Master en Informatique et Mathématiques Appliquées à la Finance et à l'Assurance

Ecole Polytech'Nice Sophia Antipolis

Stage encadré :
Mme Francoise BAUDE (OASIS - INRIA Sophia Antipolis)
Mme Mireille BOSSY (OMEGA - INRIA Sophia Antipolis)

Grille de Calcul : Prototypage d'une valorisation d'option européenne et américainne par la méthode de Monte Carlo

Viet Dung DOAN

Rapport du stage de fin d'études

Septembre 2006

Sommaire

1 Remerciements

Je voudrais exprimer mes remerciements sincères en premier lieu à Mme Françoise Baude et Mme Mireille Bossy qui ont dirigées mon travail de recherche, qui m'ont donné des conseils et des aides précieuses pendant la réalisation de mon stage au sein de l'équipe OASIS et OMEGA à l'Inira Sophia Antipolis.
Tous mes sincères remerciements aux Professeurs de l'Ecole Polytech Nice Sophia Antipolis pour les connaissances qu'ils m'ont données pendant mes études à l'EPU
Je remercie les stagiaires, les doctorants de l'équipe OASIS pour leur soutien pendant les cinq mois de mon stage.

2 Introduction

Ce projet s'est déroulé au sein de l'équipe OASIS (Objects actifs, sémantique, Internet et sécurité) de l'INRIA Sophia Antipolis, sous la responsabilité de Mme Francoise Baude. Ce travail a été mené en collaboration avec Mireille Bossyde l'équipe OMEGA, une équipe de recherche dans les méthodes numériques probabilistes, notamment pour les applications en finance.

2.1 L'INRIA : son organisation et quelques chiffres

L'INRIA est une organisation de recherche reconnue à l'échelon européen avec de nombreux partenariats dans le domaine scientifique :
- 3500 personnes
- 450 post-doc, stagiaires.
- 950 doctorants.
- 560 ingénieurs et techniciens.

500 personnes travaillent à l'INRIA Sophia Antipolis.

L'équipe OASIS est une équipe commune entre l'INRIA, le laboratoire I3S CNRS et l'université de Nice Sophia-Antipolis. Ses activités se concentrent sur le calcul distribué, notamment dans les grilles de calcul, et plus spécialement dans le développement de systèmes distribués basés sur la technologie des objets actifs. Le résultat de ces recherches se concrétise notamment par le développement d'une bibliothèque écrite en Java : ProActive. L'équipe OASIS participe actuellement au réseau d'excellence européen sur les grilles et les systèmes Pair-a-Pair, CoreGrid. Par ailleurs, elle participe à une action, ayant pour but d'améliorer la coordination des recherches européennes en matière de grille.

L'équipe de recherche OMEGA est bi localisée à l'INRIA Sophia-Antipolis et à l'INRIA Lorraine. L'équipe a pour objectif de développer et d'analyser les modèles stochastiques et les méthodes numériques probabilistes. Les champs de l'application actuels sont : la finance, la neurobiologie, la cinétique chimique. L'équipe OMEGA collabore avec les institutions financières et les chercheurs dans la finance et l'assurance.

2.2 GRID'5000

La communauté de recherche en France a décidé de se doter d'une grille expérimentale de 5000 processeurs reliés par des réseaux haut débit, référence [2]. L'INRIA est fortement impliqué dans cette opération, GRID'5000, lancée et financée par le ministère de la recherche avec l'INRIA, le CNRS, Renater, plusieurs universités et différents conseils régionaux et généraux. Les grilles informatiques, implémentant le principe de mémoire distribuée, répondent à un double enjeu :
1. Un besoin de ressources (calcul, stockage et communication) de plus en plus grand pour différentes applications : médecine, ingénierie, etc ;
2. Le partage des données, des programmes et des ressources en dépassant les frontières géographiques et administratives pour les membres de communautés virtuelles.

Une architecture aussi ambitieuse que simple est donc disponible pour supporter des calculs intensifs. Cependant son utilisation et sa programmation ne sont pas aisés. Ainsi l'équipe OASIS dévéloppe un environnement de programmation nommé ProActive qui favorise la portabilité, la distribution et la mobilité des calculs.

2.3 ProActive

ProActive est un environnement de développement sur grille se présentant sous la forme d'une bibliothèque Java permettant la programmation de calculs distribués. Il apporte un cadre uniforme garantissant la sécurité et la mobilité des applications. Cette bibliothèque a été développée dans le but de fournir une API complète simplifiant la programmation d'applications distribuées sur

le réseau local, sur un cluster ou sur des grilles. ProActive est construit avec des classes Java standards ce qui permet une portabilité du code sans avoir à modifier la machine virtuelle. La bibliothèque Java RMI est utilisée en tant que couche de transport par défaut.

3 Description du travail proposé

Les applications financières nécessitent de résoudre des problèmes de grande taille (portefeuille composé de plusieurs actifs, évaluations de risque de crédit, contrôle du risque global de la banque ...) aux limites des possibilités des ordinateurs actuels. Le recours au parallélisme est d'ores et déjà introduit dans ce contexte mais son utilisation est loin d'être totalement maîtrisée. Il s'agit donc de définir et développer un logiciel qui exploite le parallelès et la distribution pour des calculs financiers.

3.1 L'existant

Un premier prototype avait été développé par un stagiaire d'été en 2005, permettant de faire des simulations en utilisant un serveur et plusieurs calculateurs (les workers) qui s'occupent des calculs. Un problème de tolérance aux pannes et d'adaptabilité à des grilles de grande taille avait été identifié : un serveur ne peut s'occuper de plusieurs workers. Plus récemment ce travail à été repris par un étudiant du CNAM en stage à Supélec [4] qui collabore avec OASIS et continuait en même temps par notre projet IMAFA. Le travail, réalisé en collaboration à permis les amélisations suivantes : La mise en place d'une méthode distribuée de simulation la comparaison de deux méthodes de parallélisation à travers la simulation (en utilisant la méthode de Monte Carlo) de la volatilité d'un marché d'actions et une nouvelle architecture qui consiste en un serveur gérant plusieurs sous serveurs qui s'occupent de plusieurs workers.

3.2 L'objectif du travail

Le travail est lui-même soutenu par un projet de l'ANR, pour trois ans, notifié en décembre 2005, dans le cadre de l'appel 'Calcul Intensif et Grilles de Calcul', intitulé GCPMF : Grilles de Calcul appliquées aux Problèmes de Mathématiques Financières. Mon travail a necessité de prendre en main les méthodes, outils, techniques, logiciels des membres du projet de l'ANR (particulière le ProActive, l'architecture à plusieurs serveurs et workers de Supélec et des méthodes de valorisation des options de l'équipe OMEGA à l'Inria).

Pour évoluer la pertimence du logiciel de calcul financier sur grille, l'objectif du travail était de pouvoir prendre en compte un produit financier encontournable : les options. On a du considérer plusieurs variantes d'un tel produit, dans le cas la date d'exercice fixée (option européenne) ou non-fixée (option américaine) et leur éxotiques formes comme option à barrière, à double barrière, ou bien option sur panier. Il a donc fallu etendre la plateforme logicielle et de plus, effectuer des benchmarks pour mesurer la puissance réelle d'une telle solution distribuée et en digérer des évolutions futures. Mon travail a porté d'une part, sur un intergiciel pour programmer, puis déployer et exécuter des calculs à grande échelle d'autre part, a consisté à maitriser certains algorithmes de mathématiques financières et les 'gridifier' en vue de leur répartition et parallélisation sur les grilles. Un objectif futur est d'aboutir à une infrastructure logicielle ouverte pour tous les calculs financiers : Evaluation des prix de produit optionnels mais aussi de produit fermes.

Mais déjà, en se penchant sur les différentes formes de l'options européennes, simple, barrière, panier et les options américaines simples, on a pu voir les degrés de compléxité de chacune et comment cela s'entégrer en une bibliothèque de calcul distribué. Puis nous avons fait la comparaison des temps de calculs entre une architecture "mono-processeur" et une architecture "multi-processeurs".

4 Contexte : La plateforme ProActive

ProActive est fourni sous forme d'une API écrite en langage Java. Etant écrit dans un langage portable et orienté objet, ProActive peut être utilisé dans un environnement de machines hétérogènes. Un developpeur de systèmes distribuées devra juste connaitre les concepts de ProActive et pourra les réutiliser dans n'importe quel contexte de machines, référence [1] pour la partie de ProActive.

4.1 Les objets actifs

ProActive repose sur le concept d'objet actif. Un objet actif est un objet Java accessible à distance, tout autre objet passif (normal) ou actif peut donc faire appel à ses méthodes comme s'il se trouvait sur la même machine. Cet objet actif possède son propre thread et sa file d'exécution de requêtes. Lorsqu'un programme fait un appel de méthode sur un objet actif renvoyant un objet en retour, l'application n'est pas bloquée jusqu'à l'arrivée de la réponse, car un objet futur est envoyé tout de suite, et le résultat futur sera remplacé par l'objet attendu lorsqu'il sera disponible. Si le programme tente d'accéder à la valeur de l'objet futur (avant que sa vraie valeur ne soit connue), il sera bloqué. Ce mécanisme, appelé wait-by-necessity, permet au programme de se poursuivre jusqu'au moment où il aura réellement besoin du résultat, c'est une communication asynchrone entre objets. Le résultat doit être sérialisable pour traverser le réseau. De plus, ce mécanisme ne peut fonctionner que si l'objet résultat est réifiable (au sens de ProActive). C'est-à-dire s'il respecte les trois points suivants : il n'est pas de type primitif (boolean, int, float),la classe de l'objet n'a pas l'attribut final,il possède un constructeur vide et sans argument.

4.2 La création d'un objet actif

Un objet actif peut être crée soit par instantiation, soit à partir d'un objet passif existant.

4.2.1 La création par instanciation

Une variable params contenant les paramètres du constructeur est créee.

```
A a;
Object[] params = new Object[  new Integer (26), "astring"];
```

Nous remarquons que nous passons le type entier sous sa forme objet et nous devons faire de même avec tous les types primitifs passés en paramètres.

```
try {
a = (A) ProActive.newActive("example.A", params);
} catch (ActiveObjectCreationException e) {
e.printStackTrace();
}
catch(NodeException ex)
{
ex.printStackTrace();
}
```

Grâce à la méthode **newActive**, on crée donc un objet actif en fournissant les paramètres du constructeur de la classe A. Un objet actif doit être aussi réifiable, comme les objets résultats.

4.2.2 La création à partir d'un objet existant

De plus, nous pouvons modifier un objet passif en un objet actif. Cela permet d'utiliser, par exemple, un objet résultat envoyé par un appel de méthode dans une librairie dont nous n'avons pas accès au code.

```
A a = new A (26, "astring");
a = (A) ProActive.turnActive(a);
```

4.2.3 La spécification des noeuds d'accueil

Pour communiquer avec une JVM, ProActive a besoin que l'on définisse un noeud qui fournira les services de communication voulues. Par exemple, nous voulons créer un objet actif a de type A, sur le noeud représenté par la machine neutrino.inria.fr :

```
A a = ProActive.newActive("example.A",params,"neutrino.inria.fr");
```

Ce paramètre existe aussi avec la méthode ProActive.turnActive();

4.2.4 L'enregistrement d'un objet actif

Lorsqu'un objet actif est crée, il doit être enregistré afin que ses méthodes soient disponibles pour les objets distants. La méthode ProActive.register() doit donc être éxécuté sur la machine possédant cet objet. Pour obtenir une référence sur cet objet, une machine distante doit utiliser la méthode ProActive.lookupActive(). Le paramètre doit préciser le nom de la machine et le nom de l'objet sur cette machine. On peut donc utiliser le même nom sur des machines différentes.

