TD6

19 février 2004

Exercice 1. Allocation optimale en présence de contraintes.
On rappelle qu'un programme d'optimisation s'écrit comme

$$\mathcal{P} = \begin{cases} \max_{x \in C} f(x) & \\ g_i(x) = \alpha_i & i \in E \\ g_i(x) \leq \alpha_i & i \in I \end{cases}$$

On dit que le programme est convexe si
– C est un sous-ensemble convexe de R^n,
– f est une fonction différentiable convexe sur C,
– pour $i \in E$, $g_i(x) = (h_i, x) + \beta_i$ est une fonction affine sur C,
– pour $i \in I$, g_i est une fonction convexe différentiable sur C.

En supposant que
(*) il existe $\tilde{x} \in C$ tel que, pour tout $i \in I$, $g_i(\tilde{x}) < \alpha_i$,
nous pouvons énoncer le théorème de Kühn et Tucker :
Theorème 1. *$x \in C$ est solution su programme convexe $\mathcal{P}$ si et seulement s'il existe $\lambda \in R^{I \cup E}$ tel que*

$$\begin{aligned} f'(x) &= \sum_{i \in I \cup E} \lambda_i g_i'(x) \\ g_i(x) &= \alpha_i, \ \textit{si } i \in E \textit{ et } g_i(x) \leq \alpha_i, \ \textit{si } i \in E, \\ \lambda_i &\geq 0, \ \textit{si } i \in I \textit{ et } \lambda_i = 0 \textit{ si } i \in I \textit{ et } g_i(x) < \alpha_i. \end{aligned}$$

1. On va considérer le problème convexe suivant :

$$\begin{cases} \max_{x \in C} f(x) & \\ m_i \leq x_i, & i \in I \end{cases}$$

où $I \subseteq \{1, \ldots, n\}$. On suppose aussi que la $(*)$ est vrai de f et C. En utilisant le théorème de Kühn-Tucker, montrer le résultat suivant :

Proposition 2. *x est une solution de $\mathcal{P}$ si et seulement si $x \in C$ et :*

$$\frac{\partial f}{\partial x_i}(x) \text{ est } \begin{cases} \leq 0, & i \in I \text{ et } x_i = m_i \\ = 0, & \text{dans le autres cas.} \end{cases}$$

2. Utiliser le résultat précèdent pour proposer un algorithme de résolution du problème de l'allocation optimale dans les cas que les montants des actifs risqués ne soient pas à découvert :

$$\begin{cases} \max_{\alpha_0, \alpha \in R^{1+n}} f(\alpha_0, \alpha) \\ w = \alpha_0 + \alpha' p_t \\ k_i \leq \alpha_i p_{t,i}, \qquad\qquad i = 1, \ldots, n\,, \end{cases}$$

où

$$f(\alpha_0, \alpha) = E_t(w_{t+1}) - \frac{A}{2} V_t(w_{t+1})\,.$$

Exercice 2. Inversion de matrices

Soit $A = \big(a_{i,j}\big)$ une matrice $n \times n$; on veut calculer sont inverse $B = A^{-1}$. La formule usuelle de l'inversion donne

$$b_{i,j} = \frac{(-1)^{i+j}}{\det A} \det A_{i,j}$$

où la matrice $A_{i,j}$ est obtenue de A en supprimant sa ligne i et sa colonne j.

– Donner une estimation du coût du calcul de A^{-1} en fonction du nombre de sommations et multiplications utilisés.

Supposons maintenant que A est une matrice inférieure : si $i < j$, alors $a_{i,j} = 0$.

1. Observer que $\det(A) = \prod_i a_{i,i}$.
2. Observer que si $j \leq i$ alors $A_{i,j}$ est une matrice inférieure. Est ce que cette condition est nécessaire ? Observer que si $j < i$ alors $\det(A_{i,j}) = 0$.
3. Donner une estimation du temps nécessaire pour résoudre l'équation $Ax = b$, où A est une matrice inférieure.
4. Proposer un algorithme pour calculer A^{-1}, A étant une une matrice inférieure. Donner une estimation du coût du calcul de cet algorithme. En déduire un algorithme similaire pour une matrice supérieure.

On veut chercher une méthode pour décomposer une matrice $C = AB$ de façon que : A est une matrice inférieure telle que $a_{i,i} = 1$ pour $i = 1, \ldots, n$, et B est une matrice supérieure. De cette façon on obtiendra un méthode moins coûteux pour inverser la matrice C. Observons que

$$c_{i,j} = \begin{cases} \sum_{k \leq i} a_{i,k} b_{k,j} & i \leq j \\ \\ \sum_{k \leq j} a_{i,k} b_{k,j} & i > j\,. \end{cases}$$

De cette façon, on peut envisager l'algorithme suivant :

```
pour j = 1, ..., n faire
    pour i = 1, ..., n faire
        si i ≤ j
            b_{i,j} = c_{i,j} − Σ_{k<i} a_{i,k} b_{k,j}
        si j > i
            a_{i,j} = (1/b_{j,j}) (c_{i,j} − Σ_{k<j} a_{i,k} b_{k,j})
```

– Faire marcher l'algorithme sur la matrice

$$C = \begin{pmatrix} 1 & 0 & 3 \\ 4 & 2 & 1 \\ 2 & 1 & 1 \end{pmatrix}$$

– Donner une mesure de complexité pour cet algorithme.
– Peut l'algorithme échouer et pourquoi ?
– Proposer un algorithme efficace pour inverser une matrice quelconque.

Exercice 3. Bon modèle, mauvais modèle ...

Dans le TD4 – où nous avons considéré un portefeuille w comportant seulement deux actifs risqués – on a fait la supposition que le processus Y_t (Excess Gain) satisfait

$$E_t(Y_{t+1}) = \mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} \qquad V_t(Y_{t+1}) = \Omega = \begin{pmatrix} \omega_{11} & \omega_{12} \\ \omega_{21} & \omega_{22} \end{pmatrix}. \tag{1}$$

Nous allons faire la supposition que Y_t est un processus VAR(1) :

$$Y_t = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} + \begin{pmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \end{pmatrix} Y_{t-1} + \epsilon_t \qquad V(\epsilon_t) = \begin{pmatrix} \omega_{11} & \omega_{12} \\ \omega_{21} & \omega_{22} \end{pmatrix}. \tag{2}$$

– Écrire un nouvel programme sas pour le calcul de l'allocation optimale du portefeuille, qui tient compte de l'hypothèse (2). Pour cela, on pourra faire estimer à VARMAX le vecteur μ et les matrices Φ et Ω (voir les tableaux ODS Constant, ARCoef, CovInnov).
– Comparer les résultat de la gestion du mois de janvier (TD5) obtenus en faisant l'hypothèse (1) avec les résultats obtenus en faisant l'hypothèse (2). Commenter les résultats.
– Simuler un processus VAR(1) bivarié Y_t, pour t du 2 novembre 2003 jusqu'au 31 janvier 2004. A partir du processus bivarié Y_t, que l'on suppose être le processus des Excess Gains, calculer le processus bivarié p_t des prix des deux actifs risqués, étant donné un taux fixe r_t. Avec les données simulés, comparer les résultat de la gestion du mois de janvier sous l'hypothèse (1) avec les résultats sous l'hypothèse (2).