**Projet VBA :**

**Pricer d’Options Vanille**

**Projet VBA : Pricer d’Options Vanille**

Table des matières

[I) Introduction 3](#_Toc293235419)

[II) Modèle de Black & Scholes 4](#_Toc293235420)

[III) Explication des calculs et du code 8](#_Toc293235421)

[1) Basic 8](#_Toc293235422)

[2) OptionPage 17](#_Toc293235423)

[3) OptionsStrategies 21](#_Toc293235424)

[4) StrategyGraphs 21](#_Toc293235425)

[IV) Guide d’utilisation 22](#_Toc293235426)

# 

# Introduction

Un opérateur de marché au sein d’une banque a un besoin indispensable d’utiliser des outils de pricing fiables et rapides pour son activité. Cependant, malgré tous les différents outils proposés en matière de pricing par les éditeurs de logiciels, chaque opérateur a des besoins spécifiques auxquels il doit répondre par ses propres développements (VBA, C++).

En effet, la technicité des produits financiers traités sur les marchés et la complexité de leurs gestions, génèrent des besoins trop pointus pour que des spécialistes du développement puissent y répondre rapidement. Il devient donc indispensable d’être capable de produire des outils de pricing sur mesure.

Nous avons donc souhaité concevoir un outil de pricing afin de s’exercer à répondre à nos futurs besoins en salle des marchés. Nous avons voulu pricer des options vanilles européennes et étudier les grecs associés par la méthode Black and Scholes.

Nous allons dans un premier temps présenter le modèle de pricing d’options de Black and Scholes et ses formules mathématiques pour ensuite expliquer le code vba et les calculs que nous avons utilisés, et enfin voir le guide d’utilisation afin de présenter le fonctionnement de cet outil.

# Modèle de Black & Scholes

Le modèle Black-Scholes repose sur un certain nombre de conditions :

* le prix de l'actif sous-jacent *St* suit un mouvement brownien géométrique avec une volatilité σ constante et une dérive μ constante

 dS_t = \mu S_t\,dt + \sigma S_t\,dW_t \, , où W_t \, est un processus de Wiener.

* il n'y a pas d'opportunités d'arbitrage,
* le temps est une fonction continue,
* il est possible d'effectuer des ventes à découvert,
* il n'y a pas de coûts de transactions,
* il existe un taux d'intérêt sans risque, connu à l'avance et constant,
* tous les sous-jacents sont parfaitement divisibles (on peut par exemple acheter 1/100e d'action),
* dans le cas d'une action, celle-ci ne paie pas de dividendes entre le moment de l'évaluation de l'option et l'échéance de celle-ci.

Chacune de ces hypothèses est nécessaire pour être dans le cas Black-Scholes.

La formule de Black-Scholes permet de calculer la valeur théorique d'une option européenne à partir des données suivantes :

B&S nous donne les formules suivantes :

C:\Users\Haydens\Desktop\devoir eslsca\VBA\img\prix call.jpg

C:\Users\Haydens\Desktop\devoir eslsca\VBA\img\prix put.jpg

Avec:





Variables :

-  valeur d’un call

- 0 valeur actuelle du sous-jacent en 

-  taux de dividende sur la période de vie de l’option (dividend yield)

-  le taux d’intérêt sans risque

-  la date de maturité de l’option

-  le prix d’exercice de l’option

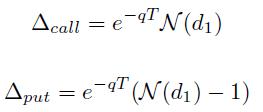
- la volatilité retenue sur la période de vie de l’option

Les grecs pour le Call et le Put :

Ces indicateurs calculent l'impact sur le prix de l'option d'une variation des paramètres qui le forment.

**Calcul du delta :**

Le delta d'une option mesure la sensibilité de son prix à une variation donnée du cours du sous-jacent. (ex: de combien le prix de l’option va varier pour une hausse de 1% de l’action)

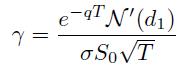


**Calcul du gamma :**

Le gamma mesure la sensibilité du delta aux variations de l’actif sous-jacent. Il indique si le prix de l'option a tendance à évoluer plus ou moins vite que le prix du sous-jacent.

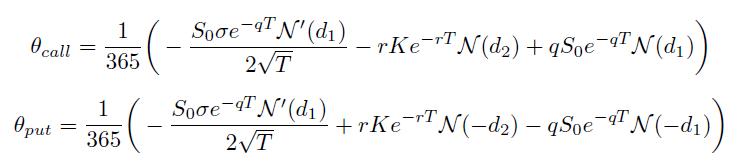
Il indique aussi le sens d'évolution du delta en fonction du prix du sous-jacent. Un gamma positif indique que prix du sous-jacent et delta évoluent dans le même sens, alors qu'un gamma négatif montre le contraire.

Le Gamma est la dérivé du delta. Il représente donc la dérivé seconde du prix de l’option par rapport au sous-jacent. Par analogie, on peut comparer le delta à la vitesse et le gamma à l'accélération.



**Calcul du thêta :**

Le thêta est le coût (ou le gain) du temps qui passe sur un portefeuille d'options. Il évalue combien le passage du temps influe sur la valeur d'une option. Une position longue d'options (gamma positive) sera thêta négative. Le trader devra veiller tous les jours à payer son thêta journalier en profitant de sa position longue en gamma.



**Calcul du Véga :**

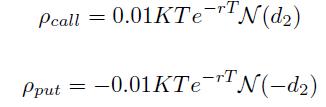
Le véga est une mesure de la sensibilité à la volatilité implicite.

