Université de Montréal

Problème de Snell et application aux options bermudiennes

par

Vincent Gagnon

Département de mathématiques et de statistique

Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures

en vue de l'obtention du grade de

Maître ès sciences (M.Sc.)
en Probabilité

15 avril 2008

Université de Montréal

Faculté des études supérieures

Ce mémoire intitulé

Problème de Snell et application aux options bermudiennes

présenté par

Vincent Gagnon

a été évalué par un jury composé des personnes suivantes :

_____
(président-rapporteur)

Bruno Rémillard
_____
(directeur de recherche)

_____
(membre du jury)

Mémoire accepté le:

_____

REMERCIEMENTS

J'aimerais remercier mon directeur de recherche, Bruno Rémillard, pour son aide dans le choix d'un sujet de recherche aussi intéressant tout comme pour avoir dirigé mes travaux. Je remercie aussi Louis Doray et Manuel Morales pour leur lecture attentive du mémoire.

SOMMAIRE

Dans le cadre de ce mémoire, on présente une résolution du problème de Snell dans le cas où les temps d'arrêts ont des valeurs comprises dans l'ensemble suivant $\{0, 1, \ldots, n\}$. Ce résultat est par la suite utilisé dans un contexte d'options bermudiennes. Premièrement, on déterminera une couverture de risque possible dans le cadre des options bermudiennes. Deuxièmement, on évaluera aussi la valeur de ces options à l'aide de simulations Monte Carlo basées sur la résolution du problème de Snell effectuée précédemment. L'algorithme Monte Carlo présenté dans ce mémoire conserve les propriétés de convexité et monotonicité que peuvent avoir certaines options, et ce, en plus de pouvoir servir à implanter une autre méthode pour obtenir des bornes plus précises.

Mots Clés : Options bermudiennes, problème de Snell, région d'exercice.

SUMMARY

In this paper we solve the Snell problem in the particular case where the stopping times take values in a set $\{0, 1, \ldots, n\}$. This result is then used in the case of Bermudans options. First, we are using it to hedge this kind of option, and then we use a Monte Carlo algorithm based on this result to evaluate the price of the option. The algorithm in this paper includes the special feature of preserving the convexity and monotonicity of the options, if the option has these characteristics. At the same time, this algorithm can be used in conjunction with another algorithm to get sharper bounds on the value of the option.

Key words : Bermudans options, Snell problem, exercise region

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION

En ingénierie financière, l'évaluation d'options est un sujet où il y a eu plusieurs développements au cours des dernières décennies. D'un point de vue théorique, le problème d'optimisation (problème de Snell) résolu dans le livre de Neveu (1975) est central dans l'évaluation d'options américaines ou bermudiennes, car il permet de définir un temps d'arrêt afin d'exercer l'option au moment où le contrat a la plus grande valeur théorique. Bien que le problème de Snell ait été introduit dans le livre de Neveu (1975) à l'aide d'exemples sur les jeux de hasard, il est évident que ce problème est exactement le même que celui que l'on rencontre avec les options d'achats et de ventes.

Malgré le fait que le problème ait été résolu théoriquement, les chercheurs ont dû trouver des méthodes pratiques afin d'être en mesure de calculer la valeur de l'option. Une des premières méthodes de résolution mise sur pied a été proposée par Brennan et Schwartz (1977). Ces derniers se sont basés sur la résolution d'équations différentielles pour parvenir à la résolution du problème de Snell. Par la suite, les méthodes de résolution en arbre ont été introduites par Cox et al. (1979) pour évaluer la valeur d'options américaines. Malgré le fait que les méthodes de simulation Monte Carlo aient été discutées par Boyle (1977) pour les options européennes, il a fallu attendre jusqu'en 1993 avec la parution de l'article de Tilley (1993) pour voir la première évaluation d'options basée sur des simulations. Cet article a marqué le début d'une effervescence de la méthode Monte Carlo dans ce domaine. Le domaine de l'évaluation d'options a par la suite vu naître plusieurs articles améliorant les résultats de Tilley (1993), tels que Carriere (1996), Broadie et Glasserman (1997), Longstaff et Schwartz (2001), Broadie et

Glasserman (2004). Plus récemment, des méthodes permettant l'évaluation de bornes inférieures et supérieures de la valeur de l'option ont été développées par Rogers (2002) et Andersen et Broadie (2004).

Dans certains cas d'options particulières, les fonctions représentant les valeurs possèdent des propriétés particulières telles la monotonicité et la convexité, comme dans le cas d'options américaines d'achat ou de vente. L'algorithme présenté dans le présent mémoire permet de conserver les propriétés de monotonicité et de convexité, ce que les méthodes Monte Carlo précédemment énumérées ne permettaient pas de manière générale. Ce mémoire a donc comme objectif d'énoncer les résultats théoriques permettant la réalisation de cet algorithme et de comparer les résultats à d'autres algorithmes.

Dans la première section de ce mémoire, on retrouve la démonstration de la résolution du problème de Snell pour le cas particulier des options bermudiennes. Ensuite, ce résultat est utilisé pour démontrer comment on peut faire une couverture de risque sur une telle option. Suit la description de l'algorithme préservant les propriétés de monotonicité et de convexité. Finalement on présente les résultats de cette procédure lorsqu'elle est appliquée à des options bermudiennes. Étant donné que l'algorithme développé permet l'application des résultats d'Andersen et Broadie (2004), il y a aussi une description des algorithmes de cet article et quelques résultats numériques intéressants. Plus précisément, les algorithmes de "borne supérieure" et "borne inférieure" de l'article de Andersen et Broadie (2004) sont aussi décrits dans le chapitre 3.

Chapitre 1

PROBLÈME DE SNELL

Soit $(\Omega, \mathcal{F}, P)$ un espace probabilisé.

1.1. Définitions

Les définitions suivantes seront utiles afin de bien comprendre les énoncés du présent chapitre. On peut aussi retrouver les définitions dans Billingsley (1995).

Définition 1.1.1. *On dit que* $\mathbb{F} = \{\mathcal{F}_t; t = 0, 1, 2, \ldots\}$ *est une filtration si pour tout* $t \in \{0, 1, 2, \ldots\}$, $\mathcal{F}_t$ *est une suite de sous-tribus de* $\mathcal{F}$ *telle que*

$$\mathcal{F}_t \subseteq \mathcal{F}_{t+1}.$$

Définition 1.1.2. *Une variable aléatoire* τ, *prenant des valeurs dans* $\mathbb{N} \cup \{+\infty\}$, *est un temps d'arrêt par rapport à la filtration* $\mathbb{F}$ *si pour tout* $t \in \{0, 1, 2, \ldots\}$,

$$\{\tau \leq t\} \in \mathcal{F}_t.$$

Définition 1.1.3. *Soit* $X = (X_t)_{t \geq 0}$ *un processus stochastique, où* t *est discret et* $\mathbb{F}$ *une filtration. On dit que* X *est un processus* $\mathbb{F}$*-adapté si chaque* $t \in \{0, 1, 2, \ldots\}$, X_t *est* $\mathcal{F}_t$*-mesurable.*

Définition 1.1.4. *Soit X un processus $\mathbb{F}$-adapté où $\mathbb{F}$ est une filtration. Alors $(X_t, \mathcal{F}_t)_{t\geq 0}$ est une martingale si pout tout $t \in \{0, 1, 2, \ldots\}$,*

- $E(|X_t|) < \infty$,
- $E(X_{t+1}|\mathcal{F}_t) = X_t$ *p.s.*

De plus, on dit que la suite $(X_t, \mathcal{F}_t)_{t\geq 0}$ est une supermartingale si

- $E(|X_t|) < \infty$,
- $E(X_{t+1}|\mathcal{F}_t) \leq X_t$ *p.s.*

1.2. Énoncé du problème de Snell

Soit $X = (X_t)_{t=1}^n$ un processus adapté à la filtration $\mathbb{F} = \{\mathcal{F}_t; t = 0, \ldots, n\}$, tel que $E(|X_t|) < \infty$ pour tout $t \leq n$, et soit $\mathbb{T}$ l'ensemble des temps d'arrêt à valeur dans $\mathcal{T} = \{0, 1, \ldots, n\}$. On cherche à résoudre le problème suivant, dit problème de Snell : est-il possible de trouver un temps d'arrêt $\tau^* \in \mathbb{T}$, tel que

$$E(X_{\tau^*}) = \sup_{\tau\in\mathbb{T}} E(X_\tau). \tag{1.2.1}$$

Pour en faire la résolution, nous allons étudier un problème qui à prime abord est un peu plus complexe. Il s'agit de trouver $U = (U_t)_{t=0}^n$ tels que

$$U_t = \sup_{\tau\in\mathbb{T},\, \tau\geq t} E(X_\tau|\mathcal{F}_t), \quad t = 0, \ldots, n. \tag{1.2.2}$$

Nous allons montrer que dans le cas présent, la solution est donnée par

$$U_t = \begin{cases} X_n & \text{si } t = n, \\ \max\{X_t, E(U_{t+1}|\mathcal{F}_t)\} & \text{si } t < n, \end{cases} \tag{1.2.3}$$

et que $\tau_0(\omega) = min\{s \geq 0 | U_s(\omega) = X_s(\omega)\} \in \mathbb{T}$ satisfait l'équation (1.2.1).

Le processus U est bien adapté à la filtration $\mathcal{F}$. Dans le cas où $t = n$, $U_n = X_n$ et X est adapté à la filtration $\mathcal{F}$. Dans notre dernier cas, où $t \leq n$, comme X_{t_j} et $E(U_{t_{j+1}}|\mathcal{F}_{t_j})$ sont tous les deux $\mathcal{F}_t$-mesurables, le maximum l'est aussi.

1.3. Lien avec les options bermudiennes

Le problème de Snell tel qu'énoncé précédemment n'est qu'un problème théorique, mais il est tout de même possible de l'appliquer aux options bermudiennes. Si on reprend l'énoncé de départ, on peut redéfinir le tout en termes financiers. Par exemple X_t est la valeur d'exercice de l'option au temps t, et donc le processus stochastique U représente l'évolution de la valeur de l'option au cours du temps.

Si on se place dans la peau du détenteur de l'option, l'option bermudienne nous permet à certains moments prédéfinis de choisir entre exercer son droit ou bien attendre le prochain moment où un tel choix sera possible. Le problème de Snell tente en fait de répondre à cette question : Dois-je exercer maintenant mon droit ou attendre encore ? En d'autres mots, à quel moment dans le temps est-ce que la valeur d'exercice de mon option sera la plus grande ?

Le problème de Snell cherche en fait le temps d'arrêt optimal τ^*. C'est-à-dire, à quel moment doit-on exercer l'option pour en retirer un maximum de profit ? Cette décision se fait en fonction d'un certain critère qui nous permet à chaque moment de décider si la valeur d'exercice de l'option au temps présent est plus grande que celle qu'elle aura dans le futur. Comme on ne peut prédire ce qui se passera dans le futur avec la valeur de l'option, on compare ce qu'on a présentement avec l'espérance de la valeur de l'option dans le futur à chaque moment d'exercice possible. La fonction U_t, aussi appelée l'enveloppe de Snell, est la fonction qui permet cette comparaison puisqu'elle représente la valeur actualisée de l'option au temps t.

1.4. Résolution

Nous allons résoudre ce problème en quatre étapes. Les trois premières consistent en trois lemmes qui seront énoncés puis démontrés. La quatrième étape sera, quant à elle, la concaténation de nos résultats précédents afin d'arriver à notre conclusion. La démonstration des lemmes et du résultat principal est fortement inspirée des notes de cours de Gauthier (2006).

Définition 1.4.1. *Pour nous aider dans les résultats qui suivent, nous allons introduire la notation suivante.*

$$\Lambda_t = \{\tau : \Omega \to \{t, t+1, \ldots, n\} | \tau \text{ est un temps d'arrêt}\},$$

$t \in \mathcal{T}$.

Λ_t *représente donc l'ensemble des temps d'arrêt prenant des valeurs plus grandes ou égales à* t.

Soit U le processus défini par (1.2.3).

Lemme 1.4.1. *Pour tout* $t \in \{0, \ldots, n\}$ *et pour tout* $\tau \in \Lambda_t$

$$U_t \geq E(X_\tau | \mathcal{F}_t). \tag{1.4.1}$$

Démonstration. Pour démontrer ce lemme nous allons procéder par induction inversée sur t. Commençons par $t = n$. Lorsque $t = n$, Λ_t ne contient que le temps d'arrêt $\tau = n$, ce qui entraîne que $E(X_\tau | \mathcal{F}_n) = E(X_n | \mathcal{F}_n) = X_n$. Puisque, par définition, $U_n = X_n$, on a donc

$$U_n = X_n \geq E(X_\tau | \mathcal{F}_n).$$

Notre égalité est donc vérifiée pour $t = n$.

