procédures citées.

```
Sub monteCarloEuropeanMertonJumpDiffusionCall()

Dim numberSteps%, numberSimulations%, randomSeed%, i%, j%, spot#, strike#, volatility#, riskFreeRate#
Dim life#, timeStep#, S#, jj#, z#, u#, priceCum#, kappa#, drift#, dividend#, lambdaJump#, muJump#
Dim sigmaJump#, N_tt As Variant

Workbooks("vba_project").Sheets("Monte_Carlo").Activate

numberSteps = ActiveSheet.Range("C13").Value 'Nombre d'itérations
numberSimulations = ActiveSheet.Range("C14").Value 'Nombre de simulations
spot = ActiveSheet.Range("C5").Value 'Prix au comptant
volatility = ActiveSheet.Range("H13").Value 'Volatilité
riskFreeRate = ActiveSheet.Range("C6").Value 'Taux sans risque
life = ActiveSheet.Range("H5").Value 'Maturité
strike = ActiveSheet.Range("H6").Value 'Prix d'exercice
randomSeed = ActiveSheet.Range("C15").Value 'Fixation de l'aléa
dividend = ActiveSheet.Range("C16").Value 'Dividende
lambdaJump = ActiveSheet.Range("H14").Value 'Fréquence de saut
muJump = ActiveSheet.Range("H15").Value 'Moyenne liée au saut
sigmaJump = ActiveSheet.Range("H16").Value 'Volatilité liée au saut

timeStep = life / numberSteps 'Le pas d'itération

kappa = Exp(muJump) - 1

drift = riskFreeRate - dividend - (lambdaJump * kappa) - (0.5 * (volatility ^ 2)) 'La tendance

' Fixe le générateur de nombre aléatoire (optionnel)
If randomSeed <> 0 Then
   Randomize ([randomSeed])
Else
   Randomize
End If
u = Rnd(-randomSeed)
```

Dans cette première partie du code, les variables déclarées prennent les valeurs associées. La fonction « Randomize » a la même utilité que précédemment.

```
' Tableau contenant le passage du temps
Dim tableauTime() As Double
ReDim tableauTime(0 To numberSteps)

tableauTime(0) = 0 'Initialisation
For i = 1 To numberSteps
    tableauTime(i) = tableauTime(i - 1) + timeStep
Next i

Dim tableau() As Double
ReDim tableau(1 To numberSimulations, 0 To numberSteps)
```

Nous déclarons ensuite deux tableaux. Le premier tableau contient le passage du temps à chaque itération. Tandis que, le second tableau contient l'évolution du prix pour chaque simulation.

```
' Simulation de Monte Carlo avec le modèle de Merton
For i = 1 To numberSimulations
    tableau(i, 0) = spot
    For j = 1 To numberSteps
        jj = 0
        If lambdaJump <> 0 Then
           N_tt = PoissonInv(u, lambdaJump * timeStep)
           If N_tt(0) > 0 Then
              For S = 1 To N_tt(0)
                  u = Rnd()
                  jj = jj + Application.WorksheetFunction.Norm_Inv(u, muJump - ((sigmaJump ^ 2)) / 2, sigmaJump)
              Next S
           End If
        End If
    u = Rnd()
    z = Application.WorksheetFunction.Norm_Inv(Rnd(), 0, 1)
    tableau(i, j) = tableau(i, j - 1) * Exp((drift * timeStep) + (volatility * Sqr(timeStep) * z) + jj)
    Next j
Next i

' Affiche le passage du temps sur la feuille Excel
For i = 0 To numberSteps
    ActiveSheet.Cells(i + 32, 2) = tableauTime(i)
Next i

' Affiche les valeurs sur la feuille Excel jusqu'à 10 simulations si le nombre de simulations est inférieur ou égale à 10
' Affiche le prix et l'écart type des simulations affichées sur la feuille Excel
If numberSimulations <= 10 Then
   For i = 1 To numberSimulations
       ActiveSheet.Cells(32, i + 2) = tableau(i, 0)
       For j = 1 To numberSteps
           ActiveSheet.Cells(j + 32, i + 2) = tableau(i, j)
       Next j
       ActiveSheet.Cells(29, i + 2) = Application.WorksheetFunction.Max(tableau(i, j - 1) - strike, 0) * Exp(-riskFreeRate * life)
       ActiveSheet.Range("D25") = ActiveSheet.Range("D25") + ActiveSheet.Cells(29, i + 2)
   Next i
ActiveSheet.Range("D25") = ActiveSheet.Range("D25") / numberSimulations
ActiveSheet.Range("D26") = Application.WorksheetFunction.StDev(Range(Cells(29, 3), Cells(29, 2 + numberSimulations))) / Sqr(numberSimulations)
ActiveSheet.Range("H19") = ActiveSheet.Range("D25")
ActiveSheet.Range("H20") = ActiveSheet.Range("D26")
End If
```

