

# Optimisation transport optimal martingale

Latypov Schwartzman Righenzi Benmoussa

CentraleSupélec

Soutenance finale 27/01/2022

# Introduction

# Introduction

- Explosion du marché des options 1970 : besoin de couverture, appétit pour la spéculation, arbitrage croissants

# Introduction

- Explosion du marché des options 1970 : besoin de couverture, appétit pour la spéculation, arbitrage croissants
- Il est essentiel de valoriser correctement une option : valeur intrinsèque + valeur du temps

# Introduction

- Explosion du marché des options 1970 : besoin de couverture, appétit pour la spéculation, arbitrage croissants
- Il est essentiel de valoriser correctement une option : valeur intrinsèque + valeur du temps
- Modèles classiques : Black&Scholes&Merton, arbre binomiaux

# Introduction

- Explosion du marché des options 1970 : besoin de couverture, appétit pour la spéculation, arbitrage croissants
- Il est essentiel de valoriser correctement une option : valeur intrinsèque + valeur du temps
- Modèles classiques : Black&Scholes&Merton, arbre binomiaux
- Risque de modèle : prix suit une distribution log-normale

# Introduction

- Explosion du marché des options 1970 : besoin de couverture, appétit pour la spéculation, arbitrage croissants
- Il est essentiel de valoriser correctement une option : valeur intrinsèque + valeur du temps
- Modèles classiques : Black&Scholes&Merton, arbre binomiaux
- Risque de modèle : prix suit une distribution log-normale
- Model independent finance : résolution du problème de transport optimal martingale pour valoriser une option call

# Problème initial

## Définition

On considère un processus  $(S_n)_{n=0,1,\dots,N}$  correspondant au prix d'un actif sous-jacent quelconque. On considère une option d'achat (call) d'une maturité  $T$  et de strike price  $K$  portant sur le sous-jacent défini précédemment. On rappelle que le profit réalisé par le détenteur du call est :

$$\max(S_N - K, 0)$$

puisque l'option n'est exercée que si  $S_N > K$ .



## Cas $N = 2$

### Définition

On considère alors  $S_0$ ,  $S_1$  et  $S_2$  les trois prix du sous-jacent aux différents instants considérés.

### Prix du call

Dans la suite on va supposer que le prix de l'option aux instants  $n = 1$  et  $n = 2$  est connu. On a ainsi :

$$\text{Prix}((S_1 - K)^+) = C_1(K)$$

$$\text{Prix}((S_2 - K)^+) = C_2(K)$$

## Cas $N = 2$

### Combinaison de calls

Pour toutes fonctions du type  $f(S_1)$  ou  $g(S_2)$ , ces options peuvent être approchées arbitrairement bien par des combinaisons linéaires de calls et donc achetées ou vendues sur des marchés à des prix respectifs

$$\int f(s) \mu_1 ds$$

et

$$\int g(s) \mu_2 ds$$

## Cas $N = 2$

### Optimisation du prix (admis)

Optimiser  $p = F(S_1, S_2)$ ,  $F$  fonction de prix, tel qu'il existe une stratégie menant au profit est équivalent à ce que le processus  $(S_0, S_1, S_2)$  soit une martingale sous une mesure martingale  $\mathbb{Q}$ .

### Optimisation du prix (admis)

Soit  $\mathbb{Q}$  une mesure martingale, alors  $\mathbb{Q}$  est consistante avec le marché si et seulement si

$$\forall i \in \mathbb{N} \quad S_i \sim \mu_i$$

où  $\mu_i$  est une mesure.

