M2 IF Apprentissage projet composants

Spécifications composant 4

|  |
| --- |
| **Groupe 2** |
| Nicolas EZE |
| Vincent JOLY |
| Ricchie PHILIPPE |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| 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 | 24/02/2015 | Groupe 2 | Initialisation des spécifications |
| 1.3 | 26/02/2015 | Groupe 2 | Modifications mineures |
| 1.4 | 01/03/2015 | Groupe 2 | Modifications mineures |
| 1.5 | 03/04/2015 | Groupe 2 | Modifications majeures |
| 1.6 | 03/04/2015 | Groupe 2 | Modifications mineures |
| 1.7 | 03/04/2015 | Groupe 2 | Modifications mineures |
|  |  |  |  |

SOMMAIRE

Contexte 3

Fonction du composant 3

Cas d’utilisation 1 : Récupération de la matrice de volatilité 4

Entrées 4

Sorties 4

Fonctionnement 4

Cas d’utilisation 2 : Détermination de la volatilité 4

Entrées 4

Sorties 5

Fonctionnement 5

Précisions 6

Interpolation par splines cubiques: 6

Interpolation linéaire entre deux points: 8

Flat extrapolation: 8

Headers 9

Annexes 9

# Contexte

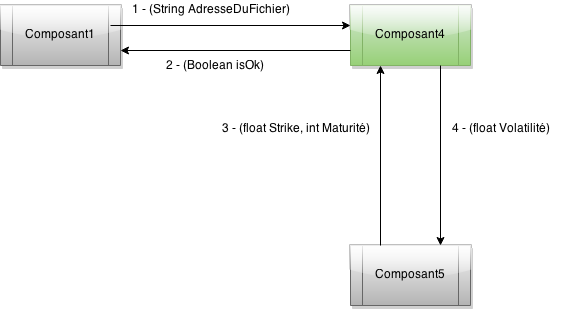
L’objectif final est de valoriser des options européennes ou bermudéenes via un algorithme de Monte Carlo. Cet outil est divisé en 7 composants interagissant les uns avec les autres.

# Fonction du composant

A partir d’un fichier de volatilité, le composant 4 a pour objectif de calculer une volatilité implicite pour n’importe quel couple de Strike/Maturité donné, et ce par un procédé d’interpolation ou d’extrapolation.

Compte-tenu du contexte, il est nécessaire de définir deux cas d’utilisations au composant 4 :

* Le composant 1 dialogue avec le composant 4 afin de vérifier que le fichier de volatilité initial est correct.
* Le composant 5 dialogue avec le composant 4 afin de connaître la valeur d’une volatilité implicite pour un Strike et une maturité donnée.



# 

# Cas d’utilisation 1 : Récupération de la matrice de volatilité

## Entrées

|  |  |  |
| --- | --- | --- |
| **Description** | Format | Commentaire |
| Adresse absolue du fichier de volatilité | String | Exemple : ‘K:\fichierVolatilite.data’ |

## Un exemple du fichier de volatilité fourni :

## 

## Sorties

|  |  |  |
| --- | --- | --- |
| **Description** | Format | Commentaire |
| Bonne récupération du fichier de volatilité | Booléen | Renvoie True si le fichier a été correctement récupéré et est lisible  False sinon |

## Fonctionnement

Le composant 1 dialogue avec nous en nous faisant parvenir l’adresse absolue du prétendu fichier de volatilité. Notre composant aura alors pour mission d’explorer cette adresse afin de vérifier que ce fichier existe bien et qu’il est lisible.

Un fichier est lisible s’il est non-vide et que toutes les lignes sont au format suivant :

float strike+(double espace)+int maturité+(double espace)+float volatilité

Le strike, la maturité et la volatilité doivent être non-négatifs et séparés d’un double espace.

Si c’est le cas, il répond au composant 1 par un booléen TRUE, et FALSE sinon.

# Cas d’utilisation 2 : Détermination de la volatilité

## Entrées

|  |  |  |
| --- | --- | --- |
| **Description** | Format | Commentaire |
| Strike | Float | Obligatoire |
| Maturité (en années) | Integer | Obligatoire |

## Sorties

|  |  |  |
| --- | --- | --- |
| **Description** | Format | Commentaire |
| Valeur de volatilité | Float |  |

## Fonctionnement

**Dans un premier temps**, le composant devra déterminer *n-1* splines cubiques par Strike fourni dans le fichier de volatilité initiale.  
*n* correspondant au nombre de maturité distinctes disponibles dans le fichier de volatilité initiale. Par exemple, avec la matrice de volatilité implicite suivante :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | Maturité (en années) | | |
| 1 | 2 | 3 |
| Strike (en %) | 120 | **12.2** | **8.2** | **7.5** |
| 100 | **10.1** | **7** | **6.2** |
| 80 | **8.7** | **6.4** | **4.9** |

On aura pour mission de déterminer 2 splines cubiques par Strike, donc un total de 6 splines cubiques.

Pour des questions d’optimisation, il est nécessaire de calculer tous les splines dès le chargement du fichier, et de les conserver.

**Dans un second temps**, le composant devra être capable de fournir une volatilité implicite *v* pour un couple quelconque de strike *s* et de maturité *T*.

* À partir du strike fourni, nous déterminons les strikes et les plus proches de .
* Nous en déduisons les deux splines cubiques et déjà calculées, qui sont respectivement associés à et et ce pour l’intervalle de maturité contenant la maturité *T* fournie en paramètre.
* Nous procédons par interpolation linéaire afin de déterminer la fonction (linéaire) passant par et .

