CLAIRE PELTIER 2007

Plan du premier trimestre :

Cours 1 : INITATION VBA

1- PRESENTATION
2- ACCES VISUAL BASIC EDITOR
3- Les variables
4- Boucles et conditions.
A-Les conditions
B- Les boucles
C- Les boucles conditionnelles
5- Les tableaux (vecteur, matrice)
A- Déclaration :
B -Initialisation :
C- Courbe des taux,

Cours 2

1- Appel des Fonctions Préprogrammées
2- Black-Scholes Généralisé
3- Obligation TF,TV sans risque de défaut

Cours 3

4- Simulation Monte Carlo : Option Call Put sur Action (Boyle 1977)

Cours 4

5- Modèle Binomial : Cox Ross Rubinstein (1979)
6- Call Put Européen Américain

Cours 5

7- SWAP, SWAPTION
8- CAP FLOOR

CLAIRE PELTIER 2007

COURS FINANCE ET INFORMATIQUE

Cours 1 : INITATION VBA

1- PRESENTATION

Visual Basic pour Applications est le langage de programmation des applications de Microsoft Office. VBA permet d'automatiser les tâches, de créer des applications complètes, de sécuriser vos saisies et vos documents, de créer de nouveaux menus et de nouvelles fonctions pour booster efficacement votre logiciel.

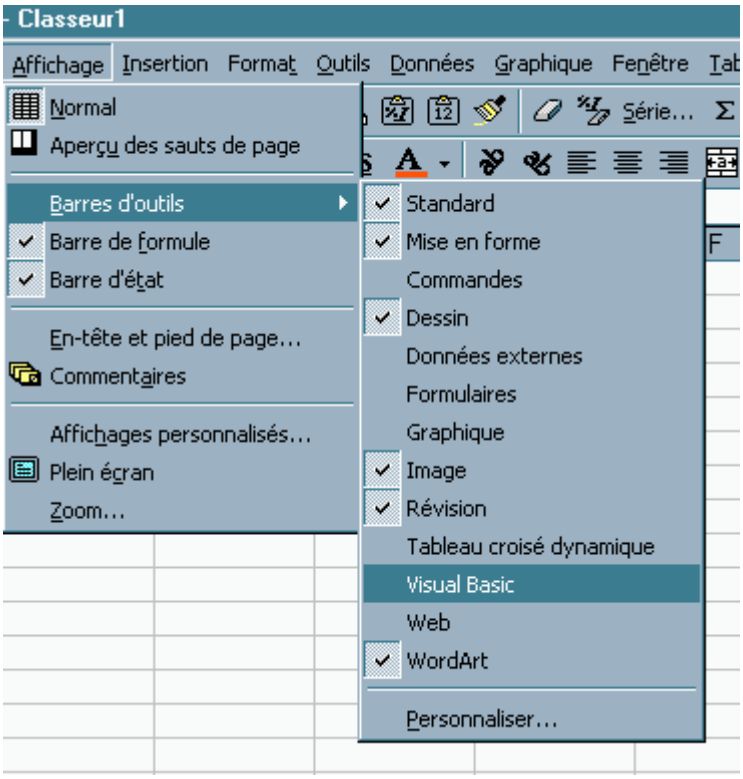
VBA utilise le même langage que Microsoft Visual Basic. La différence entre VB et VBA est que VB est un ensemble complet qui permet de développer des applications indépendantes et librement distribuables alors qu'une application réalisée en VBA est complètement liée au logiciel sous lequel elle a été créée (une application VBA créée sous Excel ne pourra pas se lancer sur un poste si Excel n'est pas installé).

Avant qu'Excel n'utilise ce langage de programmation, le logiciel utilisait son propre langage de programmation et une application était appelée « macro ». Ce terme est resté, mais une macro Excel réalisée en VBA n'est rien d'autre qu'une procédure telle qu'elles sont réalisées sous VB. Un programmeur sous VBA n'a aucun problème pour passer à VB et vice-versa.

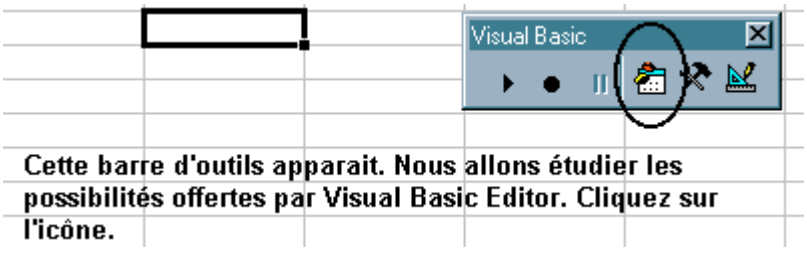
Le langage VBA est accessible à tous. Cependant, une bonne connaissance d'Excel est nécessaire avant de se lancer dans la création d'application. En effet, il est important de bien maîtriser les principaux objets que manipule VBA, comme les objets Workbook (classeur), Worskheet (Feuille de calcul), Range(plage de cellule), etc…

2- ACCES VISUAL BASIC EDITOR

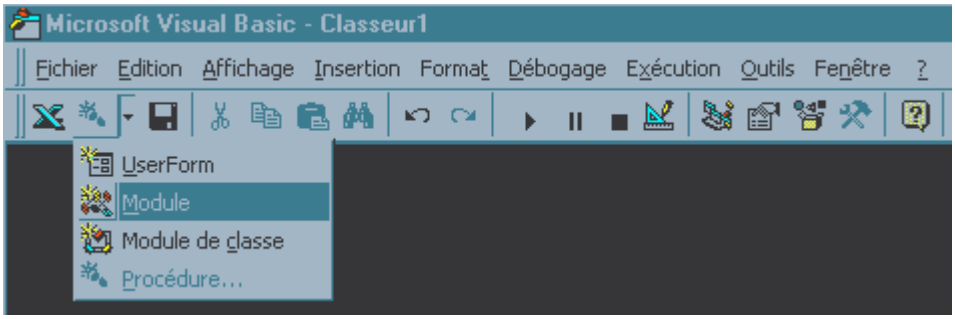
Pour accéder à Visual Basic Editor
faites apparaître la barre d'outils Visual Basic



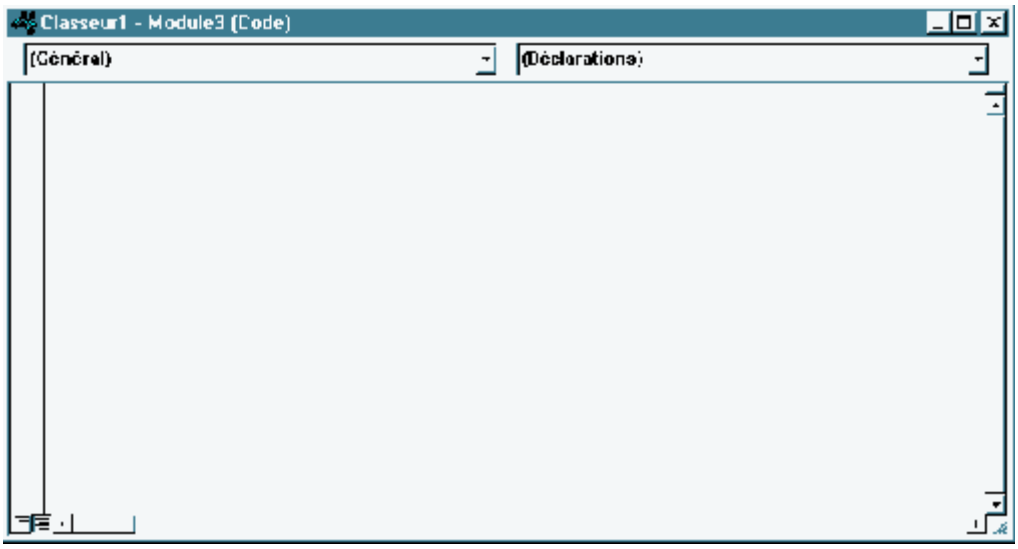
une fois la barre d'outils en place



Une fois dans Visual Basic Editor, cliquez sur la 2ème icône



Un module vierge apparaît



Une procédure est une suite d'instructions effectuant des actions. Elle commence par Sub + NomDeLaProcédure et se termine par End Sub. Le nom des procédures ne doit pas contenir d'espaces. Utilisez le caractère de soulignement pour séparer les mots. Je vous conseille de les écrire comme des noms propres.