Le code ci-dessus simule dans un premier temps l'évolution du prix de l'action pour chaque simulation en partant du prix au comptant de l'action. La fonction « PoissonInv() » est appelée, permettant de simuler l'évolution du prix de l'action sous le modèle à sauts de Merton. Cette fonction est décrite à la page 38. De plus, toutes les évolutions du prix de l'action sont stockées dans le tableau « tableau ». Par la suite, à chaque itération le passage du temps est affiché sur la feuille Excel. La dernière partie du code affiche sur la feuille au maximum les dix simulations si le nombre de simulations est inférieur ou égal à dix. Si c'est le cas, il affiche également le prix de l'option pour chacune des dix ou moins premières simulations, ainsi que l'écart type associé à ces simulations.

```
' Affiche les 10 premières valeurs sur la feuille Excel si le nombre de simulations est supérieure à 10
' Affiche le prix et l'écart-type des 10 premières simulations affichées sur la feuille Excel
If numberSimulations > 10 Then
   For i = 1 To 10
       ActiveSheet.Cells(32, i + 2) = tableau(i, 0)
       For j = 1 To numberSteps
           ActiveSheet.Cells(j + 32, i + 2) = tableau(i, j)
       Next j
       ActiveSheet.Cells(29, i + 2) = Application.WorksheetFunction.Max(tableau(i, j - 1) - strike, 0) * Exp(-riskFreeRate * life)
       ActiveSheet.Range("D25") = ActiveSheet.Range("D25") + ActiveSheet.Cells(29, i + 2)
   Next i
ActiveSheet.Range("D25") = ActiveSheet.Range("D25") / 10
ActiveSheet.Range("D26") = Application.WorksheetFunction.StDev(Range(Cells(29, 3), Cells(29, 2 + 10))) / Sqr(10)
End If

' Calcul et affiche sur la feuille Excel le prix final de l'option et l'écart-type de la simulation
If numberSimulations > 10 Then
   Dim tableauResults() As Double
   ReDim tableauResults(1 To numberSimulations)
   For i = 1 To numberSimulations
       tableauResults(i) = Application.WorksheetFunction.Max(tableau(i, numberSteps) - strike, 0) * Exp(-riskFreeRate * life)
       priceCum = priceCum + tableauResults(i)
   Next i
   priceCum = priceCum / numberSimulations
   ActiveSheet.Range("H19") = priceCum
   ActiveSheet.Range("H20") = Application.WorksheetFunction.StDev(tableauResults) / Sqr(numberSimulations)
End If
End Sub
```