Pour la suite, nous allons supposer que l'inégalité (1.4.1) est vérifiée pour t et nous allons démontrer le lemme pour $t-1$. Choisissons un temps d'arrêt $\tau \in \Lambda_{t-1}$ et évaluons l'espérance conditionnelle par rapport à la filtration au temps $t-1$.

$$\begin{aligned} E(X_\tau | \mathcal{F}_{t-1}) &= E(X_\tau 1_{\{\tau = t-1\}} | \mathcal{F}_{t-1}) + E(X_\tau 1_{\{\tau > t-1\}} | \mathcal{F}_{t-1}) \\ &= E(X_{t-1} 1_{\{\tau = t-1\}} | \mathcal{F}_{t-1}) + E(X_{\tau \vee t} 1_{\{\tau > t-1\}} | \mathcal{F}_{t-1}). \end{aligned} \tag{1.4.2}$$

Nous allons maintenant considérer le premier terme de l'équation (1.4.2). Nous savons que

$$U_{t-1} = \max\{X_{t-1}, E(U_t | \mathcal{F}_{t-1})\} \geq X_{t-1},$$

et donc

$$\begin{aligned} E(X_{t-1} 1_{\{\tau = t-1\}} | \mathcal{F}_{t-1}) &\leq E(U_{t-1} 1_{\{\tau = t-1\}} | \mathcal{F}_{t-1}) \\ &= U_{t-1} 1_{\{\tau = t-1\}}. \end{aligned}$$

Il nous reste à considérer le deuxième terme de l'équation (1.4.2).

Comme le maximum entre deux temps d'arrêt est lui aussi un temps d'arrêt, $\tau \vee t$ est donc un temps d'arrêt. De plus, $\tau \vee t$ est élément de Λ_t, ce qui nous permet d'appliquer l'hypothèse d'induction suivante : $U_t \geq E(X_{\tau \vee t}|\mathcal{F}_t)$.

On peut ainsi développer le deuxième terme de l'équation comme suit :

$$\begin{aligned} E(X_{\tau\vee t}1_{\{\tau>t-1\}}|\mathcal{F}_{t-1}) &= 1_{\{\tau>t-1\}}E(X_{\tau\vee t}|\mathcal{F}_{t-1}) \\ &= 1_{\{\tau>t-1\}}E(E(X_{\tau\vee t}|\mathcal{F}_t)|\mathcal{F}_{t-1}) \\ &\leq 1_{\{\tau>t-1\}}E(U_t|\mathcal{F}_{t-1}) \\ &\leq 1_{\{\tau>t-1\}}\max\left\{X_{t-1}, E(U_t|\mathcal{F}_{t-1})\right\} \\ &= 1_{\{\tau>t-1\}}U_{t-1}, \end{aligned}$$

car $\tau \in \Lambda_{t-1}$.

Donc $U_{t-1} \geq E(X_\tau|\mathcal{F}_{t-1})\ \forall \tau \in \Lambda_{t-1}$, ce qui complète la preuve. □

Lemme 1.4.2. *Soit le temps aléatoire suivant :*

$$\tau_t(\omega) = min\left\{s \geq t | U_s(\omega) = X_s(\omega)\right\}, \tag{1.4.3}$$

Alors $\tau_t(\omega)$ est un temps d'arrêt, c'est-à-dire que $\forall s \geq t$,

$$\{\omega \in \Omega : \tau_t(\omega) = s\} \in \mathcal{F}_s.$$

Démonstration. Pour un $s \geq t$,

$$\begin{aligned} &\{\omega \in \Omega : \tau_t(\omega) = s\} \\ &= \{\omega \in \Omega : \forall u < s, U_u(\omega) \neq X_u(\omega) \text{ et } U_s(\omega) = X_s(\omega)\} \\ &= \{\omega \in \Omega : \forall u < s, U_u(\omega) > X_u(\omega) \text{ et } U_s(\omega) = X_s(\omega)\} \\ &= \bigcap_{u<s} \underbrace{\{\omega \in \Omega : U_u(\omega) > X_u(\omega)\}}_{\in \mathcal{F}_u \subseteq \mathcal{F}_s} \bigcap \underbrace{\{\omega \in \Omega : U_s(\omega) = X_s(\omega)\}}_{\in \mathcal{F}_s} \in \mathcal{F}_s \end{aligned}$$

□

Remarque 1.4.1. *Nous venons de démontrer que τ_t est un temps d'arrêt, mais il est bien de spécifier dans sa définition qu'il est élément de Λ_t.*

Lemme 1.4.3. *Pour tout* $t \in 0, ..., n$

$$E(X_{\tau_t}|\mathcal{F}_t) = U_t.$$

DÉMONSTRATION. Comme au lemme 1.4.1, nous allons procéder par induction inversée sur t.

Posons t = n.

$$\begin{aligned}\forall\omega, \tau_n(\omega) &= \min\{s \geq n | U_s(\omega) = X_s(\omega)\} \\ &= n \\ &\Rightarrow E(X_{\tau_n}|\mathcal{F}_n) = E(X_n|\mathcal{F}_n) = X_n = U_n.\end{aligned}$$

L'égalité est donc vérifiée pour $t = n$.
Supposons le lemme 3 vrai pour t et vérifions le pour $t-1$.

$$\begin{aligned}E(X_{\tau_{t-1}}|\mathcal{F}_{t-1}) &= E(X_{\tau_{t-1}}1_{\{\tau_{t-1}=t-1\}}|\mathcal{F}_{t-1}) + E(X_{\tau_{t-1}}1_{\{\tau_{t-1}>t-1\}}|\mathcal{F}_{t-1}) \\ &= E(X_{t-1}1_{\{\tau_{t-1}=t-1\}}|\mathcal{F}_{t-1}) + E(X_{\{\tau_{t-1}\vee t\}}1_{\{\tau_{t-1}>t-1\}}|\mathcal{F}_{t-1}) \quad (1.4.4)\end{aligned}$$

Pour le premier terme de l'équation (1.4.4), on a

$$\begin{aligned}E(X_{t-1}1_{\{\tau_{t-1}=t-1\}}|\mathcal{F}_{t-1}) &= X_{t-1}1_{\{\tau_{t-1}=t-1\}} \\ &= U_{t-1}1_{\{\tau_{t-1}=t-1\}},\end{aligned}$$

la dernière égalité venant du fait que si le temps d'arrêt $\tau_{t-1} = t-1$, cela veut dire que $X_{t-1} = U_{t-1}$.
Traitons maintenant le deuxième terme de l'équation (1.4.4). Montrons que $\tau_{t-1} = \tau_t$ lorsque $\tau_{t-1} > t-1$. En effet, dans ce cas,

$$\tau_{t-1} = \min\{s \geq t | U_s(\omega) = X_s(\omega)\} = \tau_t.$$

Revenons au deuxième terme

$$\begin{aligned}E(X_{\{\tau_{t-1}\vee t\}}1_{\{\tau_{t-1}>t-1\}}|\mathcal{F}_{t-1}) &= E(X_{\tau_t}1_{\{\tau_{t-1}>t-1\}}|\mathcal{F}_{t-1}) \\ &= 1_{\{\tau_{t-1}>t-1\}}E(X_{\tau_t}|\mathcal{F}_{t-1})\end{aligned}$$

$$= 1_{\{\tau_{t-1}>t-1\}} E(E(X_{\tau_t}|\mathcal{F}_t)|\mathcal{F}_{t-1}) \tag{1.4.5}$$

$$= 1_{\{\tau_{t-1}>t-1\}} E(U_t|\mathcal{F}_{t-1}) \tag{1.4.6}$$

$$= 1_{\{\tau_{t-1}>t-1\}} U_{t-1}. \tag{1.4.7}$$

Le passage de l'équation (1.4.5) à l'équation (1.4.6) se fait à l'aide de l'hypothèse d'induction.

Lorsque $\tau_{t-1} > t-1 \Rightarrow U_{t-1}(\omega) > X_{t-1}(\omega), \forall\omega$

$$\begin{aligned} U_{t-1} &= \max\{X_{t-1}(\omega), E(U_t|\mathcal{F}_{t-1})\} \\ &= E(U_t|\mathcal{F}_{t-1}), \end{aligned}$$

Maintenant que nous venons de justifier les passages des équations (1.4.5) à (1.4.6) et (1.4.6) à (1.4.7), on peut recoller les morceaux de l'équation (1.4.4) pour obtenir le résultat final du lemme 1.4.3 :

$$\begin{aligned} E(X_{\tau_{t-1}}|\mathcal{F}_{t-1}) &= E(X_{\tau_{t-1}} 1_{\{\tau_{t-1}=t-1\}}|\mathcal{F}_{t-1}) + E(X_{\tau_{t-1}} 1_{\{\tau_{t-1}>t-1\}}|\mathcal{F}_{t-1}) \\ &= U_{t-1} 1_{\{\tau_{t-1}=t-1\}} + U_{t-1} 1_{\{\tau_{t-1}>t-1\}} \\ &= U_{t-1} 1_{\{\tau_{t-1}\geq=t-1\}} \\ &= U_{t-1}. \end{aligned}$$

□

Nous venons de terminer la démonstration des trois lemmes qui nous serviront à démontrer le théorème principal, soit celui relié au problème de Snell.

PROBLÈME DE SNELL. Si on applique les trois lemmes, on trouve que

$$\sup_{t\in\Lambda_t} E(X_\tau|\mathcal{F}_t) \leq U_t = E(X_{\tau_t}|\mathcal{F}_t) \leq \sup_{t\in\Lambda_t},$$

d'où

$$\sup_{t\in\Lambda_t} E(X_\tau|\mathcal{F}_t) = U_t = E(X_{\tau_t}|\mathcal{F}_t) E(X_\tau|\mathcal{F}_t).$$

En particulier, lorsque $t = 0$, on obtient

$$E[X_{\tau_0}|\mathcal{F}_0] = U_0 = \sup_{t\in\Lambda_0} E(X_\tau).$$

Donc τ_0 est bel et bien un temps d'arrêt optimal que nous recherchions pour le problème de Snell (équation (1.2.1)). On conclut aussi que le U défini par l'équation (1.2.3) est bien la solution de (1.2.2) que l'on cherchait.

□

1.5. Propriétés de la solution

Une propriété mathématique de la solution obtenue dans ce chapitre est que U est en fait la plus petite supermartingale qui domine X. On entend par le mot "domine", que $U_t \geq X_t$, pout tout $t = 0, \ldots, n$. En voici la démonstration.

Démonstration. La définition de U_t nous dit que pour $t = n$, $U_n = X_n$ et pour les autres $t = 1, \ldots ..., n$,

$$U_{t-1} \geq X_{t-1}$$

et

$$U_{t-1} \geq E(U_t|\mathcal{F}_{t-1}).$$

Ceci montre bien que U_t domine X_t et que U est une supermartingale.

Il reste à prouver que U_t est la plus petite supermartingale qui domine X_t. Comme dans la preuve principale de ce chapitre, on retrouve l'utilisation de l'induction inversée sur t, soit en commençant par $t = n$ et en diminuant vers $t - 1$. Pour ce faire, posons Y une autre supermartingale qui domine X. Ceci voudrait dire que $Y_n \geq X_n = U_n$. Supposons donc que $Y_t \geq U_t$ et montrons que nécessairement, $Y_{t-1} \geq U_{t-1}$.

À l'aide de la propriété de supermartingale de Y, on a

$$Y_{t-1} \geq E(Y_t|\mathcal{F}_{t-1}) \geq E(U_t|\mathcal{F}_{t-1}).$$

Comme Y domine X, on a donc

$$Y_{t-1} \geq \max(X_{t-1}, E(U_t|\mathcal{F}_{t-1})) = U_{t-1}.$$

On peut donc conclure que Y domine U, ce qui implique que U est la plus petite supermartingale qui domine X. □

1.6. Exemple

Dans cette section, un exemple reflétant le problème de Snell et sa solution est présenté. Considérons une option de vente où la valeur au temps t est

$$X_t = B_t \cdot \max\{100 - S_t, 0\},$$

où S_t est la valeur du sous-jacent au temps t. Le taux d'actualisation sera dans cet exemple $B_t = \frac{1}{1{,}05^t}$.