C:\Users\Haydens\Desktop\devoir eslsca\VBA\img\vega.jpg

Contrairement au gamma et au thêta, le véga est une fonction croissante de la maturité. Ainsi une augmentation parallèle de la volatilité aura plus d'impact sur les options dont la date d'échéance est éloignée que sur celles dont elle est proche. Une position généralement appréciée des traders et des market makers est alors d'avoir une position globalement gamma positive (sensible aux grands mouvements de marché) et véga négative, qui consiste à acheter des options courtes et à vendre des options longues.

**Calcul du Rhô :**

Le Rhô est le taux de variation de la valeur de la prime en fonction du taux d'intérêt.



# Explication des calculs et du code VBA

Dans cette partie nous allons expliquer feuille par feuille les formules et le code utilisé.

## Basic

Dans cette feuille nous avons les données sur le sous jacent de l’option et les variables qui vont impacter le prix de l’option.

Nous retrouvons les valeurs à entrer :

* Sous-jacent (Underlying Price)
* Prix d’exercice (Exercise Price, Strike)
* Date d’aujourd’hui (Today's Date)
* Maturité (Expiry Date)
* Volatilité historique (Historical Volatility)
* Taux sans risque (Risk Free Rate)
* Taux du rendement de l’action (Dividend Yield)
* Maturité en année (DTE Date to expiration)

Par la suite la feuille calcule à partir de ces données, le prix théoriques ainsi que les grecs de l’option.

Une procédure *Function* est similaire à une procédure *Sub*, à la différence près qu'elle peut retourner une valeur.

La déclaration d'une procédure *Function* se fait entre les insctructions *Function* et *End Function* selon la syntaxe suivante :

**Function** Nom\_De\_La\_Procedure(argument1, argument2, ...)

Liste d'instructions

**End Function**

La procédure *Function* peut renvoyer une valeur, de type Variant (long, single, double…) en affectant une valeur dans une ou plusieurs de ses instructions à une variable possédant le même nom qu'elle.

**Calcul Prix du Call :**

Nous allons donc déclarer une function que nous nommons « CallOption » :

Le prix du Call en C16 va prendre les arguments : = CallOption(C3;C4;C10;C8;C7;C9)

**Code : « CallOption »**

*Function CallOption(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)*

*CallOption = Exp(-Dividend \* Time) \* UnderlyingPrice \* Application.NormSDist(dOne(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)) - ExercisePrice \* Exp(-Interest \* Time) \* Application.NormSDist(dTwo(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)*

*End Function*

Avec dans l’ordre les arguments (UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)

qui correspondent à (C3;C4;C10;C8;C7;C9), nous les avons renommé pour être plus clair dans le code.

🡪 C3 = UnderlyingPrice ; C4 = ExercisePrice ;C10 = Time; C8 = Interest; C7 = Volatility; C9 = Dividend

Formule de B&S pour le prix du call : C:\Users\Haydens\Desktop\devoir eslsca\VBA\img\prix call.jpg

**Ici e-qT =Exp(-Dividend \* Time)**

Nous avons utilisé la fonction Exponentielle de VBA « Exp » que nous pouvons trouver dans l’explorateur d’objets (F2) dans la bibliothèque VBA.

🡪 Function Exp(Number As Double) As Double

- Rappel: le Double est un sous-type de variant qui contient un nombre à virgule en double précision, dont la valeur est comprise entre -1,79769313486232E308 et -4,94065645841247E-324 pour les valeurs négatives ; de 4,94065645841247E-324 et 1,79769313486232E308 pour les valeurs positives

- Variant : le type d'une variable correspond à la manière dont l'ordinateur stocke la variable en mémoire, c'est-à-dire la succession de 0 et de 1 (système binaire) dans les cases mémoire.

C:\Users\Haydens\Desktop\devoir eslsca\VBA\img\prix call.jpg

**Avec**  = Application.NormSDist(dOne(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend))

Nous utilisons ici la fonction « NormSDist(z)» qui nous donne la probabilité qu’une variable aléatoire (z) qui suit une loi normale centré réduite (0,1) de moyenne 0 et de variance 1 soit inférieur ou égal à z.

🡪 Function NormSDist(Arg1 As Double) As Double

(Explorateur d’objets (F2) dans la bibliothèque Excel (WorksheetFunction).

Ici dans la Function NormSDist nous avons mis comme argument « dOne » (d1 de B&S).

Où « dOne » est aussi une fonction :

Code:

*Function dOne(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)*

*dOne = (Log(UnderlyingPrice / ExercisePrice) + (Interest - Dividend + 0.5 \* Volatility ^ 2) \* Time) / (Volatility \* (Sqr(Time)))*

*End Function*



Function dOne(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend),

reprend les mêmes données saisies (Prix du sous-jacent, Strike, Maturité (DTE), volat, taux div, taux rf) que la « fonction CallOption ».

ln () = Log(UnderlyingPrice / ExercisePrice)

Nous avons utilisé la fonction « Log » qui renvoie le Logarithme népérien d'un nombre. Nous pouvons la trouver dans l’explorateur d’objets (F2) dans la bibliothèque VBA (Math)

= (Interest - Dividend + 0.5 \* Volatility ^ 2) \* Time)

= (Volatility \* (Sqr(Time)))

Utilisation de la fonction racine; Function Sqr(Number As Double) As Double (biblio VBA (Math)) pour avoir la volatilité sur la période T.