Le code ci-dessus correspond à la dernière partie de la procédure décrite « monteCarloEuropeanMertonJumpDiffusionCall ». Ce code fait la même chose que le code précédent mais dans le cas où le nombre de simulations est supérieur à dix. Il va afficher les dix premières simulations, le prix de l'option pour chacune des dix premières simulations, ainsi que l'écart type associé à ces simulations. Il affiche également le prix de l'option et l'écart type correspondant à la simulation complète.

```
Function PoissonInv(prob As Double, Mean As Double) As Variant
  ' shg 2011, 2012, 2014, 2015-0415

  ' For a Poisson process with mean Mean, returns a three-element array:
  '   o The smallest integer N such that POISSON(N, Mean, True) >= Prob
  '   o The CDF for N-1 (which is < Prob)
  '   o The CDF for N (which is >= Prob)

  ' E.g., POISSON(5, 10, TRUE) returns 0.067085962
  ' PoissonInv(0.067085962, 10) returns 5

  ' Returns a descriptive error if Prob <= 0 or Prob >= 1
  ' If Mean >= 100, then uses a normal approximation

  Dim n             As Long     ' number of events
  Dim cdf           As Double   ' cumulative distribution function at N
  Dim CDFOld        As Double   ' CDF at N-1

  ' These two variables are used to simplify the probability mass
  ' function summation CDF = CDF + Exp(-Mean) * Mean ^ N / N!

  Dim dExpMean      As Double   ' =Exp(-Mean)
  Dim dK            As Double   ' incremental power & factorial

  If prob <= 0 Or prob >= 1 Then
    PoissonInv = "Prob ]0,1["

  ElseIf Mean < 100 Then
    dExpMean = Exp(-Mean)
    dK = 1#
    cdf = dExpMean

    Do While cdf < prob - 0.000000000000001
      CDFOld = cdf
      n = n + 1
      dK = dK * Mean / n
      cdf = cdf + dExpMean * dK
    Loop

    PoissonInv = Array(n, CDFOld, cdf)
  Else
    ' Plan B, for large means; approximate the Poisson as a normal distribution
    ' http://en.wikipedia.org/wiki/Continuity_correction#Poisson
    Dim iInv        As Long

    With WorksheetFunction
      iInv = .Ceiling(.Norm_Inv(prob, Mean, Sqr(Mean)) - 0.5, 1)
      PoissonInv = Array(iInv, _
                         .Norm_Dist(iInv - 0.5, Mean, Sqr(Mean), True), _
                         .Norm_Dist(iInv + 0.5, Mean, Sqr(Mean), True))
    End With
  End If
End Function
```

Cette fonction « PoissonInv() » correspond à la loi de Poisson inversée. Cette fonction est nécessaire pour modéliser le modèle à sauts de Merton. Elle prend en paramètre une certaine probabilité et le nombre moyen d'occurrences de l'événement, c'est-à-dire le nombre moyen de sauts du prix de l'action dans un certain laps de temps. Et, renvoie le nombre d'occurrences selon la probabilité et le nombre moyen d'occurrences fixés.

III. Evaluation des obligations

Cette troisième application concerne l'évaluation des obligations. L'utilisateur a le choix entre quatre périodicités différentes concernant les versements des coupons. Il peut choisir entre des périodicités annuelles, semestrielles, trimestrielles ou encore mensuelles. En outre la détermination du prix de l'obligation, le taux de rendement actuariel, la duration, la sensibilité et la convexité associés à l'obligation sont également calculés. L'interface de cette application se présente sous la forme suivante,

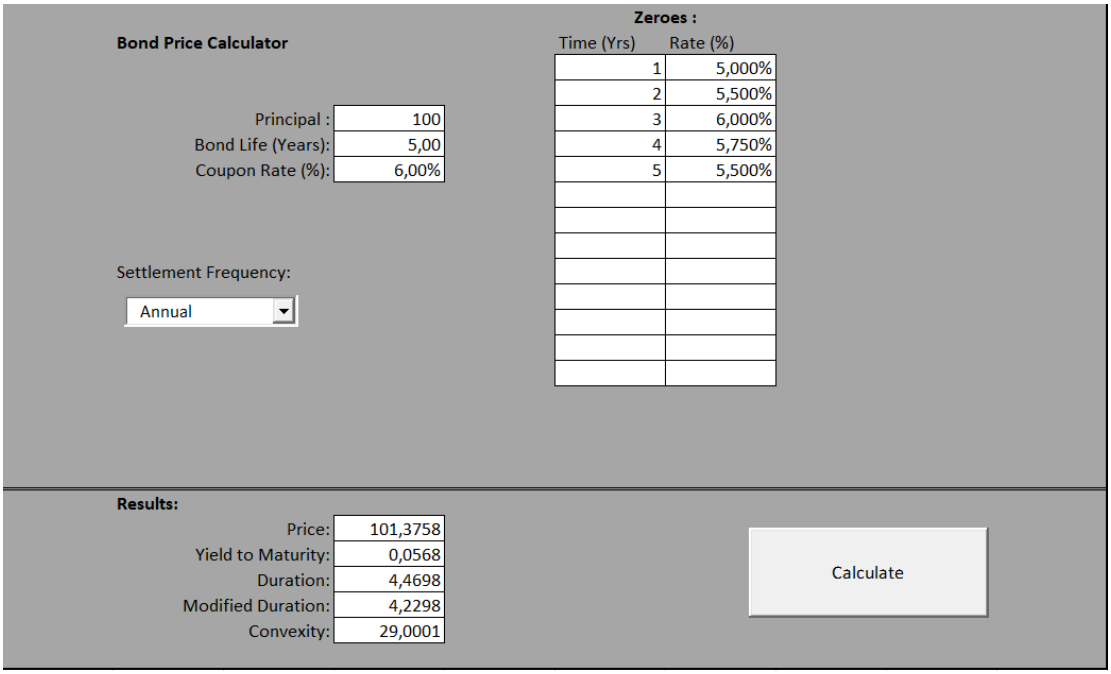

Illustration 12.