Pour déclarer une procédure, taper Sub et son nom puis taper Entrée. VBE ajoute automatiquement les parenthèses et la ligne End Sub.
Exemple de Procédure nommée Essai :

```
Sub Essai()
    MsgBox "Bonjour"
End Sub
```

Il faut ensuite compiler : Debogage + Compiler VBA Project ensuite exécuter Execution+ Executer la Macro (ou F5)

Une fonction est une procédure qui renvoie une valeur.
Exemple de fonction nommée Calcul :

```
Function Calcul(Nbre1 As Integer, Nbre2 As Integer) As Integer
    Calcul = Nbre1 + Nbre2
End Function
```

Il faut ensuite compiler : Debogage + Compiler VBA Project et retourner sur la feuille Excel et appeler la fonction : Insertion + fx fonction +Personnalisees+Nomde la fonction crée : une boîte de dialogue apparaît : à vous de remplir les données

3- Les variables

Lors d'une procédure, les variables servent à stocker toutes sortes de données (des valeurs numériques, du texte, des valeurs logiques, des dates ...). Elles peuvent également faire référence à un objet.
Suivant les données que la variable recevra, on lui affectera un type différent. Les différents types de variables de VB sont :

Type de données:	*Mot clé :*	Espace occupé	Plage de valeur
Logique	Boolean	2 octets	True(1) ou False(0)
Entier	Integer	2 octets	Entier de -32 768 à 32 768
Entier Long	Long	4 octets	Entier de -2 147 483 648 à 2 147 483
Décimal Double	Double	8 octets	-1,79769313486231E308 à -4,94065645841247E-324 pour les valeurs négatives 4,94065645841247E-324 et 1,79769313486231E308 pour les valeurs positives
Date	Date	8 octets	1er Janvier 1900(1) au 31 décembre 9999
Chaîne de caractères à longueur fixe	String	Longueur de la chaîne	1 à 65 400 caractères
Variant avec chiffres	Variant	16 octets	Ce type de données peut contenir des données de toutes sortes, à l'exception des données de type String .
Défini par l'utilisateur	Type	Variable	Identique au type de données.

Un octet est une unité de mesure en informatique mesurant la quantité de données. Un octet est lui-même composé de 8 bits, soit 8 chiffres ou lettres...
1000 octets sont 1 Kilo octet Ko, Un mégaoctet représente un million d'octets. Un gigaoctet représente un milliard d'octets. Un hexaoctet représente un milliard de milliards d'octets.
Une image GIF prend environ 50 Ko.

Attention au nombre de lignes dans une feuille d'Excel, elles sont de 65536 depuis la version 97, donc si vous devez lire chaque ligne d'une feuille à l'aide d'une variable, celle-ci ne peut être de type *Integer*, mais doit être de type Long.

Pour rendre obligatoire la déclaration de variables, placez l'instruction "Option Explicit" sur la première ligne du module.
La déclaration explicite d'une variable se fait par le mot Dim (abréviation de Dimension). Le nombre maximum de caractères du nom de la variable est de 255. Il ne doit pas commencer par un chiffre et ne doit pas contenir d'espaces.

La syntaxe est "Dim NomDeLaVariable As Type".

Vous pouvez également déclarer vos variables sur une même ligne :

```
Function Calcul_plus(Nbre1 As Integer, Nbre2 As Integer) As Integer

    Dim SommeVal As Integer, Val1 As Integer, Val2 As Integer

    Val1 = 5
     Val2 = 2
    SommeVal = Val1 + Val2

    Calcul_plus = Nbre1 + Nbre2+SommeVal

 End Function
```

La portée d'une variable est différente suivant l'endroit et la façon dont elle est déclarée.

Une variable déclarée à l'intérieur d'une procédure est dite "Locale". Elle peut-être déclarée par les mots Dim ou Static. Dès que la procédure est terminée, la variable n'est plus chargée en mémoire sauf si elle est déclarée par le mot Static. Une variable Locale est généralement placée juste après la déclaration de la procédure.

Une variable peut garder toujours la même valeur lors de l'exécution d'un programme. Dans ce cas, elle est déclarée par les mots Const ou Global Const.

```
Option Explicit
Global Const Pi = 3.14159265358979
```

4- Boucles et conditions.

A-Les conditions :

Les conditions sont très courantes dans les applications VB. Elles peuvent déterminer la valeur que prennent les variables, arrêter une procédure, appeler une procédure, quitter une boucle, atteindre une étiquette.
Les exemples suivants vont déterminer la valeur que prendra la variable Mention par rapport à des notes.
Le tableau des notes est :

Notes :	*Mention* :
0	Nul
1 à 5	Moyen
6 à 10	Passable
11 à 15	Bien
16 à 19	Très bien
20	Excellent

L'instruction la plus courante dans VB est la condition

If *condition* Then *faire une opération* :

```
'Pour trouver la valeur de la mention, on pourrait écrire :

Function Mention(Note As Double) As String
    If Note = 0 Then Mention = "Nul"
    If Note >= 1 And Note <6 Then Mention = "Moyen"
    If Note >= 6 And Note <11 Then Mention = "Passable"
    If Note >= 11 And Note <16 Then Mention = "Bien"
    If Note >= 16 And Note <20 Then Mention = "Très Bien"
    If Note = 20 Then Mention = "Excellent"
End Function
```

Si la valeur vraie possède plusieurs lignes d'instructions, la syntaxe devient

If *Condition* Then
faire une opération
faire une seconde opération
End If

Pour calculer la valeur de la mention, on utilisera plus facilement la syntaxe
If *Condition* Then
faire une opération
Elsef *Condition* Then
Faire une autre opération
End If

On peut ajouter autant de fois que nécessaire l'instruction ElseIf.

```
Function Mention2(Note As Double) As String
    If Note = 0 Then
       Mention2 = "Nul"
    ElseIf Note >= 1 And Note <6 Then
       Mention2 = "Moyen"
    ElseIf Note >= 6 And Note <11 Then
       Mention2 = "Passable"
    ElseIf Note >= 11 And Note <16 Then
       Mention2 = "Bien"
    ElseIf Note >= 16 And Note <20 Then
       Mention2 = "Très Bien"
    Else
       Mention2 = "Excellent"
    End If
End Function
```

Dans le cas de conditions multiples, comme dans notre exemple, on préférera le bloc d'instruction

```
Select Case Valeur à tester
        Case Valeur  1
                faire une opération
        Case Valeur  2
                faire une opération
        Case Valeur  n
                faire une opération
End Select
```

Exemple :

```
Function Mention3(Note As Double) As String

    Select Case Note
       Case 0
          Mention3 = "Nul"
       Case 1 To 5
          Mention3 = "Moyen"
       Case 6 To 10
          Mention3 = "Passable"
       Case 11 To 15
          Mention3 = "Bien"
       Case 16 To 19
          Mention3 = "Très Bien"
       Case Else
          Mention3 = "Excellent"
    End Select
End Function
```

B- Les boucles :

Les boucles le plus souvent utilisés sont les boucles For ... Next. Elles permettent de répéter un nombre de fois défini un bloc d'instructions. Elles utilisent une variable qui est incrémentée ou décrémentée à chaque répétition

```
Function Sum_entier (n as integer) as integer

    'La boucle suivante va calculer la somme des entiers jusqu'à n
    Dim Resultat as integer
    Dim i as integer
    Resultat = 0

    For i  = 1 To n
       Resultat = Resultat + i
    Next i

    Sum_entier = Resultat

End Function
```

La variable peut être incrémentée d'une valeur différente de 1 par le mot Step.

```
Function Sum_entier2 (n as integer) as integer

    'La boucle suivante va calculer la somme des entiers paires jusqu'à n
    Dim Resultat as integer
    Dim i as integer
    Resultat = 0

    For i  = 0 To n Step 2
       Resultat = Resultat + i
    Next i

    Sum_entier2 = Resultat

End Function
```

La variable peut également être décrémentée. Dans ce cas, le mot Step est obligatoire.

```
Function Sum_entier3 (n as integer) as integer

    'La boucle suivante va calculer la somme des entiers jusqu'à n
    Dim Resultat as integer
    Dim i as integer
    Resultat = 0

    For i  = n To 1 Step -1
       Resultat = Resultat + i
    Next i

    Sum_entier3 = Resultat

End Function
```

A l'intérieur d'un bloc d'instruction For Next, l'instruction Exit For, peut quitter la boucle avant que la variable n'est atteint sa dernière valeur

```
Function Sum_entier_12(n As Integer) As Integer
    'La boucle suivante va calculer la somme des entiers jusqu'à n<=12
    Dim Resultat As Integer
    Dim i As Integer
    Resultat = 0
        For i = 1 To n
                If i > 12 Then
                Exit For
                End If
         Resultat = Resultat + i
        Next i
    Sum_entier_12 = Resultat

End Function
```

C- Les boucles conditionnelles:

Les boucles While *condition* Wend exécutent un bloc d'instruction tout pendant que la condition est vraie.

```
    Function Sum_entier_carre(n As Integer) As Integer

    Dim Resultat As Integer
    Dim i As Integer
    Resultat = 0
    i=0
    'Le bloc d'instruction suivant va additionner les
    ' nombres de 1 à 10 au carré Tant que la valeur de Compteur est inférieur n

    While i <= n
       Resultat = Resultat + i^2
       'Ne pas oublier d'incrémenter le compteur sinon
       'la boucle ne pourra pas s'arrêter.
       i = i + 1
    Wend
    Sum_entier_carre = Resultat

    End Function
```

Remarque : pour élever un nombre à une puissance décimale utiliser un point :
i^(2.5)

5- Les tableaux

Contrairement aux variables classiques qui contiennent une seule valeur, un tableau est une variable qui peut contenir un ensemble de valeurs de même type.