# Problème de transport optimal martingale

## Transport optimal martingale

On considère une option call portant sur un actif sous-jacent. Sa fonction de prix est donnée par la fonction  $s$ . On considère le prix du sous-jacent aux instants 1 et 2, respectivement  $s_1$  et  $s_2$ . On définit  $\mathcal{M}(\mu_1, \mu_2) = \{\mathbb{Q} : \text{mesure martingale telle que } s_i \sim \mu_i\}$  et on veut trouver la solution du problème de maximisation (upper price bound problem) suivant

$$\max_{\mathbb{Q} \in \mathcal{M}(\mu_1, \mu_2)} \mathbb{E}_{\mathbb{Q}}(F(s_1, s_2))$$

Avec  $F$  la fonction de coût associée.

# Problème discret

## Paramètres

On considère que les distributions marginales sont discrètes :

$$\mu = \sum_{i=1}^m \alpha_i \delta_{x_i}$$

avec pour tout  $i = 1, 2, \dots, m$  :  $\alpha_i > 0$ ,  $x_i \in \mathbb{R}$  et  $\sum_{i=1}^m \alpha_i = 1$ .

$$\nu = \sum_{j=1}^n \beta_j \delta_{y_j}$$

avec pour tout  $j = 1, 2, \dots, n$  :  $\beta_j > 0$ ,  $y_j \in \mathbb{R}$  et  $\sum_{j=1}^n \beta_j = 1$ .

# Problème discret

## Problème discret

On peut utiliser la condition martingale pour établir le problème discret :

$$\left\{ \begin{array}{l} \max \sum_{i=1}^m \sum_{j=1}^n q_{i,j} c_{i,j} \\ q_{i,j} \geq 0 \\ \sum_{j=1}^n q_{i,j} = \alpha_i \\ \sum_{i=1}^m q_{i,j} = \beta_j \\ \sum_{j=1}^n q_{i,j} (y_j - x_i) = 0 \end{array} \right.$$

# Résolution du problème

## Contraintes

On définit ainsi nos espaces de contraintes :  $C_1, C_2, C_3$  :

$$C_1 = \{P \in \mathbb{R}_+^{m \times n} : \sum_{j=1}^n P_{i,j} = \alpha_i, 1 \leq i \leq m\}$$

$$C_2 = \{P \in \mathbb{R}_+^{m \times n} : \sum_{i=1}^m P_{i,j} = \beta_j, 1 \leq j \leq n\}$$

$$C_3 = \{P \in \mathbb{R}_+^{m \times n} : \sum_{j=1}^n y_j P_{i,j} = x_i \alpha_i, 1 \leq i \leq m\}$$

# Résolution du problème

## Démarche

On va dans la suite considérer  $\mathbb{P}$  comme une matrice  $(P_{i,j})_{(i,j)}$  et nous allons nous intéresser au problème de maximisation d'une fonction linéaire  $L : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}$  définie telle que :

$$L : P \mapsto \sum_j \sum_i c_{i,j} P_{i,j}$$

## Définition

La fonction strictement concave  $H : \mathbb{R}_+^{m \times n} \rightarrow \mathbb{R}$  est définie de la manière suivante :

$$H : P \mapsto H(P) = - \sum_{i=1}^m \sum_{j=1}^n P_{i,j} \times (\log(P_{i,j}) - 1)$$



# Résolution du problème

## Définition de $L_\epsilon(P)$

Pour tout  $\epsilon > 0$  et tout  $P \in \mathbb{R}_+^{m \times n}$  on va définir :

$$L_\epsilon(P) = L(P) + \epsilon \times H(P)$$

Alors :

- $P \mapsto H(P)$  est strictement convexe
- $P \mapsto L_\epsilon(P)$  est strictement concave

Ces deux points nous permettent dans la suite d'estimer le point qui annule la dérivée et d'utiliser la méthode des multiplicateurs de Lagrange.