C:\Users\Haydens\Desktop\devoir eslsca\VBA\img\prix call.jpg

Enfin = - ExercisePrice \* Exp(-Interest \* Time) \*Application.NormSDist(dTwo(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend) - Volatility \* Sqr(Time))

Où la fonction “dTwo” (d2 de B&S) :

Code :

*Function dTwo(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)*

*dTwo = dOne(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend) - Volatility \* Sqr(Time)*

*End Function*



**Calcul Prix du Put :**

De même pour pricer l’option Put nous utilisons la fonction « PutOption » :

Le prix du Put en D16 va prendre les arguments : = PutOption(C3;C4;C10;C8;C7;C9)

(C3 = UnderlyingPrice ; C4 = ExercisePrice ;C10 = Time; C8 = Interest; C7 = Volatility; C9 = Dividend)

**Code : « PutOption »**

*Function PutOption(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)*

*PutOption = ExercisePrice \* Exp(-Interest \* Time) \* Application.NormSDist(-dTwo(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)) - Exp(-Dividend \* Time) \* UnderlyingPrice \* Application.NormSDist(-dOne(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend))*

*End Function*

C:\Users\Haydens\Desktop\devoir eslsca\VBA\img\prix put.jpg

C:\Users\Haydens\Desktop\devoir eslsca\VBA\img\prix put.jpgAvec =

ExercisePrice \* Exp(-Interest \* Time) \* Application.NormSDist(dTwo(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend))

C:\Users\Haydens\Desktop\devoir eslsca\VBA\img\prix put.jpgEt =

Exp(-Dividend \* Time) \* UnderlyingPrice \* Application.NormSDist(-dOne(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend))

**Calcul du Delta Call / Put:**

Le delta Call en C17 va prendre les arguments : = CallDelta(C3;C4;C10;C8;C7;C9)

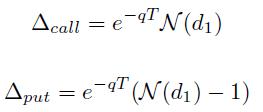
(C3 = UnderlyingPrice ; C4 = ExercisePrice ;C10 = Time; C8 = Interest; C7 = Volatility; C9 = Dividend)

**Code : « CallDelta»**

*Function CallDelta(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)*

*CallDelta = Exp(-Dividend \* Time) \* Application.NormSDist(dOne(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend))*

*End Function*

 =

Exp(-Dividend \* Time) \* Application.NormSDist(dOne(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend))

Le delta Put en D17 va prendre les arguments : = PutDelta(C3;C4;C10;C8;C7;C9)

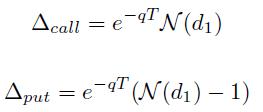
(C3 = UnderlyingPrice ; C4 = ExercisePrice ;C10 = Time; C8 = Interest; C7 = Volatility; C9 = Dividend)

**Code : « PutDelta»**

*Function PutDelta(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)*

*PutDelta = Exp(-Dividend \* Time) \* (Application.NormSDist(dOne(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)) - 1)*

*End Function*



=

Exp(-Dividend \* Time) \* (Application.NormSDist(dOne(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)) - 1)

**Calcul du Gamma Call / Put:**

Le Gamma Call en C18 et Put D18 vont avoir les arguments ; = Gamma(C3;C4;C10;C8;C7;C9)

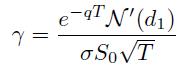
(C3 = UnderlyingPrice ; C4 = ExercisePrice ;C10 = Time; C8 = Interest; C7 = Volatility; C9 = Dividend)

**Code : « Gamma»**

*Function Gamma(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)*

*Gamma = Exp(-Dividend \* Time) \* NdOne(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend) / (UnderlyingPrice \* Volatility \* Sqr(Time))*

*End Function*



= Exp(-Dividend \* Time) \* NdOne(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend) / (UnderlyingPrice \* Volatility \* Sqr(Time))

N’(D1) est calculé grace à notre fonction NdOne :

*Function NdOne(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)*

*NdOne = Exp(-(dOne(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend) ^ 2) / 2) / (Sqr(2 \* 3.14159265358979))*

*End Function*

**Calcul du Thêta Call / Put:**

Le Thêta Call en C19 va prendre les arguments : = CallTheta(C3;C4;C10;C8;C7;C9)

(C3 = UnderlyingPrice ; C4 = ExercisePrice ;C10 = Time; C8 = Interest; C7 = Volatility; C9 = Dividend

**Code : « CallTheta»**

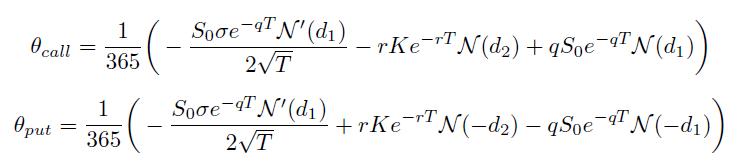
*Function CallTheta(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)*

*CT = -(UnderlyingPrice \* Volatility \* Exp(-Dividend \* Time) \* NdOne(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)) / (2 \* Sqr(Time)) - Interest \* ExercisePrice \* Exp(-Interest \* (Time)) \* NdTwo(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend) + Dividend \* UnderlyingPrice \* NdOne(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend) \* Exp(-Dividend \* Time)*

*CallTheta = CT / 365*

*End Function*

Où NdTwo = Application.NormSDist(dTwo(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend))



= -(UnderlyingPrice \* Volatility \* Exp(-Dividend \* Time) \* NdOne(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)) / (2 \* Sqr(Time)) - Interest \* ExercisePrice \* Exp(-Interest \* (Time)) \* NdTwo(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend) + Dividend \* UnderlyingPrice \* NdOne(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend) \* Exp(-Dividend \* Time)

CallTheta = CT / 365

Où la variable CT est égale à la formule du thêta call entre les parenthèses.