Le code permettant l'évaluation de l'obligation est présenté ci-dessous.

```
Sub bonds_compute()

Dim nb_zeroes%, nb_périodes%, compteur_périodes!, compteur_périodes_2!, compteur_périodes_3!
Dim fréquence$, i%, j%, k%, l%, principal#, bond_life#, coupon#, TRA#, taux_zc#
Dim price#, num_duration#, num_conv#, duration#, mod_duration#, convexité#, multiplicateur_périodes%

Workbooks("vba_project").Sheets("bonds_calculator").Activate

fréquence = ActiveSheet.ComboBox1.Value 'Fréquence des intérêts

compteur_périodes = 0
price = 0

principal = Range("D5").Value 'Principal (valeur faciale)
bond_life = Range("D6").Value 'Maturité
coupon = Range("D7").Value 'Coupon

Select Case fréquence
       Case Is = "Monthly"
            multiplicateur_périodes = 12
       Case Is = "Quarterly"
            multiplicateur_périodes = 4
       Case Is = "Semi-Annual"
            multiplicateur_périodes = 2
       Case Is = "Annual"
            multiplicateur_périodes = 1
End Select

nb_périodes = bond_life * multiplicateur_périodes

nb_zeroes = Application.WorksheetFunction.CountA(Range(Cells(3, 7), Cells(15, 7)))
```

Dans cette première partie du code, la variable « multiplicateur_périodes » (de type Integer) est affectée d'un nombre selon la périodicité des coupons choisie par l'utilisateur. Elle permet l'utilisation du même code pour l'évaluation de l'obligation, peu importe sa périodicité.

```
If nb_périodes <> nb_zeroes Then
   MsgBox "Number of Zeroes different from number of periods" 'Contrôle nombre de taux zéro-coupons
Else
    'Calcul prix
    For i = 1 To nb_périodes
        compteur_périodes = compteur_périodes + (1 / multiplicateur_périodes)
        taux_zc = Cells(i + 2, 7).Value
    If compteur_périodes <> bond_life Then
       price = price + (coupon / multiplicateur_périodes) * Exp(-taux_zc * compteur_périodes)
    Else
       price = price + ((principal / 100) + coupon / multiplicateur_périodes) * Exp(-taux_zc * compteur_périodes)
    End If
    Next i

    Range("D21").Value = price * 100

    'Calcul Yield To Maturity
    Dim tableau_flux() As Double
    ReDim tableau_flux(0 To nb_périodes)
    tableau_flux(0) = -price
    For k = 1 To nb_périodes - 1
        tableau_flux(k) = coupon / multiplicateur_périodes
    Next k

    tableau_flux(nb_périodes) = coupon / multiplicateur_périodes + principal / 100
    TRA = Application.WorksheetFunction.IRR(tableau_flux)
    TRA = TRA * multiplicateur_périodes
    Range("D22").Value = TRA
```