4.3 Les groupes d'objets

ProActive permet de créer des groupes d'objets. Ces objets réifiables peuvent être actifs ou passifs. Un groupe est déclaré avec un seul type. Mais il peut contenir des objets de même type ou des objets dérivés de la classe déclarée. Chaque objet du groupe pourra être enregistré sur un noeud différent. Lorsqu'un groupe sera donc crée, on pourra faire des appels de méthodes communes à chaque objet et obtenir un groupe de résultats qui seront gérés en utilisant les objets futurs.
La méthode ProActive.newGroup() permet de créer un groupe et d'avoir une representation fonctionnelle, permettant d'executer les méthodes du groupe.

```
A ag1=(A)ProActiveGroup.newGroup("A", params[],rmi://globus1.inria.fr/Node1);
/* on execute la méthode foo() */
ag1.foo();
```

La méthode ProActive.getGroup() permet d'obtenir une representation de gestion du groupe.

```
A a1 = new A();
A a2 = (A) ProActive.newActive("A", paramsA[], rmi://globus1.inria.fr/Node1);
/* on récupère le groupe crée avec la méthode getGroup() */
Group gA = ProActiveGroup.getGroup(ag1);
gA.add(a1);
gA.add(a2);
/* on utilise ag1 pour éxecuter une méthode */
ag1.foo();
```

4.4 Le descripteur de déploiement

Pour déployer une application distribuée, il faut définir des noeuds virtuels que l'on utilisera dans le code de l'application. Un noeud virtuel représente au moins une machine virtuelle Java. Chaque JVM est associé à une machine de la grille informatique. Pour cela, il faut passer par un fichier XML, le descripteur de déploiement. Le fichier XML utilisera un schéma XML propre à ProActive(DescriptorSchema.xsd).

4.5 L'exemple

Pour bien comprendre et surtout rassembler la totalité des concepts, il faut passer par un exemple. Nous allons créer une classe Bonjour, qui va permettre à chaque noeud de dire "Bonjour je m'appelle " associé au nom du noeud. Pour cela, nous allons créer une méthode parler() qui va retourner

un objet réifiable ContainerString au lieu d'un String. Le type String qui possède l'attribut *final* ne permettant pas de faire des communications asyndrrones ProActive.

```
public class ContainerString implements Serializable {
private String string;
public ContainerString() { }
public ContainerString (String string){
this.string=string; }
public String getString() {return(string);}
}
```

Puis dans la méthode **main** de la classe Bonjour on associe le fichier XML à une variable ProActive (pad = getProactiveDescriptor()). Ensuite on active la création des noeuds distants indiqués dans le descripteur de déploiement (pad.activateMappings()). On récupère le noeud virtuel (pad.getVirtualNode()), et on l'utilise comme argument pour créer un groupe d'objets actifs, un objet actif étant créé sur chaque noeud associé au noeud virtuel. Dans le descripteur de déploiement trois noeuds sont associés au noeud virtuel obtenu, donc trois objets actifs sont créés. On crée aussi un objet passif et un objet actif en local, puis on les ajoute au groupe grace à la représentation standard de groupe. Après avoir lancé la méthode parler() sur le groupe, on obtient un groupe de messages, on récupère chaque membre du groupe de résultat avec la méthode get(), ensuite avec la méthode getString() de ContainerString on peut obtenir le message de chaque noeud. La méthode killall(false) associé au descripteur ProActive pad permet de détruire tous les noeuds et machine virtuelles créés.

```
try {
ProActiveDescriptor pad = ProActive.getProactiveDescriptor(
"/user/ProActive/descriptors/essai.xml");
pad.activateMappings();
VirtualNode noeudEssai = pad.getVirtualNode("NoeudVirtuel");
Bonjour groupBonjour = (Bonjour) ProActiveGroup.newGroup(
Bonjour.class.getName(), new Object[] , noeudEssai);
Bonjour bonjour1 = new Bonjour();
Bonjour bonjour2 = (Bonjour) ProActive.newActive( Bonjour.class.getName(), new
Object[] );
Group gS = ProActiveGroup.getGroup(groupBonjour);
gS.add(bonjour1);
gS.add(bonjour2);
ContainerString groupeMessage = groupBonjour.parler();
Group gM = ProActiveGroup.getGroup(groupBonjour);
for(int i=0;i<gM.size();i++)
{ System.out.println( ((ContainerString)gM.get(i)).getString()); }
pad.killall(false);
}
catch (Exception e) {
System.err.println("Error : " + e.getMessage()); e.printStackTrace();
}
```

Un autre exemple consiste à obtenir un tableau des noeuds associés à un noeud virtuel grace à la méthode getNodes(). On peut alors créer un objet actif sur un des noeuds obtenus, et appeler sa méthode appeler() même s'il est sur une machine distante.

```
VirtualNode noeudEssai = pad.getVirtualNode("NoeudVirtuel");
Node[] nodes = NoeudSalut.getNodes();
Bonjour bonjour = (Bonjour) ProActive.newActive( Bonjour.class.getName(), new
```

Object[] , nodes[0]) ;
System.out.println(bonjour.parler().getString()) ;

Voici le résultat :
Bonjour je m'appelle //meije.inria.fr/noeudBonjour-1078
Bonjour je m'appelle //sidonie.inria.fr/noeudBonjour-5541
Bonjour je m'appelle //noadcoco.inria.fr/noeudBonjour-4781
Bonjour je m'appelle LOCAL
Bonjour je m'appelle //orchidee.inria.fr/Node-4781

5 Description du travail réalisé

Les acteurs (ou intervenants) d'un marché financier s'échangent des titres financiers de différents type suivant le marché considéré. Un titre est exprimé sous forme de contrat. Nous pouvons généralement classer ces titres financiers en quatre groupes :
- Les actions donnent un droit sur la gestion de l'entreprise, sur les bénéfices réalisés et sur l'actif social.
- Les obligations sont des titres de créances représentatives de dettes. Une obligation donne droit au paiement d'un intérêt en général annuel et au remboursement du capital.
- Les matières premières et les devises sont des contrats de propriété sur des éléments comme l'or, le café, le dollar ou l'euro.
- Les produits dérivés.

En finance, un produit dérivé est un contrat entre deux parties, un acheteur et un vendeur, qui fixe des flux financiers futurs basés sur ceux d'un actif sous-jacent (support), réel ou théorique, généralement financier. Depuis les années 80, les transactions sur les produits dérivés restent en forte croissance et représentent aujourd'hui l'essentiel de l'activité des marchés financiers.

Le rôle des produit dérivés est de permettre la réalisation d'une transaction
- qu'il serait plus difficile ou coûteux de réaliser sur l'actif sous-jacent lui-même, pour des raisons qui peuvent être d'origine réglementaire, comptable, fiscale, ou financière.
- voire qu'il serait impossible de réaliser parce que l'actif en question n'existe encore que théoriquement, son existence étant contingente à la réalisation éventuelle d'un évènement.

On distingue deux types de produits dérivés : les produits fermes (ou à terme) et les produits optionnels. Nous nous intéresserons à ces derniers produits.

5.1 La valorisation d'options européennes

Les options se négocient sur les marchés financiers. Mais à l'inverse des actions, devises, matières premières, la valeur d'une option peut se déterminer par un raisonnement d'arbitrage et résulte de la nécessité de couvrir le risque endossé par la signature du contrat. On parle de prix d'arbitrage ou encore de prime comme pour un contrat d'assurance. La théorie financière de l'arbitrage fixe la valeur d'une option comme étant l'espérance mathématique du payoff de l'option, calculé dans l'unité monétaire de la date de signature (c'est à dire $t = 0$). C'est le taux d'intérêt court terme r, supposé constant sur la période de vie de l'option, qui paramètre le changement de numéraire, référence [3], [10] et [8].
Prime initiale d'une option de vente (ou "Put" en anglais) :

$$\text{Prime à } t \text{ égale } 0 = \mathbb{E}^* \left[\exp(-rT) \max(K - S_T, 0) \right]. \tag{1}$$

Prime initiale d'une option d'achat (ou "Call" en anglais) :

$$\text{Prime à } t \text{ égale } 0 = \mathbb{E}^* \left[\exp(-rT) \max(S_T - K, 0) \right]. \tag{2}$$

où

– S_T : prix du sous-jacent à l'échéance T du contrat.
– K : prix d'exercice de l'option.
– r : le taux d'intérêt.
– T : la date d'échéance.
Ainsi, pour calculer la prime d'une option, il faut connaître la distribution de probabilité du prix S_T, sachant uniquement des informations disponibles à la date de signature c'est à dire à la date $t = 0$. Il faut donc se donner un modèle stochastique pour l'évolution temporelle du sous-jacent.
Un des modèles les plus célèbres de la finance est celui proposé par Black et Scholes, dès 1973, asses riche pour contenir l'essentiel du risque, assez simple pour permettre, au moins dans les cas d'options put et call sur un seul sous-jacent de type action, d'avoir des formules de primes explicites.
Dans le modèle de Black-Scholes, le prix du sous-jacent à chaque instant est donné par :

$$S_t = S_0 \exp\left((r - \sigma^2/2)t + \sigma W_t\right) \tag{3}$$

où $(W_t, t \geq 0)$ est un mouvement brownien standard de dimension un et où σ est un paramètre appelé *volatilité du sous-jacent*. Notons que S_t est solution de l'équation différentielle stochastique

$$dS_t = S_t(rdt + \sigma dW_t), S_{t=0} = S_0 \text{ donné .}$$

qui se discrétise en temps, pour la simulation d'une trajectoire de prix de sous-jacent, par

$$S_{(n+1)\Delta t} = S_{n\Delta t}\left(1 + (r\Delta t + \sigma\sqrt{\Delta t}G_n\right) \tag{4}$$

où les $(G_n, n \geq 1)$, sont une famille indépendante de variables aléatoires gaussiennes centrées et réduites. Dans le cas du modèle de Black et Scholes unidimensionnel, on peut utiliser la discrétisation exacte

$$S_{(n+1)\Delta t} = S_{n\Delta t} \exp\left((r - \sigma^2/2)\Delta t + \sigma\sqrt{\Delta t}G_n\right). \tag{5}$$

5.2 La formule de valorisation de Black-Scholes

La formule de Black-Scholes permet de calculer la valeur théorique d'une option à partir des cinq données S_0,K,r,σ et T, référence [8] et [9].
Le prix de l'option d'achat est donné par l'espérance sous la probabilité risque neutre du payoff terminal actualisé

$$Call = \mathbb{E}^*[max(S_T - K, 0).exp(-r.T)],$$

, soit la formule de Black-Scholes :

$$Call = S_0.N(d_1) - Ke^{-rT}N(d_2) \tag{6}$$

Le prix de l'option de vente est donné par l'espérance sous probabilité risque neutre du payoff terminal actualisé

$$Put = \mathbb{E}^*[max(K - S_T, 0).exp(-r.T)],$$

, soit la formule de Black-Scholes :

$$Put = Ke^{-rT}N(-d_2) - S_0.N(-d_1) \tag{7}$$

où N(x) est la fonction de répartition de la loi normale centrée réduite N(0,1)

$$N(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\prod}} e^{\frac{-1}{2}u^2} du \tag{8}$$

et

$$d_1 = \frac{ln(S/K) + (r + \sigma^2/2)(T-t)}{\sigma\sqrt{T-t}},\ d_2 = d_1 - \sigma\sqrt{T-t} \tag{9}$$

Nous utilisons la formule de Black-Scholes pour calculer explicitement la valeur du prix de l'option européenne et nous l'utilisons pour vérifier la convergence du résultal obtenu par la méthode de Monte Carlo décrite dans la section suivante.