En debut de programme, il faut choisir la valeur du premier incrément des tableaux :

Option Base 0 L'indice de la première case du tableau est 0

Option Base 1 L'indice de la première case du tableau est 1

A- Déclaration :

Vecteur de Double de n dimension :
Au debut de la fonction :
Dim Vecteur() As Double
Après toutes les déclaration et avant le début du programme
ReDim Vecteur(n)

Matrice de Double de n, m dimension :
Au debut de la fonction :
Dim Matrice() As Double
Après toutes les déclaration et avant le début du programme
ReDim Matrice(n, m)

B- Initialisation :

Exemple initialiser un vecteur de double à 4%

```
Si Option Base 0
        For i = 0 To n-1
         Vecteur(i)= 0.04
        Next i

Si Option Base 1
        For i = 1 To n
         Vecteur(i)= 0.04
        Next i
```

Exemple initialiser une matrice de double à i+j , i numéro de ligne, j numéro de colonne

```
Si Option Base 0
        For i = 0 To n-1
                For j = 0 To n-1
                matrice(i , j)= i+j+2
                Next j
        Next i

Si Option Base 1
        For i =  To n
                For j = 1 To n
                matrice(i , j)= i +j
                Next j
        Next i
```

C- Courbe des taux

Le but de ce paragraphe est d'aller chercher sur Excel des données qui seront utilisées par la suite dans le programme sous forme de matrice :

Exemple : courbe de taux
courbe zéro-coupon

5-nov-08	3,00%
31-déc-07	3,03%
31-mars-08	3,06%
30-juin-08	3,08%
30-sept-08	3,09%
31-déc-08	3,11%
31-mars-09	3,13%
30-juin-09	3,15%
30-sept-09	3,17%
31-déc-09	3,18%
31-mars-10	3,19%
30-juin-10	3,20%
30-sept-10	3,20%
31-déc-10	3,21%
31-mars-11	3,22%
30-juin-11	3,24%
30-sept-11	3,25%
30-déc-11	3,26%
30-mars-12	3,28%
29-juin-12	3,29%
28-sept-12	3,31%
31-déc-12	3,32%
28-mars-13	3,33%
28-juin-13	3,35%
30-sept-13	3,36%
31-déc-13	3,38%
30-juin-14	3,40%
31-déc-14	3,43%
30-juin-15	3,45%
31-déc-15	3,48%
30-déc-16	3,53%
29-déc-17	3,58%
31-déc-18	3,62%
31-déc-19	3,66%
31-déc-20	3,70%

La première colonne : les dates

La seconde colonne : les taux

Attention : Les TAUX doivent comporter des VIRGULES et non des points !

Cette procédure initialise la matrice aTab aux valeurs de la feuilles Excel quel que soit l'endroit où se situe les valeurs de la page précédente et sa taille

```
Sub loadZCRange (aValuedate As Date, aRange As Range, aTab() As Double)

    Dim lNbRow As Integer, lNbColumn As Integer
    Dim I As Integer, J As Integer
    Dim lTemp() As Date

    lNbRow = aRange.Rows.Count
    lNbColumn = aRange.Columns.Count

    ReDim aTab(1 To lNbRow, 1 To lNbColumn)
    ReDim lTemp(1 To lNbRow)

    For J = 1 To lNbRow
        lTemp(J) = aRange.Cells(J, 1)
        aTab(J, 1) = (lTemp(J)-aValuedate)/365.25
        aTab(J, 2) = aRange.Cells(J, 2)
    Next

End Sub
```

Pour ensuite utiliser cette courbe de taux dans un autre programme il suffit de :

- Déclarer comme variable dans la fonction le tableau de la courbe des taux :

```
Function discount_factor(aValuedate As Date, aMaturity As Date, aZCrange As Range) As Double
```

- Déclarer le nom de la matrice d'où est sauvegarder la courbe pour les calculs dans les premières déclarations :

```
Dim lZcTab() As Double
```

***Remarque* : on choisit un nom différent pour ne pas modifier les valeurs initiales**

- Au début de la fonction, juste après la déclaration des variables on écrit :

```
Call loadZCRange(aValuedate, aZCrange, lZcTab)
```

La courbe et ses dates sont maintenant dans la matrice lZcTab et on peut faire tous les calculs demandés par le programme.

Exemple Discount Factor :

```
Function discount_factor(aValuedate As Date, aMaturity As Date, aZCrange As Range) As Double
    Dim lMAT As Double
    Dim lZcTab() As Double
    Dim lNbRow As Integer, I As Integer
    Dim ltaux As Double

     lMAT = (aMaturity- aValuedate)/365.25

    Call loadZCRange(aValuedate, aZCrange, lZcTab)

    lNbRow = aZCrange.Rows.Count

    For I = 1 To (lNbRow - 1)
       If (lZcTab(I, 1) < lMAT) Then
          If (lMAT <= lZcTab(I + 1, 1)) Then
           ltaux = interpolation(lZcTab(I, 1), lZcTab(I + 1, 1), lZcTab(I, 2), lZcTab(I + 1, 2), lMAT)
          End If
       End If
    Next I

    discount_factor = 1 / (1 + ltaux) ^ lMAT

End Function
```

avec

```
Function interpolation(aX1 As Double, aX2 As Double, aY1 As Double, aY2 As Double, aX As Double)

    interpolation = (aY1 - aY2) / (aX1 - aX2) * (aX - aX2) + aY2
End Function
```

CLAIRE PELTIER 2007

***Exercice : Prix d'une obligation Taux Fixe en utilisant la fonction* discount_factor**

Données : Courbe des taux, Dates de tombées de coupons, Coupons, Notionnel

Notionnel	Date de tombée de coupon	Coupon	Discount Factor	Prix actualisé	Prix Obligation
10 000,00 €	5 novembre 2007	5,00%			
10 000,00 €	5 novembre 2008	6,00%			
10 000,00 €	5 novembre 2009	3,00%			
10 000,00 €	5 novembre 2010	3,50%			
10 000,00 €	5 novembre 2011	5,00%			
10 000,00 €	5 novembre 2012	5,00%			
10 000,00 €	5 novembre 2013	5,00%			
10 000,00 €	5 novembre 2014	5,00%			
10 000,00 €	5 novembre 2015	5,00%			
10 000,00 €	5 novembre 2016	5,00%			
10 000,00 €	5 novembre 2017	5,00%			

CLAIRE PELTIER 2007

COURS FINANCE ET INFORMATIQUE

Cours 2

9- Appel des Fonctions Préprogrammées
10-Black-Scholes Généralisé
11-Obligation TF,TV sans risque de défaut

1- Appel des fonctions Excel

Pour pouvoir appeler les fonctions de Excel en VBA il faut paramétrer VBA :

- **Aller sur Excel+ Outils+MacroCompémentaire + Cocher Utilitaires d'analyse et Utilitaire d'analyse VBA**

- **Aller sur sur VBA+Outils+reference+ Cocher ATBVBAEN.xls + solver.xls**

***The Excel function is part of the Analysis Tool Pak add-in. To use it in VBA, you need to first load the "Analysis Tool Pak - VBA" add-in in Excel. Then, open your VBA project, go to the Tools menu, choose Reference, and put a check next to ATBVBAEN.xls item. Once you do this, you can access the function directly*.**

Accès aux fonctions de VBA :

Appel direct dans le programme de la fonction.

Liste des fonction :

Function **Abs**(Number)
Membre de VBA.Math

Function **Cos**(Number As Double) As Double
Membre de VBA.Math

Function **Exp**(Number As Double) As Double
Membre de VBA.Math

Function **Log**(Number As Double) As Double
Membre de VBA.Math

Function **Sin**(Number As Double) As Double
Membre de VBA.Math

Function **Sqr**(Number As Double) As Double
Membre de VBA.Math

Function **DateDiff**(Interval As String, Date1, Date2, [FirstDayOfWeek As VbDayOfWeek = vbSunday], [FirstWeekOfYear As VbFirstWeekOfYear = vbFirstJan1])
Membre de VBA.DateTime

Dans VBA, the **DateDiff** function retourne la différence entre des dates, suivant un intervalle spécifique. It

CLAIRE PELTIER 2007

La syntaxe de DateDiff function est:

DateDiff(interval, date1, date2, [firstdayofweek], [firstweekofyear])

interval est l'intervalle de temps utilisé pour calculer cette différence.