Cette seconde partie du code introduit une instruction conditionnelle veillant à ce que l'utilisateur ait rentré un nombre de taux zéro-coupons cohérent avec le nombre de périodicités. Si cette condition est respectée, le calcul du prix de l'obligation est exécuté et le prix est affecté à la cellule D21 de la feuille Excel « bonds_calculator ». Le calcul du taux de rendement actuariel est par la suite exécuté. Ce taux est ensuite affecté à la cellule D22 de la présente feuille.

```
    'Calcul Duration
    For j = 1 To nb_périodes
       compteur_périodes_2 = compteur_périodes_2 + (1 / multiplicateur_périodes)
       taux_zc = Cells(j + 2, 7).Value
       If compteur_périodes_2 <> bond_life Then
          num_duration = num_duration + (compteur_périodes_2 * coupon / multiplicateur_périodes) * Exp(-taux_zc * compteur_périodes_2)
       Else
          num_duration = num_duration + compteur_périodes_2 * ((principal / 100) + coupon / multiplicateur_périodes) * Exp(-taux_zc * compteur_périodes_2)
       End If
    Next j
    duration = num_duration / price
    Range("D23").Value = duration

    'Calcul sensibilité
    mod_duration = duration / (1 + TRA)
    Range("D24").Value = mod_duration

    'Calcul convexité
    For l = 1 To nb_périodes
        compteur_périodes_3 = compteur_périodes_3 + (1 / multiplicateur_périodes)
        taux_zc = Cells(j + 2, 7).Value
        If compteur_périodes_3 <> bond_life Then
           num_conv = num_conv + (compteur_périodes_3 ^ 2 * coupon / multiplicateur_périodes) * Exp(-taux_zc * compteur_périodes_3)
        Else
           num_conv = num_conv + compteur_périodes_3 ^ 2 * ((principal / 100) + coupon / multiplicateur_périodes) * Exp(-taux_zc * compteur_périodes_3)
        End If
    Next l
    convexité = (1 / (1 + TRA) ^ 2) * (duration + num_conv / price)
    Range("D25").Value = convexité
End If

End Sub
```

La dernière partie de cette procédure permet de déterminer la duration, la sensibilité et la convexité associées à l'obligation. Elles sont affectées respectivement aux cellules D23, D24 et D25.

Le prochain chapitre introduit une application permettant l'évaluation des swaps.

IV. Evaluation des swaps

Cette dernière application concerne l'évaluation des swaps. L'utilisateur a la possibilité de choisir la maturité du swap, celle-ci pouvant être au minimum de deux années et au maximum de cinq années. Cet outil d'évaluation est capable de valoriser un swap uniquement avec une périodicité trimestrielle des taux variables. Cette fréquence correspond à l'une des plus utilisées sur les marchés financiers. Ce calculateur permet de valoriser un swap en pourcentage et en euros (selon un notionnel fixé par l'utilisateur). Il permet également de déterminer la sensibilité du prix du swap par rapport à une augmentation de 0.01% de la courbe des taux. L'interface de cette application se présente sous la forme suivante,