La volatilité *v* recherchée est alors le résultat de l’équation.

**Cas particulier:** S’il arrive que la maturité *T* ou le strike *s* fourni soit en dehors des bornes du fichier de volatilité initiale, il s’agira alors de déterminer la volatilité implicite par une *extrapolation flat*.

## Précisions

### Interpolation par splines cubiques:

L’objectif est d’interpoler un nombre *n* de points par une courbe lisse.

Il aurait été possible d’effectuer cette interpolation à l’aide d’un polynôme de degré *n* – 1, mais la courbe produite par cette méthode est généralement « trop ondulée ».

Avec les splines cubiques, l’idée est de calculer un polynôme unique pour chaque sous intervalle entre les points. Alors, la courbe formée par l’agrégation de ces polynômes devra ressembler à une seule et même courbe. Ce qui signifie, mathématiquement parlant, que deux polynômes successifs devront avoir les mêmes dérivées premières et secondes au niveau de leur jointure.

**Méthode de construction de spline cubique entre deux points:**

*Il ne s’agit ici que d’une proposition, n’importe quelle méthode ayant pour but de construire les splines cubiques sera acceptée.*

Soit sont *n*+1 points tels que *i* = 0, 1, …, *n*.

**Alors, on a *n* polynôme du troisième degré** :

**(1) :** avec

qui interpole dans l’intervalle pour de telle sorte que pour

**avec**

**(2) :**

**(3) :** 

**(4) :** 

pour i = 1, …, *n,* avec

**(5) :** 

**(6) :**

**(7) :**  

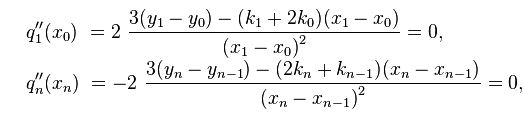
**Ainsi, pour déterminer un polynôme , l’unique difficulté réside dans la variable représentant la 1ère dérivée du polynôme au point *i*, le reste est connu.**

Si la séquence de , …, est telle que  pour *i* = *1, …, n-1* la fonction qui en résulte aura même une dérivée seconde continue.

C’est le cas pour **(3)** et **(4)** si et seulement si :

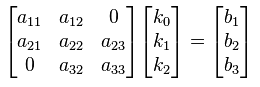
**(8) :**  

Afin d’obtenir des splines « naturels », on définit que pour et , on a :

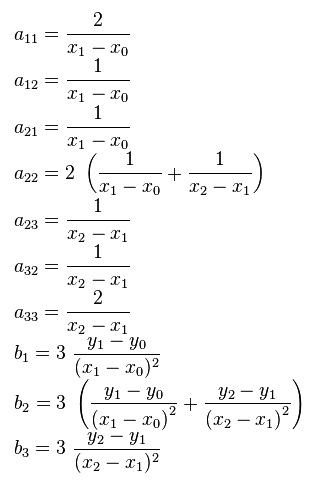
**(9) :**  

**C’est à l’aide des équations (8) et (9) qu’on pourra déterminer les . En effet, on peut alors mettre en place *n*+1 équations linéaires définissant les *n*+1 variables , …, .**

On obtient alors finalement un système linéaire tri-diagonal qu’on résoudra à l’aide d’une méthode de Gauss. Par exemple, avec 3 points, on peut trouver et avec le système suivant :



Avec



Avec les points (-1, 0.5), (0, 0), (3, 3), en résolvant le système, on obtient :

et .

### Interpolation linéaire entre deux points:

Entre deux points et de coordonnées respectives et , l'interpolation est donnée par la formule suivante :

Avec la pente *p* qui s'exprime comme :

### Flat extrapolation:

Soit un point *i* de coordonnées (*x*, *y)*, alors la *flat extrapolation* définit que tous les points *n* d’abscisse *z* < *x* ont pour ordonnée *y*.

Par exemple, avec la matrice de volatilité implicite suivante :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | Maturité (en année(s)) | | |
| 1 | 2 | 3 |
| Strike (en %) | 120 | **12.2** | **8.2** | **7.5** |
| 100 | **10.1** | **7** | **6.2** |
| 80 | **8.7** | **6.4** | **4.9** |

Par *flat extrapolation*, pour un strike de 120% :

* Pour toutes les maturités < 1 an, l’unique volatilité implicite est 12.2.
* Pour toutes les maturités > 3 an, l’unique volatilité implicite est 7.5.

Par *flat extrapolation*, pour une maturité de 2 ans:

* Pour tous les strike > 120%, l’unique volatilité implicite est 8.2.
* Pour tous les strike < 80%, l’unique volatilité implicite est 6.4.

# Headers

|  |  |  |
| --- | --- | --- |
| Méthode | **Retour** | **Commentaires** |
| loadVolatilityData(String nomFichier) | Booléen | Retourne TRUE si le fichier est chargé correctement, FALSE sinon. |
| findVolatility(float strike, float maturity) | float | Retourne la valeur de volatilité pour un strike et une maturité donnée |

# Annexes

Vous trouverez ci-dessous, en guise d’exemple, un exemple de fichier de volatilité implicite que le composant 5 sera capable de fournir après avoir dialogué avec notre composant dans le but qu’on lui fournisse les volatilités implicites dont il a besoin :