5.3 La méthode de Monte Carlo

5.3.1 Une vue d'ensemble de la méthode appliquée à la valorisation des options

La méthode de Monte Carlo consiste à approcher l'espérance mathématique dans les formules (2) et (1) par une moyenne d'un nombre fini $nbMC$ de réalisations aléatoires indépendantes du prix S_T [10] :

$$\mathbb{E}^* \left[\exp(-rT)\max(K - S_T, 0)\right] \simeq \frac{1}{nbMC} \sum_{i=1}^{nbMC} \left[\exp(-rT)\max(K - S_T^i, 0)\right]. \tag{10}$$

Pour cela, on simule des réalisation S_T^i par la méthode de discrétisation (4) ou (5) qui induit une erreur d'ordre Δt. Ces réalisations sont indépendantes entre elles, si la succession des variables aléatoires G_k sont indépendantes entre elles et entre les trajectoires simulées.

Ayant toutes ces espérances, on calcule une moyenne des simulations pour obtenir la valeur du prix d'une option. De plus, on pourra aussi calculer la variance de la variable aléatoire simulée, afin d'obtenir un ordre de grandeur de la précision numérique obtenue, pour un échantillon de taille $nbMC$ de simulations. Donc, à la fin des simulations par la méthode de Monte Carlo, il faut vérifier cette propriété - de convergence (10).

5.3.2 La convergence et l'intervalle de confiance

La qualité de la méthode Monte Carlo est évaluée par sa vitesse de convergence et son erreur. Comme nous pouvons d'observer, on ne peut pas estimer le prochain nombre aléatoire. Cependant, avec un grand ou très grand nombre de variables aléatoires on peut diminuer l'effect de l'aléatoire. Pour décrire la convergence, on peut utiliser les théorèmes suivants :
- Théorème de la forte loi des grands nombres
- Théorème de la limite centrale
- Théorème du logarithme itéré

Par la loi des grands nombres :

$$\lim_{nbMC\to\infty} \frac{1}{nbMC} \sum_{i=1}^{nbMC} \exp(-rT)\max(K - S_T^i, 0) = I \tag{11}$$

Par la théorème de la limite centrale

$$\lim_{nbMC\to\infty} \frac{\sqrt{nbMC}}{\delta} \mathbb{E}^* \left[\exp(-rT)\max(K - S_T^i, 0) - I\right] = 0 \tag{12}$$

Donc, combien de réalisation doit être choisi pour obtenir un résultat proche de I ?
Grâce aux documents donneés par M. Bossy, référence [3], on peut :
- On utilise la notion d'intervalle de confiance pour exprimer l'erreur de la méthode Monte Carlo et par conséquent on peut avoir une confiance à 95% de la valeur obtenue après les simulation de Monte Carlo (10), si I est dans l'intervalle suivante :

$$[I_{nbMC} - 1.96\frac{\delta}{\sqrt{nbMC}} \ ; \ I_{nbMC} + 1.96\frac{\delta}{\sqrt{nbMC}}] \tag{13}$$

– La vitesse de convergence de cette méthode est en $\frac{\delta}{\sqrt{nbMC}}$

– En finance, un ordre de convergence de 10^3 est souhaitable qui impose que nbMC = $\frac{10^6}{\delta^2}$

5.4 Le générateur des nombres aléatoires en parallèle

Pour obtenir une meilleur précision, pour un meilleur temps de calcul, les simulations de Monte Carlo sont executées en parallèle sur plusieur d'ordinateurs. Chaque simulation de Monte Carlo a besoin que la génération pseudo-aléatoire avec des bonnes propriétés d'indépendance et de convergence. Ainsi les générateurs aléatoires ne peuvent pas simplement être distribués par ensemble de worker travaillant.

Une recherche bibliographique, référence [5] et [14], nous a permis d'odentifier plusieurs méthodes pour produire des sequences indépendantes de nombres aléatoires : il y a fondamentalement deux types de méthodes. Un type est basé sur *'skipping ahead'* dans les séquences , et l'autre utilise de différents points de départ pour un générateur congruential linéaire.

Dans notre cas, on a utilisé le premier type *'skipping ahead'* dans un générateur congruential linéaire. Parfois il est utile de produire des sous-séquences séparés et indépendants avec le même générateur. Pour produire des sous-séquences séparées, ce n'est généralement pas une bonne idée de choisir deux racines pour les sous-séquences parce que nous n'avons généralement aucune information sur les rapports entre les racines. En fait, un choix malheureux des racines pourrait générer des sous-séquences recourrantes. Une meilleure manière consiste à fixer la racine d'un subsequence et puis de sauter en avant d'une distance déterminée pour commencer la deuxième sous-séquence.

Proposons un générateur congruential linéaire :

$$x_{i+1} = ax_i \, mod \, m \tag{14}$$

ou on peut re-écrire :

$$x_{i+k} = a^k x_i \, mod \, m \tag{15}$$

La première séquence peut générer :

$$x_s, x_{s+1}, x_{s+2}, ...$$

Concernant la séquence non-overlapping suivante nous pouvons générer la suite :

$$x_{s+k}, x_{s+k+1}, x_{s+k+2}, ...$$

si on choisit la racine initiale : x_{s+k-1} = (a^k mod m) x_0 mod m
ou une autre suite

$$x_{s+k}, x_{s+2k}, x_{s+3k}, ...$$

si on choisit la racine initiale : x_{i+1} = (a^k mod m) x_i mod m

5.5 Les options européennes simples

Une option d'achat (resp. de vente) européenne, de prix d'exercice K (*strike* en anglais), d'échéance T sur un actif désigné (sous-jacent) est un contrat qui donne à son détenteur le droit, mais non l'obligation, d'acheter (resp. de vendre) cet actif au prix K à la date T. Le prix K est fixé à la signature du contrat. Suivant la terminologie anglo-saxonne, on désigne par *call* l'option d'achat et par *put*, l'option de vente. On note généralement S_t, pour $t \in [0, T]$, le prix de l'actif sous-jacent, tel qu'il est coté sur un marché à un instant t de la période $[0, T]$. A la date d'échéance, l'option peut être exercée ou non, suivant le gain obtenu à partir du sous-jacent.

- L'acheteur d'un call ne décide d'exercer (acheter les titres) que si le cours (prix à l'échéance) du sous-jacent est supérieur à son prix d'exercice K.
- De même, l'acheteur d'un put ne décide d'exercer (vendre les titres) que si le cours du support est inférieur à son prix d'exercice.

Exemple :

Prenons le cas d'une option de vente sur une action, un contrat est signé pour une date d'échéance 1 an, avec un prix d'exercice de 45 euros. A l'instant de la signature du contrat, le prix de l'action est de 40 euros. Un an plus tard, le prix du sous-jacent est de 42 euros. L'option est exercée car elle a une valeur de 3 euros.

Ainsi, à l'échéance du contrat, l'exercice (ou non) de l'option, dégage un flux financier, toujours positif ou nul, qu'on appelle *payoff*.

Le payoff du call : $(S_T - K)^+ = \max(S_T - K, 0)$.

Le payoff du put : $(K - S_T)^+ = \max(K - S_T, 0)$.

raisonnement d'arbitrage. On parle de valorisation d'option.

5.5.1 L'algorithme de la valorisation d'option européenne simple en parallèle

Une option simple est un put ou call européen. On veut simuler $nbMC$ valeurs indépendantes du prix du sous-jacent entre la date de signature de l'option et la date d'échéance. On appellera trajectoire la courbe $t \longrightarrow S_t$, du prix du sous-jacent entre la date $t = 0$ et la date $t = T$ (date d'échéance du contrat). Pour appliquer l'algorithme à la valorisation des options à barrières et des options sur panier (que l'on expliquera plus tard), il est nécessaire de discrétiser en temps comme dans (4) ou (5). La discrétisation se fait avec un pas de temps constant égal à $\Delta t = T/N$.
La variable $nbMC$ désigne le nombre de simulations de trajectoires du prix (le nombre de Monte Carlo). Dans le cas d'une architecture non distribuée, les algorithmes sont séquentiels. Pour pouvoir profiter d'une architecture distribuée, il faut se baser sur des calculs que l'on peut distribuer. Dans le cas présent, les calculs de chaque simulation se font indépendamment les uns des autres. Il suffit donc de répartir ce calcul des trajectoires.

Dans notre application, nous avons un serveur qui gère plusieurs sous-serveurs et chaque sous-serveur gère un groupe de plusieurs worker qui sont des simulateurs. Le sous-serveur va s'occuper de répartir les calculs aux simulateurs. Les simulateurs vont retourner les résultats de chaque simulation. Pour éviter des temps de communication inutiles, nous allons répartir les simulations par lot de 1000 ou 5000.

Nous déléguons la gestion des simulateurs aux sous-serveurs, au lieu du serveur ; cela permet d'éviter un goulet d'étranglement dans les communications réseau vers le serveur.
Voici l'algorithme d'un simulateur :

1: ***/* À la réception d'un pacquet, chaque simulateur calcule M simulations de Monte Carlo */***
2: **for** $i = 0$ to $M - 1$ **do**
3: **for** $t = 0$ to nbIntervallleTemps-1 **do**

Tirage d'une valeur $\epsilon_i \sim$ ***N(0,1) de loi gaussienne centrée réduite***

$S_t = S_{t-1} exp((r - \sigma^2/2)\Delta t + \sigma\epsilon_i\sqrt{\Delta t})$

end for

$X = max(K - S, 0)$

$sumX += X$

$sumX^2 += X * X$

end for

/* Chaque simulateur renvoie les sommes, et le server calcule le prix d'option */

$moyenne = \frac{1}{M * nbPacket} * \sum_1^{nbPaket} sumX$

$variance = \frac{1}{M * nbPacket} * \sum_1^{nbPaket} sumX^2 - moyenne^2$ ***/* Ici variance*** $= \delta^2$ ****/***

$put = e^{-rT} * moyenne$

$call = put - Ke^{(-rT)} + S_0$

/* Intervalle de confiance : C'est une concéquence du théorème de la limite centrale qui permet de mesure la précision du calcul, regardant l'intervalle (13) */

$Borne\_inferieure = put - 1.96 * \sqrt{(\frac{variance}{M*nbPacket})}$

$Borne\_superieure = put + 1.96 * \sqrt{(\frac{variance}{M*nbPacket})}$

5.6 Les options européennes à barrières

Une option à barrière est une option européenne (i.e. dont l'exercice n'est autorisée qu'à l'échéance) qui s'active ou se désactive suivant que le prix du sous-jacent franchie (ou non) une barrière, pendant la durée de vie de l'option, c'est à dire entre les instants $t = 0$ et $t = T$. Pour une option européenne simple, il y a donc quatre options à simple barrière associées.

Les options à barrière ont été introduites pour disposer d'option de type call et put, moins chères que les options simples, car ne couvrant pas toutes les situations de prix sur le sous-jacent, mais seulement des tranches possibles de valeurs. Ainsi, l'option barrière permet de sélectionner l'amplitude du mouvement du sous-jacent que l'on souhaite couvrir. Par exemple, si vous croyez que l'action IBM montera cette année, mais vous êtes disposé à parier qu'elle n'ira pas plus haut que 100 euros, alors vous pouvez acheter une option de barrière 100 et payer moins que la prime que l'option simple [8].