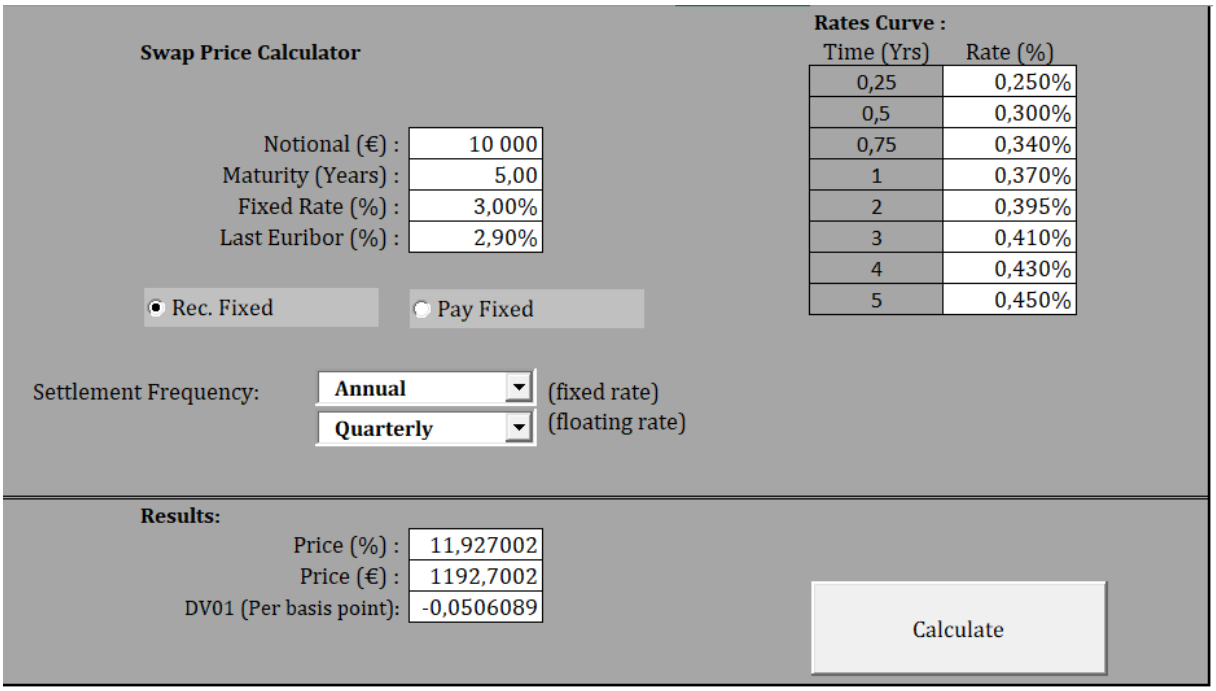

Illustration 13.

Le lancement du calcul se fait lorsque l'utilisateur appuie sur le bouton « Calculate ». Cette procédure appelle la procédure « swaps_compute ».

```
' Cette procédure lance le programme qui calcul le prix du swap et la sensibilité
Private Sub CommandButton1_Click()

Workbooks("vba_project").Sheets("swap_calculator").Activate

swaps_compute

End Sub
```

Le code correspondant à la procédure « swaps_compute » est le suivant,

```
Sub swaps_compute()

Dim notional#, maturity#, fixedRate#, lastEuribor#, delta#, NPV_Swap_perc#, NPV_Swap_abs#

Workbooks("vba_project").Sheets("swap_calculator").Activate

notional = ActiveSheet.Range("D5").Value 'Notionnel
maturity = ActiveSheet.Range("D6").Value 'Maturité
fixedRate = ActiveSheet.Range("D7").Value 'Taux fixe
lastEuribor = ActiveSheet.Range("D8").Value 'Dernier taux variable Euribor

If maturity <= 1 Or maturity > 5 Then
   alertMaturitySwap.Show 0
ElseIf ActiveSheet.OptionButton1.Value = True Then
     NPV_Swap_perc = NPV_swap(notional, maturity, fixedRate, lastEuribor, 1) * 100
     NPV_Swap_abs = NPV_Swap_perc * notional / 100
     delta = DV01_swap(notional, maturity, fixedRate, lastEuribor, 1) * 100 - NPV_Swap_perc
     ActiveSheet.Range("D18").Value = NPV_Swap_perc
     ActiveSheet.Range("D19").Value = NPV_Swap_abs
     ActiveSheet.Range("D20").Value = delta
Else
     NPV_Swap_perc = NPV_swap(notional, maturity, fixedRate, lastEuribor, 0) * 100
     NPV_Swap_abs = NPV_Swap_perc * notional / 100
     delta = DV01_swap(notional, maturity, fixedRate, lastEuribor, 1) * 100 - NPV_Swap_perc
     ActiveSheet.Range("D18").Value = NPV_Swap_perc
     ActiveSheet.Range("D19").Value = NPV_Swap_abs
     ActiveSheet.Range("D20").Value = delta
End If

End Sub
```

Le partie essentielle du code ci-dessus correspond à l'instruction conditionnelle « If ». La première partie de cette instruction vérifie que l'utilisateur ait bien rentré une maturité supérieure à une année ou inférieure ou égale à cinq années. Si ce n'est pas le cas, le programme va envoyer un signal d'alerte sous la forme d'un « UserForm » (voir illustration 14).

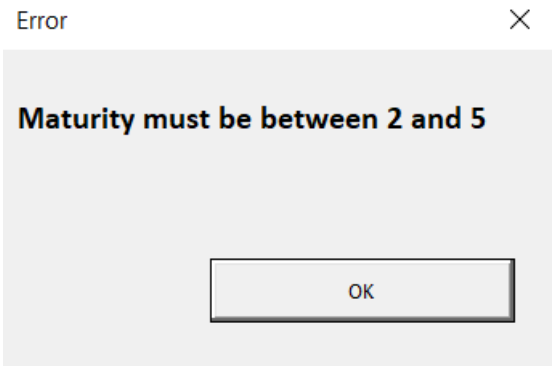

Illustration 14.

Si la condition citée précédemment est respectée, alors, le code va rentrer dans l'instruction « ElseIf » si l'utilisateur a déterminé qu'il est receveur du taux fixe. Dans le cas contraire, l'utilisateur a fixé qu'il est payeur du taux fixe et le code va rentrer dans l'instruction « Else ». Lorsque le code rentre dans l'une des deux dernières instructions il fait appel à la fonction permettant de calculer le prix du swap « NPV_swap » (située dans le module « Swap_functions »).