# Résolution du problème

## Comparaison de $L(P)$ et $L_\epsilon(P)$

On peut montrer qu'il est possible d'encadrer l'erreur faite en approchant  $L(P)$  par  $L_\epsilon(P)$  :

$$\left| \max_{P \in C_1 \cap C_2 \cap C_3} L_\epsilon(P) - \max_{P \in C_1 \cap C_2 \cap C_3} L(P) \right| \leq C\epsilon$$

Avec  $C = 1 - \sum_{i=1}^m \alpha_i \log(\alpha_i) + m \log(n)$  une constante.  
Cette erreur dépend de  $\epsilon$  que l'on peut réduire pour augmenter la précision.

# Divergence de Kullbeck-Leibler

## Divergence de Kullbeck-Leibler

On définit la divergence de Kullbeck-Leibler de la manière suivante pour  $P, q \in \mathbb{R}_+^{m \times n}$  :

$$KL(P|q) = \sum_i \sum_j P_{i,j} (1 - \log(\frac{P_{i,j}}{q_{i,j}}))$$

# Modification du problème

## Nouvelle formulation des fonctions du problème

Dans la suite, nous définissons pour tout  $i = 1, 2, \dots, m$  et tout  $j = 1, 2, \dots, n$  :

$$q_{i,j} = \exp\left(\frac{C_{i,j}}{\epsilon}\right) \iff C_{i,j} = \epsilon \log(q_{i,j})$$

On peut alors montrer que la fonction  $L_\epsilon$  peut s'exprimer sous la forme :

$$L_\epsilon(P) = \epsilon KL(P|q)$$

## Optimisation de la divergence

On peut alors modifier notre problème d'optimisation à  $\epsilon$  fixé :

$$\max_{P \in C_1 \cap C_2 \cap C_3} KL(P|q)$$

# Projection et suite de compacts

## Définition de la projection

Soient  $P_* \in \mathbb{R}_+^{m \times n}$  et  $k \in \mathbb{N}$ , on définit la projection de  $P_*$  sur  $C_k$  par :

$$\text{Proj}_{C_k}(P_*) = \operatorname{argmax}_{P \in C_k} KL(P|P_*)$$

## Suite de compacts

Pour tout  $k \in \mathbb{N}^*$ , on définit la suite de compacts  $(C_k)_{k \geq 1}$  :

$$\begin{cases} C_1, C_2, C_3 \\ C_{k+3} = C_1 \\ C_{k+4} = C_2 \\ C_{k+5} = C_3 \end{cases}$$

# Algorithme de projection itérative

## Théorème de projection itérative de Bregman

Soit  $(P^{(k)})_{k \geq 0}$  une suite dans  $\mathbb{R}_+^{m \times n}$  définie par

- $P^{(0)} = q \in \mathbb{R}_+^{m \times n}$
- $\forall k \geq 1, P^{(k)} = \text{Proj}_{C_k}(P^{(k-1)})$

Alors

$$\lim_{k \rightarrow \infty} P^{(k)} = \operatorname{argmax}_{P \in C_1 \cap C_2 \cap C_3} KL(P|q)$$

# Calcul explicite des projecteurs

## Méthode

Par exemple, pour l'espace  $C_1$ , on pose le multiplicateur de Lagrange :

$$L(P, q, \lambda_1, \dots, \lambda_m) = KL(P|q) + \sum_{i=1}^m \lambda_i \times \left( \sum_{j=1}^n P_{i,j} - \alpha_i \right)$$

On dérive ensuite par rapport à chaque variable pour dériver l'expression du projecteur sur  $C_1$ . On procède de la même manière pour les autres espaces de contrainte.

# Calcul explicite des projecteurs

## Espace $C_1$

Pour tout  $i = 1, 2, \dots, m$  et  $j = 1, 2, \dots, n$  le maximiseur est :

$$P_{i,j}^{(l+1)} = \frac{\alpha_j P_{i,j}^{(l)}}{\sum_{j=1}^n P_{i,j}^{(l)}}$$

## Espace $C_2$

Pour tout  $i = 1, 2, \dots, m$  et  $j = 1, 2, \dots, n$  le maximiseur est :

$$P_{i,j}^{(l+1)} = \frac{\beta_j P_{i,j}^{(l)}}{\sum_{i=1}^m P_{i,j}^{(l)}}$$



# Calcul explicite des projecteurs

## Espace $C_3$

Pas d'expression fermée. La solution est trouvable en calculant les coefficients  $\lambda_i$  de sorte que :

$$P_{i,j} = q_{i,j} \exp(\lambda_i y_j)$$

On calcule les  $\lambda_i$  en tant que racines d'un polynôme déterminé par les données du problème. On peut par exemple utiliser la méthode de Newton.