Les options à barrières peuvent être désactivantes (Out) ou activantes (In). L'option disparaît ou apparaît dès que le cours du sous-jacent atteint, pendant la période de référence, la barrière fixée préalablement.

- Option down : l'option est désactivée ou activée lorsque le cours du sous-jacent franchit la barrière à la baisse.
- Option up :l'option est désactivée ou activée lorsque le cours du sous-jacent franchit la barrière à la hausse.

Exemple 1 :

Prenons le cas d'une option d'achat up-in sur une action, un contrat est signé pour une date d'échéance $T = 1$ an. A l'instant de la signature du contrat, le prix de l'action S est de 40 euros, le prix d'exercice à la date d'échéance T est $K = 43$ euros et la barrière UP (le seuil haut est supérieure au prix de l'action) est fixé à 44 euros. Le sous-jacent, qui part de 40 à $t = 0$ doit franchir le niveau 44, à un moment ou un autre des $t \in [0, T]$, pour que l'option soit prise en compte. Une fois la barrière franchie, le sous-jacent peut redescendre en dessous de 44.
Ainsi, si l'option expirait avec le prix de l'action à 43.5, mais qui n'a pas atteint la barrière de 44 pendant la durée du contrat de l'option, l'option barrière n'est pas activée alors que le call simple aurait rapporté 0.5 euros. Par contre, si le prix de l'action atteint la barrière de 44 pendant la

durée du contrat de l'option, cette option est activée (IN) et l'option dégage 0.5 euros comme le call simple. La prime à payer pour ce Call UP and IN est :

$$\text{Prime à la date } (t=0) = \mathbb{E}\left(\exp(-rT)(S_T-K)^+ \mathbb{1}_{(\max\limits_{t\in[0,T]} S_t > 44)}\right).$$

où $(S_T - K)^+$ signifie la partie positive de $S_T - K$ ou encore $\max(S_T - K, 0)$ et où l'indicatrice d'ensemble $\mathbb{1}_A$ vaut 1 si la condition A est vrai et 0 si la condition A est fausse, voir la Figure (1).

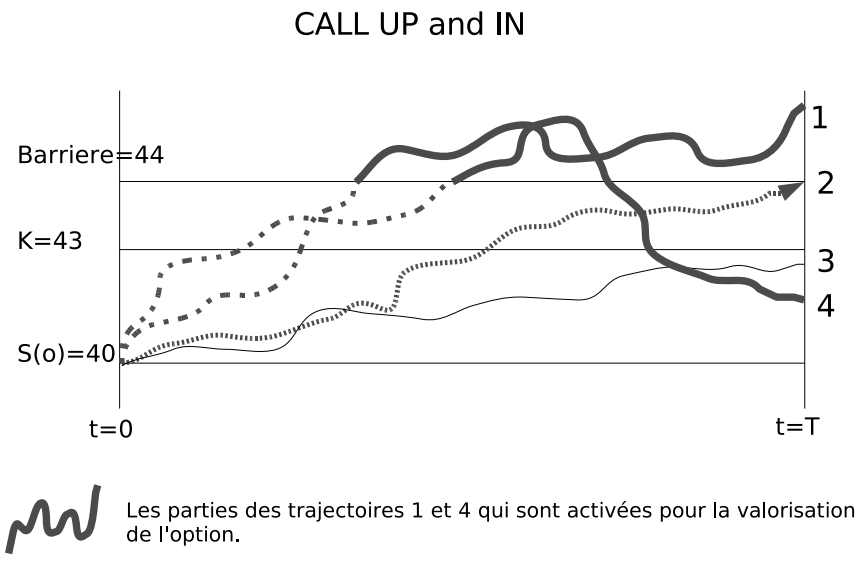

FIG. 1 – La simulation d'un Call UP and IN

Exemple 2 :

Prenons le cas d'une option de vente down-out sur une action, un contrat est signé pour une date d'échéance $T = 1$ an. A l'instant de la signature du contrat, le prix de l'action S est de 43 euros et le prix d'exercice à la date d'échéance T est $K = 44$ euros et la barrière DOWN (le seuil bas est inférieur au prix de l'action) à 42 euros. Cette option se comporte comme une option de vente simple si le prix du sous-jacent ne va jamais en dessous de 42. Mais si le prix du sous-jacent descend en dessous de 42, l'option s'annule. On dit que cette option est désactivée (OUT). Notez que l'option ne se réactive pas même si le prix du sous-jacent remonte au-dessus de 42 encore. Une fois qu'elle est OUT, elle est OUT pour toujours.

La prime à payer pour ce Put DOWN OUT est :

$$\text{Prime à la date } (t=0) = \mathbb{E}\left((K - S_T)^+ \mathbb{1}_{(\min_{t\in[0,T]} S_t > 42)}\right).$$

voir la Figure (2) pour référence.

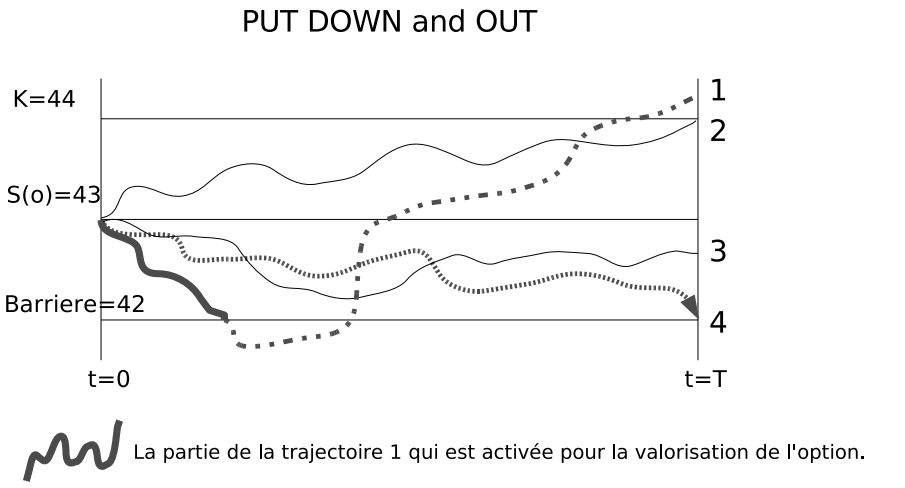

FIG. 2 – La simulation d'un Put DOWN and OUT

5.6.1 L'algorithme de la valorisation d'option européenne à barrière en parallèle

Dans notre application, nous avons developpé 4 algorithmes de valorisation de l'option à barrière (UP IN, UP OUT, DOWN IN et DOWN OUT).
Nous ne donnons que 2 des 4 algorithmes (ceux qui correspondent aux 2 exemples UP IN et DOWN OUT). Les 2 autres sont pratiquement identiques, en notant qu'il n'y a plus de parité call-put comme dans la cas des options simples.

Valorisation l'option européenne à barrière UP IN

/* À la réception d'un pacquet, chaque simulateur calcule M simulations de Monte Carlo */

2: **for** $i = 0$ to $M - 1$ **do**

Tirage d'une valeur $\epsilon_i \sim$ N(0,1) suivant la loi gaussienne centrée réduite

ETAT=VIDE
/* Le prix initial du sous jacent est inférieur au seuil, l'option est UP IN donc on ne comptabilise que la trajectoire où le prix de sous jacent est supérieur au seuil */
while ETAT = VIDE AND t < nbIntervalleTemps + 1 **do**
  $S_t = S_{t-1}exp((r - \sigma^2/2)\Delta t + \sigma\epsilon_i\sqrt{\Delta t})$
  if $S_t >$ Seuil **then**
    ETAT=ACTIVE
  end if
end while
if t = nbIntervalleTemps AND ETAT = VIDE **then**
  $S_t = K$
end if
while ETAT = ACTIVE et t < nbIntervalleTemps + 1 **do**
  $S_t = S_{t-1}exp((r - \sigma^2/2)\Delta t + \sigma\epsilon_i\sqrt{\Delta t})$
end while
$X = max(K - S_t, 0)$
$Y = max(S_t - K, 0)$
$sumX+ = X$
$sumX^2+ = X * X$
$sumY+ = Y$
$sumY^2+ = Y * Y$
end for
/* Chaque simulateur renvoie des sommes et le server calcule le prix d'option */
$moyennePut = \frac{1}{M * nbPacket} * \sum_1^{nbPaket} sumX$
$variancePut = \frac{1}{M * nbPacket} * \sum_1^{nbPaket} sumX^2 - moyennePut^2$
$moyenneCall = \frac{1}{M * nbPacket} * \sum_1^{nbPaket} sumY$
$varianceCall = \frac{1}{M * nbPacket} * \sum_1^{nbPaket} sumY^2 - moyenneCall^2$
$put = e^{-rT} * moyennePut$
$call = e^{-rT} * moyenneCall$

Valorisation l'option européenne à barrière DOWN OUT

/* À la réception d'un pacquet, chaque simulateur calcule M simulations de Monte Carlo */
for $i = 0$ to $M - 1$ **do**
  Tirage d'une valeur $\epsilon_i \sim$ ***N(0,1) suivant la loi gaussienne centrée réduite***
  ETAT=VIDE
  /* Le prix initial du sous jacent est supérieur au seuil, l'option est DOWN OUT donc on ne comptabilise que la trajectoire où le prix de sous jacent est supérieur au seuil */
  while ETAT = VIDE et t < nbIntervalleTemps + 1 **do**
    $S_t = S_{t-1}exp((r - \sigma^2/2)\Delta t + \sigma\epsilon_i\sqrt{\Delta t})$
    if $S_t <$ Seuil **then**
      ETAT = DESACTIVE
    end if
  end while
  $X = max(K - S_t, 0)$
  $Y = max(S_t - K, 0)$
  $sumX+ = X$
  $sumX^2+ = X * X$

15: $sumY += Y$
$sumY^2 += Y * Y$
end for
/* Chaque simulateur renvoie des sommes et le server calcule le prix d'option */
18: $moyennePut = \frac{1}{M * nbPacket} * \sum_1^{nbPaket} sumX$
$variancePut = \frac{1}{M * nbPacket} * \sum_1^{nbPaket} sumX^2 - moyennePut^2$
$moyenneCall = \frac{1}{M * nbPacket} * \sum_1^{nbPaket} sumY$
21: $varianceCall = \frac{1}{M * nbPacket} * \sum_1^{nbPaket} sumY^2 - moyenneCall^2$
$put = e^{-rT} * moyennePut$
$call = e^{-rT} * moyenneCall$

5.7 Les options européennes sur panier

Une option sur panier est une option sur un ou plusieurs actifs (on considère un panier avec un actif de poids = 1 comme étant possible). La composition du panier est décrite par la donnée des pourcentages de chaque actif considéré sachant que l'on doit avoir un poids de 1 en additionnant ces poids. Les actifs composant le panier sont corrélés aux autres.