Pour simplifier les calculs, le sous-jacent suivra un modèle à deux périodes où les seuls événements possibles sont les suivants. (On retrouve un exemple semblable dans Gauthier (2006).) Il est possible d'exercer l'option au temps 1 et 2.

ω	S_0	S_1	S_2	$P(\omega)$
ω_1	100	115	123,25	30%
ω_2	100	115	117	20%
ω_3	100	95	106,25	30%
ω_4	100	95	90	20%

Remarque 1.6.1. *La mesure de probabilité P dans le tableau fait en sorte que le prix actualisé du sous-jacent est une martingale.*

La filtration dans ce problème est :

- $\mathcal{F}_0 = \{\emptyset, \Omega\}$,
- $\mathcal{F}_1 = \sigma\{\{\omega_1, \omega_2\}, \{\omega_3, \omega_4\}\}$,
- $\mathcal{F}_1 = \sigma\{\omega_1, \omega_2, \omega_3, \omega_4\}$.

Le tableau suivant représente la valeur d'exercice X actualisée pour toutes les valeurs possibles du sous-jacent.

ω	X_0	X_1	X_2
ω_1	0	0	0
ω_2	0	0	0
ω_3	0	$\frac{5}{1{,}05} \approx 4,76$	0
ω_4	0	$\frac{5}{1{,}05} \approx 4,76$	$\frac{10}{1{,}05^2} \approx 9,07$

On pourrait choisir comme temps aléatoire celui où l'on exercerait l'option de vente au moment où la valeur serait à son maximum pour chaque ω. Ce qui donne dans notre exemple le temps aléatoire suivant :

$$(\tau(\omega_1), \tau(\omega_2), \tau(\omega_3), \tau(\omega_4)) = (2, 2, 1, 2).$$

Malheureusement, ce temps aléatoire n'est pas un temps d'arrêt car

$$\{\omega \in \Omega : \tau(\omega) = 1\} = \{\omega_3\} \notin \mathcal{F}_1.$$

Donc pour employer cette stratégie, il faudrait être capable de prédire l'avenir, ce qui est bien sûr impossible. Cependant, la solution du problème de Snell que nous avons développée dans ce chapitre nous permet de maximiser notre espérance de gain en se basant sur les informations disponibles au moment de la prise de décision.

ω	U_0	U_1	U_2
ω_1	2,38	0	0
ω_2	2,38	0	0
ω_3	2,38	$\max(4,76; 3,628) = 4,76$	0
ω_4	2,38	$\max(4,76; 3,628) = 4,76$	9,07

Suite à ces calculs, on obtient que le τ_0 de la solution du problème de Snell est $\tau_0 = (1, 1, 1, 1)$. τ_0 est choisi pour chaque ω comme étant le premier temps pour lequel $U_t = X_t$. La valeur associée à ce temps d'arrêt est de $2,38$. On peut voir que ce n'est pas le seul temps d'arrêt pour lequel on atteint cette valeur. En effet, $(2, 2, 1, 1)$ est un autre temps d'arrêt qui a la même valeur, mais la solution du problème de Snell nous assure que nous obtenons la valeur maximale avec τ_0.

1.7. Cas markovien

Ici on suppose que $X_t = f_t(S_t)$, où S est une chaîne de Markov pour la filtration $\mathbb{F}$ et f_t est une fonction mesurable. Dans ce cas, il est facile de voir que $U_t = u_t(S_t)$, où $u_n(s) = f_n(s)$ et pour tout $0 \leq t < n$,

$$u_t(s) = \max\left(f_t(s), E(u_{t+1}(S_{t+1})|S_t = s\right).$$

Cela montre bien que dans le cas markovien décrit ci-dessus, la solution du problème de Snell dépend d'une fonction déterministe $u(t, s)$.

Chapitre 2

COUVERTURE DU RISQUE

2.1. Situation-problème

Comment est-il possible de faire une couverture de risque ("hedging") sur les options bermudienne étudiées dans cet article ?

Supposons qu'une option bermudienne peut-être exercée au temps $t_0 = 0 < t_1 < ... < t_n = T$ et notons la fonction de la valeur de l'option actualisée au temps t_k, par f_k.

Supposons aussi que nous sommes dans un contexte de Black-Scholes, où la valeur du sous-jacent S_t satisfait l'équation suivante sous la mesure neutre au risque Q ,

$$dS_t = (r - \delta)S_t dt + \sigma S_t dW_t \tag{2.1.1}$$

où r est le taux d'intérêt et δ est le taux des dividendes.

Grâce aux résultats obtenus dans le chapitre 1, on sait que la valeur actualisée de l'option au temps t_k est donnée par l'équation suivante :

$$U_k(S_{t_k}) = max\left\{f_k, E_Q[U_{k+1}(S_{t_{k+1}})|F_{t_k}]\right\} \text{ pour } 0 \leq t < n$$

et pour t = n, $U_n = f_n$. Il a aussi été question du plus petit temps d'arrêt optimal, qui a été donné par

$$\tau = min\left\{k \geq 0; f_k = U_k\right\}.$$

Maintenant que la situation est établie, notons la valeur du portefeuille Π_t nécessaire pour faire la couverture de risque de l'option bermudienne. L'équation du portefeuille est :

$$\Pi_t = \phi_t e^{rt} - \psi_t S_t$$

Par le fait même, lorsqu'il y a un changement dans le temps :

$$d\Pi_t = d(\phi_t e^{rt}) - \psi_t dS_t,$$

ϕ_t est la valeur de l'option au temps t, qui est multiplié par le facteur d'actualisation et dans l'équation de la valeur du portefeuille, ψ_t est égale à la valeur de l'option (actualisée) divisée par la valeur du sous-jacent afin d'annuler la valeur de l'option. Il est à noter que dans la seconde équation, ψ_t n'est qu'un nombre qu'il faut déterminer afin que $d\Pi_t$ soit encore nul.

D'après la formule de Ito, et en remplaçant $\phi_t e^{rt}$ par V(S,t), on obtient :

$$dV = \frac{\delta V}{\delta t}dt + \frac{\delta V}{\delta S}dS + \frac{1}{2}\sigma^2 S^2 \frac{\delta^2 V}{\delta S^2}dt,$$

ce qui nous donne comme équation pour le delta du portefeuille en entier :

$$d\Pi_t = \frac{\delta V}{\delta t}dt + \frac{\delta V}{\delta S}dS + \frac{1}{2}\sigma^2 S^2 \frac{\delta^2 V}{\delta S^2}dt - \psi_t dS_t,$$

Donc, comme notre but est d'éliminer l'effet du hasard causé par la valeur du sous-jacent, il faudrait pouvoir annuler les termes dS. Ceci peut être atteint si l'on pose

$$\psi_t = \frac{\delta V}{\delta S},$$

En choisissant cette valeur pour ψ_t, il reste toujours

$$d\Pi_t = \frac{\delta V}{\delta t}dt + \frac{1}{2}\sigma^2 S^2 \frac{\delta^2 V}{\delta S^2}dt,$$

En supposant qu'il n'y a pas d'arbitrage possible, nous avons la relation suivante

$$d\Pi_t = r\Pi_t dt,$$

En effectuant ensuite les substitutions appropriées, nous arrivons au résultat

$$\begin{aligned} d\Pi_t &= \frac{\delta V}{\delta t}dt + \frac{1}{2}\sigma^2 S^2 \frac{\delta^2 V}{\delta S^2}dt \\ r\Pi_t dt &= \frac{\delta V}{\delta t}dt + \frac{1}{2}\sigma^2 S^2 \frac{\delta^2 V}{\delta S^2}dt \\ r\Pi_t &= \frac{\delta V}{\delta t} + \frac{1}{2}\sigma^2 S^2 \frac{\delta^2 V}{\delta S^2} \\ r(V - \frac{\delta V}{\delta S}S_t) &= \frac{\delta V}{\delta t} + \frac{1}{2}\sigma^2 S^2 \frac{\delta^2 V}{\delta S^2} \\ -\frac{\delta V}{\delta t} &= -rV + \frac{1}{2}\sigma^2 S^2 \frac{\delta^2 V}{\delta S^2} + r\frac{\delta V}{\delta S}S_t \end{aligned}$$

Nous reconnaissons ici l'équation différentielle partielle de Black-Scholes. Ce qui nous permet de dire que la relation est conservée si nous prenons $\psi_t = \frac{\delta V}{\delta S}$.

2.2. Résolution

Avec les démarches de la section précédente, on voudrait pouvoir évaluer la valeur de ψ_t afin de se couvrir face au risque de l'option bermudienne. Comme cette valeur est dynamique, c'est-à-dire qu'elle change constamment dans le temps, il ne serait pas possible de la calculer à chaque instant pour faire l'action nécessaire pour faire la couverture de risque du portefeuille. Cependant, une des façons de faire serait de l'évaluer à certain temps fixe $t_1, t_2, ...t_n$ et de faire un rebalancement à ces moments précis.

Afin de simplifier la suite, introduisons la notation suivante :

Définition 2.2.1.

$$h_k(t, S_t) = E_Q[U_k(S_{t_k})|F_t], \ \textit{pour } 0 \leq t \leq t_k,$$

Dans les cas où l'option est exercée, il n'est plus nécessaire de continuer la couverture de risque.

Maintenant, supposons que l'option n'est pas exercée au temps t=0 sinon l'exemple serait trivial, donc $f_0 < U_0(S_0)$ pour tout t entre 0 et t_1, et

$$\psi(t, S_t) = e^{rt}\frac{\delta h_1(t,s)}{\delta s}|_{s=S_t},$$

Encore une fois, si l'option est exercée au temps t_1, il n'y a plus de couverture de risque à faire. Autrement, on a que $f_1(S_{t_1}) < U_1(S_{t_1})$, ce qui implique que $h_1(t_1, S_{t_1}) = h_2(t_1, S_{t_1})$. En effet,

$$\begin{aligned} h_1(t_1, S_{t_1}) &= E_Q[U_1(S_{t_1})|F_{t_1}] \\ &= E_Q[max\left\{f_1, E_Q[U_2(S_{t_2})|F_{t_1}]\right\}|F_{t_1}] \\ &= E_Q[E_Q[U_2(S_{t_2})|F_{t_1}]|F_{t_1}] \\ &= E_Q[U_2(S_{t_2})|F_{t_1}] \\ &= h_2(t_1, S_{t_1}), \end{aligned}$$

Il s'en suit que pour $t_1 \leq t < t_2$,

$$\psi(t, S_t) = e^{rt}\frac{\delta h_2(t,s)}{\delta s}|_{s=S_t},$$

Si on poursuit ce procédé pour chaque temps d'exercice possible, on obtiendrait que si l'option n'est pas exercée au temps $= t_k$, pour $0 <$ k $<$ n, on a $f_k(S_{t_k}) < U_k(S_{t_k})$ et $h_k(t_k, S_{t_k}) = h_{k+1}(t_k, S_{t_k})$. Il s'en suit que pour $t_k \leq t < t_{k+1}$,

$$\psi(t, S_t) = e^{rt}\frac{\delta h_{k+1}(t,s)}{\delta s}|_{s=S_t},$$

Comme $S_{t_k} = S_t e^{(r-\delta)(t_k-t)+\sigma(W_{t_k}-W_t)}$ (résultat obtenu lorsque l'on résout l'équation 2.1.1) est indépendant de $\mathcal{F}_t$, il s'en suit que l'on peut écrire h_k comme

$$\begin{aligned} h_k(t, S_t) &= E\left[U_k(se^{(r-\delta)(t_k-t)+\sigma(W_{t_k}-W_t)}\right] \\ &= E\left[U_k(se^{(r-\delta)(t_k-t)+\sigma\sqrt{t_k-t}Z}\right], \end{aligned}$$

où Z est la loi Normale(0,1). Donc,

$$\psi(t, S_t) = E\left[se^{(r-\delta)(t_k-t)+\sigma\sqrt{t_k-t}Z}U_k'(se^{(r-\delta)(t_k-t)+\sigma\sqrt{t_k-t}Z})\right],$$

Il est donc possible maintenant d'évaluer ψ à l'aide de simulations Monte Carlo, et de pouvoir effectuer la couverture de risque de l'option bermudienne.

