Université Paris-Dauphine
Département MIDO
DUMI2E troisième année

Optimisation numérique
Gérard Lebourg
année 2010/2011

Séance de Travaux Pratiques n°2

Objectif

Étant donné un panier de N actifs dont le vecteur des rendements moyens r et la matrice des variances-covariances Q sont connus, déterminer la composition d'un portefeuille constitué à partir de ces N actifs assurant un rendement minimal anticipé ρ en minimisant, pour ρ et c fixés, la fonction barrière logarithmique :

$$f(x,\mu,\rho) = \frac{1}{2}\, x^T Q\, x - \mu \sum_{i=1}^{N} \ln(x_i) - \mu \ln\left(r^T x - \rho\right),$$

sous la contrainte $\sum_{i=1}^{N} x_i = 1$, avec μ un paramètre réel strictement positif destiné à tendre vers zéro.

1. Préliminaires

Télécharger le fichier-archive `optinum-tp2.tgz` à l'adresse suivante :

`http://docs.ufrmd.dauphine.fr/optinum/tps/optinum-tp2.tgz`

puis extraire les fichiers qui seront utilisés durant la séance dans le répertoire `optinum-tp2`. Lancer alors MATLAB et se placer dans le répertoire `optinum-tp2`.

2. Écriture d'un oracle

Le fichier `lbpfun.m` contenu dans le répertoire `optinum-tp2` contient le modèle d'un oracle associé au critère du problème. Le code, reproduit ci-après, déclare quatre variables *globales* `Q`, `r`, `mu`, et `rho` destinées à contenir respectivement la matrice des variances-covariances des rendements du panier d'actifs considéré et le vecteur des rendements moyens, ainsi que les valeurs des paramètres μ et ρ :

```
lbpfun[f,g,H]=function(x)
global Q r mu rho
excess=r'*x-rho ;
if all(x>0)&&(excess>0) % lbpfun defined at x
   f=0.5*x'*Q*x-mu*[sum(log(x))+log(excess)] ;
   if nargout>1
      g=
      if nargout>2
         H=
      end
   end
else % lbpfun not defined at x
   f=inf ; g=r ; g(:)=NaN ; H=Q ; H(:,:)=NaN ;
end
```

Éditer le fichier `lbpfun.m`, compléter le code manquant dans l'écriture de la fonction (c'est-à-dire les définitions de `g` et de `H`), et sauvegarder.

3. Acquisition des données

1. Entrer la commande `global Q r mu rho pb`, qui déclare les variables globales `Q`, `r`, `mu`, et `rho`, ainsi q'une variable `pb` destinée à contenir une *structure* rassemblant la totalité des informations utiles sur le problème à résoudre, dont les champs seront :
   (a) `pb.crit` : une « *poignée* » (handle) `@lbpfun` sur la fonction `lbpfun`,
   (b) `pb.init` : l'initialisation `x0` choisie pour l'algorithme,
   (c) `pb.eqc` : une structure `pb.eqc={pb.eqc.mat,pb.eqc.rhs}` contenant la contrainte d'égalité
   `pb.eqc.mat*x=pb.eqc.rhs`,
   soit encore, pour le problème traité, simplement `pb.eqc.mat=[1,1,...,1]`, et `pb.eqc.rhs=1`.

Trois jeux de données pourront être utilisés. Ils sont stockés dans un fichier `data.m` contenant une *structure* `stocks`. Les trois jeux de données `stocks(pos)`, `pos=1,2,3`, peuvent être automatiquement chargés en utilisant la commande `initialize(pos)` avec la valeur 1, 2, ou 3 de l'argument `pos`. Cette commande affecte une valeur aux variables globales `Q`, `r`, `mu`, `rho`, et `pb`. Les valeurs par défaut de `rho` et `mu` sont `rho=0` [(1)], et `mu=0.1`. Elles peuvent être modifiées à tout moment par une nouvelle affectation

2. Entrez par exemple la commande `initialize(1)` qui charge un premier jeu de données : c'est un exemple « jouet » de dimension trois.
3. Entrez alors les commandes `Q`, puis : `r` – sans la faire suivre d'un point-virgule – pour « voir » les valeurs des données correspondantes.