On choisit de modéliser les corrélations entre les titres de façon indirecte. On modélise le prix de chaque titre du panier avec le modèle de Black-Sholes. On note N_p le nombre d'actifs dans le panier, alors pour $i = 1, \ldots, N_p$,

$$dS_t^i = S_t^i(rdt + \sigma^i dW_t^i)$$

Le vecteur $W_t = (W_t^1, \ldots, W_t^{N_p})$ est obtenu par la formule

$$W_t = LB_t \tag{16}$$

où le vecteur $B_t = (B_t^1, \ldots, B_t^{N_p})$ est un vecteur de mouvements browniens indépendants entre eux, et où L est une matrice de mélange de dimension $N_p \times N_p$. On a l'habitude de se donner la matrice de corrélations entre les W_1^i sous la forme d'une matrice

$$Cor = \begin{pmatrix} 1 & \rho & \ldots & \rho \\ \rho & 1 & \ldots & \rho \\ \ldots & \ldots & \ldots & \ldots \\ \rho & \rho & \ldots & 1 \end{pmatrix}$$

formée de 1 sur la diagonale et de coefficients constants, tous égaux à $0 \leq \rho \leq 1$, partout ailleurs. Si $\rho < 1$, cette matrice, à diagonale dominante, est définie positive. Elle admet une *racine*, c'est à dire qu'il existe une matrice L, triangulaire inférieure, telle que, (en notant L^t la transposée) [6],

$$Cor = L^t \times L.$$

Dans la forme à temps discret du modèle de Black Scholes, le prix d'un actif dépend, au cours du temps, des prix des actifs à l'instant précédent. On remarque donc que les calculs doivent être effectués en même temps et que l'on ne peut pas distribuer le calcul de la trajectoire du prix du panier. On restera donc sur la distribution du nombre de simulations.
On va calculer chaque trajectoire du panier qui sera la somme pondérée des trajectoires des prix de chaque actif du panier. Le prix de chaque actif devra utiliser dans sa formule de calcul le mélange des browniens, c'est à dire la matrice L, par la formule (16).
Les hypothèses et notations
– Un panier de N_p actions.

– Un vecteur de prix du sous jacent des actifs sur panier S_t^i.
– Un vecteur de volatilité des actifs du panier σ^i.
– Un vecteur de poids des actifs sur panier p_i.
– Une matrice de corrélation entre des browniens $(Cor_{(i,j)})$, avec $i, j = 1, \ldots, N_p$.
– Le prix d'exercice K est un scalaire.
– Le taux sans risque r est constant.
– La date date d'échéance T est fixée.
Un actif ne peut plus être simulé indépendamment des autres car son mouvement brownien doit être simulé en même temps que les autres en raison de la corrélation.

La méthode de Monte Carlo pour valorisation de l'option sur panier :

$$S^i_{t+\Delta_t} = S^i_t exp((r - \sigma_i^2/2)\Delta_t + \sigma_i Z^i \sqrt{\Delta_i})$$

où

$$Z = LG$$

et G est un vecteur gaussien centré réduit, de dimension N_p. L est la matrice triangulaire inférieure, issue de la décomposition de Choleski de la matrice Cor.

5.7.1 L'algorithme de la valorisation d'options européennes sur panier en parallèle

Calculer la matrice triangulaire inférieure L par la méthode de Choleski à partir de la matrice de corrélation au server.
/* À la réception d'un pacquet, chaque simulateur calcule M simulations de Monte Carlo */
for $i = 0$ to $M - 1$ **do**
  for $t = 0$ to nbIntervalleTemps-1 **do**
    Générer un vecteur gaussien G, centré réduit.
    Calculer le vecteur Z = L*G.
    for $j = 0$ to nbActifs **do**
      $S_j = S_{j-1} exp((r - \sigma_j^2/2)\Delta t + \sigma_j * Z_j * \sqrt{\Delta t})$
    end for
  end for
  /* Calculer le prix du panier en fonction du vecteur de poids des actifs P */
  $P = \sum_1^{nbActif} S_j * P_j$
  $X = max(K - P, 0)$
  $Y = max(P - K, 0)$
  $sumX += X$
  $sumX^2 += X * X$
  $sumY += Y$
  $sumY^2 += Y * Y$
end for
/* Chaque simulateur renvoie des sommes et le server calcule le prix d'option */
$moyennePut = \frac{1}{M * nbPacket} * \sum_1^{nbPaket} sumX$
$variancePut = \frac{1}{M * nbPacket} * \sum_1^{nbPaket} sumX^2 - moyennePut^2$
$put = e^{-rT} moyennePut$
$moyenneCall = \frac{1}{M * nbPacket} \sum_1^{nbPaket} sumY$
$varianceCall = \frac{1}{M * nbPacket} \sum_1^{nbPaket} sumY^2 - moyenneCall^2$
$call = e^{-rT} moyenneCall$

/* Establisement de l'intervalle de confiance */

$$Borne\_inferieure = put - 1.96 * \sqrt{\frac{variancePut}{M * nbPacket}}$$

$$Borne\_superieure = put + 1.96 * \sqrt{\frac{variancePut}{M * nbPacket}}$$

5.8 Les options américaines simples

La seule différence entre des options européennes et des options américaines est qu'une option américaine peut être exercée n'importe quand pendant la vie de l'option. La difficulté est de déterminer simultanément la stratégie de moment d'exercice optimale pour optimiser le prix de l'option étant donneé qu'il existe plusieurs dates possibles d'exercice. Par conséquent, le détenteur de l'option doit décider, à chaque date possible d'exercice, s'il vaut mieux exercer l'option ou attendre. Cette décision dépend de la comparaison entre le montant d'argent obtenu si l'option est exercée (la valeur immédiate d'exercice φ) et le montant d'argent obtenu si l'option est exercée à une future date (la valeur de continuation u).

La décision optimale d'exercice se fonde sur l'évaluation de la valeur de continuation. Comparant ces valeurs de l'exercice immédiat avec la valeur de continuation qu'on peut identifier la règle d'arrêt optimale. Ce procédé est iteratif, commencant par la valeur (connue) à T de l'option et se propage jusqu'àu temps zero. A chaque pas de temps, en escomptant les pay-off obtenus au temps zéro, le prix de l'option américaine est trouvé.

Nous nous interessons à partir de maintenant à la valorisation d'une option américainne de type Put. Le payoff d'un put américain est calculé comme :

$$payoff = max(K - S(t), 0) \tag{17}$$

si t est la date d'exercice choisie par la détenteur. Donc, le prix d'un put américain est comme suivant :

$$put = \sup_{(0 \leq \tau \leq T)} \mathbb{E}^*[exp(-r.\tau)max(K - S(\tau), 0)] \tag{18}$$

où $S(\tau)$ est le prix du sous-jacent au temps d'exercice aléatoire τ. Par rapport aux options européenne, dans le cas américain, on doit chercher la date aléatoire optimale d'exercice pour calculer le prix de l'option. Accordez a la méthode de Monte Carlo, par rapport de nombre de simulation de Monte Carlo $nbMC$ de simulations, on peut estimer le prix d'un put américain au temps zero :

$$put_0^{nbMC} = \frac{1}{nbMC} \sum_{1 \leq i \leq nbMC} exp(-r.\tau^i)max(K - S(\tau^i), 0) \tag{19}$$

Une question se pose : comment trouver le temps optimal τ parmi les valeurs aléaotoires τ^i. Longstaff et Schwartz (2001) estiment la valeur de continuation par une régression nommée moindres carrés en tenant compte de l'information fournie par la simulation de Monte Carlo. Pour calculer la valeur de continuation ils emploient un ensemble de fonctions de base dont les arguments sont les prix des sous-jacents au pas de temps précedent. Les valeurs adaptées selon ces régressions sont prises comme valeurs de continuation. Comme nous avons dit, en comparant la valeur de continuation avec la valeur d'exercice imédiatement, on identifie la règle d'arrêt optimale τ^*.

$$\tau_N^* = N \tag{20}$$

$$\tau_j^* = j \, \mathbb{1}_{\{\varphi_j \geq u_j\}} + \tau_{j+1}^* \, \mathbb{1}_{\{\varphi_j < u_j\}} \tag{21}$$

Avec

$$\{\varphi_j < u_j\} = \{max(K - S(\tau_j), 0) < \mathbb{E}[e^{-r(T-j)}max(K - S(\tau_{j+1}^*), 0) \mid S(\tau j)]\} \tag{22}$$

Alors, τ_j^* est trouvé récursivement avec $(0 \leq j \leq N)$, N est le nombre de pas de temps. :

$$\tau_j^* = min\{i \geq j, \varphi_i = u_j\} \tag{23}$$

Nous parlerons de la méthode des moindres carrés dans la section suivante. D'abord, nous voulons présenter la façon de générer des prix de sous-jacent, et comment on obtient la valeur de payoff pour chaque pas de temps.
Comme dans le cas de la valorisation d'une option européenne, on simule des réalisation S_i^t par la méthode de discrétisation (4) ou (5). Cela veut dire que nous simulons des trajectoires du prix de sous-jacent en disrétisant le temps dans chaque trajectoire. Ca nous donne une matrice de la valeur du sous-jacent.

$$\begin{pmatrix} S_{t=1}^1 & S_{t=2}^1 & \dots & \dots & \dots & S_{t=N}^1 \\ S_{t=1}^2 & S_{t=2}^2 & \dots & \dots & \dots & S_{t=N}^2 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ S_{t=1}^i & \dots & S_{t=n}^i & S_{t=n+1}^i & \dots & S_{t=N}^i \\ \dots & \dots & \dots & \dots & \dots & \dots \\ S_{t=1}^M & S_{t=2}^M & \dots & \dots & \dots & S_{t=N}^M \end{pmatrix} \tag{24}$$

avec i = 1,...,M nombre de simulations de Monte Carlo et n = 1,...,N nombre de pas de temps.
Alors qu'une option européenne ne s'exécute qu'à la date d'échéance T, une option américaine peut s'exercer à chaque pas de temps. Le problème d'estimation de la valeur d'une option américaine revient à trouver le pas de temps où son payoff va être optimal. Pour cela, on peut dire que pour la valorisation, nous devons comparer le payoff de l'exercice immédiat au pas de temps n à la valeur de continuation du pas de temps *n+1*, considérant bien sûr que le payoff doit être positif dans les deux cas d'option Put et Call.
Nous notons la matrice de la valeur de payoff d'une option Put ainsi :

$$\begin{pmatrix} (K - S_{t=1}^1)^+ & (K - S_{t=2}^1)^+ & \dots & \dots & \dots & (K - S_{t=N}^1)^+ \\ (K - S_{t=1}^2)^+ & (K - S_{t=2}^2)^+ & \dots & \dots & \dots & (K - S_{t=N}^2)^+ \\ \dots & \dots & \dots & \dots & \dots & \dots \\ (K - S_{t=1}^i)^+ & \dots & (K - S_{t=n}^i)^+ & (K - S_{t=n+1}^i)^+ & \dots & (K - S_{t=N}^i)^+ \\ \dots & \dots & \dots & \dots & \dots & \dots \\ (K - S_{t=1}^M)^+ & (K - S_{t=2}^M)^+ & \dots & \dots & \dots & (K - S_{t=N}^M)^+ \end{pmatrix} \tag{25}$$