Interval	Explanation
yyyy	Year
q	Quarter
m	Month
y	Day of year
d	Day
w	Weekday
ww	Week
h	Hour
n	Minute
s	Second

date1 and *date2* sont les deux dates entre lesquelles on calcule la différence.

firstdayofweek est optionnel. C'est une constante qui spécifie le premier jour de la semaine. Si ce paramètre est omis, Excel suppose que le premier jour de la semaine est le dimanche.
firstweekofyear is optional. est optionnel. C'est une constante qui spécifie le premier jour de la semaine. . Si ce paramètre est omis, Excel suppose que le premier jour de l'année est le 1er janvier .

Accès aux fonctions de ATPVBAEN.XLS :

Appel direct dans le programme de la fonction.

Liste des fonctions « intéressantes »:

Function **Accrint**(issue, first_interest, settlement, rate, par, frequency, [basis])
Membre de ATPVBAEN.XLS.Fonctions et procédures VBA
Renvoie l'intérêt couru non échu d'un titre dont l'intérêt est perçu périodiquement

Function **Duration**(settlement, maturity, coupon, yld, frequency, [basis])
Membre de ATPVBAEN.XLS.Fonctions et procédures VBA
Calcule la durée d'un titre avec des paiements d'intérêts périodiques

Function **Edate**(start_date, months)
Membre de ATPVBAEN.XLS.Fonctions et procédures VBA
Renvoie le numéro de série de la date située un nombre spécifié de mois dans le passé ou le futur par rapport à une date indiquée

Function **Randbetween**(bottom, top)
Membre de ATPVBAEN.XLS.Fonctions et procédures VBA
Renvoie un nombre aléatoire entre les nombres que vous spécifiez

Function **MDuration**(settlement, maturity, coupon, yld, frequency, [basis])
Membre de ATPVBAEN.XLS.Fonctions et procédures VBA
Renvoie la durée de Macauley modifiée d'un titre, pour une valeur nominale considérée égale à 100 F

CLAIRE PELTIER 2007

Function **Price**(settlement, maturity, rate, yld, redemption, frequency, [basis])
Membre de ATPVBAEN.XLS.Fonctions et procédures VBA
Renvoie le prix d'un titre rapportant des intérêts périodiques, pour une valeur nominale de 100 F

Function **Yearfrac**(start_date, end_date, [basis])
Membre de ATPVBAEN.XLS.Fonctions et procédures VBA
Renvoie une fraction correspondant au nombre de jours séparant date_début de date_fin par rapport à une année complète

Function **Yield**(settlement, maturity, rate, par, redemption, frequency, [basis])
Membre de ATPVBAEN.XLS.Fonctions et procédures VBA
Calcule le rendement d'un titre rapportant des intérêts périodiquement

Accès aux fonctions de Excel :

Au cours d'un procédure ou d'une function VBA, pour appeler une fonction prédéfinie il faut généralement faire appel au chemin suivant :

Application.WorksheetFunction.

le menu des fonctions excel apparaît quand on tape le dernier point .

Quelques exemples de la liste exhaustive que l'on découvre en cliquant droit sur la feuille du module et en choisissant explorateur d'objets + :

Exemple :

```
Function Inv_Distribution_Normale(x As Double, moyenne As Double, ecart_type As Double) As Double
        Inv_Distribution_Normale = Application.WorksheetFunction.NormInv(x, moyenne, ecart_type)
End Function

Function CND(X As Double) As Double
CND = Application.WorksheetFunction.NormDist(X, 0, 1, True)
End Function

Function ND(X As Double) As Double
ND = Application.WorksheetFunction.NormDist(X, 0, 1, False)
End Function

Function DATE_DECALLE(aValueDate As Date, nbr_mois As Integer) As Date
DATE_DECALLE = Edate(aValueDate, nbr_mois)
End Function
```

2-Black-Scholes Généralisé

Exemple de programme : Black-Scholes Généralisé.

La formule généralisées de Black et Scholes peux être utilisée pour évaluer une option européenne sur action classique, une option européenne sur action payant un dividende proportionnel, une option sur futur ainsi que des options de change.

$$Call_{BSG} = Se^{(b-r)T} N(d_1) - Xe^{-rT} N(d_2)$$

$$PUT_{BSG} = Xe^{-rT} N(-d_2) - Se^{(b-r)T} N(-d_1)$$

avec

$$d_1 = \frac{\ln\left(\frac{S}{X}\right) + \left(b + \frac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}}$$

$$d_2 = \frac{\ln\left(\frac{S}{X}\right) + \left(b - \frac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}} = d_1 - \sigma\sqrt{T}$$

$b = r$	donne la Formule de **Black et Scholes** (1973)
$b = r - q$	donne la formule de **Merton** (1973) : option européenne sur une action payant un dividende continu q.
$b = 0$	donne la formule de **Black** (1976) : option sur futur

CLAIRE PELTIER 2007

```
'// The generalized Black and Scholes formula

Function GBlackScholes(CallPutFlag As String, S As Double, X _
        As Double, T As Double, r As Double, b As Double, v As Double) As Double

    Dim d1 As Double, d2 As Double

    d1 = (Log(S / X) + (b + v ^ 2 / 2) * T) / (v * Sqr(T))
    d2 = d1 - v * Sqr(T)

    If CallPutFlag = "c" Then
        GBlackScholes = S * Exp((b - r) * T) * CND(d1) - X * Exp(-r * T) * CND(d2)
    ElseIf CallPutFlag = "p" Then
        GBlackScholes = X * Exp(-r * T) * CND(-d2) - S * Exp((b - r) * T) * CND(-d1)
    End If

End Function
```

Exercice pour le prochain cours : Calculer les greek :

Delta : dérivée par rapport à S Sous jacent
Gamma : dérivée seconde par rapport à S
Rho : dérivé par rapport à r , le taux d'actualisation
Vega : dérivées par rapport à la volatilité σ
Theta : dérivée par rapport à la maturité T

3- OBLIGATION TF

Nécessite les fonctions déjà programmée au cours précédant ainsi que la fonction **CalculBase** que je vous donne.

A mettre dans un module UTIL

```
Function interpolation(aX1 As Double, aX2 As Double, aY1 As Double, aY2 As Double, aX As Double)
   interpolation = (aY1 - aY2) / (aX1 - aX2) * (aX - aX2) + aY2
End Function

Sub loadZCRange(aValuedate As Date, aRange As Range, aTab() As Double)
   Dim lNbRow As Integer, lNbColumn As Integer
   Dim I As Integer, J As Integer
   Dim lTemp() As Date

   lNbRow = aRange.Rows.Count
   lNbColumn = aRange.Columns.Count

   ReDim aTab(1 To lNbRow, 1 To lNbColumn)
   ReDim lTemp(1 To lNbRow)

   For J = 1 To lNbRow
         lTemp(J) = aRange.Cells(J, 1)
         aTab(J, 1) = year_frac(aValuedate, lTemp(J))
         aTab(J, 2) = aRange.Cells(J, 2)
   Next j
End Sub

Function discount_factor(aValuedate As Date, aMaturity As Date, aZCrange As Range) As Double
   Dim lMAT As Double
   Dim lZcTab() As Double
   Dim lNbRow As Integer, I As Integer
   Dim ltaux As Double

    lMAT = year_frac(aValuedate, aMaturity)

   Call loadZCRange(aValuedate, aZCrange, lZcTab)

   lNbRow = aZCrange.Rows.Count

   For I = 1 To (lNbRow - 1)
      If (lZcTab(I, 1) < lMAT) Then
         If (lMAT <= lZcTab(I + 1, 1)) Then
          ltaux = interpolation(lZcTab(I, 1), lZcTab(I + 1, 1), lZcTab(I, 2), lZcTab(I + 1, 2), lMAT)
         End If
      End If
   Next I
   discount_factor = 1 / (1 + ltaux) ^ lMAT
End Function

Function year_frac(adate1 As Date, adate2 As Date) As Double
   year_frac = (adate2 - adate1) / 365.25
End Function
```

```
Function Calculbase(adate1 As Date, adate2 As Date, abasis As String) As Double

   If abasis = "Exact/Exact" Then
CF
          ElseIf abasis = "Exact/360" Then
      Calculbase = (adate2 - adate1) / 360
   ElseIf abasis = "Exact/365" Then
      Calculbase = (adate2 - adate1) / 365
   ElseIf abasis = "30/360" Then
     Calculbase = WorksheetFunction.Days360(adate1, adate2) / 360
   End If
End Function
```