Le code de cette procédure est la suivante,

```
Function NPV_swap(notional#, maturity#, fixedRate#, lastEuribor#, X%) As Double

' Cette fonction calcule le prix d'un swap pour un taux fixe annuel donné, un premier taux varibale
' trimestriel donné et l'utilisation de la courbe de taux pour la détermination des autres taux
' variables trimestriels.
'
' INPUTS
'-----------------------------------------------------------------
' notional    : notionnel
' maturity    : maturité
' fixedRate   : taux fixe annuel
' lastEuribor : premier taux variable trimestriel
' X           : 1 si receveur du fixe, 0 sinon
'
' OUTPUT
'-----------------------------------------------------------------
'
' Prix du swap
'
'-----------------------------------------------------------------

Dim mult_period%, nb_rates%, nb_periods%, i#, j#
Dim Base#, invBase#, NPV_JF#, NPV_JV#, u#, v#, cumSumBeta#
Dim NPV_Swap_perc#, NPV_Swap_abs#, npv_swap_X#

mult_period = 4 'On assume des taux variables trimestriels

nb_periods = mult_period + maturity - 1

nb_rates = Application.WorksheetFunction.CountA(Range(Cells(3, 8), Cells(15, 8)))
```

Cette fonction prend en paramètre le montant de notionnel, la maturité, le taux fixe, le dernier taux Euribor et une dernière variable X de type « Integer ». Dans le cas, où cette variable est égale à 1 la fonction va calculer le prix du swap pour un receveur du taux fixe, sinon pour un payeur du taux fixe . La fonction « CountA » va compter le nombre de taux que l'utilisateur a rentré sur la plage de cellules H3:H10 sur la feuille « swap_calculator ».

```
If nb_periods <> nb_rates Then
   MsgBox "Number of Rates different from number of periods" 'Contrôle le nombre de taux
Else
   Dim tabRates() As Double 'Tableau contenant la courbe de taux
   ReDim tabRates(1 To nb_rates)
   For i = 1 To nb_rates Step 1
       tabRates(i) = ActiveSheet.Cells(i + 2, 8)
   Next i

   Dim tabBeta() As Double 'Tableau contenant les betas
   ReDim tabBeta(1 To nb_rates)
   For i = 0.25 To 1 Step 0.25
       j = j + 1
       tabBeta(j) = 1 / ((1 + tabRates(j)) ^ i)
   Next i
   For j = 5 To nb_rates Step 1
       cumSumBeta = cumSumBeta + tabBeta(j - 1)
       tabBeta(j) = (1 - tabRates(j) * cumSumBeta) / (1 + tabRates(j))
   Next j

   j = 0
   Dim tabZC() As Double 'Tableau contenant les taux zéro-coupon
   ReDim tabZC(1 To nb_rates)
   For i = 1 To 4 Step 1
       tabZC(i) = tabRates(i)
   Next i
   For i = 5 To nb_rates
       tabZC(i) = ((1 / tabBeta(i)) ^ (1 / (i - 3))) - 1
   Next i

   Dim tabZCFinal() As Double 'Tableau contenant les taux zéro-coupon finaux
   ReDim tabZCFinal(1 To 4 * maturity)
   For i = 1 To 4 Step 1
       tabZCFinal(i) = tabZC(i)
   Next i
   For j = 2 To maturity
       tabZCFinal((j - 1) * 4 + 1) = tabZC(j + 2) + 0.25 * (tabZC(j + 3) - tabZC(j + 2))
       tabZCFinal((j - 1) * 4 + 2) = tabZC(j + 2) + 0.5 * (tabZC(j + 3) - tabZC(j + 2))
       tabZCFinal((j - 1) * 4 + 3) = tabZC(j + 2) + 0.75 * (tabZC(j + 3) - tabZC(j + 2))
       tabZCFinal((j - 1) * 4 + 4) = tabZC(j + 3)
   Next j
```