avec i = 1,...,M nombre de simulation Monte Carlo et n = 1,...,N nombre de pas de temps.
La décision optimale doit s'exécuter si la valeur de l'exercice immédiat est positive et plus grande ou égale à la valeur de continuation. Normalement, le payoff de l'exercice immédiat au pas de temps n est connu mais comment on connait la valeur de continuation. Utilisons (20),(23),(**??**), on les solutionne par la méthode des moindres carrés. Pour calculer la valeur de continuation, il faut d'abord savoir la valeur du prix du sous-jacent au pas de temps *n+1* pour calculer la valeur de continuation au pas de temps n. On calcule les prix du sous-jacent au pas de temps N à la date d'échéance T, les S_T^i en utilisant (4) ou (5). À ce point, on se demande comment valoriser le prix du sous-jacent au pas de temps N-1, le $S_{T-\Delta t}^i$. Regardant l'équation (3), elle donne un moyen de calculer le prix du sous-jacent au pas de temps suivant. Mais ici c'est le pas de temps précédant qui nous intéresse. Nous voyons que le calcul de prix du sous-jacent S_t depend aussi du movement Brownien W_t. Nous allons voir comment un pont Brownien [6] peut être utilisé pour cela.
Supposons que W_T est le mouvement Brownien determié au pas de temps N (T = N.Δt), donc $W_{T-\Delta t}$ est trouvé par la formule de calcul de pont Brownien (26) suivant :

$$W_{T-\Delta t} = \frac{T - \Delta t}{T} W_T + \sqrt{\frac{(T - \Delta t)\Delta t}{T}} G$$

avec G une variable aléatoire gaussienne centrée et réduite. Alors :

$$W_T - W_{T-\Delta t} = \frac{\Delta t}{T} W_T - \sqrt{\frac{(T - \Delta t)\Delta t}{T}} G$$

plus généralement on a :

$$W_{T-(n+1)\Delta t} = \frac{T-(n+1)\Delta t}{T-n\Delta t} W_{T-n\Delta t} + \sqrt{\frac{(T-(n+1)\Delta t)\Delta t}{T-n\Delta t}} G \tag{26}$$

$$W_{T-n\Delta t} - W_{T-(n+1)\Delta t} = \frac{\Delta t}{T-n\Delta t} W_{T-n\Delta t} - \sqrt{\frac{(T-(n+1)\Delta t)\Delta t}{T-n\Delta t}} G \tag{27}$$

Pour calculer le prix de sous-jacent au temps $T - \Delta t$, par rapport à la formule du mouvement brownien géométrique(3), on se sert de la formule :

$$S_t = S_0 \exp\left((r - \sigma^2/2)t + \sigma W_t\right)$$

ainsi

$$S_t = S_0 \exp\left((r - \sigma^2/2)(t-0) + \sigma(W_t - W_{t=0})\right)$$

où bien sûr, le mouvement Brownien au temp zéro égal 0. Alors on déduit que :

$$S_t = S_{t-\Delta t} \exp\left((r - \sigma^2/2)(t-(t-\Delta t)) + \sigma(W_t - W_{t-\Delta t})\right)$$

et en remplacant t par T, on a :

$$S_T = S_{T-\Delta t} \exp\left((r - \sigma^2/2)\Delta t + \sigma(W_T - W_{T-\Delta t})\right)$$

Finalement, on a la formule générale de calcul récursif de sous-jacent en pas de temps :

$$S^i_{T-(n+1)\Delta t} = \frac{S^i_{T-n\Delta t}}{exp((r - \sigma^2/2)(\Delta t) + \sigma(W_{T-n\Delta t} - W_{T-(n+1)\Delta t}))} \tag{28}$$

Alors le payoff corespondant est :

$$max(K - S^i_{T-(n+1)\Delta t}, 0) \tag{29}$$

En le faisant récursivement jusqu'au pas de temps 1, on obtient la matrice de prix (24) et aussi la matrice de payoff (25). Apres avoir calculé l'ésperance conditionnelle, nous faisons la comparaison entre le payoff actuel et la valeur de continuation. Lorsque la décision optimale est prise, on marque 1 sur une matrice de temps d'arrêt optimal, dont le temps optimal est le premier 1 en partant de t = 0. À la fin de l'analyse récursieve au pas de temps 1, on a aussi une matrice du temps d'arrêt optimal Opt(i,n).
La prime à payer pour ce Put américain est de

$$\text{Prime à la date } (t=0) = \mathbb{E}\left((K - S^i_n)^+ \mathbb{1}_{(Opt_{i,n}=1)}\right). \tag{30}$$

ou bien :

$$\text{Prime à la date } (t=0) = \frac{1}{nbMC} \sum_{i=1}^{nbMC} ((K - S^i_n)^+ \mathbb{1}_{(Opt_{i,n}=1)}). \tag{31}$$

5.8.1 La méthode de moindre carré (The Least Squares Method - LSM)

Supposez une suite de couples de points (x_i, y_i) avec i = 1,2,...,M. Trouvez une fonction f pour que $f(x_i) + \epsilon \simeq y_i$. La solution de ce problème que on suppose la fonction $f(x) = ax^2 + bx + c$ où a,b,c sont des coefficients inconnus et ϵ est une perturbation aléatoire suivant la loi normale

$N(0, \sigma^2)$. La fonction *f(x)* est la fonction de base et on peut trouver des coefficients a,b,c par la minimisation de l'équation suivante :

$$min \sum_{i=1}^{M} (y_i - f(x_i))^2 \tag{32}$$

C'est pourquoi cette méthode s'appelle 'moindre carré' (Least Squares), ref [7]. Pour résoudre cette minimisation, on se sert de l'analyse de régression linéaire, car la fonction f est linéaire avec a,b,c C'est à dire :
Trouvez a,b,c pour minimiser la fonction suivante :

$$L(a, b, c) = \sum_{i=1}^{M} (y_i - (ax_i^2 + bx_i + c))^2 \tag{33}$$

Alors a,b,c sont des solutions du système [11] :

$$\frac{\partial L}{\partial a} = 0, \frac{\partial L}{\partial b} = 0, \frac{\partial L}{\partial c} = 0$$

En fait, il y a plusieurs types de fonctions de base qu'on peut choisir selon la précision voulue pour l'estimation de la valeur de continuation :

$$Lineair\ simple : f(x) = bx + c \tag{34}$$

$$Quadratic : f(x) = ax^2 + bx + c \tag{35}$$

$$Exponentiel : f(x) = be^x + c$$

$$Periodic : f(x) = bsin(ax) + c$$

La liste de ces fonctions est ordonnée en fonction du degré de difficulté croissant pour obtenir la solution. Dans notre application nous utilisons la fonction de base linéaire simple $f(x) = bx+c$ pour réduire le coût de temps de calcul tout en assurant une bonne qualité de la fonction f, référence [12].
Parce que nous devons escompter le ϵ de la formule de fonction de base, nous notons $\widehat{b}, \widehat{c}$ les 2 approximations de b et c. Donc $\widehat{f}(x) = \widehat{b}x + \widehat{c}$ est la fonction d'approximation de la fonction $f(x) = bx + c + \epsilon$ L'estimation des coefficients $\widehat{b}$,$\widehat{c}$ est trouvée facilement par résolution du système des deux équations (36), (37) :

$$\widehat{c} \sum_{i=1}^{M} x_i^2 + \widehat{b} \sum_{i=1}^{M} x_i = \sum_{i=1}^{M} x_i y_i \tag{36}$$

$$\widehat{c} \sum_{i=1}^{M} x_i + \widehat{b} \sum_{i=1}^{M} 1 = \sum_{i=1}^{M} y_i \tag{37}$$

En résolvant ce système, on a :

$$\widehat{b} = \frac{\overline{xy} - \overline{xy}}{\overline{x^2} - (\overline{x})^2} \tag{38}$$

$$\widehat{c} = \overline{y} - \widehat{b}\overline{x} \tag{39}$$

où

$$\overline{xy} = \frac{1}{M}\sum_i x_i y_i,\ \overline{x} = \frac{1}{M}\sum_i x_i \ et\ \overline{y} = \frac{1}{M}\sum_i y_i,\ avec\ i = 1, 2, ..., M$$

Ayant $\hat{b}$ et $\hat{c}$, comment estimer b et c : nous calculons la variance de $\hat{b}$ et $\hat{c}$ pour savoir l'intervalle de confiance de b et c avec la probabilité de $1-\alpha$ que b et c sont dans cet intervalle.

$$\frac{\widehat{b} - b}{Var(\widehat{b})} \sim T(M-2)\ et\ \frac{\widehat{c} - c}{Var(\widehat{c})} \sim T(M-2)$$

Avec :

$$Var(\widehat{b}) = \sqrt{\frac{\widehat{\sigma^2}/n}{\overline{x^2} - (\overline{x})^2}} \tag{40}$$

$$Var(\widehat{c}) = \sqrt{\frac{\widehat{(\sigma^2}/n)\overline{x^2}}{\overline{x^2} - (\overline{x})^2}} \tag{41}$$

$$\widehat{\sigma^2} = \frac{\sum (y_i - \widehat{f}(x_i))^2}{M-2} \tag{42}$$

et T(M-2) suit la loi Student. On a à la fin les deux coefficients dans l'intervalle de confiance

$$b \subset (\widehat{b} \pm Var(\widehat{b}) Student_{\alpha/2}^{M-2}) \tag{43}$$

$$c \subset (\widehat{c} \pm Var(\widehat{c}) Student_{\alpha/2}^{M-2}) \tag{44}$$

En appliquant cette méthode avec les valeurs du payoff $(K - S_n^i)^+$ du pas de temps n avec $i = 1, ..., M$ de la matrice des payoff (25), nous trouvons les coefficients b,c de la fonction de base f(x). De cela nous avons l'ésperance conditionnelle (ou la valeur de continuation) de ce pas de temps n, qui peut être exprimé comme combinaison linéaire des fonctions de base p(X), comme (34), (35) ...

$$\mathbb{E}[e^{-r(T-j)}(K - S(\tau_{j+1}^*))^+ \mid S(\tau_j)] = \sum_{k=0}^{\infty} a_k p_k(X) \tag{45}$$

Dans notre cas, nous utilisons les deux premières fonctions de base (34), (35), alors nous avons la formule de l'éspérance conditionelle. Si nous utilisons la première fonction de base (34), on a :

$$\mathbb{E}[e^{-r(T-j)}(K - S(\tau_{j+1}^*))^+ \mid S(\tau_j)] = b * (S\tau_j) + c \tag{46}$$

dans la cas de la seconde (35), on a :

$$\mathbb{E}[e^{-r(T-j)}(K - S(\tau_{j+1}^*))^+ \mid S(\tau_j)] = a * (S\tau_j)^2 + b * (S\tau_j) + c \tag{47}$$

avec i = 1,2,...,M nombre de simulations de Monte Carlo.

Une autre facon d'estimer la précision de la valeur de continuation, on consiste ajouter la fonction du payoff $(K - S(\tau_j), 0)$ dans la fonction de base finale. Cependant on ne l'aborde pas encore dans ce rapport.

5.8.2 L'algorithme de la valorisation de l'option américnne simple en parallèle

Quelque déclarations des variables utilisées dans l'algorithme :
Par rapport de la matrice de prix du sous-jacent (24), on a

$$\begin{pmatrix} \cdots & \cdots \\ S^i_{t=n-1} & S^i_{t=n} \\ \cdots & \cdots \\ \underbrace{\quad} & \underbrace{\quad} \\ cashflow\_new[i] & cashflow\_old[i] \end{pmatrix}$$

- Le tableau *browian_ old[i]* est utilisé pour stocker des valeurs du mouvement brownien au pas de temps n pour calculer la valeur du mouvement brownien au pas de temps n-1, par la formule du pont brownien..
- Le tableau *payoff_ optimal[i]* est utilisé pour stocker des valeurs de payoff optimal.
- Le tableau *payoff_ exercice[i]* est utilisé pour stocker des valeurs d'exercice.
- Le tableau *payoff_ continue[i]* est utilisé pour stocker des valeurs de continuation.