Calcul d'une obligation TF à coder dans un module Obligation

Variables nécéssaires

Date de valeur
Date de début (si l'obligation commence dans le futur, si non = Date de valeur)
Date de Fin (échéance)
Taux du coupon
Nombre de mois entre deux coupons
La Courbe des taux (dates et taux annuels actuariels)
Le nominal
La base de calcul entre deux coupons (exemple "Exact/Exact")
Remboursement (1 si remboursement du nominal en Fin)

Algorithme

Calcul du nombre de coupons
Calcul des dates de tombée de coupon (boucle for)
Calcul de la somme des coupons actualisés (boucle for) (attention à la base !)
Dans le cas où il y a remboursement in Fine du capital Ajouter à cette somme la valeur actualisé de ce flux

CLAIRE PELTIER 2007

Calcul d'une obligation TV à coder dans un module Obligation

Fonction nécessaire :

Calcul du taux forward à partir de la courge des Zéro Coupon

Variables nécéssaires

Date de valeur
Date de début (si l'obligation commence dans le futur, si non = Date de valeur)
Date de Fin (échéance)
Nombre de mois entre deux coupons
La Courbe des taux (dates et taux annuels actuariels)
Le nominal
La base de calcul entre deux coupons (exemple "Exact/Exact")
Remboursement (1 si remboursement du nominal en Fin)
Le Spread par rapport au taux variable

Algorithme

Calcul du nombre de coupons
Calcul des dates de tombée de coupon (boucle for)
Calcul de la somme des coupons actualisés (boucle for) (attention à la base !)
Dans le cas où il y a remboursement in Fine du capital Ajouter à cette somme la valeur actualisé de ce flux

CLAIRE PELTIER 2007

COURS FINANCE ET INFORMATIQUE

Cours 2 Correction

12-Correction : Obligation TF et TV
13-Black-Scholes Généralise : Les Greek

1- Correction : Obligation TF et TV

```
Option Explicit

Function prix_obligation_pp(aValuedate As Date, adatedebut As Date, aEndDate As Date, tauxcoupon
As Double, NBRMOISENTRECOUPON As Integer, aZCrange As Range, nominal As Double, abasis As
String, remboursement As Integer) As Double
```

```
Dim I As Integer, N As Integer
Dim datecoupon() As Date
Dim prix As Double
Dim delta As Double
Dim dateavcoupon As Date
prix = 0
```

CALCUL DU NOMBRE DE COUPONS

```
N = Int(Calculbase(adatedebut, aEndDate, "30/360") * 360 / ((NBRMOISENTRECOUPON) * 30))
```

CALCUL DES DATES DE TOMBEE DE COUPON (BOUCLE FOR)

```
ReDim datecoupon(0 To N)
For I = 0 To N
  datecoupon(I) = edate(aEndDate, -(N - I) * NBRMOISENTRECOUPON)
Next I
```

CALCUL DE LA SOMME DES COUPONS ACTUALISES (BOUCLE FOR) (ATTENTION A LA BASE !)

```
For I = 1 To N
   delta = Calculbase(datecoupon(I - 1), datecoupon(I), abasis)
   prix = prix + delta * nominal * tauxcoupon * discount_factor(aValuedate, datecoupon(I), aZCrange)
Next I
```

PREMIER COUPON

```
If (adatedebut < aValuedate) Then
  dateavcoupon = edate(datecoupon(0), -NBRMOISENTRECOUPON)
  delta = Calculbase(dateavcoupon, datecoupon(0), abasis)
prix = prix + delta * tauxcoupon * nominal * discount_factor(aValuedate, datecoupon(0), aZCrange)
End If
```

DANS LE CAS OU IL Y A REMBOURSEMENT IN FINE DU CAPITAL AJOUTER A CETTE SOMME LA VALEUR ACTUALISE

```
If (remboursement = 1) Then
prix = prix + nominal * discount_factor(aValuedate, datecoupon(N), aZCrange)
End If

prix_obligation_pp = prix

End Function
```

```
Function prix_obligation_tv(aValuedate As Date, adatedebut As Date, aEndDate As Date,
NBRMOISENTRECOUPON As Integer, aZCrange As Range, nominal As Double, abasis As String,
remboursement As Integer, spread As Double) As Double

Dim I As Integer, N As Integer
Dim datecoupon() As Date
Dim prix As Double
Dim delta As Double
Dim dateavcoupon As Date
prix = 0
```

CLAIRE PELTIER 2007

CALCUL DU NOMBRE DE COUPONS

```
N = Int(Calculbase (adatedebut, aEndDate, "30/360") * 360 / (NBRMOISENTRECOUPON * 30))
```

CALCUL DES DATES DE TOMBEE DE COUPON (BOUCLE FOR)

```
ReDim datecoupon(0 To N)
For I = 0 To N
  datecoupon(I) = edate(aEndDate, -(N - I) * NBRMOISENTRECOUPON)
Next I
```

CALCUL DE LA SOMME DES COUPONS ACTUALISES (BOUCLE FOR) (ATTENTION A LA BASE !)

```
For I = 1 To N
  delta = Calculbase (datecoupon(I - 1), datecoupon(I), abasis)
  prix = prix + delta * nominal * (taux_forward(aValuedate, datecoupon(I - 1), datecoupon(I), aZCrange,
abasis) + spread) * discount_factor(aValuedate, datecoupon(I), aZCrange)
Next I
```

CALCUL DU FIXING

```
if (adatedebut < aValuedate) Then
dateavcoupon = Edate(datecoupon(0), -NBRMOISENTRECOUPON)
delta = Calculbase(dateavcoupon, datecoupon(0), abasis)
prix = prix + delta * (tauxfixing + spread) * nominal * discount_factor(aValuedate, datecoupon(0),
aZCrange)
End If
```

DANS LE CAS OU IL Y A REMBOURSEMENT IN FINE DU CAPITAL AJOUTER A CETTE SOMME LA VALEUR ACTUALISE DE CE FLUX

```
If (remboursement = 1) Then
prix = prix + nominal * discount_factor(aValuedate, datecoupon(N), aZCrange)
End If

prix_obligation_tv = prix

End Function
```

```
Public Function taux_forward(aValuedate As Date, aStartDate As Date, aEndDate As Date, aZCrange
As Range, abasis As String)
Dim delta As Double
delta = base(aStartDate, aEndDate, abasis)
taux_forward = (discount_factor(aValuedate, aStartDate, aZCrange) / discount_factor(aValuedate,
aEndDate, aZCrange) - 1) / delta

End Function

Function oblig_coupon_couru(aValuedate As Date, adatedebut As Date, aEndDate As Date,
tauxcoupon As Double, NBRMOISENTRECOUPON As Integer, aZCrange As Range, nominal As Double,
abasis As String, spread As Double) As Double
oblig_coupon_couru = 0
Dim delta As Double
Dim dateavcoupon As Date

Dim datecoupon As Date
Dim N As Integer

N = Int(Calculbase(adatedebut, aEndDate, "30/360") * 360 / (NBRMOISENTRECOUPON * 30))
'N = coupnum(adatedebut, aEndDate, 12 / NBRMOISENTRECOUPON)
datecoupon = Edate(aEndDate, -N * NBRMOISENTRECOUPON)
oblig_coupon_couru = 0
```

```
If (adatedebut < aValuedate) Then
   dateavcoupon = Edate(datecoupon, -NBRMOISENTRECOUPON)
   delta = Calculbase(dateavcoupon, adatedebut, abasis)
   oblig_coupon_couru = (delta) * (tauxcoupon + spread) * nominal
End If

End Function
```

2-Black-Scholes Généralisé : Les Greek

'// Delta for the generalized Black and Scholes formula

Pour un call $\Delta_{call} = \frac{dc}{dS} = e^{(b-r)T} N(d_1)$

Pour un put $\Delta_{call} = \frac{dc}{dS} = -e^{(b-r)T} N(-d_1)$

```
Function GDelta(CallPutFlag As String, S As Double, X As Double, T As Double, r As Double,
          b As Double, v As Double) As Double

   Dim d1 As Double

   d1 = (Log(S / X) + (b + v ^ 2 / 2) * T) / (v * Sqr(T))

   If CallPutFlag = "c" Then
      GDelta = Exp((b - r) * T) * CND(d1)
   ElseIf CallPutFlag = "p" Then
      GDelta = Exp((b - r) * T) * (CND(d1) - 1)
   End If

End Function
```

'// Gamma for the generalized Black and Scholes formula

Pour un call et un put $\Gamma = \frac{d^2c}{d^2S} = \frac{d^2p}{d^2S} = \frac{n(d_1)e^{(b-r)T}}{S\sigma\sqrt{T}}$

```
Public Function GGamma(S As Double, X As Double, T As Double, r As Double, b As Double, v As
Double) As Double

   Dim d1 As Double

   d1 = (Log(S / X) + (b + v ^ 2 / 2) * T) / (v * Sqr(T))
   GGamma = Exp((b - r) * T) * ND(d1) / (S * v * Sqr(T))

End Function
```