Le code continue ci-dessus avec une instruction conditionnelle « If ». Il va tester si le nombre de taux rentré par l'utilisateur est cohérent avec la maturité. Si ce n'est pas le cas, le code affiche un message d'erreur via un « MsgBox » puis met fin à la procédure. Si la condition est respectée, le code va dans un premier temps créer un tableau contenant les variables de la courbe des taux « tabRates », puis un deuxième tableau « tabBeta », qui va permettre le calcul des taux zéro-coupon. Les prix des taux zéro-coupon infra-annuels sont identiques aux taux de la courbe des taux, en effet dans la boucle « For » les taux d'échéance trois mois, six mois, neuf mois et un an sont copiés à l'identique. Pour les échéances supérieures à un an, une équation doit être appliquée. Les taux zéro-coupon sont ensuite calculés puis stockés dans le tableau « tabZCFinal ». Les taux zéro-coupon infra-annuels sont identiques à ceux de la courbe des taux. Là aussi, pour les échéances supérieures à un an, une équation doit être appliquée.

```
    Dim tabBetaFinal() As Double 'Tableau contenant les betas finaux
    ReDim tabBetaFinal(1 To 4 * maturity)
    For j = 1 To maturity
        u = u + 0.25
        v = v + 1
        tabBetaFinal(v) = 1 / ((1 + tabZCFinal(v)) ^ u)
        u = u + 0.25
        v = v + 1
        tabBetaFinal(v) = 1 / ((1 + tabZCFinal(v)) ^ u)
        u = u + 0.25
        v = v + 1
        tabBetaFinal(v) = 1 / ((1 + tabZCFinal(v)) ^ u)
        u = u + 0.25
        v = v + 1
        tabBetaFinal(v) = 1 / ((1 + tabZCFinal(v)) ^ u)
    Next j

    Base = 90 / 360 'Base
    invBase = 360 / 90 'Base inversée

    Dim tabEuribor() As Double 'Tableau contenant les taux Euribor 3 mois
    ReDim tabEuribor(1 To 4 * maturity)
    tabEuribor(1) = lastEuribor
    For i = 2 To 4 * maturity
        tabEuribor(i) = ((tabBetaFinal(i - 1) / tabBetaFinal(i)) - 1) * invBase
    Next i

    For i = 4 To maturity * 4 Step 4
        NPV_JF = NPV_JF + (fixedRate * tabBetaFinal(i))
    Next i
    For i = 1 To 3
        NPV_JV = NPV_JV + (Base * tabEuribor(i)) / (1 + Base * tabZCFinal(i))
    Next i
    For i = 4 To 4 * maturity
        NPV_JV = NPV_JV + (Base * tabEuribor(i) * tabBetaFinal(i))
    Next i
```

Lorsque les taux zéro-coupon sont calculés, un tableau « tabBetaFinal » est créé. Celui-ci contient les prix zéro-coupon. Les taux Euribor sont ensuite calculés et stockés dans le tableau « tabEuribor ». Les deux boucles « For » qui suivent calculent la valeur actuelle nette de la jambe fixe et de la jambe variable de swap « NPV_JF » et « NPV_JV » respectivement. La dernière étape du code est une instruction conditionnelle « If », vérifiant si l'utilisateur est receveur ou payeur du taux fixe. Selon la sens du taux fixe la fonction renvoie la valeur du swap adéquate.

```
    If X = 1 Then
       npv_swap_X = NPV_JF - NPV_JV 'Valeur du swap si on est receveur du taux fixe
    Else
       npv_swap_X = NPV_JV - NPV_JF 'Valeur du swap si on est payeur du taux fixe
    End If

    NPV_swap = npv_swap_X 'Valeur final du swap

End If

End Function
```

Une fois la valeur du swap déterminée, la procédure « swap_compute » affecte cette valeur en pourcentage et en euros dans les variables « NPV_Swap_perc » et « NPV_Swap_abs » respectivement. La fonction « DVO1_swap » est ensuite appelée (située dans le module « Swap_functions »). Cette fonction calcule le prix du swap, si la courbe des taux augmente de 0.01%. Cette fonction est identique à celle décrite précédemment à une exception près : chaque taux que rentre l'utilisateur est majoré de 0.01%. Ainsi, la différence entre le prix du swap dont les taux sont majorés de 0.01% et le prix du swap sans majoration représente la sensibilité du swap par rapport à la courbe des taux, soit son delta. Le prix du swap en pourcentage, en euros et sa sensibilité sont respectivement affectés aux cellule « D18 », « D19 » et « D20 » de la feuille Excel « swap_calculator ».