# Implémentation du schéma numérique

- Utilisation de Python pour implémentation de la méthode numérique
- Spécifier le nombre de points de la discrétisation  $n$  et  $m$
- Spécifier les poids associées à chaque Dirac par les coefficients  $\alpha_i$  et  $\beta_j$
- Les points de discrétisation sont également à être fixés dans notre algorithme
- Choix d'une valeur initiale

# Implémentation du schéma numérique

Dans la suite nous prendrons les valeurs suivantes :

## Choix des paramètres

- $P^{(0)} = (\frac{1}{mn})_{i,j}$
- $\forall i = 1, 2, \dots, m : \alpha_i = \frac{1}{m}$
- $\forall j = 1, 2, \dots, n : \beta_j = \frac{1}{n}$
- $\forall i = 1, 2, \dots, m : x_i = \frac{i}{m}$
- $\forall j = 1, 2, \dots, n : y_j = \frac{j}{n}$

# Implémentation du schéma numérique

## Conclusion

- Code fonctionnel : convergence vers une solution
- Vitesse plus ou moins rapide selon le choix de précision
- Le temps de résolution reste relativement important pour de grandes matrices

# Implémentation directe

## Scipy et CVXPY

- Utilisation des modules Scipy et CVXPY pour résoudre directement le système linéaire
- Scipy : sortie du domaine de définition de la variable, pas d'optimalité
- CVXPY : succès pour matrices carrées, temps de convergence très rapide
- CVXPY : échec pour matrices rectangulaires. Tentative en annexe 7 de corriger le code en agrandissant les matrices et vecteurs mais pas de succès

# Conclusion

## Conclusion




- Cheminement nous a permis d'effectuer une résolution numérique avec un langage de programmation
- Implémentation du schéma réussie
- Schéma plus consistant mais moins rapide que résolution directe
- Ouverture : trouver une méthode pour réduire le nombre d'itérations et l'ordre de convergence

# Compléments

## Etudes complémentaires dans le rapport

- Discussion sur l'erreur de projection sur chaque espace de contrainte
- Discussion de l'implémentation de la méthode de Newton
- Comparaison des méthodes de recherche de zéro
- Vectorialisation du problème
- Annexes 1 à 7

# Bibliographie I

-  Steven Diamond and Stephen Boyd.  
CVXPY : A Python-embedded modeling language for convex optimization.  
*Journal of Machine Learning Research*, 17(83) :1–5, 2016.
-  Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd.  
A rewriting system for convex optimization problems.  
*Journal of Control and Decision*, 5(1) :42–60, 2018.
-  Akshay Agrawal, Steven Diamond, and Stephen Boyd.  
Disciplined geometric programming.  
*Optimization Letters*, 13(5) :961–976, 2019.



# Bibliographie II



Akshay Agrawal and Stephen Boyd.  
Disciplined quasiconvex programming.  
*Optimization Letters*, 2020.  
To appear.



Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J. Zico Kolter.  
Differentiable convex optimization layers.  
In *Advances in Neural Information Processing Systems*, pages 9558–9570, 2019.



Akshay Agrawal and Stephen Boyd.  
Differentiating through log-log convex programs.  
*arXiv*, 2020.

# Bibliographie III



Wankere R. Mekwi.

Iterative methods for roots of polynomials.

*Trinity*, 2001.



Méthode de halley.

*Wikipedia*.