/* À la réception d'un packet, chaque processeur calcule et M est le nombre de simulations de Monte Carlo d'un packet */
$N = nbIntervalle$
for $i = 0$ to $M - 1$ **do**
Tirage d'une valeur $\epsilon_i \sim$ ***N(0,1) suivant la loi gaussienne centrée réduite***
$S^i_T = S_0 exp((r - \sigma^2/2)T + \sigma\epsilon_i\sqrt{T})$
Stocker des données nécessaires
$cashflow\_old[i] = S_T$
$browian\_old[i] = \epsilon_t\sqrt{T}$
$payoff\_optimal[i] = max(K - S_T, 0)$
end for
$n = N - 1$
while n > 0 **do**
for $i = 0$ to $M - 1$ **do**
/* Calculer la différence du pont du movement brownien récursif */
Tirage d'une valeur $\epsilon_i \sim$ ***N(0,1) suivant la loi gaussienne centrée réduite***
$DifferenceBrownien = W_{old} - W_{new}$ (27)
/* Calculer le prix du sous-jacent récursif */
$S_{T-(N-n)\Delta t} = cashflow\_old[i]$
$$S^i_{T-(N-n+1)\Delta t} = \frac{S^i_{T-(N-n)\Delta t}}{exp((r - \sigma^2/2)(\Delta t) + \sigma(W_{old} - W_{new}))}$$
$cashflow\_new[i] = S^i_{T-(N-n+1)\Delta t}$
$browian\_old[i] = W_{T-(N-n+1)\Delta t}$
if $max(K - S_{T-(N-n+1)\Delta t}, 0) > 0$ **then**
$payoff\_exercice[i] = K - S_{T-(N-n+1)\Delta t}$
$payoff\_continue[i] = max(K - S_{T-(N-n)\Delta t}, 0)e^{-r*n*\Delta t}$
end if
end for
Trouver des coefficients pour la formule de l'espérance conditionelle du pas de temps n : E_n ***(??) par la méthode de moindre carré***
for $i = 0$ to $M - 1$ **do**
if $payoff\_exercice[i] > E_n(cash\_flow[i])$ **then**
$payoff\_optimal[i] = payoff\_exercice[i] * exp(-r * n * \Delta t)$
end if
end for
$n = n - 1$
end while

for $i = 0$ to $M - 1$ **do**
$sumPut = sumPut + payoff\_optimal[i]$
$sumPut\_carree = sumPut\_carree + (payoff\_optimal[i])^2$
end for
/* Chaque processeur renvoie le sumPut et sumPut_ carree et le server calcule le prix d'option */
$moyennePut = \frac{1}{M * nbPacket} * \sum_1^{nbPaket} sumPut$
$variancePut = \frac{1}{M * nbPacket} * \sum_1^{nbPaket} sumPut\_carree - moyennePut^2$
$put = moyennePut$

5.9 Les benchmarks

Pour chaque algorithme de valorisation au dessus, nous avons fait la comparaison des temps de calculs entre une architecture "mono-processeur" et une architecture "multi-processeurs".

5.9.1 Les benchmarks de la valorisation de l'option européenne simple

Nous allons montrer des applications numériques en utilisant les valeurs suivantes pour l'option :
Prix du sous jacent à t=0 : 42 euros, Prix d'execice (strike) : 43 euros
taux d'interêt r = 0.1, volatilité du marché : 0.05 et date d'échéance du contrat T = 1 an

Dans les tableaux suivants, nous donnons les temps de calcul(en secondes) du put , du call et de l'intervalle de confiance en utilisant les paramètres suivants :
nombre total de simulations : 1 000 000 et 5 000 000
nombre de calculs repartis par simulateur (un paquet) : 1000 et 5000

Nous faisons des benchmarks sur les processeurs de l'équipe OASIS et sur le cluster Grid'5000 du site de Sophia Antipolis.
Les machines d'OASIS : processeur Intel (XEON), 2.00 GHz / cache 512K / Mem 1G.
Les noeuds du cluster Grid'5000 : processeur AMD Opteron 246, 2.00 GHz / cache 1M / 400MHz / Mem 2G.

1 000 000	5 000 000
77	400

Tableau 1-1 : Temps calcul (en sec) pour un intervalle de temps mensuel ($\Delta t = \frac{1}{12}$) et un seul processeur de l'équipe OASIS

	1 000 000	5 000 000
1000	60	322
5000	17	96

Tableau 1-2 : Temps calcul (en sec) pour un intervalle de temps mensuel ($\Delta t = \frac{1}{12}$) et trois processeurs de l'équipe OASIS

1 000 000	5 000 000
3,5	15,6

Tableau 1-3 : Temps calcul (en sec) pour un intervalle de temps mensuel ($\Delta t = \frac{1}{12}$) et un processeur sur Grid'5000

	1 000 000	5 000 000
1000	6	26
5000	1,5	7,6

Tableau 1-4 : Temps calcul (en sec) pour un intervalle de temps mensuel ($\Delta t = \frac{1}{12}$) et trois processeurs sur Grid'5000

Avec un intervalle de temps $\Delta t = \frac{1}{360}$

1 000 000	5 000 000
450	2200

Tableau 2-1 : Temps calcul (en sec) pour un intervalle de temps quotidien ($\Delta t = \frac{1}{360}$) et un seul processeur de l'équipe OASIS

	1 000 000	5 000 000
1000	158	736
5000	120	600

Tableau 2-2 : Temps calcul (en sec) pour un intervalle de temps quotidien ($\Delta t = \frac{1}{360}$) et trois processeurs de l'équipe OASIS

1 000 000	5 000 000
94	440

Tableau 2-3 : Temps calcul (en sec) pour un intervalle de temps quotidien ($\Delta t = \frac{1}{360}$) et un processeur sur Grid'5000

	1 000 000	5 000 000
1000	35	164
5000	33	156

Tableau 2-4 : Temps calcul (en sec) pour un intervalle de temps quotidien ($\Delta t = \frac{1}{360}$) et trois processeurs sur Grid'5000

Les temps d'exécution augmentent linéairement lorsque le nombre de simulations ou le nombre d'intervalle de temps augmentent. Lorsque le nombre de calculs par simulateur augmente (on passe d'une taille de paquet de 1000 à 5000), les temps sont aussi diminués, ce qui prouve que les temps de communications réseau sont non négligeables et donc qu'il faut en faire le moins possible. Par contre les temps sont divisés par 3 pour 360 intervalles alors qu'ils ne le sont que par 1,5 pour 12 intervalles lorsque l'on passe de 1 à 3 processeurs car les temps de communications ont plus de poid sur les petits calculs. Autrement dit lorsque l'on multiplie par trois le nombre de processeurs, les calculs sont plus rapides, mais on peut remarquer que la baisse est plus significative avec un nombre d'intervalles élevé.

5.9.2 Les benchmarks de la valorisation de l'option européenne à barrière

Prix du sous jacent à t=0 : 43 euros, Prix d'execice (strike) : 44 euros
taux d'interêt r = 0.1, volatilité du marché : 0.05 et date d'échéance du contrat T = 1 an.

Pour l'application numérique la barrière est de 42 euros ce qui veut dire que l'on calcule des options DOWN. Les temps d'execution concernent le calcul des options DOWN OUT.

1 000 000	5 000 000
16	72

Tableau 3-1 : Temps calcul (en sec) pour un intervalle de temps mensuel ($\Delta t = \frac{1}{12}$) et un seul processeur

	1 000 000	5 000 000
1000	6	28
5000	4	21

Tableau 3-2 : Temps calcul (en sec) pour un intervalle de temps mensuel ($\Delta t = \frac{1}{12}$) et trois processeurs de l'équipe OASIS

1 000 000	5 000 000
1,7	8,4

Tableau 3-3 : Temps calcul (en sec) pour un intervalle de temps mensuel ($\Delta t = \frac{1}{12}$) et un processeur sur Grid'5000

	1 000 000	5 000 000
1000	5	25
5000	1,2	5

Tableau 3-4 : Temps calcul (en sec) pour un intervalle de temps mensuel ($\Delta t = \frac{1}{12}$) et trois processeurs sur Grid'5000

Avec un intervalle de temps $\Delta t = \frac{1}{360}$

1 000 000	5 000 000
265	1389

Tableau 4-1 : Temps calcul (en sec) pour un intervalle de temps quotidien ($\Delta t = \frac{1}{360}$) et un seul processeur de l'équipe OASIS

	1 000 000	5 000 000
1000	95	476
5000	87	451

Tableau 4-2 : Temps calcul (en sec) pour un intervalle de temps quotidien ($\Delta t = \frac{1}{360}$) et trois processeurs de l'équipe OASIS

1 000 000	5 000 000
25	120

Tableau 4-3 : Temps calcul (en sec) pour un intervalle de temps quotidien ($\Delta t = \frac{1}{360}$) et un processeur sur Grid'5000

	1 000 000	5 000 000
1000	11	60
5000	9	44

Tableau 4-4 : Temps calcul (en sec) pour un intervalle de temps quotidien ($\Delta t = \frac{1}{360}$) et trois processeurs sur Grid'5000

Nous pouvons tirer les mêmes conclusions sur les temps de calculs que dans le cas des options simples.

5.9.3 Les benchmarks de la valorisation de l'option européenne sur panier

Nous considérons un panier de 40 actifs.
- Prix d'un actif à t=0 : 42 euros
- Prix d'exercice du panier (strike) : 44 euros
- La corrélation entre les 40 actifs est 0.005
- Taux d'intérêt r constant = 0.1
- Volatilité d'un actif : 0.25
- Poids des 40 actifs est uniform
- Date d'échéance du contrat T = 1 an avec 360 pas de temps, $\Delta t = \frac{1}{360}$.

Nous éffectuons un nombre de 1000000 simulations. Normalement, la simulation avec le modèle mono-processeur (sur une machine d'OASIS, processeur Intel (XEON), 2.00 GHz, cache 512K, Mem 1G) coûte environ 9h, c'est inacceptable sur les marchés financiers. L'ideé est de trouver un moyen d'obtenir le résultat au bout d'à une minute. Successivement, nous utilisons 72, 144, 260 processeurs sur le Grid'5000 sur un ou plusieur cluster pour réduire le temps de calcul à moins d'une minute. La figure (3) présente l'influence du nombre de processeur sur le temps de valorisation total. La taille du packet est expliqué dans la figure (4).
Pour les benchmarks sur panier, les calculs sont complexifiés par le nombre d'actifs et la simulation de la corrélation. Ils prennent plus de temps que les autres calculs de valeurs d'option mais les conclusions sur la linéarité et les baisses significatives avec l'augmentation des processeurs, donc des simulateurs restent valable.