'// Theta for the generalized Black and Scholes formula

Pour un call $\theta_{call} = -\frac{dc}{dT} = -\frac{Se^{(b-r)T} n(d_1)\sigma}{2\sqrt{T}} - (b-r)Se^{(b-r)T} N(d_1) - rXe^{-rT} N(d_2)$

Pour un put $\theta_{put} = -\frac{dp}{dT} = -\frac{Se^{(b-r)T} n(d_1)\sigma}{2\sqrt{T}} + (b-r)Se^{(b-r)T} N(-d_1) + rXe^{-rT} N(-d_2)$

```
Public Function GTheta(CallPutFlag As String, S As Double, X As Double, T As Double, r As Double, b
As Double, v As Double) As Double

   Dim d1 As Double, d2 As Double

   d1 = (Log(S / X) + (b + v ^ 2 / 2) * T) / (v * Sqr(T))
   d2 = d1 - v * Sqr(T)

   If CallPutFlag = "c" Then
      GTheta = -S * Exp((b - r) * T) * ND(d1) * v / (2 * Sqr(T)) - (b - r) * S * Exp((b - r) * T) * CND(d1) - r * X
* Exp(-r * T) * CND(d2)
   ElseIf CallPutFlag = "p" Then
      GTheta = -S * Exp((b - r) * T) * ND(d1) * v / (2 * Sqr(T)) + (b - r) * S * Exp((b - r) * T) * CND(-d1) + r *
X * Exp(-r * T) * CND(-d2)
   End If

End Function
```

'// Vega for the generalized Black and Scholes formula

Pour un put comme pour un call $Vega = \frac{dc}{d\sigma} = \frac{dp}{d\sigma} = Se^{(b-r)T} n(d_1)\sqrt{T}$

```
Public Function GVega(S As Double, X As Double, T As Double, r As Double, b As Double, v As Double)
As Double

   Dim d1 As Double

   d1 = (Log(S / X) + (b + v ^ 2 / 2) * T) / (v * Sqr(T))
   GVega = S * Exp((b - r) * T) * ND(d1) * Sqr(T)

End Function
```

CLAIRE PELTIER 2007

'// Rho for the generalized Black and Scholes formula

Pour un call **si** $b \neq 0$ $\rho_{call} = \frac{dc}{dr} = TXe^{-rT} N(d_2)$

si $b = 0$ $\rho_{call} = \frac{dc}{dr} = -T \times Call$

Pour un call **si** $b \neq 0$ $\rho_{put} = \frac{dc}{dr} = -TXe^{-rT} N(-d_2)$

si $b = 0$ $\rho_{put} = \frac{dc}{dr} = -T \times put$

```
Public Function GRho(CallPutFlag As String, S As Double, X As Double, T As Double, r As Double, b As Double, v As Double) As Double

  Dim d1 As Double, d2 As Double

  d1 = (Log(S / X) + (b + v ^ 2 / 2) * T) / (v * Sqr(T))
  d2 = d1 - v * Sqr(T)
  If CallPutFlag = "c" Then
     If b <> 0 Then
        GRho = T * X * Exp(-r * T) * CND(d2)
     Else
        GRho = -T * GBlackScholes(CallPutFlag, S, X, T, r, b, v)
     End If
  ElseIf CallPutFlag = "p" Then
     If b <> 0 Then
        GRho = -T * X * Exp(-r * T) * CND(-d2)
     Else
        GRho = -T * GBlackScholes(CallPutFlag, S, X, T, r, b, v)
     End If
  End If

End Function
```

CLAIRE PELTIER 2007

COURS FINANCE ET INFORMATIQUE

Cours 3 Correction

14-Simulation Monte Carlo : Option Call Put sur Action (Boyle 1977)

3 Simulation Monte Carlo : Option Call Put sur Action

Rappels Mathématiques : Convergence et Limite de la méthode

Loi forte des Grands Nombres

Soit $(X_n)_{n\in N}$ une suite de variables aléatoires indépendantes identiquement distribuées de même espérance m

$$\text{Si } E(|X_n|) \text{ converge alors } \frac{\sum_{i=1}^{n} X_i}{n} \xrightarrow[PS]{} m = E(X_i)$$

Théorème centrale limite

Soit $(X_n)_{n\in N}$ une suite de variables aléatoires indépendantes identiquement distribuées de même espérance m et de même variance σ finie

$$\text{alors } \frac{\sum_{i=1}^{n} X_i - nm}{\sigma\sqrt{n}} \xrightarrow[loi]{} Normale(0,1)$$

c'est à dire si on nomme $ecart_n = m - \frac{\sum_{i=1}^{n} X_i}{n}$ alors

l'intervalle de confiance à 95% de cet écart vaut $\left[m - 1.96\frac{\sigma}{\sqrt{n}}, m + 1.96\frac{\sigma}{\sqrt{n}} \right]$

Description de la méthode :

Pour utiliser les méthode de Monte Carlo on doit d'abord mettre sous forme d'espérance la quantité que l'on cherche à calculer. A l'issue de cette étape, il reste à calculer une quantité de la forme $E(X)$ où X est une variable aléatoire. Pour pouvoir calculer $E(X)$ il convient de savoir simuler une variable aléatoire selon la loi de X.

Mathématiquement, cela signifie que l'on dispose de la réalisation d'une suite de variable aléatoire indépendante (X_i) suivant la loi de X.

Informatiquement, on ramène la simulation de cette loi à celle d'une variable aléatoire indépendante suivant une loi uniforme sue l'intervalle $[0,1]$

Il reste à approcher $E(X)$ par $E(X)=\dfrac{\sum_{i=1}^{n} X_i}{n}$

Ensuite on approche la variance par $\overline{\sigma_X} = \dfrac{1}{n-1}\sum_{i=1}^{n}\left(X_i - \dfrac{\sum_{i=1}^{n} X_i}{n}\right)^2$

L'intervalle de confiance vaut pour le rang n : $\left[\dfrac{\sum_{i=1}^{n} X_i}{n} - 1.96\dfrac{\overline{\sigma_X}}{\sqrt{n}}, \dfrac{\sum_{i=1}^{n} X_i}{n} + 1.96\dfrac{\overline{\sigma_X}}{\sqrt{n}}\right]$

Pour simuler une variable aléatoire de fonction de répartition F , on inverse cette fonction. En effet, la fonction de répartition donne une valeur entre 0 et 1 et est croissante continue donc inversible. Donc si on cherche des nombres aléatoire x suivant une loi F, il faut trouver une valeur u entre 0 et 1, au hasard, tel que : $u = F(x) \in [0,1]$

Ensuite, on effectue l'inversion $x = F^{-1}(x)$

Pour la loi normale, par exemple, il faut utiliser la fonction Application.NormInv(Rnd(), 0, 1))

Rappels :

$$\int_{t_1}^{t_2} \sigma dW_t \xrightarrow{loi} N\left(0, \sigma^2(t_2 - t_1)\right)$$

Pour une action de Cost and Carry b, la diffusion vaut : $S_T = S_0 EXP\left(bT - \frac{1}{2}\sigma^2 T + \int_0^T \sigma dW_t\right)$

Exercice : Simulation MC :

Option Call Put sur Action Européenne

```
'// Monte Carlo plain vanilla European option
Public Function MonteCarloStandardOption(CallPutFlag As String, S As Double, X As
Double, T As Double, r As Double, b As Double, v As Double, nSimulations As Integer)
As Double

    Dim St As Double
    Dim Sum As Double, Drift As Double, vSqrdt As Double, UNIT As Double
    Dim i As Integer, z As Integer


    Drift = (b - v ^ 2 / 2) * T
    vSqrdt = v * Sqr(T)

    If CallPutFlag = "c" Then
        z = 1
    ElseIf CallPutFlag = "p" Then
        z = -1
    End If

    For i = 1 To nSimulations
        UNIT = Application.NormInv(Rnd(), 0, 1)
        St = S * Exp(Drift + vSqrdt * UNIT)
        Sum = Sum + Application.WorksheetFunction.Max(z * (St - X), 0)
    Next i

    MonteCarloStandardOption = Exp(-r * T) * (Sum / nSimulations)

End Function
```

```
'// Monte Carlo Asian option
Public Function MonteCarloAsianOption(CallPutFlag As String, S1 As Double, X As
Double, T As Double, T1 As Double, r As Double, b1 As Double, v1 As Double, nSteps
As Integer, nSimulations As Integer) As Double

    Dim dt As Double, St1 As Double
    Dim i As Integer, j As Integer, z As Integer
    Dim Sum As Double, Drift1 As Double
    Dim v1Sqrdt As Double
    Dim Epsilon1 As Double, Average1 As Double

    If CallPutFlag = "c" Then
        z = 1
    ElseIf CallPutFlag = "p" Then
        z = -1
    End If

    dt = (T - T1) / nSteps
    Drift1 = (b1 - v1 ^ 2 / 2) * dt
    v1Sqrdt = v1 * Sqr(dt)

    For i = 1 To nSimulations
        Average1 = 0
        St1 = S1 * Exp((b1 - v1 ^ 2 / 2) * T1+ v1 * Sqr(T1) * Application.NormInv(Rnd(), 0, 1))
        For j = 1 To nSteps
            Epsilon1 = Application.NormInv(Rnd(), 0, 1)
            St1 = St1 * Exp(Drift1 + v1Sqrdt * Epsilon1)
            Average1 = Average1 + St1
Next
        Average1 = Average1 / nSteps

        Sum = Sum + Application.WorksheetFunction.Max(z * (Average1 - X), 0)
    Next

    MonteCarloAsianOption = Exp(-r * T) * (Sum / nSimulations)

End Function
```