Le Thêta Put en D19 va prendre les arguments : = PutTheta(C3;C4;C10;C8;C7;C9)

(C3 = UnderlyingPrice ; C4 = ExercisePrice ;C10 = Time; C8 = Interest; C7 = Volatility; C9 = Dividend

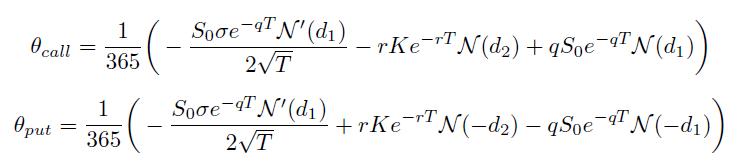
**Code : « PutTheta»**

*Function PutTheta(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)*

*PT = -(UnderlyingPrice \* Volatility \* Exp(-Dividend \* Time) \* NdOne(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)) / (2 \* Sqr(Time)) + Interest \* ExercisePrice \* Exp(-Interest \* (Time)) \* (1 - NdTwo(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)) - Dividend \* UnderlyingPrice \* (1 - NdOne(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)) \* Exp(-Dividend \* Time)*

*PutTheta = PT / 365*

*End Function*



= PT = -(UnderlyingPrice \* Volatility \* Exp(-Dividend \* Time) \* NdOne(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)) / (2 \* Sqr(Time)) + Interest \* ExercisePrice \* Exp(-Interest \* (Time)) \* (1 - NdTwo(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)) - Dividend \* UnderlyingPrice \* (1 - NdOne(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)) \* Exp(-Dividend \* Time)

PutTheta = PT / 365

Où la variable PT est égale à la formule du thêta put entre les parenthèses.

**Calcul du Véga Call / Put:**

Le Véga Call en C20 et Put D20 vont avoir les arguments ; = vega(C3;C4;C10;C8;C7;C9)

(C3 = UnderlyingPrice ; C4 = ExercisePrice ;C10 = Time; C8 = Interest; C7 = Volatility; C9 = Dividend)

**Code : « Véga»**

*Function Vega(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)*

*Vega = 0.01 \* UnderlyingPrice \* Sqr(Time) \* Exp(-Dividend \* Time) \* NdOne(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)*

*End Function*

C:\Users\Haydens\Desktop\devoir eslsca\VBA\img\vega.jpg

= 0.01 \* UnderlyingPrice \* Sqr(Time) \* Exp(-Dividend \* Time) \* NdOne(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)

**Calcul du Rhô Call / Put:**

Le Rhô du Call en C21 va prendre les arguments : = CallRho(C3;C4;C10;C8;C7;C9)

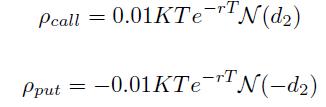
(C3 = UnderlyingPrice ; C4 = ExercisePrice ;C10 = Time; C8 = Interest; C7 = Volatility; C9 = Dividend

**Code : «CallRho»**

*Function CallRho(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)*

*CallRho = 0.01 \* ExercisePrice \* Time \* Exp(-Interest \* Time) \* Application.NormSDist(dTwo(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend))*

*End Function*

 = 0.01 \* ExercisePrice \* Time \* Exp(-Interest \* Time) \* NdTwo(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)

Le Rhô du Put en D21 va prendre les arguments : = PutRho(C3;C4;C10;C8;C7;C9)

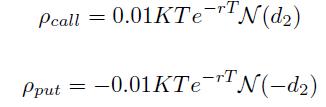
(C3 = UnderlyingPrice ; C4 = ExercisePrice ;C10 = Time; C8 = Interest; C7 = Volatility; C9 = Dividend

**Code : «PutRho»**

*Function PutRho(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend)*

*PutRho = -0.01 \* ExercisePrice \* Time \* Exp(-Interest \* Time) \* (1 - NdTwo(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend))*

*End Function*

= -0.01 \* ExercisePrice \* Time \* Exp(-Interest \* Time) \* (1 - NdTwo(UnderlyingPrice, ExercisePrice, Time, Interest, Volatility, Dividend))

## OptionPage

« OptionPage » calcule, comme la feuille « basic », le prix théorique et les grecs ainsi que la volatilité implicite des options mais pour des prix d’exercices différents.

L’objectif est donc de jouer sur les variations du strike et d’observer les impacts sur le prix et les grecs.

Ainsi nous utilisons les mêmes fonctions VBA que la feuille « basic » pour effectuer les calculs.

De la cellule A14 à A24 pour les options Call et de la cellule A29 à A39 pour les Put, nous avons les différentes valeurs des strikes avec le strike à la monnaie(ATM) (A19=A2).

Ainsi à partir du strike à la monnaie en A19, il augmente ou diminue de +/- 0,5 (ou toute autre valeur spécifiée en F7) à chaque cellule pour pouvoir calculer un nouveau prix d’option théorique.

La fonction « CallOption » sur la plage C14 à C24 et la fonction « PutOption » de C29 à C39 vont donc prendre comme argument « ExercisePrice » les nouveaux strikes en A14 à A24.

CallOption($A$2;**A14;**$A$9;$A$6;$A$4;$A$7)

De même pour toutes les autres fonctions de calcul de grecs qui prennent cet argument.

(CallDelta, PutDelta, Gamma, Vega, CallTheta, PutTheta, CallRho, PutRho)

Nous avons voulu aussi indiquer si l’option était dans la monnaie (In the money ITM), en dehors de la monnaie (Out of the money OTM) ou à la monnaie (At the money ATM).

Pour cela nous avons utilisé la fonction Test\_logique d’Excel.

Pour le Call de B14 à B24 ; exemple en B14 : SI(A14<$A$2;"ITM";SI(A14=$A$2;"ATM";"OTM"))

Donc si le strike (A14) est inférieur au spot du sous-jacent (A2) (Test\_logique) nous avons une option « ITM » gagnante (Valeur\_si\_vrai), sinon si le strike est égale au sous-jacent la fonction renvoie la valeur « ATM » et si ce n’est pas le cas la valeur « OTM » une option perdante.