Chapitre 3

SIMULATIONS

3.1. Algorithme d'évaluation de l'option

3.1.1. Raisonnement

L'algorithme présenté en premier dans ce chapitre est un algorithme présenté dans l'article de Del Moral et al. (2006) servant à déterminer la valeur des options de type bermudienne. Ainsi, la version du problème de Snell résolue au chapitre 1 nous démontre un résultat nous permettant de conclure qu'il existe une solution optimale et en même temps nous donne une piste pour trouver cette solution à l'aide d'une simulation. L'algorithme présenté ici tente de trouver une solution optimale. Les sections suivantes du chapitre décriront les algorithmes de "borne supérieure" et de "borne inférieure" associés à l'évaluation d'options dans l'article d'Andersen et Broadie (2004) ; deux procédures qui utiliserons des données calculées par l'algorithme de l'article de Del Moral et al. (2006).

En regardant les fonctions exprimant la valeur de certaines options, nous remarquons que celles-ci possèdent des propriétés de convexité. Cependant, les méthodes de simulation utilisées précédemment ne conservaient pas ces propriétés durant la simulation. L'algorithme étudié ici a pour but de conserver cette propriété de convexité tout au long du calcul, c'est-à-dire de la simulation.

Évidemment, l'algorithme proposé utilise les fondements théoriques du problème de Snell afin d'évaluer la valeur de l'option ainsi que la région d'exercice. Plus précisément, il utilise la méthode de résolution qui débute au terme de l'option et remonte le temps jusqu'au moment initial. À la différence des autres types

de simulation, cette simulation ne génère pas des chemins pour les valeurs du sous-jacent. Cette simulation fait plutôt des calculs pour plusieurs points appartenant à une grille et se sert de l'interpolation pour obtenir les résultats aux autres points désirés (autres combinaisons de valeurs des différents sous-jacents). C'est cette dernière propriété de la procédure qui permet de conserver la convexité de la fonction de la valeur de l'option, car il peut être démontré qu'en utilisant une interpolation, il est possible de conserver la convexité de l'enveloppe de Snell.

L'idée générale derrière cette simulation est tout d'abord de commencer par évaluer U_n au temps final et ce, pour plusieurs valeurs de l'option à ce moment précis. Par la suite, on remonte le temps en calculant U_{n-1} à l'aide de simulations Monte Carlo, encore une fois pour des valeurs fixes de l'option à ce moment. Nos simulations Monte Carlo nous aident à estimer la valeur de l'option au temps $t-1$ et ce, à l'aide d'une interpolation faite avec les valeurs déjà calculées au temps t. On remonte ainsi le temps jusqu'au temps 0 pour avoir la valeur initiale de l'option. Deux exemples d'algorithme évaluant la valeur de l'option sont présentés à l'annexe A. Dans ces deux exemples la simulation a été écrite à l'aide du langage C.

3.1.2. Hypothèses

Au préalable, il est primordial d'énoncer quelques hypothèses avant de se lancer dans la description d'algorithmes. Tout d'abord, on se replace dans le cas markovien discuté à la fin du chapitre 1. On suppose donc que $X_t = f_t(S_t)$, où la chaîne de Markov $(S_t)_{t\geq 0}$ prend des valeurs dans le sous-ensemble convexe $\mathcal{X} \in [0,\infty)^d$. L'hypothèse suivante est reliée aux lois des chaînes de Markov.

Hypothèse 3.1.1. *Pour tous* $1 \leq t \leq n$, *la chaîne de Markov* S_t *peut être exprimée comme suit,*

$$S_t = \pi_t(S_{t-1}, Y_t), Y_t \in \mathcal{Y},$$

où Y_t *suit la loi* μ_t *et est indépendant de* $\mathcal{F}_{t-1}$. *De plus,* $s \longmapsto \pi_t(s,y)$ *est continue sur* $\mathcal{X}$ *pour tous* $y \in \mathcal{Y}$ *fixés.*

Comme l'algorithme proposé utilise grandement les simulations Monte

Carlo, il est nécessaire de faire la prochaine hypothèse afin de s'assurer que l'enveloppe de Snell soit bien définie.

Hypothèse 3.1.2. *Pour tous* $t \in \{1, 2, ..., n\}$, $f_t(S_t)$ *est intégrable.*

3.1.3. Propriétés de U

Avant d'énoncer des propositions sur les propriétés de U, commençons par définir un autre processus. Pour un certain $t \in \{0, 1, ..., n-1\}$, V_t est défini comme suit,

$$V_t(s) = E[U_{t+1}(S_{t+1})|S_t = s].$$

Pour le même t, c'est-à-dire au même moment

$$U_t(s) = \max\{f_t(s), V_t(s)\},$$

et lorsque $t = n$, $f_n = U_n = V_n$. En termes plus concrets, U est l'enveloppe de Snell, f est la fonction de la valeur de l'option et V est la limite de la région d'exercice. La limite de la région d'exercice est en quelque sorte une limite inférieure à partir de laquelle on est mieux d'exercer l'option, c'est-à-dire que si la valeur de l'option f est supérieure à V , alors il est théoriquement plus profitable d'exercer l'option.

Les trois propositions suivantes sont des résultats sur les propriétés de U et de V. Les preuves des ces propositions sont présentées dans l'article de Del Moral et al. (2006).

Proposition 3.1.1. *Si pour tous les* $t \in \{0, ..., n\}$, f_t *est croissante (resp. décroissante) et* $\pi_t(\cdot, y)$ *est croissante (resp. décroissante) pour n'importe quel* $y \in \mathcal{Y}$, *alors* U_t *et* V_t *sont eux aussi croissantes (resp. décroissantes).*

Aussi, si f_t *est continue pour tout* $t \in \{0, ..., n\}$, *alors* U_t *et* V_t *seront eux aussi continues pour tous les* $t \in \{0, ..., n\}$.

Dans le cas où l'hypothèse sur π_t de la proposition précédente ne peut être respectée, il est nécessaire d'émettre d'autres hypothèses afin de pouvoir assurer la continuité de U et de V.

Proposition 3.1.2. *Si l'hypothèse 3.1.2 est respectée et que pour tous* $0 \leq j \leq t \leq n$, $s \mapsto E[f_t(S_t)|S_j = s]$ *et* $s \mapsto f_k(s)$ *sont continues, alors* U_t *et* V_t *seront aussi continues pour tous les* $t \in \{0, ..., n\}$

Proposition 3.1.3. *Si pour tous les* $t \in \{0, ..., n\}$, f_t *est convexe et croissante, et de plus* $\pi_t(\cdot, y)$ *est convexe et croissante pour n'importe quel* $y \in \mathcal{Y}$, *alors* U_t *et* V_t *seront convexes et continues pour tous les* $t \in \{0, ..., n\}$.

Si f_t *est convexe et décroissante et si* $\pi_t(\cdot, y)$ *est concave et croissante pour n'importe quel* $y \in \mathcal{Y}$ *et* $t \in \{0, ..., n\}$, *alors* U_t *et* V_t *seront convexes et décroissantes pour tous les* $k \in \{0, ..., n\}$

3.1.4. Algorithme

Afin de pouvoir réaliser cette procédure, il est possible de suivre les étapes de l'algorithme suivant, mais auparavant il est nécessaire d'introduire la définition suivante.

Définition 3.1.1. *Dans ce qui suit, nous allons définir le mot "Grille" comme étant un ensemble de points dans l'espace des valeurs possibles des sous-jacents. Plus précisément, chaque sous-jacent correspond à une dimension, et les points de la "Grille" ont comme dimension, le nombre de sous-jacents.*

$$Grille := \left\{ \left(x^1, \ldots, x^d\right) \mid x^i \in \left\{s_1^i, \ldots, s_{N_i}^i\right\} \text{ où les } s \text{ sont} \in \mathbb{R} \right\}$$

L'algorithme est :

- On pose $\tilde{V}_n(x) = \tilde{U}_n(x) = f_n(x)\ \forall x \in$ Grille au temps n.
- Pour $t = n, ..., 1$, on génère $\xi_1, ..., \xi_{N_t}$ selon la loi μ_t, et à chaque $x \in$ Grille, on calcule
  - $a_{t,i} = \tilde{V}_t(\pi_t(x, \xi_i))\ \forall i = 1, ..., N_t$
    (on obtient $\tilde{V}_t$ en interpolant)
  - $B_{t,i} = f_t(\pi_t(x, \xi_i))\ \forall i = 1, ..., N_t$
  - On pose

$$\tilde{V}_{t-1}(x) = \frac{1}{N_t} \sum_{i=1}^{N_t} \tilde{U}_t(\pi_t(x, \xi_i)) = \frac{1}{N_t} \sum_{i=1}^{N_t} \max(a_{t,i}, B_{t,i}).$$

Notre simulation se termine ainsi. Parmi nos résultats, nous avons les régions d'exercice aux différents temps d'exercice (voir remarque ci-bas) et par le fait

même la valeur de l'option au temps 0 pour les différentes valeurs de départs qui sont dans la grille.

Remarque 3.1.1. *En calculant $\tilde{V}$ dans cette procédure, on calcule les limites inférieures de la région d'exercice. La région d'exercice est donc toutes les valeurs qui sont supérieures à V. Plus précisément, la région d'exercice $\tilde{\mathcal{E}}$ est donnée par*

$$\tilde{\mathcal{E}} = \{(t, s) | \tilde{V}_t(s) \leq f_t(s)\}.$$

Remarque 3.1.2. *Cette simulation peut être faite pour des options à un ou plusieurs sous-jacents. Il suffit de considérer x comme un vecteur à plusieurs dimensions. Ainsi, la grille sera comprise de points de même dimension que le vecteur x.*

3.1.5. Interpolation convexe

L'interpolation utilisée dans l'algorithme précédant doit pouvoir conserver les propriétés de convexité et monotonicité. Les prochaines définitions et le prochain lemme seront cités de l'article de Del Moral et al. (2006), t serviront à encadrer l'interpolation et à préserver les deux propriétés mentionnées.

Définition 3.1.2. *Une partition P d'un ensemble convexe et compacte R, est n'importe quel ensemble fini de sous-ensembles non-vides $S_1, ..., S_n$ et tous disjoints deux à deux, tel que $R = \bigcup_{i=1}^{n} S_i$.*

Définition 3.1.3. *Soit une fonction convexe g et une partition P de R, alors une interpolation linéaire de g sur P est une fonction unique $\tilde{g}$ qui est définie de la façon suivante :*

- *Si $x \in S \subset R$, où S est l'ensemble des points $x_1, ..., x_n$,*

$$\tilde{g}(x) = \sum_{i=1}^{n} \lambda_i g(x_i),$$

  où les barycentres $\lambda_1, ..., \lambda_n$ sont la solution unique aux équations suivantes :

$$\sum_{i=1}^{n} \lambda_i x_i = x_i, \sum_{i=1}^{n} \lambda_i = 1, \lambda_i \in [0, 1].$$

- *Si $x \notin R$, on définie x_R comme étant le point de R étant le plus près de x, et on pose $\tilde{g}(x) = \tilde{g}(x)$.*

Définition 3.1.4. *Soit une fonction convexe g et une partition P de R, alors une interpolation linéaire convexe de g sur P est une fonction $\tilde{g}$ qui respecte les propriétés suivantes :*

- *o $\tilde{g}(x)$ est convexe sur R,*
- *o $g(x) = \tilde{g}(x)$, pour tout x compris dans l'ensemble des points de P,*
- *o $\forall x \in R$, il existe des $\lambda_z \geq 0$, où z est compris dans l'ensemble des points de P, qui satisfont les critères suivant :*
  - *– $\sum_z \lambda_z = 1$,*
  - *– $x = \sum_z z\lambda_z$,*
  - *– $\tilde{g}(x) = \sum_z \lambda_z g(z)$;*
- *o Pour tout $x \notin R$, $\tilde{g}(x) = \tilde{g}(y)$ où y est l'unique point tel que la distance entre x et R est égale à la distance entre x et y ($d(x, R) = d(x, y)$).*

Le lemme suivant énonce qu'une interpolation linéaire convexe existe et peux être obtenue à l'aide de la méthode du simplexe et que cette interpolation préserve aussi la même monotonicité que la fonction de départ.

Lemme 3.1.1. *Supposons que R est l'enveloppe convexe de $x_1, ..., x_n$ et que $g_1, ..., g_n$ sont des nombres réels. Pour tout $x \in C$, définissons*

$$\Lambda_x = \left\{ \lambda \in [0,1]^n ; \sum_{i=1}^{n} \lambda_i x_i = x_i, \sum_{i=1}^{n} \lambda_i = 1 \right\}.$$

Définissons $\tilde{g}$ comme suit :

$$\tilde{g}(x) = \inf_{\lambda \in \Lambda_x} \sum_{i=1}^{n} \lambda_i g_i.$$

Ceci entraîne que $\tilde{g}$ est convexe sur R et que $\tilde{g}(x_i) \leq g_i$. De plus, si g est convexe sur R tel que $g(x_i) = g_i$, alors $\tilde{g}$ est une interpolation linéaire convexe de g.
Pour ce qui est de la monotonicité, si g est monotone, et bien $\tilde{g}$ à la même monotonicité. Plus précisément, si g est non-décroissant en x_i alors $\tilde{g}$ a la même propriété pour tout les $i \in 1, 2, ..., n$.