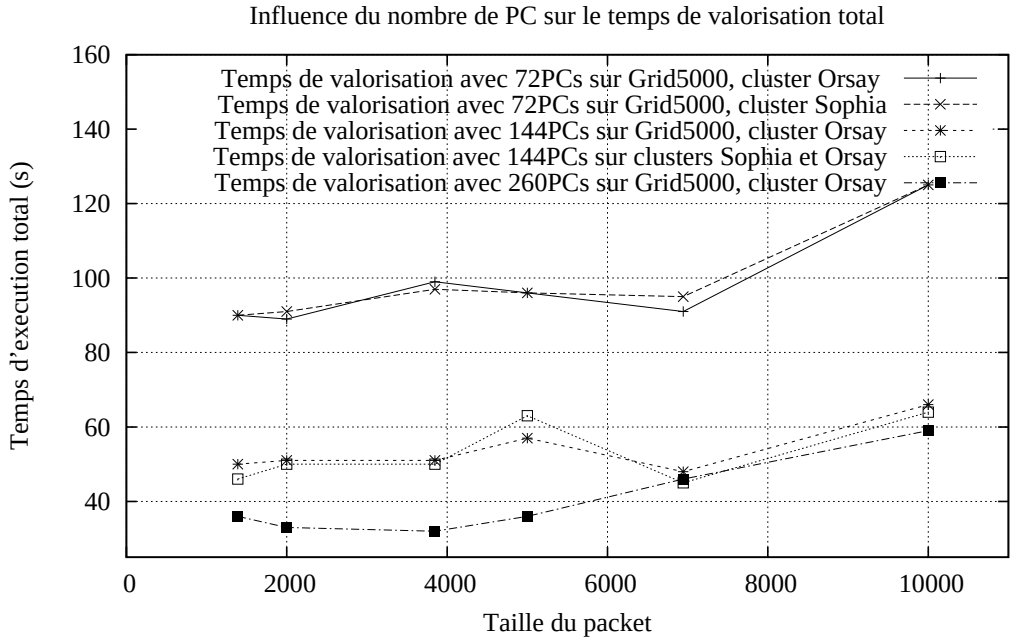

FIG. 3 – Influence du nombre de PC sur le temps de valorisation total (option européenne sur panier)

5.9.4 Les benchmarks de la valorisation de l'option américaine simple

Pour vérifier la justesse de notre implémentation, nous calculons le prix du put de l'option américaine par la méthode de simulation et comparons avec les résultats obtenus par la méthode de différence finie de Longstaff - Schwartz (2001) [7].
Nous considérons le contrat de Put suivant :
– Prix de l'actif à t = 0 : 36 euros et 44 euros
– Prix d'exercice (strike) : 40 euros
– Taux d'intérêt r sans risque et constant = 0.06
– Volatilité : 0.2
– Date d'échéance du contrat T = 1 et 2 ans. Alors $\Delta t = \frac{1}{360}$ et $\frac{2}{360}$.
La Figure numéro 5 vous montre nos résultats.

Etant donné l'objectif de notre travail, nous présentons aussi les benchmarks du modèle multi-processeurs comparer au modèle mono-processeur pour comparer les performances. La Figure 6 présente le temps total de calcul entre les deux modèles en vérifiant la variance.
Nous considérons le contrat du Put suivant :
– Prix de l'actif à t = 0 : 36 euros
– Prix d'exercice (strike) : 40 euros
– Taux d'intérêt r sans risque et constant = 0.06
– Volatilité : 0.2
– Date d'échéance du contrat T = 1, $\Delta t = \frac{1}{360}$.
Nous avons marqué que plus le nombre de simulation de Monte Carlo est élevé, plus la variance est petite, comme pour la méthode de Monte Carlo classique.

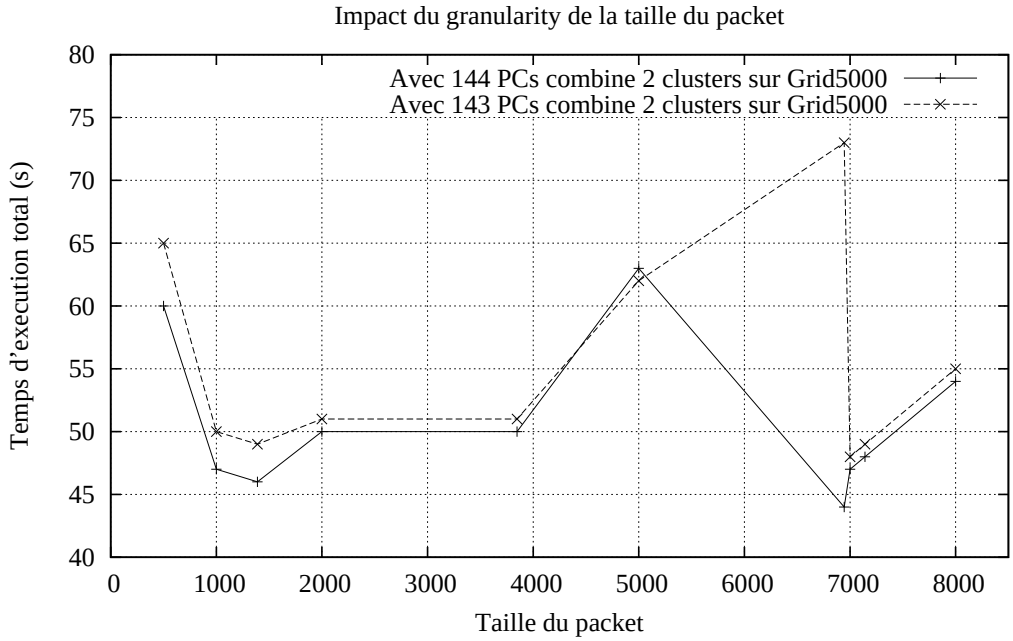

FIG. 4 – Impact de la taille du packet

5.9.5 La prolèmatique du *'load balancing'*

La Figure (4) montre les temps d'exécution de l'évaluation du prix d'une option sur panier en fonction de la taille d'un paquet de simulation sur une grille de 144 simulateurs et sur la même grille mais défectueuse de 143 simulateurs (1 simulateur en panne). Chaque simulateur récupére un paquet, il ne demande un autre qu'après avoir fini le précédent. Sur la figure (4) le temps d'exécution de petite taille est rapide, mais les nombreuses communications ralentissent pour un paquet la simulation globale. Dans le cas d'une grille de 143 simulateurs en utilisant la taille optimale (nbMC/nbSimulateur = 6945) on constrate que le temps global fait le double de celui utilisont 144 simulateurs. En effet, un des 143 simulateurs doit récupérer le travail du simulateur en panne. Si on augmente légérement la taille du paquet (par exemple jusqu'à 7000) on se rapproche du temps optimal minimal avec 144 simulateurs. Le travail futur consiste à essayer a'ajuster la taille d'un paquet automatiquement, référence [15].

6 Conclusion

6.1 Bilan du travail

D'un point de vue financier, l'application distribuée est devenue beaucoup plus réaliste et l'on peut confirmer l'intêret des infrastructure distribuées dans le monde de la finance. Dans le monde financier, être capable de prendre une décision quasiment en temps réel est crucial. L'application distribuée permet d'apporter beaucoup d'améliorations dans les temps de calculs ou de résoudre des problèmatiques qui ont besoins d'énorme calculs.
L'objectif du travail était de pouvoir prendre en compte un produit financier incontournable : les options, un très populaire produit dans le marché financier aujourd'hui, appliqué à une architecture distribuée. La valorisation des options demande des énormes calculs. Dans le cas d'option sur panier de 40 actifs, il faut 100 minutes de calcul sur un seul processeur. En premier, les résultats obtenus par des simulations nous montrent comment le temps de valorisation des options peut être amélioré par l'augmentation des ressources fournies par une grille. Mais aussi, on rencontre de nouvelles difficultés car les questions que l'on doit se poser ne sont plus seulement d'ordre fonctionnel ou technique mais aussi à la d'ordre répartition des calculs et des communications réseau. En cesens, la répartition des calculs était fixée dans le rapport, or nous avons pu de voir que si une gestion dynamique était faite par les éléments de gestion des calculs (le serveur et les sous-serveurs) alors on gagnerait en efficacité. Le point serait de trouver l'ordinateur qui calcule le plus rapidement un certain nombre de calculs donnés pour optimiser la répartition du travail de cette application distribuée . Une présentation de ce rapport fait l'objet de la publication [15].

6.2 Perspectives

Cependant, il reste à apporter des amélioration à notre architecture, au niveau de l'informatique et aussi des mathématiques financières. D'abord, pour l'informatique, nous avons dit que l'objectif futur est d'aboutir à une infrastructure logicielle ouverte pour tous les calculs financiers, dans la quelle, il sera facile d'intégrer de nouvelles méthodes de valorisation, de construire un mécanisme automatique de gestion de la taille du paquet de simulation et surtout de profiter de la capacité de communication qu'il existe entre les simulateurs grace à l'utilisation de ProActive. Ensuite, au niveau des mathématiques financières, dans le cas américain l'analyse du choix de la fonction de base utilisée pour la régression et l'ajustement de la fonction de base par la fonction du payoff ne sont pas encore examinés à fond. Par ailleurs, nous valorisons d'option américaine en parallèle, plus précisement nous valorisons plusieurs petites options du fait de la difficulté de stockage des données de chaque simulateur pendant le calcul (mémoire disponible implique que l'on peut simuler au plus 1.200.000 simulations par simulateur). En profitant de la capacité de communication des objets actifs entre eux, ce problème pourait être résolu par la combinaison des données de tous les simulateurs, nous aurions alors une méthode de valorisation des options américaines en grande taille selon le nombre de Monte Carlo (la taille souhaitée est plus de 10.000.000). Grâce à cela, nous pourons aussi savoir quel est le nombre total de simulations qui est suffisant pour la valorisation des options américaine, et ce en vérifiant la variance et la précision de résultat.

Références

[1] Référence le site web de ProActive. *ProActive* http ://www-sop.inria.fr/oasis/proactive/.

[2] Référence le site web de Grid'5000. *Grid'5000* https ://www.grid5000.fr/.

[3] M. Bossy. *Modèles mathématiques continus pour la finance et l'assurance* Master Pro IMAFA, Ecole Polytech'Nice Sophia Antipolis, France.

[4] S. Bezzine. *Rapport du stage de fin d'étude.* Supélec, été 2005 .

[5] J. Gentle. *Statistics and Computing*, chapter Random number generation and Monte Carlo methods. Springer-Verlag, 1998.

[6] P. Glasserman. *Monte Carlo methods in financial engineering*, volume 53 of *Applications of Mathematics.* Springer-Verlag, 2004.

[7] F. Lonstaff and E. Schwartz. *Valuing American option by simulation - A simple Least-Squares Approach.* Springer-Verlag, 2001.

[8] J. Hull. *Options, Futures and Other Derviatives.* Prentice Hall, 2005.

[9] J. Hull. *Risk Management and Financial Institutions.* Prentice Hall, 2006.

[10] D. Lamberton and B. Lapeyre. *Introduction to Stochastic Calculus Applied to Finance.* Chapman and Hall, 1996.

[11] Duc-Nghia. Nguyen. *Introduction to Scientific Computing.* Ecole Polytechique de Hanoi, Vietnam, 2002.

[12] M. Moreno and JF. Navas. *On the Robustness of Least-Squares Monte Carlo (LSM) for Pricing American Derivatives.* 2001.

[13] A. Newell. *Parallel Random Number Generators in Java.* PhD thesis, University of Adelaide, Australia, 2003.

[14] Référence le site web de *'Parallel Random Number Generators' The Scalable Parallel Random Number Generators Library (SPRNG) for ASCI Monte Carlo Computations.* http ://sprng.cs.fsu.edu/

[15] F. Baude, M. Bossy, VD. Doan, L. Henrio, S. Bezzine, V. Galtier, S. Vialle. *A Fault Tolerant and Multi-Paradigm Grid Architecture for Time Constrained Problems - Application to Financial Option Pricing.* 2nd IEEE International Conference on E-Science and Grid Computing, December 2006, Amsterdam, Netherlands.