Pour le Put de B19 à B29 ; exemple en B19 : =SI(L14<$A$2;"OTM";SI(L14=$A$2;"ATM";"ITM"))

Ici bien sûr le test est inversé pour que l’option Put soit gagnante et dégage un payoff il faut que le strike soit supérieur au sous-jacent ; strike (L14) est inférieur au spot du sous-jacent (A2) (Test\_logique), l’option est « OTM »(Valeur\_si\_vrai), sinon la fonction renvoie la valeur « ATM » si Strike = Sous-jacent (L14 = A2) et si ce n’est pas le cas la valeur «ITM » Strike> sous-jacent (L14>A2) dans le cas d’une option Put gagnante.

La présence d’une colonne Market Price (D14 :D24) (D19 :D29) nous permet d’insérer les prix réels des options observés sur le marché.

Mais nous avons surtout voulu calculer les volatilités implicite des options Call et Put grâce à ces nouveaux prix du marché.

Colonne « Implied Volatility » de E14 à E24 et E19 à E29.

En effet la **volatilité implicite** reflète le « prix du risque » attaché à une option.  
La volatilité implicite représente les anticipations du marché sur les variations futures du sous-jacent.

Dans le modèle de Black et Scholes, le prix d'une option européenne ne dépend que de la volatilité du sous-jacent (B&S prend une volatilité constante et historique du sous-jacent).

Le modèle de Black-Scholes (B&S) suppose une volatilité constante or cela n’est absolument pas le cas dans la réalité.

Il est possible, connaissant le prix de marché du call/put de déduire une valeur unique pour la volatilité.

On peut donc, en inversant la formule à partir de la valeur de la prime et par approximation successives, retrouver une volatilité implicite de l'actif, qui est celle qui correspond aux anticipations du marché.

Ainsi, à toute prime est associé un niveau de volatilité implicite.

**Calcul de la Volatilité Implicite Call / Put:**

Nous allons donc déclarer une function que nous nommons «ImpliedCallVolatility»

La volatilité implicite du Call (E14:E24) va prendre les arguments :

= ImpliedCallVolatility($A$2;A14;$A$9;$A$6;D14;$A$7)

(A2 =UnderlyingPrice ; A14 =ExercisePrice ;A9 =Time; A6 =Interest; D14 =Target (Mkt Price); A7 =Dividend)

**Code : «ImpliedCallVolatility»**

*Function ImpliedCallVolatility(UnderlyingPrice, ExercisePrice, Time, Interest, Target, Dividend)*

*High = 5*

*Low = 0*

*Do While (High - Low) > 0.0001*

*If CallOption(UnderlyingPrice, ExercisePrice, Time, Interest, (High + Low) / 2, Dividend) > Target Then*

*High = (High + Low) / 2*

*Else: Low = (High + Low) / 2*

*End If*

*Loop*

*ImpliedCallVolatility = (High + Low) / 2*

*End Function*

Ici nous avons utilisé des instructions de répétition Do While (condition à vérifier) et Loop.

Les boucles Do Loop, associées aux mots clés While permettent de répéter une ou plusieurs actions jusqu'à ce qu'une condition soit remplie ;

**Do While** condition [Faire] [Tant que] condition  
Actions Les actions à effectuer  
**Loop** [Recommencer]

De même la fonction «ImpliedPutVolatility»a la même construction.

Nous allons donc déclarer une function que nous nommons «ImpliedPutVolatility»

La volatilité implicite du Put (P14:P24) va prendre les arguments :

= ImpliedPutVolatility($A$2;L14;$A$9;$A$6;O14;$A$7)

(A2 =UnderlyingPrice ; L14 =ExercisePrice ;A9 =Time; A6 =Interest; O14 =Target (Mkt Price); A7 =Dividend)

**Code : «ImpliedPutVolatility»**

*Function ImpliedPutVolatility(UnderlyingPrice, ExercisePrice, Time, Interest, Target, Dividend)*

*High = 5*

*Low = 0*

*Do While (High - Low) > 0.0001*

*If PutOption(UnderlyingPrice, ExercisePrice, Time, Interest, (High + Low) / 2, Dividend) > Target Then*

*High = (High + Low) / 2*

*Else: Low = (High + Low) / 2*

*End If*

*Loop*

*ImpliedPutVolatility = (High + Low) / 2*

*End Function*

Les 3 premières lignes High, low et la condition de la boucle while sont modifiables.

High et low fixent les valeurs d’encadrement (ici 5 et 0).

Dans cette configuration, le solveur est capable de trouver les valeurs de volatilité implicite comprises entre 0 et 500%. Augmenter la valeur de la borne haute permet des une plus grande plage de résultats possible, mais ralentit l’exécution des calculs.

Le résultat de la condition (ici 0.0001) donne la précision avec laquelle est calculée la volatilité implicite. Le solveur teste des valeurs jusqu’à se rapprocher à 0.0001 du prix observé de l’option.  
De la même façon que les bornes, on peut augmenter la précision en donnant un nombre plus petit, mais cela ralentit les calculs.

Nous pouvons donc maintenant comparer volatilité historique et volatilité implicite pour déterminer les sur et sous-évaluations des options Put et Call.

Ainsi, la volatilité implicite permet de mesurer la « cherté » d’une option.

On distingue :

* La cherté absolue : Comparaison par rapport à la volatilité historique
* La cherté relative : Comparaison des volatilités implicites de différentes séries au sein d’une même classe (option du même type couvrant le même actif)

Sur le marché action, les mouvements baissiers sont accompagnés d’une volatilité implicite forte (pessimisme) et inversement d’une faible volatilité implicite pour les mouvements haussiers (optimisme).

Les options très dans la monnaie (very deep ITM) et très en dehors de la monnaie (very deep OTM) sont peu sensibles aux variations de volatilité.

