M2 IF Apprentissage projet composant

Spécifications composant 2

Boucle de Monte Carlo

|  |
| --- |
| **Groupe 3** |
| Lauren BARTHELEMY |
| Maxime BETTY |
| Long DO |
| Thardsajini SATKUNARAJAH |

|  |  |  |  |
| --- | --- | --- | --- |
| Version doc | Date | Auteur(s) | Modifications |
| 1.0 | 27/01/2015 | Jose Luu | Version initiale |
| 1.1 | 27/01/2015 | Jose Luu | Modification pour exemple |
| 1.2 | 23/02/2015 | Groupe 3 | Création des spécifications du composant « Monte Carlo » |
| 1.3 | 07/04/2015 | Groupe 3 | Mise à jour des fonctions et erreurs |
| 1.4 | 12/04/2015 | Groupe 3 | Mise à jour des erreurs |
| 1.5 | 15/04/2015 | Groupe 3 | Mise à jour des erreurs et Plan de tests |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Ce document a pour but de rassembler les spécifications concernant le composant 2 « Boucle Monte Carlo » du projet de classe.

1. Fonctionnement du composant

Les fonctions du module Monte-Carlo sont la gestion des itérations et du séquençage. L’objectif du composant est de calculer l’espérance du PayOff à partir de la valeur finale des N itérations. Ce composant est en interactivité avec d’autres composants. Voici un schéma explicatif du mode de fonctionnement du composant demandé :

 Composant 1 : XLL Interface

Composant 6 : Constructeur de chemin

Composant 3 : GNA Gaussien

Composant 7 : Echantillonnage

1

2

3

4

5

6

Composant 2 : Monte Carlo

*Calcul de l’espérance du Payoff*

7

1 – Réception de l’appel doMonteCarlo() du composant 1

2 – Appel getPath() du composant 6

3 – Retour de getPath() : vecteurs de valeurs du composant 6

4 – Appel de pricePath() du composant 7

5 – Retour de pricePath() du composant 7

6 – Le composant 2 calcul l’espérance du Payoff

7 – Retour de doMonteCarlo() au composant 1

Dans la suite du document, chacune des étapes du schéma sont explicitées afin de mieux comprendre le fonctionnement du composant et son mode de réalisation.

1. Les étapes
   1. Réception du composant 1

Notre module reçoit du composant “XLL interface” un certain nombre de paramètres :

* le nombre d’itérations
* la valeur du Strike
* le numéro du PayOff correspondant au type du PayOff (Américain, Européen …)
* La maturité fixée à 2 ans avec une période journalière soit 252 jours par an et 504 jours au total

double doMonteCarlo(string typePayoff, double maturity, double strike, int nbTrials) ;

* 1. Récupération des n valeurs

Pour récupérer n valeurs de PayOff, nous allons itérer n fois le processus suivant :

***2.2.1 Appel du composant 6***

Notre module appelle le constructeur du chemin (Composant 6) pour obtenir le vecteur des 504 valeurs aléatoires gaussiennes normalisées. En amont, C6 a reçu ces informations du composant 3 « *GNA gaussien* ».

Double\* getPath( double maturite, double spot) ;

***2.2.2 Appel du composant 7***

Notre module envoie le numéro du PayOff, le vecteur des valeurs, le strike et la maturité au composant 7 « *Echantillonnage* ». En retour, il nous fournit la valeur du PayOff. Nous la stockons.

double pricePath(string typePayoff, vector <double> path, double strike, double maturity) ;

* 1. Calcul de l’espérance

A partir des N PayOff reçus, nous calculons l’espérance du PayOff dans la méthode doMonteCarlo().

* 1. Envoi au module 1

Le résultat est renvoyé au module 1 « *XLL Interface* ».

Retour de la fonction doMonteCarlo(string typePayoff, double maturity, double strike, int nbTrials) ;

1. Description des erreurs

Pour le composant, chacune des erreurs est gérée par les exceptions (throws) suivantes :

* 1. Etape 2.1

|  |  |
| --- | --- |
| Nom Fonction : *doMonteCarlo (string typePayoff, double maturity, double strike,  int nbTrials)* | |
| *Erreur :* données négatives | *Negative value* |
| *Erreur :* données manquantes | *Missed data* |

* 1. Etape 2.2

***4.3.1 Etape 2.2.1***

|  |  |
| --- | --- |
| Nom Fonction : *getPath(double maturity, double spot )* | |
| *Erreur :* données négatives | *Negative value* |
| *Erreur :* données manquantes | *Missed data* |

*4.3.2 Etape 2.2.2*

|  |  |
| --- | --- |
| Nom Fonction : *pricePath(string typePayoff, vector<double> path, double strike,  double maturity)* | |
| *Erreur :* données négatives | *Negative value* |
| *Erreur :* données manquantes | *Missed data* |
| *Erreur :* valeur supérieure à 1 000 000 | *Value greater than 1 000 000* |

1. Plan de tests
   1. Interaction vis-à-vis des autres composants

Les composants 1, 6 et 7 sont créés dans une version simplifiée reprenant les fonctions essentielles. Ces composants renvoient des valeurs qui nous permettent de tester les différents scénarios.

* 1. Test du composant

Un main fait appel aux fonctions des différents composants et compare le comportement du composant avec celui attendu. Si le composant n’a pas le comportement souhaité pour ce cas de test, il y aura un message nous indiquant l’erreur. Si tout est correct, il n’y aura donc pas de message affiché d’erreur affiché.

NB : Pour les trois fonctions listées ci-dessous, il est possible qu’une exception soit retournée alors qu’elle n’était pas attendue. Dans ce cas, le message d’erreur suivant apparaitra :  
"FAILED: ‘XXX’ EXCEPTION TRIGGERED WHEREAS DATA ARE CORRECT";  
XXX : Name of the exception

* Getpath :

Si une des valeurs est négative alors le message d’erreur suivant doit être renvoyé :   
"NEGATIVE VALUE";

Si une donnée manque dans le tableau alors le message d’erreur suivant doit être renvoyé :   
"MISSED DATA";

* DoMonteCarlo :

Si une des valeurs (maturity, nombre de test, strike) est négative alors le message d’erreur suivant doit être renvoyé :   
"NEGATIVE VALUE EXCEPTION UNTRIGGERED ";

Si une donnée n’a pas le bon type, le message d’erreur suivant doit être renvoyé :   
"WRONG DATA EXCEPTION UNTRIGGERED ";

* PricePath :

Si une donnée manque dans le tableau alors le message d’erreur suivant doit être renvoyé :   
"MISSING DATA EXCEPTION UNTRIGGERERD ";

Si une des valeurs est négative alors le message d’erreur suivant doit être renvoyé :   
"NEGATIVE VALUE EXCEPTION UNTRIGGERED ";

Si une valeur ressorti est trop importante, le message d’erreur suivant doit être renvoyé :   
"FAILED : VALUE > 1.10^6 EXCEPTION UNTRIGGERERD ";