La démonstration de ce lemme est présente dans l'annexe C de l'article de Del Moral et al. (2006).

3.1.6. Implantation

Pour l'implantation de cet algorithme, voici quelques éléments à considérer :

– Comment choisir le domaine de la grille de points ?

– Comment réaliser l'interpolation de V ?

3.1.6.1. *Domaine de la grille*

Il est bien important de définir le domaine de la grille pour avoir une bonne précision dans notre résultat et afin d'optimiser le temps de calcul. On ne voudrait pas avoir une grille qui serait trop petite et qui ne contiendrait qu'un faible pourcentage des points ξ générés au cours de notre simulation, car ceci nuirait à la précision de notre résultat. De façon opposée, une grille trop vaste augmenterait notre temps de calcul. Une méthode suggérée pour contrer ces effets indésirables est de calculer un intervalle pour lequel le prix du ou des sous-jacents serait à 95% durant la période du contrat, et ainsi ces bornes deviendraient les bornes de notre grille.

En ayant les bornes de la grille, on peut séparer chaque dimension en N intervalles de même longueur pour ainsi obtenir une grille, par exemple, de N^2 points pour une option de deux sous-jacents.

3.1.6.2. *Interpolation*

Il existe bien des interpolations qui conservent la convexité globale d'une fonction convexe ; cependant ce que nous allons utiliser ici est un algorithme qui conserve la convexité locale seulement afin de réduire le temps de calcul. Pour en savoir plus sur les preuves d'existence, de convexité et de monotonicité d'une telle interpolation, il est recommandé de consulter la section précédente et l'annexe C de l'article écrit par Del Moral et al. (2006).

Supposons que notre fonction g est définie dans le rectangle $R = [0, 1]^n$ et que la fonction g est connue à tous les sommets de ce rectangle. Ce que nous cherchons à interpoler est la valeur de la fonction g en un point $x = (x_1, x_2, ..., x_n) \in R$ à l'aide des valeurs aux points connus du rectangle.

Voici la procédure

- Faire une permutation π des éléments du vecteur x, de telle façon qu'ils soient placés en ordre croissant $x_{\pi_1} \leq x_{\pi_2} \leq \ldots \leq x_{\pi_n}$. Notons par P la matrice associée à cette permutation.
- Poser $y = Px$.
- Calculer $\lambda_1 = y_1$ et $\lambda_{n+1} = y_1$ et pour $i \in \{2, 3, ..., d\}$, $\lambda_i = y_i - y_{i-1}$.
- Calculer la valeur finale de l'interpolation $g(x) = \sum_{j=1}^{d} \lambda_j g(P^T u_j) + \lambda_{d+1} g(0)$.

Dans la simulation, la majorité des interpolations ne se font pas à l'intérieur d'un rectangle dont les extrémités sont soient 0 ou 1. Alors, voici comment appliquer la procédure à des rectangles de la forme $\mathrm{R} = \prod_{i=1}^{d} [a_i, b_i]$. Avant de commencer l'interpolation, il faut calculer les $x^{'}$ qui seront égaux à $x'_j = (x_j - a_j)/(b_j - a_j)$ pour tous les $j = 1, ..., d$ et ensuite faire les trois premières étapes.

Pour la dernière étape, il faut modifier $g(P^T u_j)$ par $g(a + DP^T u_j)$ et $g(0)$ par $g(a)$ où a est le vecteur des a_i et D est la matrice diagonale $b - a$ (où b est le vecteur des b_i).

3.2. Algorithme de la "borne inférieure"

3.2.1. Raisonnement

L'algorithme pour la borne inférieure décrit dans Andersen et Broadie (2004)est en fait très simple. Il suffit simplement de se donner un temps d'arrêt quelconque et de simuler un chemin pour la valeur du ou des sous-jacents jusqu'au moment ou ce temps d'arrêt nous dit d'arrêter, ce qui représente dans notre cas le premier moment tel que la valeur de l'option est dans la région d'exercice $\mathcal{E}$. Il ne nous reste plus qu'à calculer la valeur associée à ce temps d'arrêt.

3.2.2. Algorithme

- Pour chaque $i = 1, ..., N_0$, simuler un chemin $S_i = (S_{i,0}, ..., S_{i,n})$, où $S_{i,0} = S_0$ $\forall i$.

- Définir le moment où S_i doit arrêter en fonction du temps d'arrêt suivant

$$\tau_{\mathcal{E},i} = \min\{j \geq 0; (j, S_{i,j}) \in \mathcal{E}\}$$

– Calculer la borne inférieure (BI)

$$BI = \frac{1}{N_0} \sum_{i=1}^{N_0} f_{\tau_{\mathcal{E},i}}(S_{i,\tau_{\mathcal{E},i}}).$$

Remarque 3.2.1. *La région d'exercice $\mathcal{E}$ utilisée dans le critère de ce temps d'arrêt peut provenir de la simulation précédente, dans laquelle, en plus d'évaluer la valeur de l'option, nous avions calculé en plus la région d'exercice $\tilde{\mathcal{E}}$.*

3.3. Algorithme de la "borne supérieure"

3.3.1. Raisonnement

Les fondements théoriques de l'algorithme suivant permettant de trouver une borne supérieure à $U_0(S_0)$ proviennent d'un résultat démontré dans dans l'article de Rogers (2002), qui stipule que pour toute martingale M,

$$U_0(S_0) \leq M_0 + E\left[\max_{0\leq t\leq n} f_t(S_t) - M_t\right].$$

Il y a égalité dans le cas où M est la martingale associée avec la décomposition de Doob-Meyer de la supermartingale $U_t(S_t)$.

Par la suite est venu le résultat de Andersen et Broadie (2004) qui suggère une méthode pour construire une martingale M se rapprochant de la décomposition de Doob-Meyer et ce basée sur les régions d'exercices. Ce sont les méthodes de l'article de Andersen et Broadie (2004) qui seront présentées ici afin de simuler la borne supérieure. Pour tout $t \in \{0, \ldots, n\}$, l_t est la fonction indicatrice de l'ensemble $\{(t, S_t) \in \epsilon\}$, c'est-à-dire l_t est égale à 1 si notre stratégie nous dit d'exercer notre option à ce moment et est égale à 0 si, selon notre stratégie toujours, nous devons continuer. Dans ce qui suit, le temps d'arrêt est $\tau_{\mathcal{E},t} = \min\{j \leq t; (j, S_t) \in \mathcal{E}\}$ et on définit $L_t = E\left(Z_{\tau_{\epsilon,t}}|\mathcal{F}_t\right)$ où $Z_t = f_t(S_t)$. Comme pour la "borne inférieur", pour définir la région d'exercice $\mathcal{E}$, on peut utiliser les limites V calculées dans l'algorithme de Del Moral et al. (2006).

La simulation nous permettra de calculer les éléments de l'équation suivante,

$$M_t = M_{t-1} + L_t - L_{t-1} - l_{t-1}E\left(L_t - L_{t-1}|\mathcal{F}_{t-1}\right) \tag{3.3.1}$$

pour $t = 1, ..., n$ et dans le cas de $t = 0$, $M_0 = L_0 = U_0(X_0)$. Par la suite, la borne supérieure (BS) sera définie par

$$BS = L_0 + E_0 \left[\max_{t \in \mathcal{T}} (f_t(S_t) - M_t) \right] \tag{3.3.2}$$

3.3.2. Algorithme

(1) Pour chaque $i = 1, ..., N_0$, simuler un chemin $S_i = (S_{i,0}, ..., S_{i,n})$, où $S_{i,0} = S_0 \ \forall i$.

(2) À chaque étape du chemin simulé en (1), vérifier s'il faut exercer l'option en fonction de la stratégie d'exercice spécifiée précédemment, c'est à dire $l_t = 1$. Si tel est le cas, faire l'étape 2(b). Dans l'autre cas, où l_t=0 (la stratégie indique qu'il ne faut pas exercer l'option), suivre l'étape 2(a).

(a) ($l_t = 0$) Débuter une simulation pour évaluer

$$L_k = E \left[f_{\tau_t}(S_{\tau_t}) | \mathcal{F}_t \right].$$

- Pour chaque $j = 1, ..., N_1$, simuler un chemin $S_j = (S_{j,t}, ..., S_{j,n})$, où $S_{j,t} = S_t \forall j$ que vous arrêtez en fonction du temps d'arrêt τ_t.
- Calculer $L_t = \frac{1}{N_1} \sum_{j=1}^{N_1} f_{\tau_t}(S_{\tau_t})$.

(b) ($l_t = 1$) Comme on doit exercer l'option à ce moment, $L_t = f_t(S_t)$. On doit aussi calculer $E(L_{t+1} | \mathcal{F}_t)$ afin de pouvoir connaître la dernière partie de l'équation 3.3.1, c'est à dire

$$l_t E(L_{t+1} - L_t | \mathcal{F}_t).$$

- Pour chaque $j = 1, ..., N_2$, simuler un chemin $S_j = (S_{j,t}, ..., S_{j,n})$, où $S_{j,t} = S_t \forall j$ que vous arrêtez en fonction du temps d'arrêt τ_{t+1}.
- Calculer $L_{t+1} = \frac{1}{N_2} \sum_{j=1}^{N_1} f_{\tau_{k+1}}(S_{\tau_{t+1}})$.

(3) À l'aide des étapes précédentes et de l'équation 3.3.1, on peut maintenant évaluer $M_{i,0}$ à $M_{i,n}$ $\forall i = 1, ..., N_0$ et calculer

$$\frac{1}{N_0} \sum_{i=1}^{N_0} \left(\max_t \{ f_t(S_t) - M_{i,t} \} \right).$$

(4) La borne supérieure (BS) sera donc égale à

$$BS = L_0 + \frac{1}{N_0} \sum_{i=1}^{N_0} \left(\max_t \{ f_t(S_t) - M_{i,t} \} \right).$$

Chapitre 4

RÉSULTATS NUMÉRIQUES

4.1. Contexte

Quelques résultats numériques seront présentés dans les sections suivantes pour l'évaluation d'une option à un et deux sous-jacents. Dans le cas de l'option à un sous-jacent, il y aura des résultats sur la valeur même de l'option, ainsi que des résultats de bornes inférieures et supérieures . Pour ce qui est de l'option à deux sous-jacents, nous avons évalué deux types d'option, c'est-à-dire deux fonctions représentant des prix d'exercice différents.

4.1.1. Résultats pour une option à un sous-jacent

Pour cet exemple, le prix des sous-jacents satisfait l'équation **??** avec les paramètres utilisés dans Anderson et Broadie (2004) afin de pouvoir faire des comparaisons. Les paramètres utilisés sont r = 0.05, $\delta = 0.1$, $\sigma = 0.2$, le temps du contrat est de trois ans et le prix initial du sous-jacent, tout comme le prix de levée, est de 100$. L'option est une option d'achat sur le sous-jacent ayant un prix d'exercice égale à $f(S) = max(S - K, 0)$.

Les calculs sont faits en supposant que la valeur cherchée du sous-jacent au moment initial est égale au prix de levée, soit à 100$. Ainsi, notre grille de points peut se situer entre 23$ et 230$ car la probabilité que la valeur du sous-jacent soit en dehors de ces bornes au terme du contrat est petite. En effet

TABLEAU 4.1. mettre un titre

Résultats pour un sous-jacent et n =2

Résultat	Algorithme	Arbre binomial
Valeur de l'option	7.1776	7.18
Borne inférieure	7.1779 ($\sigma = 0,1363$)	-
Borne supérieure	7.1784 ($\sigma = 0,2256$)	-

$P[S_T \in Grille | S_0 = 100] \geq 0.997$, ce qui peut être vérifié à l'aide de l'équation **??**.

Pour interpréter les résultats de la borne inférieure et de la borne supérieure, il est aussi important de prendre en considération l'écart-type de ceux-ci, soit σ_{BI} et σ_{BS}. À l'aide ces valeurs, il sera possible d'obtenir un intervalle de confiance de 95% pour la valeur de U_0. On peut calculer l'intervalle à l'aide de la formule suivante :

$$\left[BI - 1.96\frac{\sigma_{BI}}{\sqrt{N_{BI}}}, BS + 1.96\frac{\sigma_{BS}}{\sqrt{N_{BS}}} \right],$$

où N_{BI} et N_{BS} représentent respectivement le nombre d'itérations utilisé pour le calcul de la borne inférieure et de la borne supérieure.