4. GUMP : une boîte à outils commode

Pour pouvoir comparer simplement l'application de différentes procédures de recherche linéaire et/ou de différents choix de la direction de descente, on a codé une procédure `gump` (`GUMP` est l'acronyme pour **G**eneralized **U**nconstrained **M**inimization **P**rocedure) prenant pour argument deux structures :

- `pb`, qui contient le problème à résoudre,
- `algo`, qui contient la description de l'algorithme utilisé.

1. Pour comprendre l'utilisation de `gump`, entrez d'abord la commande `help gump`, et lire attentivement l'aide en ligne retournée par MATLAB.
2. Entrez alors – sans la faire suivre d'un point virgule – la commande `algo=gradopt`, qui charge dans la structure `algo` l'algorithme de Gradient à pas optimal, puis la commande – suivie cette fois d'un point-virgule – `pts=gump(pb,algo,100)`.
3. Suivre le comportement de l'algorithme avec la commande `plots(pts)`, qui trace différentes quantités permettant de juger de la convergence. Vous pourrez tracer ces quantités par la suite pour observer les différences de convergence des algorithmes.
4. Récupérez le dernier point calculé par l'algorithme en tapant la commande `x=pts(:,end)`, qui extrait la dernière colonne de la matrice `pts`[(2)], et vérifiez à l'aide des variables globales `r` et `Q` que le résultat obtenu fournit un portefeuille de rendement anticipé supérieur à 5, et de risque (écart-type ?) associé inférieur à 2.

1. 0n cherche donc *a priori* le portefeuille de variance minimale sans contrainte de rendement.
2. Rappelez-vous la signification des deux points.

Bien, vous avez découvert le principe de diversification du risque qui a fait la fortune de H. Markowitz, lauréat du prix Nobel d'économie en 1990[3]. Passons maintenant aux choses sérieuses...

5. Variations sur la recherche linéaire

1. Entrez la commande `initialize(3)` qui charge dans `pb` le problème de dimension 12 associé à `stocks(3)`. Les données correspondantes sont tirées de l'article : `http://www.geo.ut.ee/nbc/paper/lapsina.htm` et concernent douze actifs financiers, dont les actions les plus attractives des plus grandes compagnies – les *"Blue Chips"* – du marché russe en 1997[4].
2. Entrez la commande `gump(pb,algo,100);`. Qu'observe-t-on ? Expliquez.
3. Entrez alors la commande `algo.stepsize=@(phi)goldensearch(phi,1e-6);` qui met la tolérance utilisée par `goldensearch` à 10^{-6}. Tapez `algo` pour visualiser le changement effectué dans la définition de l'algorithme, puis `pts=gump(pb,algo,1000);`. **Attention :** n'oubliez-pas un zéro dans `itermax` et n'oubliez pas le point-virgule !
4. Vous avez trouvé un résultat raisonnable, mais pouvait-on faire mieux ? Entrez maintenant la commande `algo.stepsize=@backtrack`. Puis essayez à nouveau `gump(pb,algo,1000);`. Qu'en pensez-vous ?
5. À l'été 1997, à partir des estimations obtenues, quel portefeuille de variance minimale eussiez-vous conseillé de constituer avec les *Blue Chips* du marché russe ? Donnez le rendement moyen et la variance de ce portefeuille.

Maintenant, c'est à vous de jouer.