CLAIRE PELTIER 2007

COURS FINANCE ET INFORMATIQUE

Cours 4

15-Modèle Binomial : Cox Ross Rubinstein (1979)
16-Call Put Européen Américain

1-Modèle Binomial : Cox Ross Rubinstein (1979)

Soit b le cost and carry exponentiel supposé constant

$b = r$	donne la Formule de **Black et Scholes** (1973)
$b = r - q$	donne la formule de **Merton** (1973) : option européenne sur une action payant un dividende continu q.
S	la valeur de l'action initiale

Pour une action de Cost and Carry b, la diffusion vaut : $S_T = S_0 EXP\left(bT - \frac{1}{2}\sigma^2 T + \int_0^T \sigma dW_t \right)$

Pour la première période de l'arbre :

Sens de construction de l'arbre :

S → u S avec la probabilité p

S → d S avec la probabilité 1-p

Dans l'univers risque neutre,

$$E\left(\frac{S_{dt} - S_0}{S_0}\right) = e^{bdt} - 1 = \frac{uS - S}{S} p + \frac{dS - S}{S}(1-p) = (u-1)p + (d-1)(1-p)$$

On en déduit $\boxed{p = \frac{e^{bdt} - d}{u - d}}$

Dans l'univers risque neutre :

$$E(\ln\left(\frac{S_{dt}}{S_0}\right)^2) = E\left(bdt - \frac{1}{2}\sigma^2 dt + \sigma W_{dt}\right)^2 = \sigma^2 dt$$

donc

$$\sigma^2 dt = \left(Ln\frac{S_u}{S}\right)^2 p + \left(Ln\frac{S_d}{S}\right)^2 (1-p) = (\ln u)^2 p - (\ln d)^2 (1-p)$$

$$\sigma^2 dt = \left(Ln\frac{S_u}{S}\right)^2 p + \left(Ln\frac{S_d}{S}\right)^2 (1-p) = (\ln u)^2 p - (\ln d)^2 (1-p)$$

en générale on choisit $u = \frac{1}{d}$

$$\sigma^2 dt = (\ln u)^2 p - (\ln \frac{1}{u})^2 (1-p) = (\ln u)^2 \Rightarrow \ln u = \sigma\sqrt{dt}$$

$$u = \frac{1}{d} = e^{\sigma\sqrt{dt}}$$

Pour le call :

Le call est une fonction du sous jacent, donc sa probabilité de hausse et de baisse est identique (distribution d'un fonction d'une variable aléatoire)

Sens de construction de l'arbre :

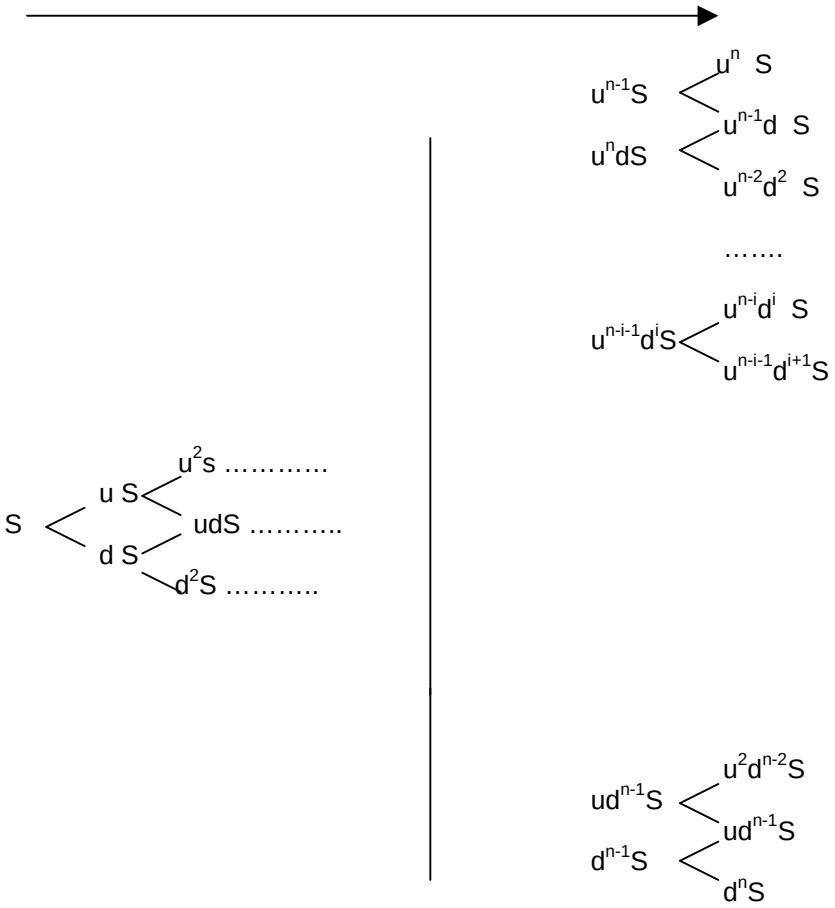
C ← **C_U = Max(u S - K , 0) avec la probabilité p**

C_d = Max(d S - K , 0) avec la probabilité 1-p

Donc $C = e^{-rdt}\left(pC_u + (1-p)C_d\right)$ avec $p = \frac{e^{bdt} - d}{u - d}$

Arbre à n pas (dt = T/n) Sous Jacent

SENS DE CONSTRUCTION DE L'ARBRE



Debut

FIN (n+1 branche)

CLAIRE PELTIER 2007

Arbre à n pas (dt = T/n) Call Européen de strike k

SENS DE CONSTRUCTION DE L'ARBRE

$C_u^n = Max(u_n S-K,0)$

$C_u^{n-1}\, d = Max(u^{n-1} dS-K,0)$

$C_u^{n-2}\, d^2 = Max(u^{n-2} d^2\, S-K,0)$

$$C_U = e^{-rdt}\left(pC_{u^2} + (1-p)C_{ud}\right)$$

$$C = e^{-rdt}\left(pC_u + (1-p)C_d\right)$$

$$C_d = e^{-rdt}\left(pC_{ud} + (1-p)C_{d^2}\right)$$

$Cu^2 d^{n-2} = Max(u^2 d^{n-2}\, S-K,0)$

$Cud^{n-1} = Max(ud^{n-1}\, S-K,0)$

$Cd^n = Max(d^n\, S-K,0)$

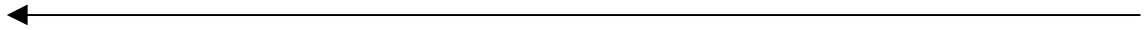
Fin

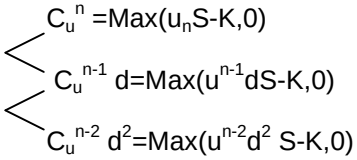
Début (n+1 branche)

CLAIRE PELTIER 2007

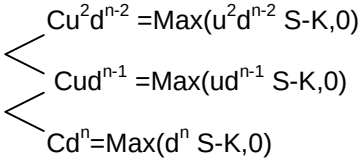
Arbre à n pas (dt = T/n) Call Américain de strike k

SENS DE CONSTRUCTION DE L'ARBRE





$$C_{u^{n-j}d^{j-i}} = Max\left[e^{-rdt}\left(pC_{u^{n-j+1}d^{j-i}} + (1-p)C_{u^{n-j}d^{j-i+1}} \right), Max\left(u^{n-j}d^{j-1}S - K\right) \right]$$