Les tableaux suivants représentent les résultats obtenus avec n = 2 et n = 10 périodes d'exercices distribuées également lors des trois années du contrat. Il y a les résultats de l'algorithme présenté dans cet article ainsi que les résultats de la méthode binomiale dans l'article de Andersen et Broadie (2004) et ce, pour la valeur de l'option.

Remarque 4.1.1. *On peut voir qu'il est possible que la valeur de la borne inférieure soit plus grande que celle de la valeur de la borne supérieure. Malgré le fait que dans l'article d'Andersen et Broadie (2004) on évoque que de tels résultats ne soit pas possible, l'article de Del Moral et al. (2006) fait la démonstration que l'argument évoqué par Andersen et Broadie (2004) ne peut pas tenir.*

TABLEAU 4.2. mettre un titre

Résultats pour un sous-jacent et n =10

Résultat	Algorithme	Arbre binomial
Valeur de l'option	7,9794	7,98
Borne inférieure	7,9939 ($\sigma = 0,000$)	-
Borne supérieure	7,9869 ($\sigma = 0,000$)	-

À l'aide de ces résultats, l'intervalle de confiance devient [7.1695; 7.1924] pour l'option à deux temps d'exercice et de [;] pour l'option à dix temps d'exercice. Comme les algorithmes de la borne inférieure et de la borne supérieure ajoute à la précision du résultat, il est possible de voir le résultat de l'algorithme principale de ce mémoire en dehors des bornes de l'intervalle de confiance.

4.1.2. Résultats pour une option à deux sous-jacents

Pour les simulations avec deux sous-jacents, les paramètres de l'équation **??** sont les mêmes que dans la simulation précédente, et ce, pour les deux sous-jacents. Il y aura des résultats pour deux types d'option d'achat, soit le premier type qui est l'option "call-on-max" où la fonction du prix d'exercice est $f(S) = max(max(S_1; S_2) - K, 0)$, et le deuxième type d'option ayant comme fonction $f(S) = max\left(\frac{S_1+S_2}{2} - K; 0\right)$.

Les calculs ont été faits en supposant que les valeurs initiales des sous-jacents étaient égales au prix de levée soit à 100$ encore une fois, mais aussi à 90$ et à 110$. Ainsi, notre grille de points peut se situer encore une fois entre 23$ et 230$ pour chaque sous-jacent car la probabilité que la valeur d'un sous-jacent soit en dehors de ces bornes au terme du contrat est très mince.

Le tableau suivant présente les résultats obtenus avec n=9 périodes d'exercices distribuées également lors des trois années du contrat. On y présente les résultats de l'algorithme présenté dans ce mémoire avec la valeur de l'option obtenue avec la méthode d'arbre binomial (résultats provenant de Andersen et Broadie (2004)). Cependant, comme pour la fonction $f(X) = max\left(\frac{X^1+X^2}{2} - K, 0\right)$, il n'y a pas

d'autres résultats dans Andersen et Broadie (2004), seulement les résultats de la méthode Monte Carlo de cet article seront présentés.

Résultats pour deux sous-jacents et n = 9

Résultat S_0 =90	Algorithme	Arbre binomial
Valeur de l'option 1	8,1346	8,075
Valeur de l'option 2	2,0289	-

Résultats pour deux sous-jacents et n = 9

Résultat S_0 =100	Algorithme	Arbre binomial
Valeur de l'option 1	13,9662	13,902
Valeur de l'option 2	4,9752	-

Résultats pour deux sous-jacents et n = 9

Résultat S_0 =110	Algorithme	Arbre binomial
Valeur de l'option 1	21,4042	21.345
Valeur de l'option 2	10,3409	-

4.2. Région d'exercice

Dans le cadre de ce mémoire, nous avons discuté d'interpolations préservant les propriétés de convexité et de monotonicité. Voici donc les graphiques des valeurs de U_0 associées aux différentes simulations effectuées. Les deux premiers graphiques représentent les résultats au temps 0 pour les options à un sous-jacent et avec une fonction pour le prix d'exercice $= \max(0; S - k)$, soit un pour le cas où il y a seulement deux temps d'exercice possible et l'autre lorsque qu'il y en a 10.

On présente ici la région d'exercice, de l'option bermudienne à 10 temps d'exercices, utilisée dans le calcul de la borne supérieure et de la borne inférieure. Cette région est calculée au cours de la simulation Monte Carlo pour obtenir la valeur de U_0. Le tableau donne la valeur minimale, à chaque temps d'exercice possible, à laquelle commence la région d'exercice. Toutes les valeurs qui sont supérieures

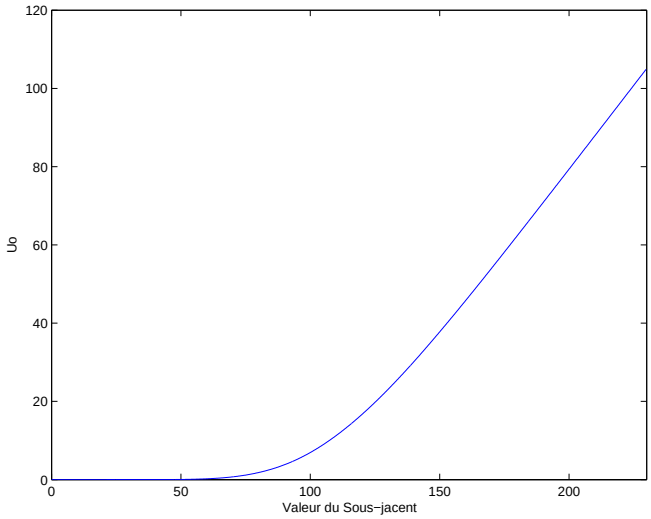


FIGURE 4.1. Valeur d'une option bermudienne d'achat au temps $t = 0$ ayant 2 périodes d'exercice.

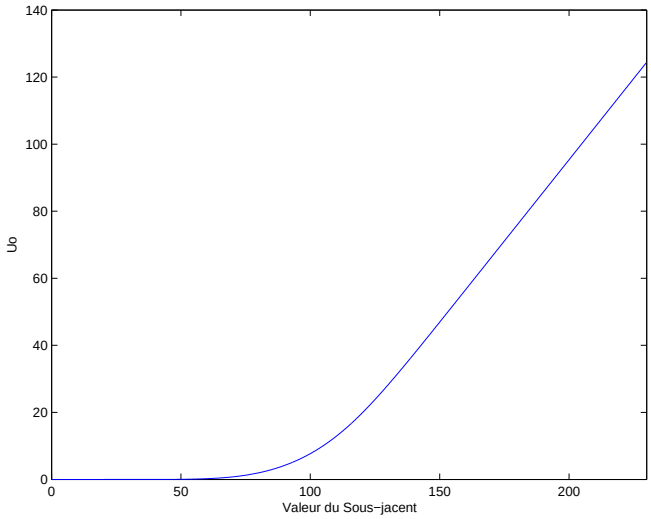


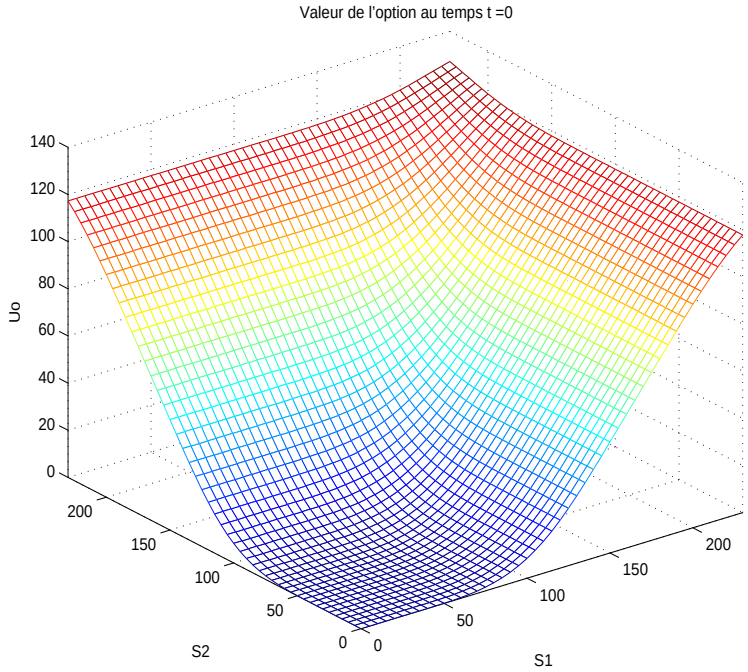
FIGURE 4.2. Valeur d'une option bermudienne d'achat au temps $t = 0$ ayant 10 périodes d'exercice.

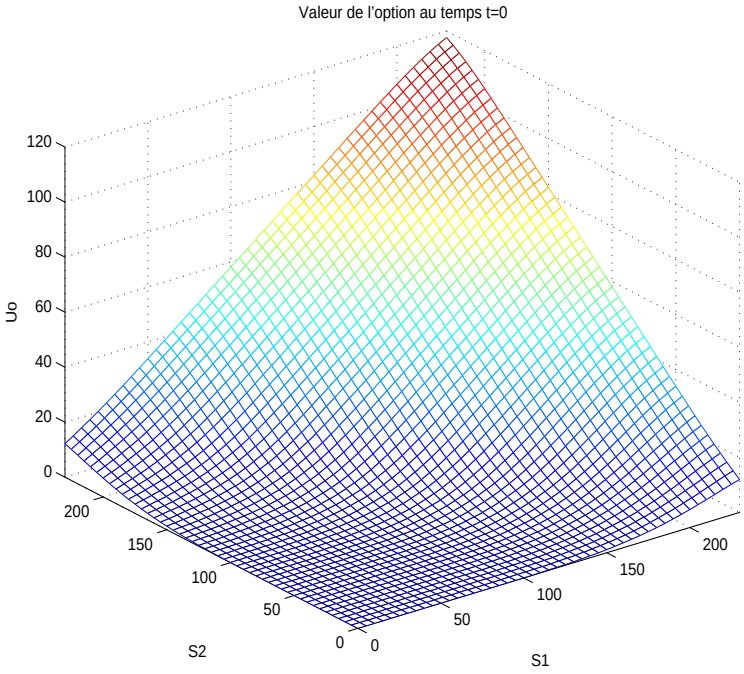
à celles dans le tableau, font partie de la région d'exercice.

Région d'exercice pour un sous-jacent

Temps d'exercice	Valeur minimale pour exercer l'option
0	123
1er	122
2e	122
3e	121
4e	120
5e	119
6e	118
7e	116
8e	114
9e	110
Échéance	101

Pour ce qui est des autres graphiques, ils représentent encore la valeur de U_0, mais cette fois pour des options avec deux sous-jacents. Le premier étant avec la fonction du prix d'exercice $= max(0; max(S_1 - k, S_2 - k))$, et le deuxième avec la fonction $= max(0; (S_1 + S_2/2 - k))$. Il est important de se rappeler que pour des raisons de rapidité des simulations, l'algorithme utilisé ici est un algorithme qui ne préserve que localement la convexité dans chaque rectangle et ne préserve pas la convexité générale, mais que la monotonicité est tout de même préservée.