6. La procédure `backtrack` utilise l'interpolation quadratique de la fonction `phi` qui lui est passée pour argument pour trouver un pas `t` vérifiant la *condition d'Armijo* : `phi(t)< phi(0)+ alpha*t*phi'(0)`.
   (a) Déterminez l'équation de la parabole passant par les points `(0,phi(0))` et `(t,phi(t))`, tangente en `t=0` au graphe de `phi`, puis l'abscisse du sommet de cette parabole en fonction de `phi(t)`, `phi(0)` et `phi'(0)`.
   (b) Éditez ensuite le fichier `backtrack.m` et expliquez pourquoi la procédure finit nécessairement par trouver un pas `t` vérifiant la condition d'Armijo (**indication** : vérifier que l'abscisse du sommet de la parabole interpolante reste toujours strictement inférieure à `t/2(1-alpha))`.
7. Entrez la commande `algo.stepsize=@Wolfe`, puis essayez à nouveau `pts=gump(pb,algo,2000);`. Éditez le fichier `Wolfe.m` et expliquez pourquoi la procédure finit toujours par trouver un pas `t` vérifiant la règle de Wolfe.
8. Pour finir, on sort le lapin du chapeau. Entrez la suite de commandes :

```
algo.memory=1;
algo.searchdir=@BarzilaiBorwein;
algo.stepsize=@backtrack;
algo.stoppingtest=@gradtest;
pts=gump(pb,algo,1000);
```

   Qu'observe-t-on ? Éditez le fichier `BarzilaiBorwein.m` et expliquez ce qui a changé dans la construction de l'algorithme.

3. En démontrant dans sa thèse, en 1952, qu'il était possible de réduire le risque par diversification, il a fait entrer le loup des mathématiques dans la bergerie de la Finance.

4. Les estimations de la moyenne et de la matrice de dispersion du vecteur des rendements moyens de ces douze actifs sont obtenues à partir du relevé des rendements hebdomadaires de ces douze actions du 14.08.1996 au 13.08.1997.

6. Variations sur la direction de recherche

1. Éditez le fichier `Cauchy.m` et expliquez le rôle de la matrice `trial.M` apparaissant dans le code de ce fichier (**indication** : souvenez-vous que l'on minimise en fait $G(u) = F(x + N\,u)$).
2. Comment calculer alors la direction de recherche du gradient conjugué ? (**indication** : la formule générale d'actualisation est $\mathtt{d}_{k+1} = -g_{k+1} + \|g_{k+1}\|^2/\|g_k\|^2\, d_k$. Le problème est de prendre en compte l'existence éventuelle de contraintes d'égalité linéaires. Par conséquant, la matrice `trial.M` doit intervenir quelquepart...).
3. Donnez le code d'une fonction `FR` prenant pour argument la structure `trial` et retournant la direction de descente de Fletcher-Reeves (**attention** : la première direction de recherche doit être la direction de Cauchy et `trial.old.g` vaut zéro à l'initialisation.).
4. Écrire alors une fonction `gradconj` sur le modèle de `gradopt`. N'oubliez pas de mettre `algo.memory` à un.
5. Entrez alors la suite de commandes : `initialize(2); algo=gradconj; pts=gump(pb,algo,1000);`, puis corrigez la valeur de `algo.stepsize` en augmentant progressivement la précision de `goldensearch` et recommencez. Qu'observe-t-on ? Expliquez.
6. Entrez la commande `algo=quasinewt` – sans la faire suivre d'un point-virgule – pour voir comment est défini l'algorithme, puis, à nouveau `pts=gump(pb,algo,1000);` et comparez avec les résultats précédents.
7. Expliquez finalement comment définir une fonction MATLAB `newtopt` permettant de charger dans la structure `algo` l'algorithme de Newton par la commande `algo=newtopt`. Justifiez le contenu de chacun des champs de la structure `algo` résultante. Détaillez en particulier le traitement de la contrainte d'égalité. Expérimentez.

7. Conclusions

Quelles sont vos conclusions ? Comment choisiriez-vous de construire un algorithme de sélection de portefeuille fondé sur la minimisation d'une barrière logarithmique ? (Le résultat peut dépendre de la dimension du problème à résoudre). Justifiez vos choix.