## OptionsStrategies

Cette feuille nous permet d’établir et d’évaluer différentes stratégies d’options et de calculer les payoffs et les variations des grecs tout au long de l’évolution du sous-jacent.

Nous utilisons les mêmes fonctions VBA complétés de fonctions de condition Excel SI, qui nous aide à déterminer par exemple si nous traitons un call ou un put, si nous somme long ou short et combien d’options nous traitons. Ce qui va bien sûr influencer les payoffs et les grecs.

Un graphique nous montre aussi l’évolution P&L Net (en bleu) et celui du P&L théorique selon les prix d’options évalués (en rose).

Il est d’autre part possible de sauvegarder puis de charger ou effacer des strategies, ce qui permet à l’utilisateur de garder ses simulations précédentes en mémoire, comme expliqué dans la dernière partie de ce document.

**Sauvegarder une stratégie :**

La sauvegarde des stratégies se fait via une fonction VBA qui stocke les données dans une feuille cachée :

*Sub SaveStrategy()*

*Dim Name As String*

*i = 1*

*j = 1*

*k = Sheets("OptionStrategies").Range("P11").Value*

*If k >= 10 Then*

*MsgBox ("You can't save more than" & k & "strategies. Please delete one before saving")*

*Exit Sub*

*End If*

*Name = InputBox("Strategy Name")*

*Sheets("SavedStrategies").Range("A" & 6 \* k + 1) = Name*

*Sheets("SavedStrategies").Range("B" & 6 \* k + 1 & ":K" & 6 \* k + 3).Value = Sheets("OptionStrategies").Range("B3:K5").Value*

*Sheets("SavedStrategies").Range("B" & 6 \* k + 5 & ":K" & 6 \* k + 5).Value = Sheets("OptionStrategies").Range("B7:K7").Value*

*Sheets("SavedStrategies").Range("N" & 6 \* k + 1 & ":N" & 6 \* k + 6).Value = Sheets("OptionStrategies").Range("N2:N7").Value*

*Sheets("OptionStrategies").Range("P11").Value = k + 1*

*End Sub*

La fonction vérifie tout d’abord par If que le nombre de stratégies sauvegardées n’atteint pas le maximum prévu (ici 10, mais ce nombre est modifiable).

Si ce nombre est atteint, un dialogue de type MsgBox informe l’utilisateur.

Dans le cas contraire, une boite de dialogue s’affiche et invite l’utilisateur à entrer un nom pour sa stratégie. Ce nom est stocké dans une variable

Ensuite, la stratégie est sauvegardée dans la feuille « SavedStrategies », a une ligne qui dépend du nombre de stratégies déjà sauvegardées dans la feuille.

**Charger ou supprimer une stratégie :**

Lorsqu’une ou plusieurs stratégies sont mémorisées, il est possible d’en effacer ou d’en charger une dans le pricer, afin de l’étudier ou de la modifier.

Ceci est réalisé par deux fonctions et un userform : Function LoadDelete(),Function LoadStrategy(number As Integer) et UserForm1 (UserForm contenant une liste et deux boutons)

La fonction LoadDelete est utilisée pour lister les stratégies sauvegardées, et les afficher dans un Userform :

*Function LoadDelete()*

*k = Sheets("OptionStrategies").Range("P11").Value*

*Dim Liste(10) As String*

*i = 0*

*While i <= k*

*Liste(i) = Sheets("SavedStrategies").Cells(i \* 6 + 1, 1).Value*

*i = i + 1*

*Wend*

*UserForm1.ListBox1.List = Liste*

*UserForm1.Show*

*End Function*

La boucle while récupère les noms des stratégies sauvegardées et les stocke dans des variables String sous forme d’array (un array est un tableau de variables possédant chacunes un nom commun et un numéro unique)

La liste du UserForm est ensuite peuplée avec les valeurs de l’array, et le UserForm est affiché.

Il est alors possible de supprimer, ou de charger une stratégie, grâce à deux bouton :

*Private Sub CommandButton1\_Click()*

*'Application.Run "LoadStrategy", Me.ListBox1.ListIndex*

*k = Me.ListBox1.ListIndex*

*Sheets("SavedStrategies").Activate*

*Sheets("OptionStrategies").Range("N2:N7").Value = Sheets("SavedStrategies").Range("N" & 6 \* k + 1 & ":N" & 6 \* k + 6).Value*

*Sheets("OptionStrategies").Range("B3:K5").Value = Sheets("SavedStrategies").Range("B" & 6 \* k + 1 & ":K" & 6 \* k + 3).Value*

*Sheets("OptionStrategies").Range("B7:K7").Value = Sheets("SavedStrategies").Range("B" & 6 \* k + 5 & ":K" & 6 \* k + 5).Value*

*Sheets("OptionStrategies").Activate*

*Me.Hide*

*End Sub*

*Private Sub CommandButton2\_Click()*

*k = Me.ListBox1.ListIndex*

*Sheets("SavedStrategies").Activate*

*Rows(k \* 6 + 1 & ":" & k \* 6 + 6).Select*

*Selection.Delete Shift:=xlUp*

*Sheets("OptionStrategies").Activate*

*Sheets("OptionStrategies").Range("P11").Value = Sheets("OptionStrategies").Range("P11").Value - 1*

*UserForm1.Hide*

*End Sub*

Charger une stratégie fonctionne de la même façon que sauvegarder une stratégie, mais copie depuis la feuille « SavedStrategies » vers « OptionStrategies.  
  
Supprimer une stratégie sélectionne dans la feuille « SavedStrategies » 6 lignes à partie de la première ligne de la stratégie à supprimer, et efface ces lignes.