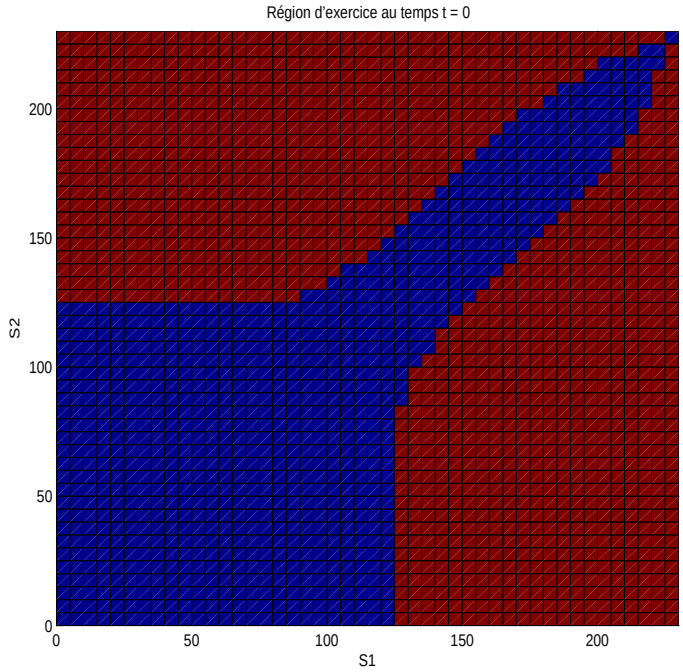


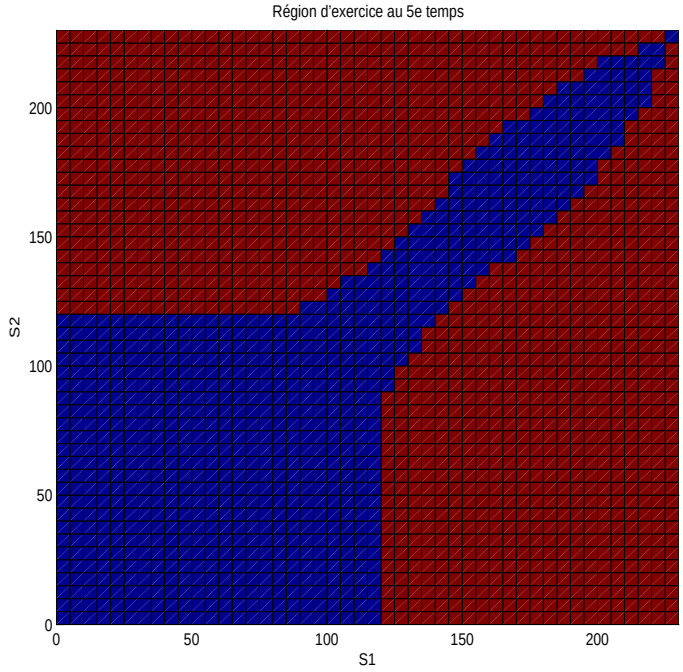


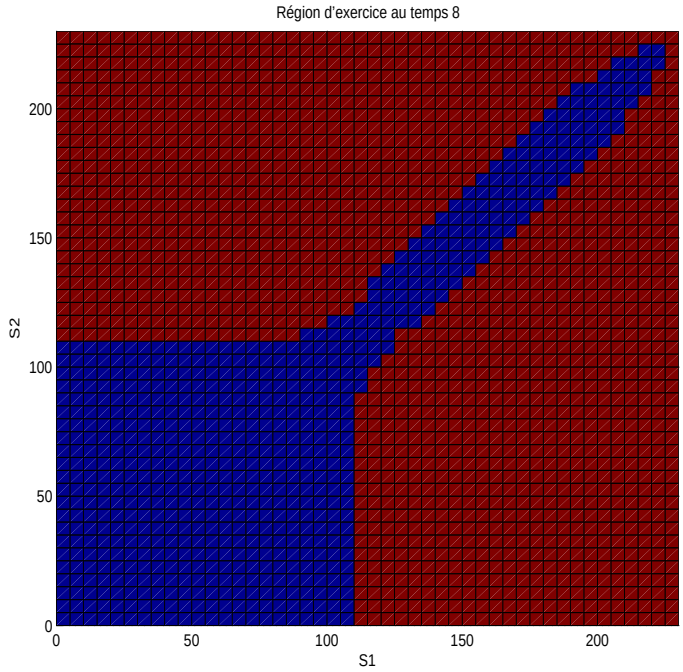
Voici maintenant comment évolue dans le temps la région d'exercice associée à la fonction de "payoff" $= max(0, max(S_1 - k, S_2 - k))$. Si on avait voulu calculer la valeur de la borne inférieure et la borne supérieure pour cette option, il aurait fallu utiliser les données de ces régions afin de savoir quand on aurait

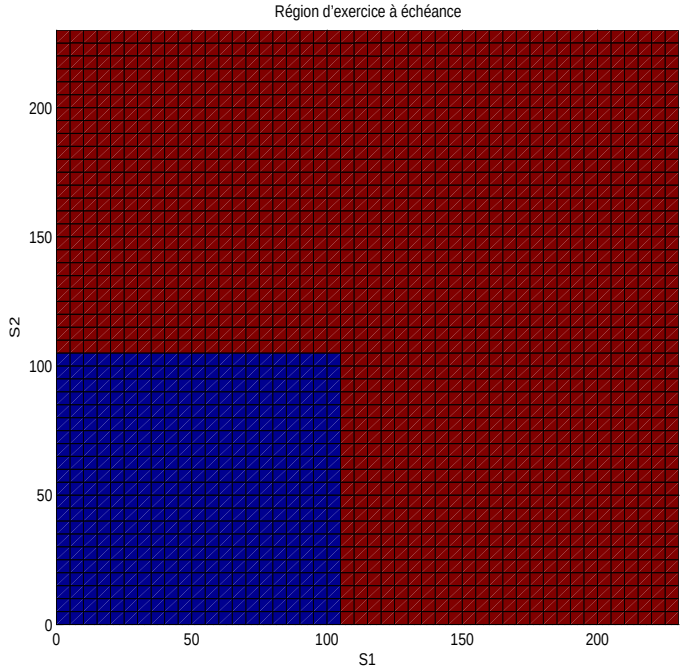
arrêté. La couleur rouge représente la région d'exercice. Les graphiques présentés sont ceux du temps initial, du 5^e, du 8^e et du dernier temps d'exercice possible. Il est à noter que lorsque la valeur de l'option est égale à la fonction représentant le prix d'exercice, le point se retrouve dans la région bleue soit la région où il est théoriquement désavantageux d'exercer notre option.

En se référant aux graphiques qui se retrouvent dans les deux prochaines pages, il est intéressant de noter que la diagonale (la droite S1 = S2), est dans tous les graphiques en grande majorité dans la région bleue jusqu'à l'échéance du contrat. De plus, on peut remarquer que la zone bleue entourant cette diagonale rétrécie lorsque l'on avance dans le temps pour finalement disparaître à la fin du contrat. Évidemment, l e carré où les options ont des valeurs inférieures à 100 est dans la zone bleu de toutes les régions.


Région d'exercice au temps t = 0
S2
S1
0
50
100
150
200


Région d'exercice au 5e temps
S2
S1
0
50
100
150
200


Région d'exercice au temps 8
S2
S1
0
50
100
150
200


Région d'exercice à échéance
S2
S1
0
50
100
150
200

CONCLUSION

Le problème de Snell est un problème central dans le domaine des options financières. Sa résolution a permis à plusieurs chercheurs d'évaluer la valeur d'options américaines et bermudiennes à l'aide de simulations Monte Carlo. Dans ce mémoire, un algorithme basé sur cette résolution a été décrit afin de conserver les propriétés de convexité et de monotonicité que peuvent avoir dans certains cas les options bermudiennes. J'ai par la suite implanté ces algorithmes en langage C pour en ressortir des résultats pratiques. Les résultats obtenus ont même démontré que les valeurs que l'on peut obtenir à l'aide de cet algorithme sont de bons estimateurs pour les options bermudiennes à un ou plusieurs sous-jacents.

Il faut noter que cet algorithme calcule aussi les régions d'exercice tel qu'illustré au chapitre 3, ce qui permet d'avoir la possibilité d'y ajouter d'autres simulations comme celle de la borne inférieure et celle de la borne supérieure présentées dans l'article d'Andersen et Broadie (2004). En plus de l'algorithme principal, j'ai donc programmé l'algorithme des bornes supérieures et inférieures et l'algorithme de couverture de risque dans le même objectif d'avoir des résultats numériques. La résolution du problème de Snell nous a aussi amené à trouver une solution au problème de la couverture de risque des options bermudiennes. Ce qui nous a montré qu'il était possible d'utiliser la méthode de simulation Monte Carlo afin de trouver les valeurs nécessaires à la résolution pratique du problème de couverture de risque.

Annexe A

PROGRAMMES EN C

A.1. Option a un sous-jacent

Algorithme pour évaluer la valeur, la borne inférieure et la borne supérieure d'une option bermudienne à un sous-jacent.

```
#include <math.h>
#include "kiss.c"
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

// la simulation est faite  pour une option venant a échéance dans 3 ans...

#define N 11 //  N -1 est le nombre de période où l'on peut exercer l'option
#define Pi  3.14159265358979323846
#define K 100      // strike price
#define MAX 230  // max que peux atteindre le prix = MAX-1*PAS
#define PAS 1// le pas d'augmentation dans l'echelle de valeur du prix
#define N1 5000// nombre de points aléatoires générés

double normale(){

double U1,U2;

      U1 = kiss(); // kiss est le generateur de nombres aléatoires
      U2 = kiss();
      return cos((double) 2.0*Pi*U1)*sqrt((double)-2.0*log((double)U2));

}// fin de la fct normale

double max(double a, double b){
       if(a>b) return a;
       else return b;}


double payoff(double valeur2, double time){

    if ( valeur2 - K <= 0)
      return 0;
    else

      return (valeur2-K)*exp(-((time*3)/(N-1))*0.05);}

  /* fonction qui genère la valeur suivante du sous-jacent */
 double genere(double valeur, double time,double u){

        double a = valeur*exp( -0.07*((time*3)/(N-1))+ 0.2*sqrt(((time*3)/(N-1)))*u);

        if ( a > 0) return a;
        else return 0;

        }

 /* fonction d'interpolation */

 double Vinter1(double valeur1,double V[MAX]) {

       int i =0;
       int j =0;
       int a1=0,b1=0;

       if( valeur1 < (MAX-1)*PAS)
           {
               while( valeur1 > (i*PAS)  )
               i++;
```

```
              a1= i-1;  b1 = i;
              if(b1==0)
                 return V[0];
              else
                 return V[a1]+ ( ( valeur1-(a1*PAS))/PAS  ) * (V[b1]-V[a1]);
           }
      else
           return V[MAX-1]+ ( valeur1 - ((MAX-1)*PAS))*((V[MAX-1]-V[MAX-2])/PAS) /*0*/ ;
      }
 /* Début de la fonction pricipale */
int main(){
double V[N][MAX];
double borneinf[N];
double gen[N1];
double U2;
double valeur1;
 int b;
double x,y,z,z1;
int g,i,j,test,l,temps,yoyo;
FILE *resultats;


long int *ii;                         // variables utilent pour normale
unsigned long int i0, j00, k0;
time_t *tloc;
int  k;


  /********** Fixation du seed **********/
      ii = (long int *)malloc(sizeof(long int));
      tloc=(time_t*)malloc(sizeof(time_t));
      *ii=time(tloc);
      j00=*ii*rand()*123;
      k0=*ii*rand()*756;
      i0=*ii*rand()*6675;
      kisset(i0,j00,k0);

resultats = fopen( "resultats_1d.txt" , "w" );

for (  i=0; i < MAX; i++) // initialisation de Vn = fn,
         V[N-1][i] = payoff(i*PAS,N-1);

for( l=0; l<N1;l++)
  { gen[l] = normale();}
for( temps=N-2 ; temps >= 0 ; temps--)// Le reste du calcul
    {for ( i=MAX-1; i >= 0; i--)
            {  V[temps][i] = 0;
                    for(  l=0; l<N1;l++)
                         {
                           valeur1 = genere(i*PAS,1,gen[l]);
                         if( valeur1 > MAX*PAS ) valeur1 = MAX*PAS;

                           x= payoff(valeur1,temps+1);
                           y= Vinter1(valeur1,V[temps+1]);
                           U2 = max( x,y);
                           V[temps][i] += U2;

                         }
              V[temps][i] = V[temps][i]/N1;

            }
      }

           /*  bien s'assurer que l'on choisit l'élément du vecteur V que l'on veut */
      fprintf(resultats,"%f",V[0][100]);

      /*   Debut de la simulation pour borne inferieure */
```

```
 borneinf[0] = 100;
 z = 0.0;


for( j=1;j < 10000;j++)
{i=1;
 test = 0;
 while( test == 0 && i< (N-1) ){
                     borneinf[i] = genere(borneinf[i-1],1,normale());

                     if( payoff(borneinf[i],i) >= Vinter1(borneinf[i],V[i]))
                        { test=1;
                          }

                     i++;
                   }

if( i ==(N-1)& test == 0)
         {    borneinf[i] = genere(borneinf[i-1],1,normale());
              z+= payoff(borneinf[i],i);
          }
else      z+= payoff(borneinf[i-1],i-1);
 }

 z = z/10000;
 fprintf(resultats,";%f\n",z);
 //fprintf(resultats ,"; %f",z);
 //scanf(" %d",&b);
 /* Fin de la simulation de la borne inferieure*/
 //
 //
 //
 /* Debut de la simulation de la borne superieur*/

 double pi[N];
 double Lk[N];// esperance conditionnelle a l'etape actuelle
 double Lk2[N];// esperance conditionnelle a l'etape precedante
 double S[N];
 int test1[N];
 double s,ax;
 int p;
 double bornesup = 0;
 for(i=0; i<N;i++)
    { Lk2[i] = 0;
      test1[i] = 0;}
for( p =0; p< 1500; p++)
{ S[0] = 100;
  s=0;
  for(i=0; i<N;i++)
      { pi[i] = 0;
        }
          // On est mieux de continuer, que d'exercer notre option au temps zero
  for( i=0; i<5000; i++)
           {
            s = S[0];
            test =0;
            j=1;
            while( test == 0 && j <10)
                    {s = genere(s,1,normale());
                     if( payoff(s,j) >= Vinter1(s,V[j]))
                         test=1;
                     j++;
                      }
            Lk[0] += payoff(s,j-1);
            }
     Lk[0] = Lk[0]/5000;

  for( i=1 ; i < N; i++)
      { S[i] = genere(S[i-1],1,normale());}
```

```
  for( i=1 ; i < N; i++)
    {  if( payoff(S[i],i) < Vinter1(S[i],V[i]))// si on est mieux de continuer
        {   test1[i] = 0;
            for( k=0; k<5000; k++)
                {
                        s = S[i];
                        test =0;
                        j=i+1;
                        while( test == 0 && j <N)
                            {s = genere(s,1,normale());
                              if( payoff(s,j) >= Vinter1(s,V[j]))
                                        test=1;
                             j++;
                            }
                  Lk[i] += payoff(s,j-1);
                }
            Lk[i] = Lk[i]/5000;
          }
        else   /* si i est un moment où on devrait exercer l'option */
        { Lk[i] = payoff( S[i],i);
          test1[i] = 1;
              if ( i < N-1)
              for( k=0; k<5000; k++)
                {
                        s = S[i];
                        test =0;
                        j=i+1;
                        while( test == 0 && j <10)
                            {s = genere(s,1,normale());
                              if( payoff(s,j) >= Vinter1(s,V[j]))
                                        test=1;
                             j++;
                             }
                  Lk2[i] += payoff(s,j-1);
                 }
              Lk2[i] = Lk2[i]/5000;
        }
    }// fin du for avec i

 pi[0] = Lk[0];
 pi[1] = Lk[1];
 for( i=2 ; i < N; i++)
   {pi[i] = pi[i-1] + Lk[i] - Lk[i-1] - test1[i-1]*(Lk2[i-1]-Lk[i-1]);
    }

s = -100;
 for(i = 0;i<N;i++)
    s = max( s, payoff(S[i],i) - pi[i]);
  bornesup +=s;
}
  bornesup = bornesup/1500;

 fprintf(resultats,";%f\n", bornesup);
 scanf(" %d",&b);
 return 1;
 }
```