Fin Rang j Début (n+1 branche)

```
'// Cox-Ross-Rubinstein binomial tree

Option Base 0          'The "Option Base" statement allows to specify 0 or 1 as the
                       'default first index of arrays.

Function CRRBinomial(AmeEurFlag As String, CallPutFlag As String, S As Double, X As Double, T As
Double, r As Double, b As Double, v As Double, n As Integer) As Double


   Dim OptionValue() As Double
   Dim u As Double, d As Double, p As Double
   Dim dt As Double, Df As Double
   Dim i As Integer, j As Integer, z As Integer

   ReDim OptionValue(n + 1)

   If CallPutFlag = "c" Then
      z = 1
      ElseIf CallPutFlag = "p" Then
      z = -1
   End If

   dt = T / n
   u = Exp(v * Sqr(dt))
   d = 1 / u
   p = (Exp(b*dt) - d) / (u - d)
   Df = Exp(-r * dt)

   For i = 0 To n
       OptionValue(i) = Max(0, z * (S * u ^ i * d ^ (n - i) - X))
   Next i

   For j = n - 1 To 0 Step -1:
      For i = 0 To j
         If AmeEurFlag = "e" Then
               OptionValue(i) = (p * OptionValue(i + 1) + (1 - p) * OptionValue(i)) * Df
         ElseIf AmeEurFlag = "a" Then
               OptionValue(i) = Max((z * (S * u ^ i * d ^ (j-i) - X)),(p * OptionValue(i + 1) + (1 - p) *
               OptionValue(i)) * Df)
         End If
      Next
   Next
   CRRBinomial = OptionValue(0)
End Function



Public Function Max(X, y)
         Max = Application.Max(X, y)
End Function
```

CLAIRE PELTIER 2007

COURS FINANCE ET INFORMATIQUE

Cours 5

17-SWAP, SWAPTION
18-CAP FLOOR

1- Swap, Swaption

Swap

S_0 **: taux forward du swap sous jacent .** S_0 **est solution de l'équation :**

$$S_0 = \frac{P(t_0,T_0) - P(t_0,T_N)}{\sum_{i=1}^{N} aP(t_0,T_i)} = S(0,T)$$

On considère un swap démarrant en T_0 **et délivrant des coupons fixes aux dates T La valeur en** t_0 **des flux de la branche fixe du swap vaut :**

$$Fixe(t_0) = aTauxfixe\sum_{i=1}^{N} P(t_0,T_i)$$

La valeur de la branche variable, quelque soit sa fréquence, vaut, en utlisant la forme double notionnel :

$$Variable(t) = P(t_0,T_0) - P(t_0,T_N)$$

Le taux forward du swap vaut donc :

$$S(t_0,T) = \frac{P(t_0,T_0) - P(t_0,T_N)}{a\sum_{i=1}^{N} P(t_0,T_i)} = \frac{X_{t_0}}{Y_{t_0}}$$

$$\frac{dS_t}{S_t} = \sigma^S(t)dW_t = [\sigma_X - \sigma_Y]dW_t$$

On note $P_k = P(t,T_k)$ **et** Γ_k **la volatilité du zéro coupon** P_k **.**

Alors $\frac{dX_t}{X_T} = \frac{dP_0 - dP_N}{P_0 - P_N}$ **donc** $\sigma_X = \frac{\Gamma_0 P_0 - \Gamma_N P_N}{P_0 - P_N}$ **.**

De même :

$$\sigma_Y = \frac{\sum_{i=1}^{N} \Gamma_i P_I}{\sum_{i=1}^{N} P_I}_N$$

donc le taux swap suit un processus de diffusion lognormal.

- Caplet

Dans le cas d'un cap simple sur swap,

$h(S(T_0,T)) = [S(T_0,T) - K]$ **et** $1_{A_{T_0}} = \{S(T_0,T) > K\}$

Le taux swap swap suivant une loi lognormale sous la probabilité Q, la valeur

CLAIRE PELTIER 2007

$$C(t) = E^{Q}\left[\exp\left(-\int_{t}^{T_0} r(s)ds\right) h[S(T_0,T)] \times 1_{A_{T_0}} / \Im_t\right]$$ **vaut :**

$$Caplet(t) = P(t,T_0)[S(t,T)N(d_1) - KN(d_2)]$$

avec

$$d_1 = \frac{\ln\left(\frac{S(t,T)}{K}\right) + \frac{\sigma^2}{2}T_0}{\sigma\sqrt{T_0}}, \quad d_2 = \frac{\ln\left(\frac{S(t,T)}{K}\right) - \frac{\sigma^2}{2}T_0}{\sigma\sqrt{T_0}}$$

- Cap

Somme de Caplets de différentes échéances

Swaption : Modèle de Marcher Black

Les swaptions européenne

$$\text{Swaption} = p(t_0,t)E\left(\sum\left[(s_t - k)\exp\left(-\int_{t}^{t_i} r(s)ds\right)\right]\right)^{+}$$

$$\text{Swaption} = p(t_0,t)E\left((s_t - k)^{+}\sum\left(P(t,t_i)\right)\right)$$

$$\text{Swaption} = \sum\left(P(t_0,t_i)\right)E\left((s_t - k)^{+}\right)$$

$$\text{Swaption} = \sum\left(P(t_0,t_i)\right)E\left((s_t - k)^{+}\right)$$

$$\boxed{\text{Swaption} = \sum\left(P(t_0,t_i)\right)blackscholes(S_t,K)}$$

PROGRAMME

```
Function caplet(cf As String, datevalo As Date, datedebut As Date, datefin As Date, strike As Double, volatilite As Double, aZCrange As Range, abasis As String) As Double

Dim tauxswapforward As Double
Dim duree As Double
Dim d1 As Double, d2 As Double

tauxswapforward = taux_forward(datevalo, datedebut, datefin, aZCrange, abasis)

duree = Calculbase(datevalo, datedebut, abasis)
d1 = (Log(tauxswapforward / strike) + volatilite ^ 2 * duree / 2) / (volatilite * Sqr(duree))
d2 = d1 - volatilite * Sqr(duree)
duree = Calculbase(datedebut, datefin, abasis)
If (cf = "c") Then
        caplet = discount_factor(datevalo, datefin, aZCrange) * (tauxswapforward * CND(d1) - strike * CND(d2))
End If
If (cf = "f") Then
        caplet = discount_factor(datevalo, datefin, aZCrange) * (-tauxswapforward * CND(-
        D1) + STRIKE * CND(-D2))
```

```
End If

End Function


Function capfloor(cf As String, datevalo As Date, datedebut As Date, datefin As Date, strike As Double, volatilite As Double, aZCrange As Range, nbrmoiscoupon As Integer, abasis As String) As Double


Dim N As Integer, I As Integer
Dim datecoupon() As Date

N = Int(Calculbase(datedebut, datefin, "30/360") * 360 / (nbrmoiscoupon * 30))


ReDim datecoupon(0 To N) As Date
For I = 0 To N
        datecoupon(I) = edate(datefin, -(N - I) * nbrmoiscoupon)
Next I

For I = 1 To N - 1
        capfloor = capfloor + caplet(cf, datevalo, datecoupon(I), datecoupon(I + 1), strike,
        volatilite, aZCrange, abasis)
Next I

End Function


Function swaptionblack(rp As String, strike As Double, datevalo As Date, datedebut As Date,
datefin As Date, aZCrange As Range, volatilite As Double, nbrmoiscouponvar As Integer,
nbrmoiscouponfixe As Integer, abasis As String)

Dim somme As Double
Dim datecoupon() As Date
Dim duree As Double
Dim N As Integer, I As Integer
Dim tauxswapforward As Double
Dim bondvar As Double
Dim BONDFIXE As Double
N = Int(Calculbase(datedebut, datefin, "30/360") * 360 / (nbrmoiscouponfixe * 30))
ReDim datecoupon(0 To N)

BONDFIXE = prix_obligation_pp(datevalo, datedebut, datefin, 1, nbrmoiscouponfixe, aZCrange,
1, abasis, 0)

bondvar = prix_obligation_tv(datevalo, datedebut, datefin, nbrmoiscouponvar, aZCrange, 1,
abasis, 0, 0, 0)

tauxswapforward = bondvar / BONDFIXE
```

```
For I = 0 To N
  datecoupon(I) = edate(datefin, -(N - I) * nbrmoiscouponfixe)
Next I
For I = 0 To N - 1
duree = Calculbase(datecoupon(I), datecoupon(I + 1), abasis)
somme = somme + duree * discount_factor(datevalo, datecoupon(I + 1), aZCrange)
Next I

duree = Calculbase(datevalo, datedebut, abasis)

Dim d1 As Double, d2 As Double
d1 = (Log(tauxswapforward / strike) + volatilite ^ 2 / 2 * duree) / (volatilite * Sqr(duree))
d2 = d1 - volatilite * Sqr(duree)

If (rp = "p") Then
   swaptionblack = somme * (tauxswapforwarD * CND(D1) - STRIKE * CND(D2))
End If
If (rp = "r") Then
   swaptionblack = somme * (-tauxswapforward * CND(-D1) + STRIKE * CND(-D2))
End If

End Function
```