## StrategyGraphs

Cette feuille reprend les données calculées pour nos stratégies de la feuille «OptionsStrategies» et les présente sous forme de graphique.

Nous avons ainsi les graphiques de l’évolution ;

* Du P&L Net à l’échéance,
* Du Delta,
* Du Gamma,
* Du Thêta
* Du Véga
* Du Rhô

# Guide d’utilisation

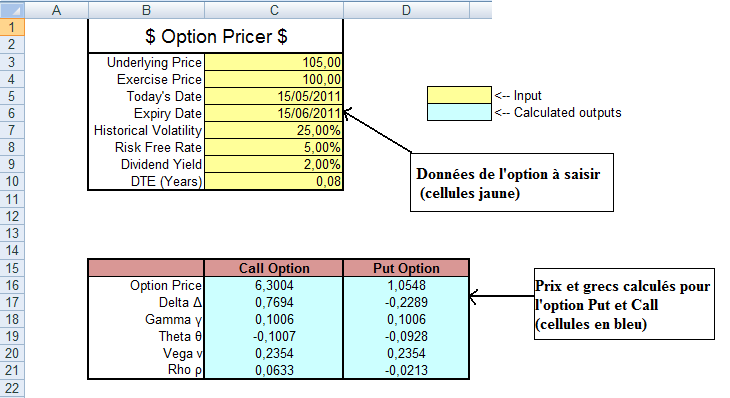
Tout au long de ce projet nous avons utilisés deux couleurs pour différencier deux types de cellule ;

- la couleur jaune pour les cellules où il faut saisir les données.

- la couleur bleu pour les données calculées par la macro VBA

**1/ Basic :**

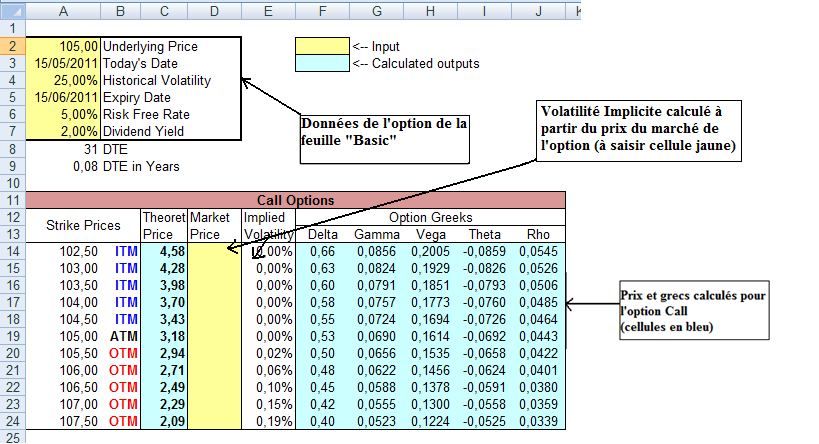
Voici notre pricer d’options principal, le reste du programme se reposera principalement sur cette feuille.



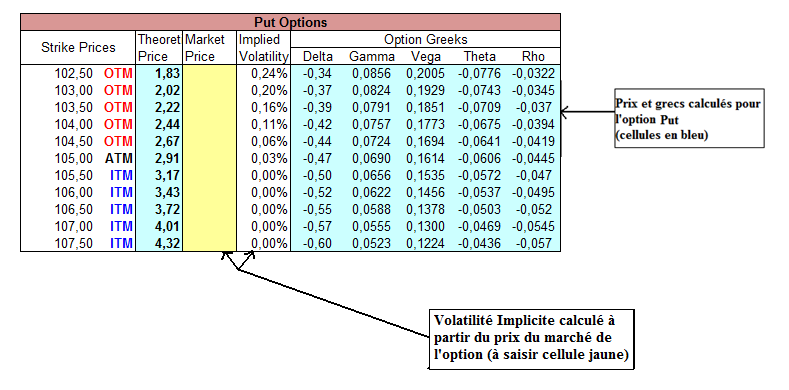
**2/ OptionPage :**

Cette page calcule le prix de l’option Call de la feuille « Basic » mais avec différents strikes.

Elle fournit les grecs ainsi que la volatilité implicite de l’option si le prix du marché est renseigné (colonne jaune, voir schéma)



De même pour l’option Put :