A.2. Option à deux sous-jacents

Algorithme pour évaluer la valeur d'une option bermudienne à deux sous-jacents.

```
 #include <math.h>
#include "kiss.c"
#include <stdio.h>
```

```
#include <time.h>
#include <stdlib.h>

#define N 10       //  N-1 = nombre de temps d'exercice possible
#define Pi  3.14159265358979323846
#define K 100      // strike price
#define MAX 115  // max que peux atteindre le prix = MAX*PAS
#define PAS  2// le pas d'augmentation dans l'echelle de valeur du prix
#define N1 5000// nombre de points aléatoires générés

double normale(){

double U1,U2;

      U1 = kiss(); // kiss est le generateur de nombres aléatoires
      U2 = kiss();
      return cos((double) 2.0*Pi*U1)*sqrt((double)-2.0*log((double)U2));

}// fin de la fct normale

double max(double a, double b){
       if(a>b) return a;
       else return b;}


double payoff(double valeur1,double valeur2, double time){

    if ( valeur1 > valeur2)
      valeur2 = valeur1;

    if ( valeur2 - K < 0)
      return 0;
    else

      return (valeur2-K)*exp(-((time*3)/9)*0.05);}

/* fonction qui genère la valeur suivante du sous-jacent */
 double genere(double valeur, double time,double u){
        double a = valeur*exp( -0.07*((time*3)/(N-1))+ 0.2*sqrt(((time*3)/(N-1)))*u);

        if ( a > 0) return a;
        else return 0;}

 double Vinter1(double valeur1,double valeur2,double V[MAX][MAX]) {
       double y[2];
       double lamda[3];
       int i =0,b;
       int j =0;
       int a1=0,a2=0,b1=0,b2=0;

       while( valeur1 > i*PAS)
            i++;
       a1= i-1;  b1 = i;

       while( valeur2 > j*PAS)
           j++;
       a2 = j-1; b2= j;


       if(b1==0)
       {valeur1=0; a1=0;}
       else
       valeur1= (valeur1- a1*PAS)/ ( b1*PAS - a1*PAS);

       if(b2 ==0)
       {valeur2 = 0; a2 =0;}
       else
       valeur2= (valeur2- a2*PAS)/ ( b2*PAS - a2*PAS);

       if( valeur1 > valeur2)
            {

                   y[0] = valeur2; y[1]=valeur1;
            lamda[0]=y[0];  lamda[1]=y[1]-y[0]; lamda[2]= 1-y[1];

            return lamda[0]*V[b1][b2]+lamda[1]*V[b1][a2]+lamda[2]*V[a1][a2];}
       else
            {
                   y[0] = valeur1; y[1]= valeur2;

            lamda[0]=y[0];  lamda[1]=y[1]-y[0]; lamda[2]= 1-y[1];

            return lamda[0]*V[b1][b2]+lamda[1]*V[a1][b2]+lamda[2]*V[a1][a2];}

       }// fin de la premiere fct d'interpolation

  double Vinter3(double valeur1,double valeur2,double V[MAX][MAX]) {
```

```
        double y[2];
        double lamda[3];
        int j =0;
        int a1,a2=0,b1,b2=0;

        a1= MAX-1;  b1 = MAX-1;
        while( valeur2 > j*PAS)
            j++;
        a2 = j-1; b2= j;
        valeur1 =1;
        if(b2 == 0)
        {valeur2 = 0; a2 =0;}
        else
        valeur2= (valeur2- a2*PAS)/ ( b2*PAS - a2*PAS);

             y[0] = valeur2; y[1]=valeur1;
             lamda[0]=y[0];  lamda[1]=y[1]-y[0];
             return lamda[0]*V[b1][b2]+lamda[1]*V[b1][a2];
        }
 double Vinter4(double valeur1,double valeur2,double V[MAX][MAX]) {
        double y[2];
        double lamda[3];
        int i =0;
        int j =0;
        int a1=0,a2=MAX-1,b1=0,b2=MAX-1;
        while( valeur1 > i*PAS)
             i++;
        a1= i-1;  b1 = i;

        if( b1==0)
        {valeur1 =0; a1=0;}
        else
        valeur1= (valeur1- a1*PAS)/ ( b1*PAS - a1*PAS);
        valeur2= 1;

             y[0] = valeur1; y[1]= valeur2;
             lamda[0]=y[0];  lamda[1]=y[1]-y[0];

             return lamda[0]*V[b1][b2]+lamda[1]*V[a1][b2];

        }

 /* fonction qui décide de quelle interpolation faire en fonction de la valeur des sous-jacents */
 double Vtest(double valeur1,double valeur2,double V[MAX][MAX]){
         int temp = MAX-1;
         if(valeur1 >= (temp)*PAS && valeur2 >=(temp)*PAS)
             return V[temp][temp];
         if(valeur1>=(temp)*PAS && valeur2 < (temp)*PAS)

             return Vinter3(valeur1,valeur2,V);
         if(valeur1<(temp)*PAS && valeur2 >= (temp)*PAS)
             return Vinter4(valeur1,valeur2,V);
         else
             return Vinter1(valeur1,valeur2,V);}



int main(){
double V[N][MAX][MAX];
double gen1[N1];
double gen2[N1];

FILE *resultats;
double U2;
double valeur1,valeur2;
 int b;
double x,y;
int g,i,j,l,temps,z,ipas,jpas,temp;
```

```
 long int *ii;                          // variable utile pour normale
 unsigned long int i0, j00, k0;
 time_t *tloc;
  int  k;


  /********** Fixation du seed **********/
      ii = (long int *)malloc(sizeof(long int));
      tloc=(time_t*)malloc(sizeof(time_t));
      *ii=time(tloc);
      j00=*ii*rand()*123;
      k0=*ii*rand()*756;
      i0=*ii*rand()*6675;
      kisset(i0,j00,k0);
resultats = fopen( "resultats_2d.txt" , "w" );

for (  i=0; i < MAX; i++) // initialisation de V_n = f_n,
      for (  j=0 ; j < MAX ; j++)
       { V[N-1][i][j] = payoff(i*PAS,j*PAS,N);


       }
for( l=0; l<N1;l++)
  { gen1[l] = normale();
    gen2[l] = normale();}
for( temps=N-2 ; temps >= 0 ; temps--)// Le reste du calcul
    {for ( i=0; i < MAX; i++)
       for ( j=MAX-1 ; j >= 0 ; j--)
            {  V[temps][i][j] = 0;
                ipas = i*PAS;
                jpas = j*PAS;
                temp = temps+1;
                     for(  l=0; l<N1;l++)
                          {
                           valeur1 = genere(ipas,1,gen1[l]);
                           valeur2 = genere(jpas,1,gen2[l]);
                          /* projection du point sur la grille */
                           if( valeur1 > MAX*PAS ) valeur1 = MAX*PAS;
                           if( valeur2 > MAX*PAS ) valeur2 = MAX*PAS;
                           x= payoff(valeur1,valeur2,temp);
                           y= Vtest(valeur1,valeur2,V[temp]);
                           U2 = max( x,y);


                           V[temps][i][j] += U2;

                         }
              V[temps][i][j] = (V[temps][i][j])/N1;

            }
       }
        /*  bien s'assurer que l'on choisit l'élément du vecteur V que l'on veut */
       fprintf(resultats,"%f;%f;%f\n",V[0][45][45],V[0][50][50],V[0][55][55]);



       scanf(" %d", &b);
       return 1;
        }
```

BIBLIOGRAPHIE

[1] Andersen, L. and Broadie, M. (2004).Primal-dual simulation algorithm for pricing multidimensionnal Americain options.Management Science, 50 :1222-1234.

[2] Billingsley, P. (1995). Probability and Measure. Wiley Inter-Science, 3e édition. New York, NY.

[3] Boyle, P.(1977).Options : A Monte Carlo approach. Journal of Financial Economics, 4 :323-338.

[4] Brennan, M. and Schwartz, E. (1977).The valutation of American put options.Journal of Finance, XXXII :449-462.

[5] Broadie, M. and Glasserman, P. (1997).Pricing American-style securities using simulation.Journal of Economics Dynamics and Control, 21 :1323-1352.

[6] Broadie, M. and Glasserman, P. (2004).A stochastic mesh methode for pricing high-dimensionnal American option. Journal of Computationnal Finance, 7 :35-72.

[7] Carriere, J.(1996). Valuation of the early-exercice price for options using simulations and non-parametric regression.Insurance : Mathematics and Economics, 19 :19-30.

[8] Cox, J., Ross, S. and Rubinstein, M. (1979).Option pricing :A simplified approach.Journal of Financial Economics, 7 :229-263.

[9] Del Moral, P., Rémillard, B., Rubenthaler, S.(2006). Convex Monte Carlo approximations of American options. Technical report, GERAD.

[10] Gauthier, G. (2006) .L'enveloppe de Snell et la tarification de droits contingents de type Américain . Notes de Cours HEC.

[11] Gentle, J. (2006).The Black-Scholes Differential Equation Delta Hedging and Continuous Dividends . http ://mason.gmu.edu/ jgentle/csi779/06f/lect11.pdf.

[12] Longstaff, F., Schwartz, E.(2001). Valuing American options by simulation : A simple least-square approach . The Review of Financial Studies, 14 :113-147.

[13] Neveu, J. (1975). Discrete-parameter Martingales . North Holland, Amsterdam.

[14] Rogers, C.(2002).Monte Carlo valuation of American options. Mathematical Finance, 12 :271-286.

[15] Steele, J.M. (2000). Stochastic Calculus and Financial Applications . Springer-Verlag, New York.

[16] Tilley, J.(1993). Valuing American options in a path simulation model. Transactions of the Society of Actuaries, 45 :83-104.