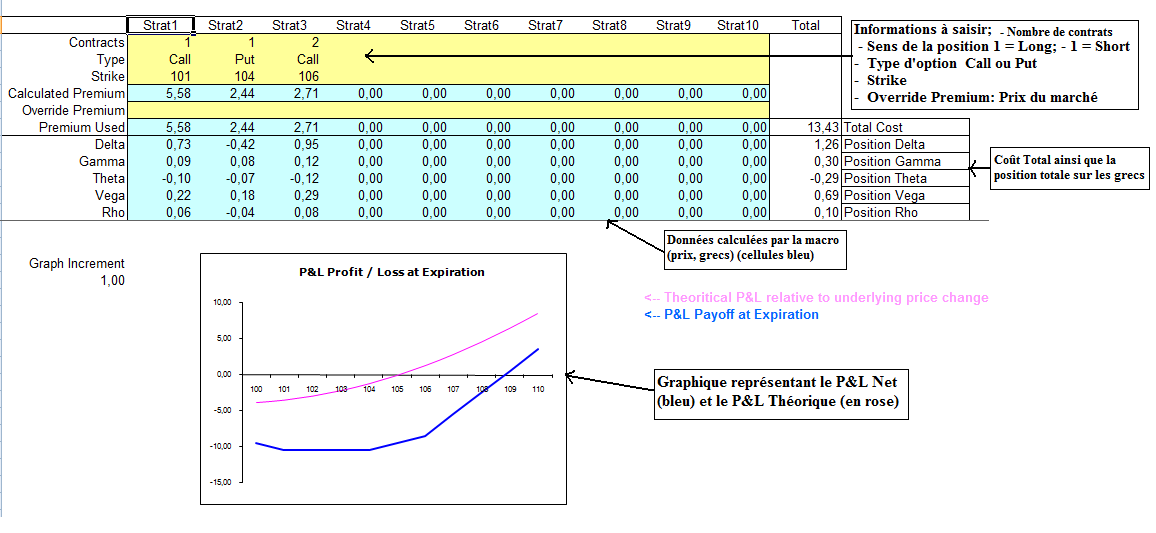


**3/ OptionStrategies :**

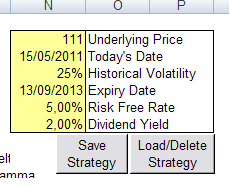
« OptionStrategies » nous permet d’établir et d’évaluer différentes stratégies d’options et de calculer les payoffs et les variations des grecs tout au long de l’évolution du sous-jacent.

Un graphique nous montre aussi l’évolution P&L Net (en bleu) et celui du P&L théorique selon les prix d’options évalués (en rose).

La cellule Graph Increment permet de modifier l’échelle du graphique (cette modification s’applique également à la feuille StrategyGraphs)

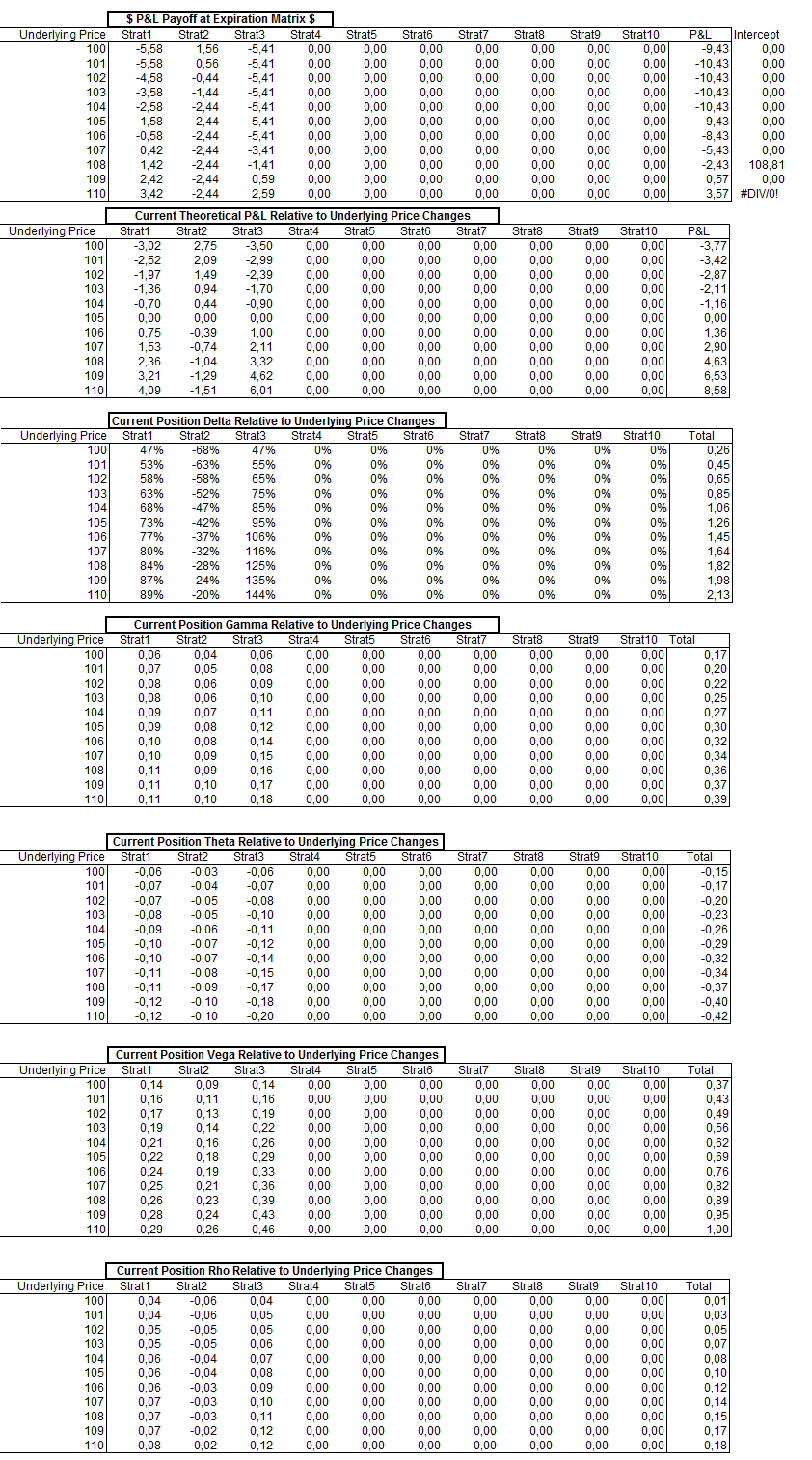


Enfin, il est possible de sauvegarder/effacer/charger des stratégies, ce qui permet de conserver son travail :



10 stratégies peuvent être sauvegardées au maximum.

Après avoir sauvegardé ou supprimé une/des stratégies, il est important de sauvegarder le fichier excel pour ne pas perdre les modifications.



Nous retrouvons ici le book de nos stratégies avec leurs payoffs et leurs grecs

**5/ StrategyGraphs :**

« StrategyGraphs » nous donne les graphiques de l’évolution de notre payoff et des grecs associés à nos stratégies d’options que nous avons établies dans « OptionStrategies » :

